1. POGLAVLJE

Uvod u Visual Basic

Ovo poglavlje sadrži informacije o instaliranju Visual Basica na vaš računalni sustav, dodavanju ili brisanju dijelova Visual Basica te izvore za učenje ili dobivanje prikladne pomoći sa Visual Basicom.

Sadržaj

- Dobro došli u Visual Basic
- Instalacija Visual Basica
- Dobivanje pomoći tijekom rada

Dobro došli u Visual Basic

Dobro došli u Microsoftov Visual Basic, najbrži i najlakši način stvaranja aplikacija za Microsoft Windowse. Bez obzira na to jeste li iskusni profesionalac ili ste potpuni početnik u Windows programiranju, Visual Basic vas opskrbljuje s kompletnim setom alata koji pojednostavljuju brz razvoj aplikacija.

Dakle, što je to Visual Basic? Riječ "Visual" (vizualan, vidni) odnosi se na metodu korištenu za stvaranje grafičkog korisničkog sučelja (graphical user interface, GUI). Umjesto pisanja bezbrojnih linija programskog koda za opisivanje ponašanja i položaja elemenata aplikacije, jednostavno ćete postaviti unaprijed definirane objekte na svoje mjesto na ekranu. Ako ste ikad koristili neku aplikaciju za crtanje kao što je Paint, imate najveći dio vještina potrebnih za stvaranje efikasnog korisničkog sučelja.

Riječ "Basic" označuje programski jezik BASIC (Beginners All–Purpose Symbolic Instruction Code), programski jezik koji koristi više programera nego bilo koji drugi jezik u povijesti računala. Visual Basic razvio se iz originalnog programskog jezika BASIC i sada sadrži više stotina izraza, naredbi i funkcija, od kojih je najveći dio direktno povezan sa Windows grafičkim sučeljem. Početnici mogu stvarati korisne aplikacije poznavajući svega nekoliko naredbi, iako snaga ovog jezika omogućuje profesionalcima da postignu sve što se može postići koristeći bilo koji drugi Windows programski jezik.

Programski jezik Visual Basic nije jedinstven samo za Visual Basic. Visual Basic programski sustav, Applications Edition uključen u Microsoft Excel, Microsoft Access, te puno drugih Windows aplikacija koristi taj isti programski jezik. Visual Basic

Scripting Edition (VBScript) je široko korišten jezik za izradu skripti i dio je Visual Basic programskog jezika. Investiranje u učenje Visual Basica omogućit će vam pristup i u ta područja.

Bez obzira na to je li vaš cilj stvaranje male uslužne aplikacije za sebe ili svoje kolege, velikog sveobuhvatnog sustava ili čak aplikacija koje ćete distribuirati širom svijeta putem Interneta, Visual Basic je alat koji trebate.

- Pristup podacima omogućuje vam kreiranje baza podataka, gotovih aplikacija i dijelova za veće sustave za sve popularnije formate baza podataka, uključujući Microsoftov SQL Server i ostale napredne baze.
- ActiveX tehnologija omogućuje vam funkcionalnost koju pružaju druge aplikacije, kao što je aplikacija za obradu teksta Microsoft Word, tablični kalkulator Microsoft Excel, te druge Windows aplikacije. Možete čak i automatizirati aplikacije i objekte kreirane korištenjem Professional ili Enterprise verzije Visual Basica.
- Internet sposobnosti omogućuju jednostavan pristup dokumentima i aplikacijama putem Interneta ili intraneta iz vaše aplikacije te dopuštaju stvaranje Internet server aplikacija.
- Vaša gotova aplikacija je stvarna .exe datoteka koja koristi Visual Basic Virtual Machine dodatak za rad i možete ju slobodno distribuirati.

Izdanja Visual Basica

Visual Basic je dostupan u tri verzije. Svaka je opremljena tako da zadovolji određen krug razvojnih zahtjeva.

- Visual Basic Learning izdanje omogućuje programerima lako stvaranje moćnih aplikacija za Microsoft Windows i Windows NT operativne sustave. Uključuje sve interne kontrole te kontrole za nadzor nad mrežom, etiketama i bazama podataka. Dokumentacija koja dolazi s ovom verzijom uključuje Learn VB Now CD te CD-e sa Microsoft Developer Network (MSDN) datotekama koje sadrže punu dokumentaciju.
- Professional izdanje pruža računalnim profesionalcima potpuno opremljen komplet alata za razvoj rješenja za druge. Uključuje sve osobine Learning izdanja te dodatne ActiveX kontrole, Internet Information Server Application Designer, ugrađene Visual Database Tools i Data Environment, Active Data Objects te Dynamic HTML Page Designer. Dokumentacija koja dolazi s Professional izdanjem uključuje i knjigu Visual Studio Professional Features i MSDN CD-e s punom dokumentacijom.
- Enterprise izdanje omogućuje profesionalcima uz pomoć razvojnih timova stvaranje snažnih aplikacija za daljnju distribuciju. Uključuje sve osobine Professional izdanja te Back Office alate kao što su SQL Server, Microsoft Transaction Server, Internet Information Server, Visual SourceSafe, SNA Server, i druge. Štampana dokumentacija koja dolazi s Enterprise izdanjem uključuje knjigu Visual Studio Enterprise Features te MSDN CD-e sa punom dokumentacijom.

Instaliranje Visual Basica

Visual Basic instalirate na svoje računalo korištenjem Setup aplikacije. Setup aplikacija instalira Visual Basic i ostale dijelove programskog paketa s CD ROM-a na vaš hard disk. Ona također instalira i datoteke potrebne za pregled dokumentacije na MSDN CD-ima. Ako želite, na svoje računalo možete instalirati samo Visual Basic dokumentaciju i primjere.

Važno Ne možete samo prekopirati datoteke sa CD ROM–a na vaš hard disk i pokrenuti Visual Basic. Morate upotrijebiti Setup aplikaciju koja će dekomprimirati i instalirati datoteke u odgovarajuće direktorije.

- Prije pokretanja Setup aplikacije – Stvari koje treba prethodno provjeriti
- Postavljanje Visual Basica – Upute za instaliranje Visual Basica

Prije pokretanja Setup aplikacije

Prije instaliranja Visual Basica, provjerite zadovoljava li vaše računalo minimalne zahtjeve i pročitajte Readme datoteku koja se nalazi u glavnom direktoriju na instalacijskom disku.

Provjera hardverskih i sistemskih zahtjeva

Za pokretanje Visual Basica morate imati odgovarajući hardver te softver već instaliran na svoje računalo. Sistemski zahtjevi su sljedeći:

- Microsoft Windows 95 operativni sustav ili viši, Microsoft Windows NT Workstation 4.0 (Service Pack 3 preporučen) operativni sustav ili viši.
- 486DX/66 MHz ili jači procesor (Pentium ili jači preporučen) ili bilo koji Alpha procesor sa Microsoft Windows NT Workstation 4.0 operativnim sustavom.
- CD-ROM pogon.
- monitor VGA ili jače rezolucije podržan od Microsoft Windowsa.
- 16 MB RAM memorije za Windowse 95/98, 32 MB RAM memorije za Windows NT Workstation
- miš ili sukladan uređaj.

Za više informacija Za više detalja o zahtjevima pogledajte “Sistemske zahtjeve za Visual Basic aplikacije” u Dodatku A, “Specifikacije, ograničenja i vrste datoteka Visual Basica”.

Pročitajte Readme datoteku

U Readme datoteci nalaze se sve promjene u dokumentaciji Visual Basica do kojih je došlo nakon ispisa. Ovu datoteku možete pronaći odabirom Read Me First opcije na početnom ekranu kod instalacije, ili u glavnom direktoriju CD ROM-a. Može joj se pristupiti i s početnog ekrana Visual Basica u dokumentaciji. Provjerite prvi dio ove datoteke gdje su navedene nove informacije o instaliranju Visual Basica.

Postavljanje Visual Basica

Kad pokrenete Setup aplikaciju, bit će stvoren direktorij za Visual Basic; nakon toga možete odabrati koje dijelove Visual Basica želite instalirati.

Izuzevši datoteke operativnog sustava u \Os direktoriju, datoteke na CD-u nisu komprimirane, tako da se mogu koristiti direktno s diska. Na primjer, na disku se nalazi bezbroj alata i dijelova u \Tools direktoriju koji mogu biti pokrenuti ili instalirani direktno s CD-a.

Instaliranje sa CD-a

1. Ubacite CD disk u CD-ROM pogon.
2. Upotrijebite odgovarajuću naredbu svog operativnog sustava za pokretanje Setup aplikacije, koja se nalazi u glavnom direktoriju prvog diska. Ako je omogućena AutoPlay opcija na vašem sustavu, Setup aplikacija će se automatski pokrenuti kad ubacite disk.
3. Odaberite **Install Visual Basic 6.0**.
4. Slijedite upute za instaliranje koje će biti ispisane na ekranu.

Za više informacija Pročitajte Readme datoteku za detaljne upute o pitanjima vezanim za instaliranje Visual Basica.

Dodavanje ili brisanje dijelova Visual Basica

Možete pokrenuti Setup aplikaciju koliko god je puta potrebno. Na primjer, možete ju pokrenuti da bi reinstalirali Visual Basic u drugi direktorij, ili da bi instalirali druge dijelove Visual Basica.

Kako dodati ili obrisati dijelove Visual Basica

1. Ubacite CD disk u CD-ROM pogon.
2. Upotrijebite odgovarajuću naredbu vašeg operativnog sustava za pokretanje Setup aplikacije, koja se nalazi u glavnom direktoriju CD-a. Ako je omogućena AutoPlay opcija na vašem sustavu, Setup aplikacija će se automatski pokrenuti kad ubacite disk.
3. Odaberite **Custom** gumb u **Microsoft Visual Basic 6.0 Setup** dijaloškom okviru.

4. Odaberite dijelove koje želite instalirati (ili označite dijelove koji će biti deinstalirani) u **Options** okviru s popisom **Custom** dijaloškog okvira.
5. Slijedite upute za instaliranje koje će biti ispisane na ekranu.

Pokretanje Visual Basica

Jednom kad završite postupak instaliranja, možete pokrenuti Visual Basic korištenjem Start gumba na Windows traci sa zadaćama. Ako je AutoPlay opcija omogućena na vašem sustavu, možete pokrenuti Visual Basic ubacivanjem Visual Basic CD-a.

Za više informacija Pogledajte 2. poglavlje, "Razvijanje aplikacije u Visual Basicu".

Dobivanje pomoći tijekom rada

Sistemska dokumentacija sadrži gotovo sve aspekte Visual Basica. Ona uključuje:

- Sve Visual Basic knjige, pružajući sadržajne informacije o korištenju mnoštva svojstava Visual Basica.
- Jezične upute (Language Reference), koje sadrže sveobuhvatne informacije o Visual Basic programskom okruženju i jeziku.
- Visual Basic veze, koje ukazuju na izvore Visual Basic informacija na Internetu.
- Servise podrške (Microsoft Product Support Services), sa informacijama odobivanju tehničke pomoći.

  Napomena Svu dokumentaciju možete pogledati sa MSDN CD-a (prije toga morate instalirati MSDN) ili možete dodatno instalirati Visual Basic dokumente i primjere na vaše računalo tijekom MSDN instalacije.

Dobijanje maksimuma iz pomoći

Pomoć sadrži nekoliko odlika dizajniranih tako da pronalaženje pomoći učine lakšim.

- Što je novo u Visual Basicu 6.0 (What's New In Visual Basic 6.0)?

  Iskoristite ovaj dio za brz pristup informacijama o novim i poboljšanim odlikama Visual Basica. Organiziran je po kategorijama, i pruža oko 200 veza prema opširnijim informacijama.

- Pronađi brzo (Find It Fast)

  Upotrijebite ovaj dio za sortiranje traženih područja iz dokumentacije. Informacije o debugiranju, na primjer, mogu se pronaći u raznim dijelovima dokumentacije, ovisno o vrsti projekta koju radite. Opisne veze u ovom dijelu olakšavaju traženje.

- Pregled tema (Overview topics)

  Primijenite ovaj dio za dobivanje informacija o temama u knjizi ili poglavlju prije nego što prijeđete na same teme. Pružanjem kratkog uvida u sadržaj svake teme, opisane veze u pregledu knjige, dijela i poglavlja štede vrijeme.

- Veze na srodne pojmove (See Also links)

  Odaberite ovu vezu ispod naslova teme i vidjet ćete naslove drugih tema do kojih također možete otići za dobivanje više pomoći ili srodnih informacija.

Pomoć osjetljiva na sadržaj

Puno dijelova Visual Basica je *osjetljivo na sadržaj*. Osjetljivo na sadržaj znači da možete direktno dobiti pomoć o tim područjima bez pozivanja Help izbornika. Na primjer, da bi dobili pomoć o bilo kojoj ključnoj riječi u Visual Basic programskom jeziku, postavite kursor na tu riječ u kodnom prozoru (Code window) i pritisnite F1.

Možete pritisnuti F1 iz bilo kojeg dijela Visual Basic sučelja koji je osjetljiv na sadržaj za dobivanje pomoći o tom dijelu. Dijelovi osjetljivi na sadržaj su:

- Svaki prozor u Visual Basicu (prozor sa svojstvima objekata, kodni prozor itd.)
- Kontrole u alatnom okviru
- Objekti na formi ili objekt s dokumentom
- Svojstva u prozoru sa svojstvima objekata
- Ključne riječi Visual Basica (naredbe, funkcije, svojstva, metode, događaji i posebni objekti)
- Poruke o pogreškama

Izvođenje programskih primjera iz pomoći

Puno tema o pomoći sadrže primjere sa programskim kodom kojeg možete izvoditi iz Visual Basica. Sljedeći postupak pokazuje vam kako kopirati i izvesti primjer programskog koda iz pomoći.

> Napomena Sljedeći primjer vrijedi za programski kod koji ne zadrži javne varijable.

Kako kopirati primjer programskog koda iz pomoći

1. Kreirajte novu formu biranjem stavke **Add Form** iz izbornika **Project** ili iskoristite već postojeću formu. (Za više informacija o stvaranju i korištenju formi, pogledajte 2. poglavlje “Razvoj aplikacije u Visual Basicu”).
2. Odaberite stavku **Index** u izborniku **Help**.

3. U pomoći, potražite pojam *graphics*, i odaberite temu nazvanu "FillColor Property".
4. U temi FillColor Property, kliknite na vezu skoka **Example**, postavljenu u fiksnom području blizu vrha prozora (*veza skoka* je riječ na koju možete kliknuti da bi došli do druge teme. Veze skoka su podvučene i ispisane obojenim tekstom).

   Označite potprogram koji je dio primjera. Uočite da prva naredba "Sub" označuje početak potprograma, a posljednja naredba "End Sub" označuje kraj potprograma.
5. Kliknite desnom tipkom miša i odaberite stavku **Copy** iz pomoćnog izbornika. Označeni tekst bit će kopiran u međuspremnik.
6. Vratite se na formu koju ste kreirali i dvoklikom na formu pozovite kodni prozor.
7. Postavite kursor ispod postojećeg programskog koda u kodnom prozoru.
8. U izborniku **Edit** odaberite stavku **Paste**. Kopirani primjer će se pojaviti u kodnom prozoru.
9. U izborniku **Run** odaberite stavku **Start**, ili pritisnite F5.
10. Kliknite na formu za pokretanje koda iz primjera.

    Napomena Neki primjeri programskog koda traže od vas stvaranje kontrole na formi. Za više informacija o stvaranju kontrola, pogledajte 3. poglavlje "Forme, kontrole i izbornici".

Veze za interaktivnu pomoć

Ako imate modem ili neko drugo sredstvo pristupa Internetu, dostupne su vam dodatne informacije o Visual Basicu.

Microsoftova Web stranica

Microsoftova Web stranica sadrži nekoliko područja koja su zanimljiva Visual Basic programerima. Polazna stranica Visual Basica nalazi se na http://www.microsoft.com/vbasic/. Podaci dostupni na ovom mjestu sadrže:

- Ažurirane informacije o novim svojstvima, izdanjima, srodnim proizvodima, seminarima i posebnim događanjima.
- Dodatne informacije o svojstvima Visual Basica, uključujući predloške, savjete, priručnike i sredstva za vježbanje.
- Preuzimanje novih datoteka uključujući ažurirane programske i pomoćne datoteke, upravljačke aplikacije i ostale datoteke vezane uz Visual Basic.

  Savjet Microsoftova Web stranica također sadrži i posebno područje za registrirane vlasnike (Owner's Area) koje sadrži puno besplatnih primjera, dijelova, alata i drugoga. Zašto ne odsurfate na http://www.microsoft.com/vbasic/owners/ i odmah ne registrirate svoju kopiju Visual Basica?

Kako pristupiti Microsoftovoj Web stranici

1. Odaberite stavku **Microsoft on the Web** iz izbornika **Help**.
2. Odaberite odgovarajuću stavku o podizbornicima.

   Napomena Morate imati instaliran Internet preglednik i uspostavljenu vezu s Internetom da bi ove opcije radile. Neki sadržaji na Microsoftovoj Web stranici su prilagođeni Microsoft Internet Exploreru i možda neće biti potpuno vidljivi drugim preglednicima. Odavde možete preuzeti posljednju verziju Internet Explorera.

Microsoftovi servisi za podršku proizvoda

Microsoft nudi razne oblike podrške kako bi vam pomogao da dobijete najviše od Visual Basica.

Ako imate pitanje o ovom proizvodu, najprije pogledajte u dostupnu dokumentaciju. Ne pronađete li odgovor, kontaktirajte Microsoftove servise za podršku.

Servisi za podršku dostupni su iz Sjedinjenih Američkih Država te pomoću podružnica širom svijeta. Za kompletne detalje, pogledajte stavku Technical Support u izborniku Help.

Recite nam što mislite

Microsoft se obvezao na pružanje najboljih mogućih proizvoda svojim kupcima. Sa svakom novom verzijom, Visual Basic se razvijao udovoljavajući novim zahtjevima Windows programera.

Uvijek nas zanima čuti ivaše mišljenje. Ako imate bilo kakve prijedloge ili komentare glede unapređivanja ili svojstava koje biste željeli vidjeti u budućim verzijama Visual Basica, dajte nam do znanja. Svoje savjete možete poslati putem e-pošte na *vbwish@microsoft.com* ili kontaktirajući s podružnicom Microsofta u Hrvatskoj.

Razvijanje aplikacije u Visual Basicu

Potrebno je samo nekoliko minuta za stvaranje vaše prve Visual Basic aplikacije. Najprije trebate stvoriti korisničko sučelje "povlačenjem" kontrola na formu kao što su okviri s tekstom i naredbeni gumbi. Sljedeći korak je određivanje svojstava forme i kontrole kojima ćete odrediti vrijednosti kao sadržaj, boja i veličina. Na kraju, potrebno je napisati programski kod koji će oživjeti aplikaciju. Osnovni koraci koje ćete poduzeti pri stvaranju prve aplikacije pokazat će vam načela koja ćete koristiti pri razvojima sljedećih aplikacija.

Ovo poglavlje pruža pregled postupka razvijanja aplikacije, opisuje termine i vještine koje trebate za korištenje Visual Basica, i vodi vas korak po korak kroz nekoliko jednostavnijih aplikacija.

Sadržaj

- Pojmovi Visual Basica
- Elementi ugrađene razvojne okoline
- Vaša prva Visual Basic aplikacija

Primjer aplikacije: Firstapp.vbp

Neki primjeri programskog koda u ovom poglavlju preuzeti su iz primjera aplikacije Firstapp.vbp čiji ispis je dan u direktoriju Samples.

Pojmovi Visual Basica

Za razumijevanje postupka razvijanja aplikacije, korisno je razumjeti neke ključne zamisli na temelju kojih je kreiran Visual Basic. Budući da je Visual Basic programski jezik za Windows okruženje, nužna je sličnost s Windows okolinom. Ako ste početnik u Windows programiranju, trebate biti svjesni nekih temeljnih razlika između programiranja pod Windowsima u usporedbi s drugim okruženjima.

Kako Windowsi rade: prozori, događaji i poruke

Kompletna rasprava o sustavu rada Windowsa zahtijevala bi cijelu knjigu. Potpuno razumijevanje svih tehničkih detalja nije potrebno. Pojednostavljena verzija rada Windows operativnog sustava uključuje tri osnovna pojma: prozore, događaje i poruke.

Shvatite prozor jednostavno kao pravokutno područje s vlastitim granicama. Vjerojatno ste već upoznali nekoliko različitih vrsta prozora: prozor Explorera u Windowsima, prozor s dokumentom u vašoj aplikaciji za obradu teksta ili dijaloški okvir koji se pojavljuje da vas podsjeti na neki sastanak. Iako su to najčešći primjeri, zapravo postoji puno drugih vrsta prozora. Naredbeni gumb je prozor. Ikone, okviri s tekstom, gumbi izbora i trake s izbornicima su također prozori.

Operativni sustav Microsoft Windows upravlja mnoštvom tih prozora dodjeljujući svakome od njih jedinstven identifikacijski broj (window handle ili hWnd). Sustav neprekidno prati svaki od tih prozora tražeći znakove aktivnosti ili događaja. Događaji se mogu pojaviti zahvaljujući akciji korisnika kao što je klik mišem ili pritisak na tipku, tijekom programirane kontrole, ili čak kao rezultat akcije nekog drugog prozora.

Svaki put kad se pojavi događaj, uzrokuje poruku koja se šalje operativnom sustavu. Sustav obrađuje poruku i prosljeđuje je drugim prozorima. Svaki prozor zatim može poduzeti odgovarajuću akciju temeljenu na vlastitim uputama kako bi obradio tu određenu poruku (na primjer, ponovno se iscrtati ako je bio pokriven drugim prozorom).

Kao što pretpostavljate, obrada svih mogućih kombinacija prozora, događaja i poruka može biti pravo opterećenje. Na sreću, Visual Basic vas oslobađa potreba baratanja svim manje važnim porukama. Većinu poruka Visual Basic obrađuje automatski; ostale vam se daju na znanje kao Event postupci. To vam omogućuje brzo stvaranje moćnih aplikacija bez potrebe za obradom nebitnih detalja.

Razumijevanje modela upravljanog događajima

U uobičajenim ili “proceduralnim” aplikacijama, sama aplikacija kontrolira po kojem redu će se izvesti pojedini dijelovi programskog koda. Izvođenje započinje prvom linijom programskog koda i slijedi unaprijed određenu putanju kroz aplikaciju, pozivajući potprograme kad je potrebno.

U aplikaciji upravljanoj događajima, programski kod ne slijedi unaprijed određenu putanju - takve aplikacije izvode različite dijelove programskog koda kao odgovor na događaje. Događaji mogu biti pokrenuti akcijom korisnika, porukama sistema ili drugih aplikacija, ili čak same aplikacije. Redoslijed tih događaja određuje putanju kojom će programski kod biti izveden, pa je putanja izvođenja kroz programski kod drugačija svaki put kad se aplikacija pokrene.

Budući da ne možete predvidjeti redoslijed događaj, vaš programski kod mora poduzeti neke pretpostavke o "općem stanju" kad se počne izvoditi. Kad pretpostavite neke stvari (na primjer, da prazno polje mora sadržavati neku vrijednost prije pokretanja postupka koji će tu vrijednost obraditi), trebate sastaviti svoju aplikaciju tako da budete sigurni da će ta pretpostavka uvijek biti ostvarena (na primjer, onemogućavanje naredbenog gumba koji pokreće postupak sve dok polje za upis podatka ne sadrži neku vrijednost).

Tijekom izvođenja vaš programski kod možete pokrenuti i neke događaje. Na primjer, programirana promjena teksta u okviru s tekstom uzrokovat će pokretanje Change svojstva. Ako pretpostavite da će taj događaj pokrenuti samo akcija korisnika, možete vidjeti neočekivane rezultate. Zbog toga je potrebno razumjeti model pokretanja događajima i imati to na umu tijekom razvoja vlastite aplikacije.

Interaktivno okruženje

Postupak razvijanja uobičajenih aplikacija može biti rastavljen na tri različita koraka: pisanje, prevođenje i ispitivanje programskog koda. Za razliku od tradicionalnih programskih jezika, Visual Basic koristi interaktivni pristup razvoju, smanjujući razlike između ova tri koraka.

Kod većine programskih jezika, ako napravite pogrešku u pisanju programskog koda, nju će otkriti prevoditelj (compiler) kad započnete prevođenje aplikacije. Nakon toga trebate pronaći i ispraviti pogrešku i ponovno započeti s prevođenjem, ponavljajući taj postupak za svaku pronađenu pogrešku. Visual Basic prevodi vaš programski kod dok ga upisujete, hvatajući i označujući u hodu sve sintaksne ili pravopisne pogreške. To je skoro kao da imate stručnjaka iza leđa dok unosite programski kod.

Osim otkrivanja pogrešaka u hodu, Visual Basic također djelomično prevodi programski kod za vrijeme upisivanja. Kad ste spremni pokrenuti i ispitati svoju aplikaciju, potrebno je vrlo malo vremena do kraja prevođenja. Ako prevoditelj pronađe pogrešku, ona će biti označena u programskom kodu. Pogrešku možete odmah ispraviti i nastaviti s prevođenjem bez potrebe pokretanja cijelog postupka ispočetka.

Zbog interaktivne prirode Visual Basica, često ćete pokretati svoju aplikaciju dok ju pišete. Na taj način možete isprobati efekte programskog koda za vrijeme rada što je bolji izbor od čekanja na prevođenje na kraju programiranja.

Elementi ugrađene razvojne okoline

Radna okolina Visual Basica se često označuje kao ugrađena razvojna okolina (integrated development environment, IDE) jer objedinjuje puno različitih funkcija kao što su oblikovanje, uređivanje, prevođenje i ispravljanje pogrešaka u poznatom okruženju. Kod većine uobičajenih razvojnih alata, svaka od tih funkcija bila bi izvođena kao posebna aplikacija, svaka sa svojim sučeljem.

Pokretanje Visual Basic razvojne okoline

Kad pokrenete Setup aplikaciju Visual Basica, ona vam omogućuje postavljanje dijelova Visual Basica u već postojeću programsku grupu ili kreiranje nove programske grupe i prečica za Visual Basic u Windowsima. Nakon toga možete početi rad u Visual Basicu iz Windowsa.

Kako pokrenuti Visual Basic iz Windowsa

1. Kliknite gumb **Start** na traci sa zadaćama.
2. Odaberite grupu **Programs**, te zatim **Microsoft Visual Basic 6.0**.

   – ili –

   Kliknite na gumb **Start**.

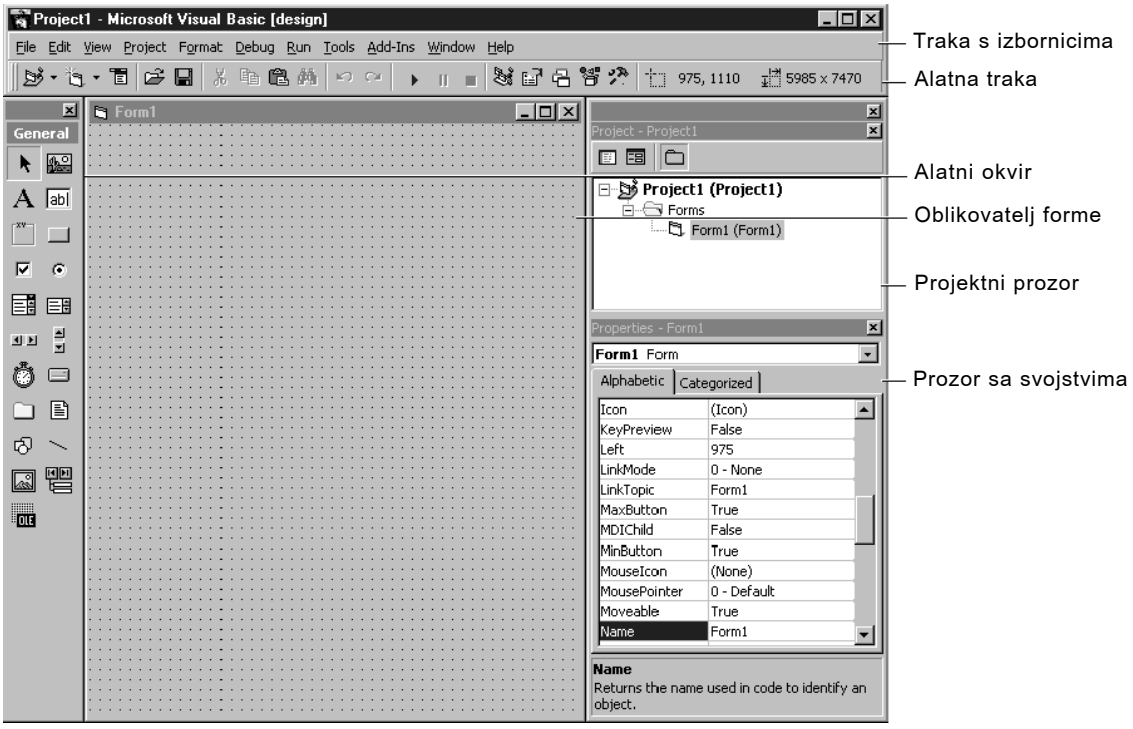
   Odaberite grupu **Programs**.

   Iskoristite **Windows Explorer** za pronalaženje izvršne datoteke Visual Basica.
3. Dvaput kliknite na ikonu Visual Basic.

Također možete kreirati prečicu do Visual Basica, i dvoklikom je pokrenuti.

Kad prvi put pokrenete Visual Basic, vidjet ćete sučelje ugrađene razvojne okoline, kako je prikazano na slici 2.1.

Slika 2.1 Ugrađena razvojna okolina Visual Basica



Elementi ugrađene razvojne okoline

Ugrađena razvojna okolina Visual Basica (IDE) sastoji se od sljedećih elemenata.

Traka s izbornicima (Menu Bar)

Sadrži naredbe koje koristite tijekom rada s Visual Basicom. Osim uobičajenih izbornika File, Edit, View, Window i Help, izbornici omogućuju pristup funkcijama specifičnim za programiranje kao što su Project, Format ili Debug.

Pomoćni izbornici (Context Menus)

Sadrže prečice do često izvođenih akcija. Za otvaranje pomoćnog izbornika, kliknite desnom tipkom miša na objekt koji koristite. Specifična lista prečica dostupna iz tako otvorenog izbornika ovisi o dijelu sučelja na koji ste kliknuli desnom tipkom miša. Na primjer, pomoćni izbornik koji ćete dobiti desnim klikom na alatni okvir omogućit će vam poziv dijaloškog okvira za prikaz sastavnih dijelova, sakrivanje alatnog okvira, sidrenje alatnog okvira ili dodavanje nove kartice u alatni okvir.

Alatne trake (Toolbars)

Omogućuju brz pristup obično korištenim naredbama u razvojnom okruženju. Potrebno je kliknuti jednom na gumb na alatnoj traci za pokretanje akcije predstavljene tim gumbom. Kao standardna postava pri pokretanju Visual Basica prikazana je alatna traka Standard. Dodatne alatne trake za uređivanje, oblikovanje forme i ispravljanje pogrešaka mogu biti uključene ili isključene preko naredbe Toolbars u izborniku View.

Alatne trake mogu biti usidrene ispod trake s izbornicima ili mogu "lebdjeti" ako odaberete okomitu traku na lijevom rubu alatne trake i odvučete je trake s izbornicima.

Alatni okvir (Toolbox)

Pruža niz alata koje koristite pri stvaranju za postavljanje kontrola na formu. Kao dodatak standardnom izgledu alatnog okvira, možete kreirati vlastite dodatne kartice odabirom stavke Add Tab iz pomoćnog izbornika te postaviti kontrole na novodobivenu karticu.

Za više informacija Ako želite naučiti više o određenim kontrolama, pogledajte 3. poglavlje " Forme, kontrole i izbornici" i 7. poglavlje "Korištenje standardnih kontrola Visual Basica". Za informacije kako dodati kontrole u alatni okvir, pogledajte "Dodavanje kontrola u projekt" u 4. poglavlju "Upravljanje projektima".

Projektni prozor (Project Explorer Window)

Sadrži popis svih forma i modula u vašem trenutnom projektu. *Projekt* je skup datoteka koje koristite za izgradnju aplikacije.

Za više informacija Za informacije o projektima pogledajte 4. poglavlje Upravljanje projektima".

Prozor sa svojstvima (Properties Window)

Sadrži popis svih svojstava za odabranu kontrolu ili formu. *Svojstvo* je osobina objekta, kao veličina, sadržaj ili boja.

Za više informacija Za informacije o svojstvima pogledajte "Tumačenje svojstava, postupaka i događaja" u 3. poglavlju " Forme, kontrole i izbornici"

Pretraživač objekata (Object Browser)

Ispisuje objekte dostupne za korištenje u vašem projektu i omogućuje vam brzo upravljanje kroz programski kod. Pretraživač objekata možete iskoristiti za istraživanje objekata u Visual Basicu i drugim aplikacijama, možete vidjeti koji postupci i svojstva su dostupni za te objekte, te možete iskopirati dijelove programskog koda u vašu aplikaciju.

Za više informacija Za više informacija o korištenju pretraživača objekata za pregled potprograma, pogledajte "Odgonetavanje objekata" u 9. poglavlju "Programiranje objektima". Za detalje o korištenjima dodataka za proširenje razvojne okoline Visual Basica, pogledajte "Korištenje čarobnjaka i dodataka" u 4. poglavlju "Upravljanje projektima".

Oblikovatelj forme (Form Designer)

Služi kao prozor kojeg prilagođavate pri oblikovanju sučelja vaše aplikacije. Dodajete kontrole, grafiku i slike na formu kako bi stvorili izgled koji želite. Svaka forma u vašoj aplikaciji ima svoj prozor za oblikovanje.

Za više informacija Da bi naučili kako dodati kontrole u aplikaciju, pogledajte "Vaša prva Visual Basic aplikacija" kasnije u ovom poglavlju. Za učenje više detalja o oblikovanju sučelja, pogledajte 6. poglavlje "Stvaranje korisničkog sučelja".

Kodni prozor (Code Editor Window)

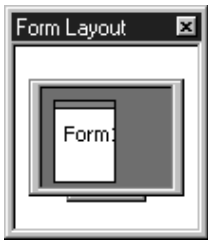
Služi kao prozor za upis programskog koda aplikacije. Za svaku formu ili modul vaše aplikacije stvara se posebni kodni prozor.

Za više informacija Za daljnje učenje o tome kako upisati programski kod i koristiti kodni prozor, pogledajte 5. poglavlje "Osnove programiranja".

Prozor s položajem forme (Form Layout Window)

Prozor s položajem forme (slika 2.2) omogućuje vam određivanje položaja forme u vašoj aplikaciji koristeći umanjen izgled ekrana.

Slika 2.2 Prozor s položajem forme



Dodatni prozori (Immediate, Locals, and Watch Windows)

Ovi dodatni prozori služe za korištenje pri traženju pogrešaka u vašim aplikacijama. Dostupni su samo kad izvršavate aplikaciju unutar korisničkog sučelja.

Za više informacija Kako bi naučili više o dodatnim prozorima, pogledajte 13. poglavlje "Traženje i obrada pogrešaka"

Napomena Pomoćne dodatke sučelja Visual Basica možete dodati koristeći aplikaciju nazvanu *add–in*. Ova aplikacija dostupna je od neovisnih proizvođača softvera i pruža dodatne mogućnosti kao na primjer, kontrola izvornog koda, što vam može pomoći pri timskom razvijanju projekata.

Mogućnosti sučelja

Visual Basic omogućuje veliku elastičnost, dopuštajući vam uobličavanje radnog sučelja tako da najbolje odgovara vašem osobnom stilu. Možete birati između sučelja sa jednim ili više dokumenata, te možete podesiti veličinu i položaj raznih elemenata korisničkog sučelja. Raspored koji stvorite ostat će zapamćen i kod idućih pokretanja Visual Basica.

SDI ili MDI sučelje

U korisničkom sučelju Visual Basica na raspolaganju su vam dva različita stila: sučelje s jednim dokumentom (single document interface, SDI) ili sučelje s više dokumenata (multiple document interface, MDI). Sa SDI sučeljem, svi prozori radne okoline mogu se slobodno pomicati bilo gdje po ekranu; sve dok god je Visual Basic trenutno aktivna aplikacija, svi njezini prozori ostat će iznad prozora drugih aplikacija. S MDI izborom, svi prozori radnog sučelja nalaze se unutar glavnog prozora promjenljive veličine.

Prijelaz između SDI i MDI načina

1. Odaberite stavku **Options** iz izbornika **Tools**.

   Prikazat će se dijaloški okvir **Options**.

2. Odaberite karticu **Advanced**.

3. Potvrdite ili odznačite kontrolnu kućicu **SDI Development Environment**.

   Korisničko sučelje pokrenut će se s odabranim stilom idući put kad pokrenete Visual Basic.

   – ili –

   Pokrenite Visual Basic iz naredbene linije s parametrima /sdi ili /mdi.

Usidreni prozori

Većina prozora korisničkog sučelja može biti usidrena ili povezana, međusobno ili s rubom ekrana. Ovo uključuje alatni okvir, prozor s položajem forme, projektni prozor, prozor sa svojstvima, paletu boja, te prozor za neposredan upis naredbi, prozor s lokalnim varijablama i nadgledni prozor.

Sa MDI stilom, prozori mogu biti usidreni uz bilo koji rub glavnog prozora; kod SDI stila mogu biti usidreni samo ispod trake s izbornicima. Usidrenost prozora uključuje se ili isključuje potvrđivanjem odgovarajuće kontrolne kućice na stranici Docking u dijaloškom okviru Options, koji je dostupan pozivom naredbe Options iz izbornika Tools.

Kako usidriti ili odsidriti prozor

1. Odaberite prozor koji želite usidriti ili odsidriti.
2. Odvucite prozor do željenog položaja držanjem pritisnute lijeve tipke miša.
3. Dok povlačite, okvir prozora će biti vidljiv.
4. Otpustite tipku miša.

Vaša prva Visual Basic aplikacija

Stvaranje aplikacije u Visual Basicu je jednostavno. Koliko jednostavno? Kao odgovor, isprobajte Zdravo, Visual Basic i Firstapp aplikacije koje slijede.

Zdravo, Visual Basic

Postoje tri glavna koraka pri stvaranju aplikacije u Visual Basicu:

1. Stvaranje sučelja
2. Postavljanje svojstava
3. Pisanje programskog koda.

Da biste vidjeli kako se to radi, pratite korake iz sljedećih postupaka za stvaranje jednostavne aplikacije koja se sastoji od okvira s tekstom i naredbenog gumba. Kad kliknete na naredbeni gumb, poruka “Zdravo, svijete!” će se pojaviti u okviru s tekstom.

Stvaranje sučelja

Forme su temelj za stvaranje korisničkog sučelja neke aplikacije. Možete koristiti forme za dodavanje prozora i dijaloških okvira svojoj aplikaciji. Možete ih iskoristiti i kao spremnike za dijelove koji nisu vidljivi dio korisničkog sučelja. Na primjer, u vašoj aplikaciji možete imati formu koja će služiti kao spremnik za grafiku koju namjeravate prikazati na drugim formama.

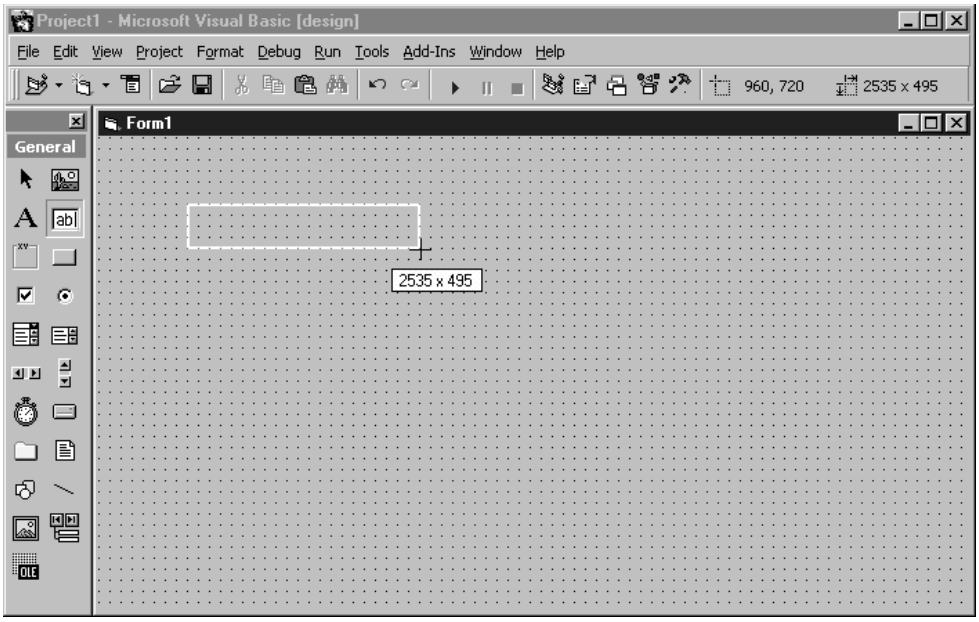
Prvi korak u izgradnji Visual Basic aplikacije je stvaranje formi koje će biti temelj sučelja aplikacije. Nakon toga potrebno je kreirati objekte koji će opremiti stvorene forme. Za ovu prvu aplikaciju, koristit ćete dvije kontrole koje se nalaze u alatnom okviru.

gumb	kontrola
	okvir s tekstom
	naredbeni gumb

Kreiranje kontrole korištenjem alatnog okvira

1. Kliknite alat kontrole koju želite kreirati – u ovom slučaju **okvir s tekstom (text box)**.
2. Pomaknite pokazivač na vašu formu. Pokazivač će imati izgled križa, kao što je prikazano na slici 2.3.

Slika 2.3 Kreiranje okvira s tekstom te alatni okvir



3. Dovedite križ na mjesto gdje želite postaviti gornji lijevi kut kontrole.
4. Povlačite križ sve dok kontrola ne bude imala željenu veličinu (*povlačenje* znači držanje pritisnute lijeve tipke miša dok pomičete objekt mišem).
5. Otpustite tipku miša.

Kontrola će se pojaviti na formi.

Drugi jednostavan način dodavanja kontrole na formu je dvoklik na gumb za željenu kontrolu u alatnom okviru. Ovaj postupak će kreirati kontrolu unaprijed određene veličine na sredini forme; nakon toga možete pomaknuti kontrolu na željeni položaj na formi.

Promjena veličine, pomicanje i zaključavanje kontrola

Uočite da se na kutovima kontrole pojavljuju mali pravokutni okviri koji se zovu *hvataljke za određivanje veličine*; koristit ćete ih u sljedećem koraku za promjenu veličine kontrole. Možete koristiti i miša, tipkovnicu i naredbe izbornika za pomicanje kontrola, zaključavanje i otključavanje položaja kontrola, te za podešavanje njihovih položaja.

Kako promijeniti veličinu kontrole

1. Odaberite kontrolu kojoj želite promijeniti veličinu tako da mišem kliknete na nju.
2. Postavite pokazivač na hvataljku, i povucite je dok kontrola ne bude željene veličine.

   Hvataljke na kutovima mijenjaju veličinu kontrole vodoravno i okomito, dok hvataljke na rubovima mijenjaju veličinu kontrole samo u jednom smjeru.
3. Otpustite tipku miša.

   – ili –

   Upotrijebite SHIFT zajedno sa kursorskim tipkama za promjenu veličine odabrane kontrole.

Kako pomaknuti kontrolu

- Upotrijebite miša za povlačenje kontrole do novog položaja na formi.

  – ili –

  U prozoru sa svojstvima objekata promijenite vrijednosti svojstava **Top** i **Left**.

Kad je kontrola odabrana, možete upotrijebiti CTRL s kursorskim tipkama za pomicanje kontrole za jednu jedinicu mreže (grid unit). Ako je mreža isključena, kontrola će se micati po jedan piksel.

Kako zaključati pozicije svih kontrola

- U izborniku **Format** odaberite stavku **Lock Controls**.

  – ili –

  Kliknite na gumb **Lock Controls Toggle** koji se nalazi na alatnoj traci Form Editor.

Ovaj postupak zaključat će sve kontrole na formi na njihovim položajima u tom trenutku tako da ih ne možete slučajno pomaknuti jednom kad ste ih postavili na željeno mjesto. Samo kontrole na odabranoj formi bit će zaključane; kontrole na drugim formama bit će nedirnute. Ovo je uključeno – isključeno naredba, pa je možete iskoristiti i za otključavanje položaja kontrola.

Kako namjestiti položaj zaključanih kontrola

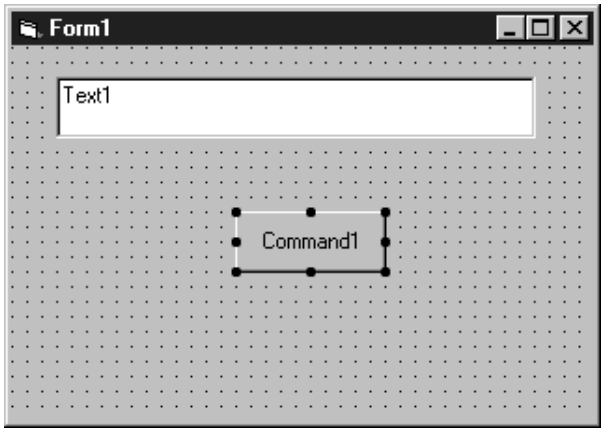
- Možete "gurnuti" kontrolu koja ima fokus držanjem pritisnute CTRL tipke i pritiskom na odgovarajuću kursorsku tipku.

  – ili –

  Možete promijeniti svojstva kontrole **Top** i **Left** u prozoru sa svojstvima.

Sad imate sučelje za "Zdravo, svijete!" aplikaciju kao što je prikazano na slici 2.4.

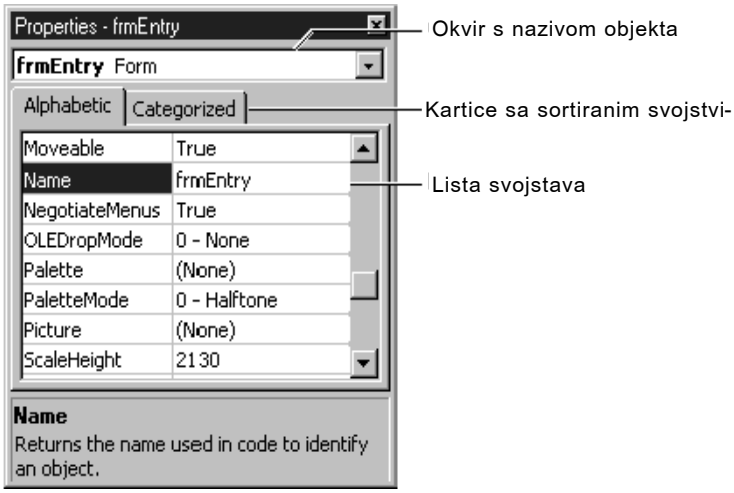
Slika 2.4 Sučelje aplikacije "Zdravo, svijete!"



Postavljanje svojstava

Sljedeći korak je određivanje svojstava kontrola koje ste kreirali. Prozor sa svojstvima (slika 2.5) omogućuje jednostavan način postavljanja svojstava za sve objekte na formi. Za otvaranje prozora sa svojstvima, odaberite naredbu Properties Window u izborniku View, kliknite na gumb Properties Window na alatnoj traci, ili iskoristite pomoćni izbornik kontrole.

Slika 2.5 Prozor sa svojstvima



Prozor sa svojstvima sastoji se od sljedećih dijelova:

- Okvir s nazivom objekta – Prikazuje ime objekta kojem ćete odrediti svojstva. Kliknite strelicu na desnoj strani ovog okvira za prikaz liste svih objekata trenutno odabrane forme.
- Kartice sa sortiranim svojstvima – Odaberite između abecednog popisa svojstava ili hijerarhijskog popisa podijeljenog po logičkim kategorijama, kao što su upravljanje izgledom, pismima ili položajem.
- Lista svojstava – Lijeva kolona prikazuje sva svojstva odabranog objekta. U desnoj koloni možete vidjeti i promijeniti njihove vrijednosti.

Kako odrediti svojstva iz prozora sa svojstvima

1. U izborniku **View** odaberite stavku **Properties**, ili kliknite na gumb **Properties** na alatnoj traci.

   Prozor sa svojstvima prikazat će svojstva odabrane forme ili kontrole.

2. Sa liste svojstava odaberite ono koje želite promijeniti.
3. U desnoj koloni, upišite ili odaberite novu vrijednost svojstva.

   Neka svojstva imaju već unaprijed određenu listu mogućih vrijednosti koje su poredane po rednim brojevima. Ovu listu možete vidjeti klikom na strelicu na desnoj strani okvira u kojem je ispisana vrijednost, ili možete kružiti kroz listu dvoklikom na taj okvir.

Za primjer aplikacije “Zdravo, svijete!” trebat ćete promijeniti vrijednosti tri svojstva. Za ostala svojstva ostavite već određene vrijednosti.

Objekt	Svojstvo	Vrijednost
forma	Caption	Zdravo, svijete!
okvir s tekstom	Text	(prazno)
naredbeni gumb	Caption	OK

Određivanje svojstva Icon

Sve forme u Visual Basicu imaju opću, unaprijed određenu ikonu koja se pojavljuje kad smanjite tu formu. Unatoč tome, vjerojatno ćete promijeniti tu ikonu u neku drugu koja će prikazivati primjenu te forme u vašoj aplikaciji. Za dodjelu ikone formi, promijenite vrijednost svojstva Icon za tu formu. Možete koristiti 32 x 32 piksela velike ikone koje su standardne za 16-bitne verzije Microsoft Windowsa, a koriste se i u Windowsima 95/98 te Windowsima NT, kao i 16 x 16 piksela velike ikone koje se koriste u Windowsima 95.

Pisanje programskog koda

Kodni prozor (Code Editor Window) je mjesto gdje upisujete programski kod svoje aplikacije. Programski kod sastoji se od naredbi, konstanti i definicija. Korištenjem kodnog prozora možete brzo pregledati i promijeniti bilo koji dio programskog koda svoje aplikacije.

Kako otvoriti kodni prozor

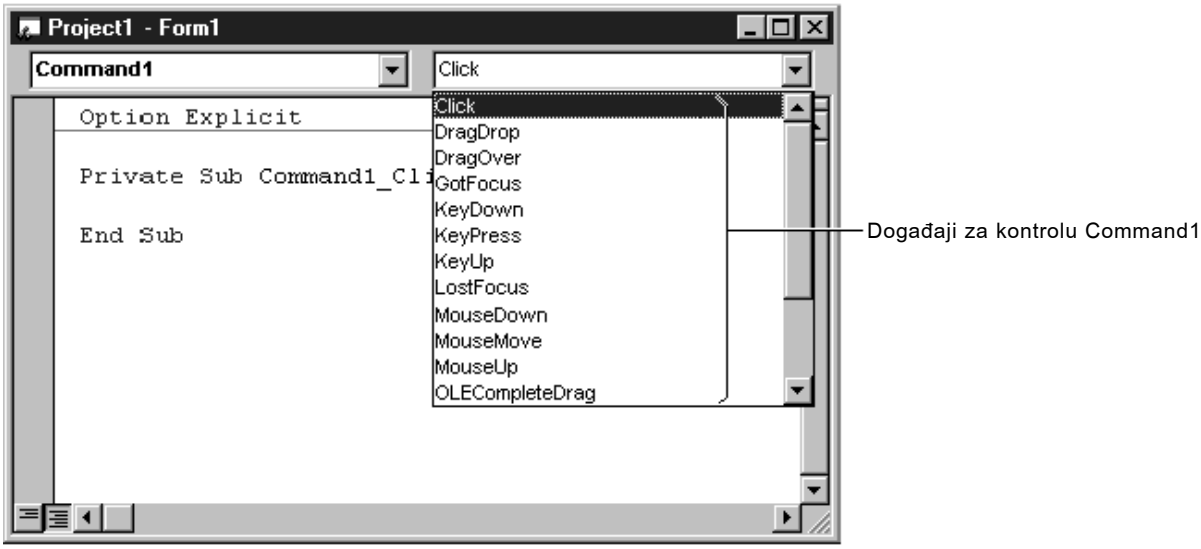
- Dvoklikom na formu ili kontrolu kojoj želite upisati programski kod.

  – ili –

  U projektnom prozoru odaberite ime forme ili modula, pa kliknite na gumb **View Code**.

Slika 2.6 prikazuje kodni prozor koji će se pojaviti kada dvaput kliknete na naredbeni gumb, te događaji za ovu kontrolu.

Slika 2.6 Kodni prozor



Možete odabrati želite li prikazati sve potprograme u istom kodnom prozoru, ili će biti prikazana samo jedan potprogram.

Kako prikazati sve potprograme u jednom kodnom prozoru

1. U izborniku **Tools** odaberite stavku **Options**.
2. Na kartici **Editor** u dijaloškom okviru **Options**, potvrdite kontrolnu kućicu lijevo od opcije **Default to Full Module View**. Kontrolna kućica lijevo od opcije **Procedure Separator** dodaje ili briše liniju između potprograma.

   – ili –

   Kliknite na gumb **Procedure View** u donjem lijevom kutu kodnog prozora.

Kako prikazati samo jedan potprogram u kodnom prozoru

1. U izborniku **Tools**, odaberite stavku **Options**.
2. Na kartici **Editor** u dijaloškom okviru **Options**, odznačite kontrolnu kućicu lijevo od opcije **Default to Full Module View**.

   – ili –

   Kliknite na gumb **Procedure View** u donjem lijevom kutu kodnog prozora.

Kodni prozor uključuje sljedeće elemente:

- Okvir s listom objekata – Prikazuje ime odabranog objekta. Kliknite na strelicu na desnoj strani okvira za prikaz svih objekata na aktivnoj formi.
- Okvir s listom svojstava – Popis potprograma ili događaja za odabrani objekt. Okvir prikazuje ime odabranog potprograma – u ovom slučaju, Click. Kliknite na strelicu na desnoj strani okvira za prikaz liste svih potprograma vezanih za odabrani objekt.

Kreiranje događajem pokretanih potprograma

Programski kod u Visual Basic aplikaciji je podijeljen u manje blokove koji se nazivaju *potprogrami*. Događajem pokretan potprogram, kao onaj kojeg ćete kreirati, sadrži programski kod koji će biti izvršen kad se javi događaj (na primjer, kad korisnik klikne na gumb). Događajem pokretan potprogram za kontrolu sastoji se od imena kontrole (određenog u svojstvu Name), podvlake (_) te imena događaja. Na primjer, ako želite da naredbeni gumb s imenom Command1 pokrene događajem pokretan potprogram kad korisnik klikne na njega, koristit ćete potprogram Command1_Click.

Kako kreirati događajem pokretan potprogram

1. U okviru s popisom **Object**, odaberite ime objekta na aktivnoj formi (*aktivna* forma je ona koja trenutno ima fokus).

   U ovom primjeru, odaberite naredbeni gumb **Command1**.

2. U okviru s popisom **Procedure** odaberite ime događaja za odabrani objekt.

   U ovom primjeru, potprogram **Click** je već odabran, jer je to standardni potprogram za naredbeni gumb. Uočite da se sad u kodnom prozoru pojavljuje *predložak* za ovaj događajem pokretan tip potprograma.

3. Upišite sljedeći programski kod između naredbi Sub i End Sub:

   ```
   Text1.Text = "Zdravo, svijete!"
   ```

   Događajem pokretan potprogram trebao bi izgledati ovako:

   ```
   Private Sub Command1_Click ()
      Text1.Text = "Zdravo, svijete!"
   End Sub
   ```

Uočit ćete da ovaj programski kod zapravo mijenja svojstvo Text kontrole imena Text1 u "Zdravo, svijete!". Sintaksa ovog primjera ima oblik *objekt.svojstvo* gdje je `Text1` objekt, a `Text` svojstvo. Ovu sintaksu možete koristiti za promjenu svojstava bilo koje forme ili kontrole kao odgovor na događaje koji će se pojaviti tijekom izvođenja aplikacije.

Za više informacija Za informacije o kreiranju ostalih tipova potprograma, pogledajte "Uvod u potprograme" u 5. poglavlju "Osnove programiranja".

Pokretanje aplikacije

Za pokretanje aplikacije, odaberite stavku Start u izborniku Run, kliknite na gumb Start na alatnoj traci, ili pritisnite F5. Kliknite na naredbeni gumb koji ste kreirali na formi, i vidjet ćete tekst "Zdravo, svijete!" ispisan u okviru s tekstom.

Primjer aplikacije Firstapp

Visual Basic vam omogućuje bogatstvo alata naprednijih od onih koje ste koristili u prvoj aplikaciji, pa ćete uskoro koristiti puno drugih mogućnosti za obradu i prilagodbu svojih aplikacija. Pregled aplikacija danih kao primjer može biti odličan način daljnjeg učenja Visual Basica. Sljedeći primjer pokazuje kako na jednostavan način kreirati korisnu aplikaciju u Visual Basicu.

Aplikacija Firstapp pokazuje kako se mogu iskoristiti *kontrola podataka* (data control) i *kontrola mreže* (grid control) za prikaz tablice informacija iz baze podataka. Visual Basic omogućuje jednostavan pristup informacijama u bazi podataka iz vaše aplikacije. Kontrola podataka ima mogućnost kretanja kroz slogove u bazi podataka, usklađujući prikaz zapisa u kontroli podataka s položajem u slogu.

Aplikacija se sastoji od kontrole podataka, kontrole MSFlexGrid, okvira s listom i dva naredbena gumba. Mreža prikazuje tabelu informacija o proizvodima dobivenu iz Northwind baze podataka. Dok korisnik odabire stavku koristeći gumbe za kretanje u kontroli podataka, ime odabranog proizvoda će biti prikazano u toj kontroli. Korisnik može i dodati proizvode na "listu za kupovanje" u okviru s listom dvoklikom na trenutno odabranu stavku u mreži.

Za dodavanje stavke u okvir s listom, koristit ćete postupak AddItem (*postupak* je funkcija Visual Basica koja djeluje na određeni objekt, u ovom slučaju na okvir s listom). Sintaksa za određivanje postupka (*objekt.postupak*) je slična sintaksi određivanja svojstva (*objekt.svojstvo*). Postupak AddItem vam dopušta dinamičko dodavanje stavki u okvir s listom tijekom izvođenja aplikacije. Obratno tome, postupak Clear se koristi za brisanje svih stavki u okviru s listom.

Za više informacija Da biste naučili više o postupcima, pogledajte "Tumačenje svojstava, postupaka i događaja" u 3. poglavlju "Forme, kontrole i izbornici".

Kreiranje projekta

Kreiranje aplikacije započinjete odabirom stavke New Project u izborniku File, zatim odabirete Standard EXE u dijaloškom okviru New Project (kad prvi put pokrenete Visual Basic, biti će prikazan dijaloški okvir New Project). Visual Basic kreira *novi projekt* i prikazuje novu formu. Da biste kreirali sučelje, postavite kontrolu podataka, kontrolu MSFlexGrid, okvir s listom i dva naredbena gumba. Kontrola MSFlexGrid ne postoji u standardnom alatnom okviru, pa ćete je dodati na sljedeći način:

Kako dodati kontrolu u alatni okvir

1. Odaberite stavku **Components** iz pomoćnog izbornika (da biste ga dobili, kliknite desnom tipkom miša unutar alatnog okvira).

   Prikazat će se dijaloški okvir **Components**.

2. Pronađite stavku MSFlexGrid (Microsoft Flex Grid 6.0) u okviru s listom **Components** i potvrdite kontrolnu kućicu s lijeve strane stavke.

3. Kliknite na gumb **OK**.

   Ikona za kontrolu **MSFlexGrid** će se pojaviti u alatnom okviru.

Upotrijebite alatni okvir za kreiranje kontrole podataka, kontrole MSFlexGrid, okvira s listom i dva naredbena gumba na formi. Ako se ne sjećate kako, pogledajte "Stvaranje sučelja" ranije u ovom poglavlju.

Određivanje svojstava

U prozoru sa svojstvima, promijenite svojstva objekata prema sljedećoj tabeli. Za ostala svojstva ostavite već postavljene vrijednosti.

objekt	svojstvo	postavka
Form	Caption	Proizvodi
Data1	DatabaseName RecordSource	*staza*\Biblio.mdb All Titles
DataGrid1	DataSource	Data1
Command1	Caption	Brisanje
Command2	Caption	Izlaz

Svojstvo DatabaseName kontrole podataka mora uključivati i stvarnu stazu do baze podataka. Standardno je baza podataka Biblio.mdb instalirana u istom direktoriju gdje i Visual Basic. Kad odaberete svojstvo DatabaseName u prozoru sa svojstvima, možete kliknuti gumb na desnoj strani okvira za prikaz standardnog dijaloškog okvira File-Open uz pomoć kojeg možete potražiti tu datoteku. Kad je svojstvo DatabaseName određeno, svojstvo RecordSource u prozoru sa svojstvima će sadržavati listu tabela ili slogova odabrane baze podataka. Postavljanje vrijednosti svojstva DataSource za kontrolu MSFlexGrid na Data1 će automatski povezati mrežu s kontrolom podataka.

Pisanje programskog koda

Sav programski kod ove aplikacije se nalazi u događajem pokretanim potprogramima Command1_Click, Command2_Click, Data1_Reposition i MSFlexGrid1_DblClick. Dvaput kliknite na formu ili kontrolu za dobivanje kodnog prozora, i upišite programski kod u odgovarajuće potprograme.

Sljedeći programski kod upišite u Command1_Click potprogram i on će obrisati okvir s listom kad korisnik klikne na ovaj naredbeni gumb:

```
Private Sub Command1_Click ()
   List1.Clear            ‘ brisanje okvira s listom
End Sub
```

U gornjoj naredbenoj liniji pozivate postupak Clear okvira s listom List1. Postupak Clear će obrisati sadržaj okvira s listom.

Ovaj programski kod treba upisati u potprogram Command2_Click za brisanje forme iz memorije i završetak rada aplikacije:

```
Private Sub Command2_Click ()
   Unload Form1
   End                    ‘ kraj rada aplikacije
End Sub
```

U gornjem potprogramu, prva linija poziva postupak Unload za formu. Ako je potrebno obaviti neku funkciju pri gašenju aplikacije, kao što je snimanje datoteke, taj programski kod ćete smjestiti u potprogram Unload forme. Druga linija poziva funkciju End, koja završava izvođenje aplikacije.

Sljedeći programski kod treba upisati u potprogram Data1_Reposition kako bi se obnovio sadržaj svaki put kad se odabere zapis:

```
Private Sub Data1_Reposition ()
   Data1.Caption = Data1.Recordset(“ProductName”)
End Sub
```

U gornjoj naredbenoj liniji dodjeljujete vrijednost na desnoj strani (sadržaj polja Title u objektu RecordSet kontrole podataka) svojstvu na lijevoj strani (svojstvo Caption kontrole podataka).

Ovaj programski kod dodajte u potprogram MSFlexGrid_DblClick kako bi aplikacija dodala novu stavku u okvir s listom kad korisnik dvoklikom potvrdi odabrani red:

```
Private Sub MSFlexGrid_DblClick ()
   List1.AddItem MSFlexGrid.Text
End Sub
```

U gornjoj naredbenoj liniji pozivate postupak AddItem okvira s listom (List1). Tekst koji će biti dodan u okvir s listom nalazi se u *argumentu* postupka; u ovom slučaju sadržaj polja Title u zapisu kontrole podataka. Prosljeđivanje vrijednosti argumentu slično je dodjeljivanju vrijednosti svojstvu; za razliku od naredbe dodjeljivanja, znak jednakosti nije potreban.

Snimanje projekta

Vaš posao na izradi aplikacije završit ćete biranjem stavke Save Project u izborniku File. Visual Basic će vas odvojeno upitati za snimanje forme te zatim i projekta. Jedno od mogućih imena projekta bilo bi "Lista za kupovanje". Windows 95/98 i Windows NT operativni sustavi dopuštaju vam imena datoteka do 255 znakova duljine, a imena datoteka mogu sadržavati i znakove razmaka. Starije verzije Microsoft Windows operativnog sustava ograničuju vas na imena datoteka s osam karaktera, sa sufiksom od tri karaktera.

Poboljšanje vaše aplikacije

Upravo ste završili svoju prvu Visual Basic aplikaciju; ona izvodi jednostavnu, ali korisnu funkciju. Ovu aplikaciju možete iskoristiti kao temelj za dodavanje sličnih mogućnosti u druge aplikacije, koristeći svoje vlastite podatke umjesto onih iz Biblio.mdb datoteke. Naravno, da bi ovu aplikaciju napravili stvarno korisnom, možda poželite dodati mogućnost snimanja ili tiskanja sadržaja okvira s listom, dodavanja novih podataka kao što su cijena i dostupnost, ili čak dobivanje podataka o kreditnoj kartici i prenošenje narudžbe putem Interneta. U nastavku ove knjige, pronaći ćete primjere kako učiniti to i puno drugih stvari.

3. POGLAVLJE

Forme, kontrole i izbornici

Prvi korak u kreiranju aplikacije Visual Basicom je stvaranje sučelja, vidljivog dijela aplikacije s kojim će korisnik surađivati. Forme i kontrole su osnovni dijelovi stvaranja sučelja; to su objekti kojima ćete izgraditi svoju aplikaciju.

Forme su objekti koji imaju svojstva koja određuju njihovo ponašanje, te događaje koji određuju interakciju s korisnikom. Postavljanjem svojstava forme i pisanjem Visual Basic programskog koda koji će odgovarati na događaje, prilagodit ćete objekt zahtjevima vaše aplikacije.

Kontrole su objekti koji su sadržani u objektima forme. Svaki tip kontrole ima svoj komplet svojstava, metoda i događaja koje ga čine prikladnim određenoj namjeni. Neke kontrole koje možete koristiti u svojim aplikacijama najbolje su opremljene za unos i prikaz teksta. Druge kontrole omogućit će vam pristup podacima i postupcima drugih aplikacija kao da su one, iako udaljene, dio vašeg programskog koda.

Ovo poglavlje uvodi vas u temeljne pojmove rada s formama i kontrolama te njihovim pridruženim svojstvima, metodama i događajima. Razrađeno je puno standardnih kontrola, te neki dijelovi specifični za forme kao što su izbornici i dijaloški okviri.

Sadržaj

- Tumačenje svojstava, postupaka i događaja
- Oblikovanje forme
- Pokretanje akcija klikom na gumb
- Kontrole za prikaz i unos teksta
- Kontrole s izborom
- Kontrole za prikaz slika i grafike
- Dodatne kontrole
- Razumijevanje fokusa
- Postavljanje tabulatornog reda
- Osnove izbornika
- Upit korisniku dijaloškim okvirom

Primjer aplikacije: Controls.vbp

Primjeri programskog koda u ovom poglavlju preuzeti su iz primjera aplikacije Controls.vbp čiji ispis je dan u direktoriju Samples.

Tumačenje svojstava, postupaka i događaja

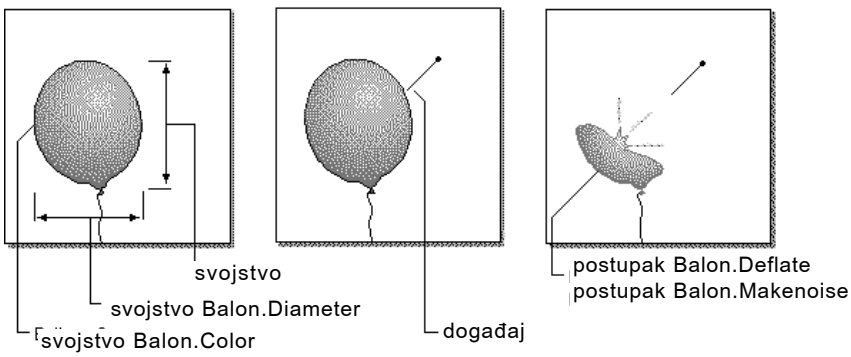
Forme i kontrole Visual Basica su objekti koji pokazuju vlastita svojstva, postupke i događaje. Svojstva možete shvatiti kao osobine objekta, postupke kao akcije, a događaje kao odgovore objekta.

Svakodnevni objekt kao dječji balon također ima svoja svojstva, postupke i događaje. Svojstva balona uključuju vidljive osobine kao što su visina, promjer i boja. Ostala svojstva opisuju njegova stanja (napuhan ili ispuhan), ili osobine koje nisu vidljive, kao starost. U pravilu, svi baloni imaju ta svojstva; vrijednosti tih svojstava se razlikuju od balona do balona.

Balon također ima nerazdvojive postupke ili događaje koje može izvesti. Postoji postupak napuhavanja (ubacivanja zraka u balon), postupak ispuhavanja (izbacivanje zraka) i postupak dizanja (ako ga pustite iz ruke). I ovdje, svi baloni sposobni su obaviti ove postupke.

Baloni također imaju unaprijed određene odgovore na neke vanjske događaje. Na primjer, balon će na događaj bušenja odgovoriti ispuhavanjem, ili će na događaj puštanja odgovoriti dizanjem.

Slika 3.1 Objekti imaju svojstva, odgovaraju na događaje i izvode pos-



tupke

Kad bi mogli programirati balon, programski kod Visual Basica bi izgledao kako slijedi. Za postavljanje svojstava balona:

```
Balon.Color = Red
Balon.Diameter = 10
Balon.Inflated = True
```

Uočite sintaksu koda – imenu objekta (Balon) i svojstva (.Color, boja) dodijeljena je vrijednost (Red, crvena). Boju balona možete promijeniti u programskom kodu ponavljanjem ove naredbene linije i dodjeljivanjem druge vrijednosti. Svojstva također mogu biti postavljena u prozoru sa svojstvima tijekom izrade aplikacije.

Postupci balona mogu biti pozivani na sljedeći način:

```
Balon.Inflate              ‘ napuši balon
Balon.Deflate              ‘ ispuši balon
Balon.Rise 5               ‘ podigni balon za 5
```

Sintaksa je slična dodjeljivanju svojstava – ime u objekta (imenica) dodan je postupak (glagol). U trećoj liniji, postoji i dodatna vrijednost, nazvana *argument*, koja određuje visinu podizanja. Neke metode će imati jedan ili više argumenata za daljnji opis akcije koja treba biti obavljena.

Balon može odgovoriti na događaj na sljedeći način:

```
Sub Balon_Puncture()
    Balon.Deflate                ‘ ako je balon probušen ispuši ga
    Balon.MakeNoise “Bang”       ‘ stvori zvuk
    Balon.Inflated = False       ‘ nije napuhan
    Balon.Diameter = 1           ‘ promjer = 1
End Sub
```

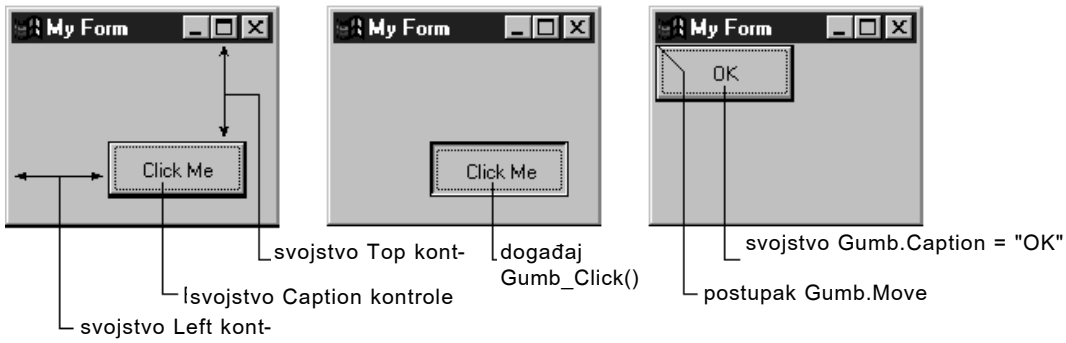
U ovom slučaju, programski kod opisuje ponašanje balona kad se pojavi događaj bušenja: pozivamo postupak Deflate (ispuhavanje), te postupak MakeNoise (stvaranje zvuka) sa argumentom “Bang” (tip zvuka koji će biti napravljen). S obzirom na to da balon više nije napuhan, svojstvo Inflated dobiva vrijednost False (netočno), a svojstvo Diameter (promjer) dobiva novu vrijednost.

Iako u stvarnosti ne možete programirati balon, možete programirati formu ili kontrolu Visual Basica. Kao programer, vi vršite nadzor. Vi odlučujete koja svojstva trebaju biti promijenjena, koji postupci pozvani i koji će događaji biti odgovor tako da postignete željeni izgled i ponašanje aplikacije.

Oblikovanje forme

Forme su temeljni objekti izgradnje aplikacije Visual Basica, zapravo prozori sa kojima korisnik surađuje tijekom izvođenja aplikacije. Forme imaju vlastita svojstva, događaje i postupke s kojima možete kontrolirati njihov izgled i ponašanje.

Slika 3.2 Forme i kontrole imaju vlastita svojstva, događaje i postupke



Prvi korak u kreiranju forme je određivanje njezinih svojstava. Svojstva forme možete odrediti tijekom *vremena izrade* aplikacije (*design time*) u prozoru sa svojstvima, te tijekom *vremena izvođenja* (*run time*) pisanjem programskog koda.

> Napomena Radit ćete s formama i kontrolama, određivati njihova svojstva i pisati programski kod tijekom *vremena izrade* aplikacije, a to je sve ono vrijeme kad stvarate aplikaciju u okruženju Visual Basica. *Vrijeme izvođenja* je vrijeme kad zapravo izvodite aplikaciju i djelujete kao što bi trebao i korisnik.

Postavljanje svojstava forme

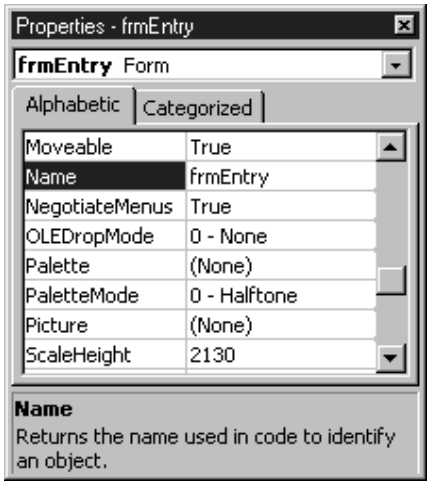
Većina svojstava forme djeluje na njzien izgled. Svojstvo Caption određuje tekst koji je ispisan u naslovnoj traci forme; svojstvo Icon određuje ikonu koja će biti iscrtana kad smanjite formu. Svojstva MaxButton i MinButton određuju može li forma biti smanjena ili povećana. Mijenjanjem vrijednosti svojstva BorderStyle možete odrediti ponašanje forme prilikom promjene njezine veličine.

Svojstva Height i Width određuju početnu veličinu forme; svojstva Left i Top određuju položaj forme u odnosu na gornji lijevi kut ekrana. Uz pomoć svojstva WindowState možete odrediti hoće li se forma pojaviti u smanjenoj, povećanoj ili normalnoj veličini.

Svojstvo Name određuje ime forme s kojim će se ona pozivati iz programskog koda. Po standardu, kad prvi put dodate formu projektu, njezino ime će biti Form1, Form2, i tako dalje. Dobra je ideja promijeniti formi ime uz pomoć svojstva Name u nešto smislenije, kao “frmUpis” formi za upis podataka.

Najbolji način upoznavanja s mnoštvom svojstava forme je isprobavanje. Promijenite neka svojstva forme u prozoru sa svojstvima (slika 3.3), pokrenite aplikaciju i pogledajte rezultate. Više o svakom svojstvu možete saznati ako ga odaberete i pritisnete F1 za pregled pomoći osjetljive na sadržaj.

Slika 3.3 Prozor sa svojstvima



Događaji i postupci forme

Kao i objekti i forme mogu izvoditi postupke i odgovarati na događaje.

Događaj forme Event pokreće se svaki put kad forma mijenja veličinu, akcijom korisnika ili programskim kodom. Zahvaljujući tome, možete poduzimati akcije kao pomicanje ili promjena veličine kontrola na formi kad se njena veličina promijeni.

Događaj Activate pojavljuje se svaki put kad forma postane aktivna; događaj Deactivate pojavljuje se kad neka druga forma ili aplikacija postanu aktivne. Ovi događaji su uobičajeni za pokretanje i zaključivanje ponašanja forme. Na primjer, u potprogram događaja Activate možete staviti programski kod koji će istaknuti tekst u nekom okviru s tekstom; potprogramom događaja Deactivate mogli bi snimiti promjene u datoteku ili bazu podataka.

Kako bi učinili formu vidljivom, trebate pozvati postupak Show:

```
Form2.Show
```

Pozivanje postupka Show ima jednak rezultat kao i postavljanje vrijednosti svojstva forme Visible na True.

Većina postupaka forme poziva tekst ili grafiku. Postupci Print, Line, Circle i Refresh korisni su kod ispisa ili crtanja direktno na površinu forme. O ovim postupcima više se raspravlja u 12. poglavlju “Rad sa tekstom i grafikom”.

Za više informacija Za dodatne informacije o formama, pogledajte “Više o formama” u 6. poglavlju “Stvaranje korisničkog sučelja”.

Pokretanje akcija klikom na gumb

Najlakši način kako omogućiti korisniku rad s aplikacijom je stvoriti mu gumb koji treba kliknuti. Možete iskoristiti kontrolu naredbenog gumba koja postoji u Visual Basicu, ili možete stvoriti vlastiti "gumb" koristeći kontrolu slike koja će sadržavati grafiku, kao ikona.

Korištenje naredbenih gumbâ

Većina aplikacija stvorenih u Visual Basicu ima naredbene gumbe koji omogućuju korisniku pokretanje akcija jednostavnim klikom. Kad korisnik odabere gumb, taj gumb ne poduzima samo prikladnu akciju, nego i izgleda kao da je pritisnut i pušten. Kad god korisnik klikne gumb, poziva se potprogram postupka Click. Programski kod koji treba biti izvršen postavite u taj potprogram.

Postoji puno načina kako odabrati naredbeni gumb tijekom izvođenja aplikacije:

- Upotrijebite miša za klik na gumb.
- Dovedite fokus na gumb pritiskanjem na tipku TAB, i nakon toga odaberite gumb pritiskom na tipke SPACEBAR ili ENTER (pogledajte "Razumijevanje fokusa" u nastavku ovog poglavlja).
- Pritisnite kombinaciju tipki za aktiviranje pristupnog znaka (ALT + podvučeno slovo) na naredbenom gumbu.
- U programskom kodu promijenite vrijednost svojstva Value naredbenog gumba u True:

  ```
  cmdClose.Value = True
  ```

- U programskom kodu pozovite događaj Click naredbenog gumba:

  ```
  cmdClose_Click
  ```

- Ako je naredbeni gumb *podrazumijevani naredbeni gumb* forme, pritisak na ENTER će odabrati gumb, čak i ako pomaknete fokus na neku drugu kontrolu. Tijekom izrade aplikacije možete odrediti podrazumijevani naredbeni gumb postavljanjem svojstva Default tog gumba na True.
- Ako je naredbeni gumb podrazumijevani *gumb za opoziv* (Cancel button) forme, pritisak na ESC će odabrati taj gumb, čak i ako je fokus na nekoj drugoj kontroli. Tijekom izrade aplikacije možete odrediti gumb za opoziv postavljanjem njegovog svojstva Cancel na True.

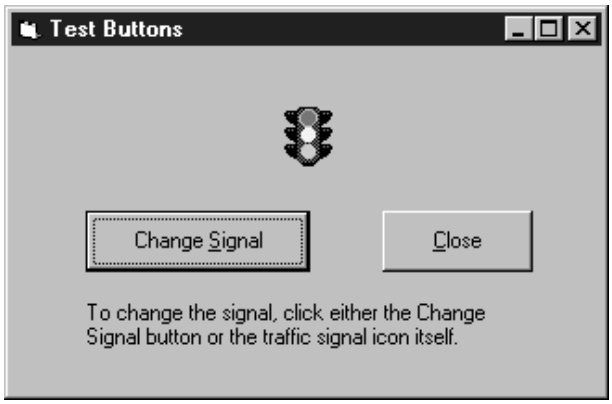
Sve ove akcije su povod Visual Basicu za poziv potprograma Click.

Aplikacija Test Buttons

Svojstvo Caption možete upotrijebiti za ispis teksta na naredbenom gumbu kako bi korisniku dali do znanja što taj gumb radi. Na slici 3.4, primjer s naredbenim gumbima iz aplikacije Controls prikazuje naredbeni gumb sa sadržajem svojstva Caption

postavljenim na “Change Signal” (za radnu verziju ovog primjera, pogledajte Button.frm u aplikaciji Controls.vbp). Uočite da podvučeno slovo “S” predstavlja pristupni znak ovom naredbenom gumbu. Ubacivanje znaka & u tekst svojstva Caption označava slovo koje slijedi pristupnim znakom (na primjer, Change &Signal).

Slika 3.4 Naredbeni gumb s naslovom



Kad korisnik klikne naredbeni gumb, izvršit će se programski kod koji se nalazi u njegovom potprogramu Click. U ovom primjeru, svaki put kad korisnik klikne gumb, pojavit će se drugačija ikona semafora.

Za više informacija O ostalim svojstvima naredbenog gumba, pogledajte 7. poglavlje “Korištenje standardnih kontrola Visual Basica”.

Kontrole za prikaz i unos teksta

Kontrole natpisa (label) i okvira s tekstom (text box) koriste se za ispis i unošenje teksta. Natpise možete koristiti kad želite da aplikacija prikazuje tekst na formi, a okvire s tekstom kad želite omogućiti korisniku da unese tekst. Natpisi sadrže tekst koji se može samo čitati, dok okviri s tekstom sadrže tekst koji može biti mijenjan.

za pružanje ovih mogućnosti	iskoristite ovu kontrolu
Tekst kojeg korisnik može mijenjati, na primjer polje za upis narudžbe ili lozinke	okvir s tekstom
Tekst koji se samo prikazuje, na primjer opis polja na formi ili ispis uputa korisniku	natpis

Korištenje natpisa za prikaz teksta

Kontrola natpisa sadrži tekst koji korisnik ne može direktno mijenjati. Možete iskoristiti natpise za opis kontrola, kao što su okviri s tekstom i trake za pomicanje, koje nemaju vlastito svojstvo Caption. Trenutni tekst ispisan u natpisu određen je vrijednošću svojstva Caption, i može biti postavljen tijekom izrade aplikacije uz pomoć prozora sa svojstvima ili tijekom izvođenja aplikacije postavljanjem iz programskog koda.

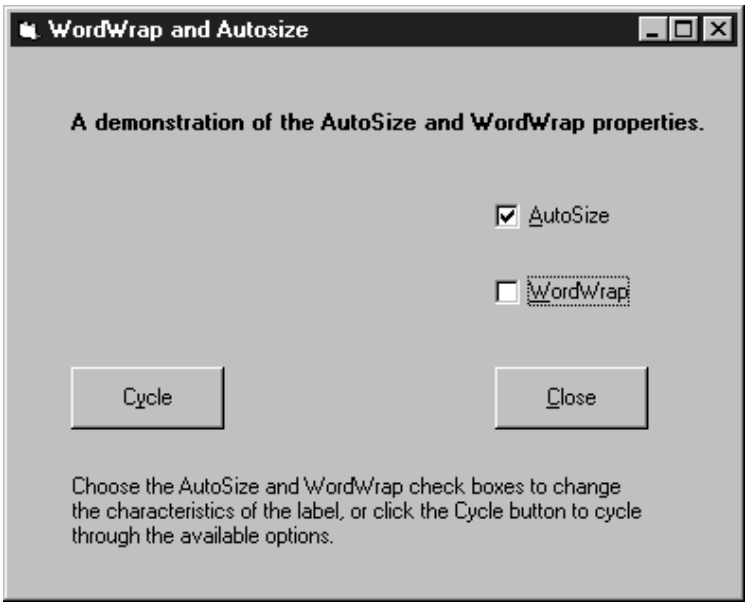
U pravilu, tekst je jedini vidljivi dio kontrole natpisa. Svejedno, ako vrijednost svojstva BorderStyle postavite na 1 (što možete napraviti tijekom izrade aplikacije), natpis će se pojavljivati s okvirom – izgledat će slično okviru s tekstom. Izgled kontrole natpisa možete promijeniti postavljanjem drugačijih vrijednosti u svojstva BackColor, BackStyle, ForeColor i Font.

Veličina natpisa prilagođena sadržaju

Jednolinijski tekstovi kao sadržaji natpisa mogu biti određeni tijekom izrade aplikacije u prozoru sa svojstvima. Što ako želite upisati dulji tekst ili ga promijeniti tijekom izvođenja aplikacije? Natpis ima dva svojstva koja će vam pomoći u promjeni veličine kontrole tako da odgovaraju duljem ili kraćem sadržaju: AutoSize i WordWrap.

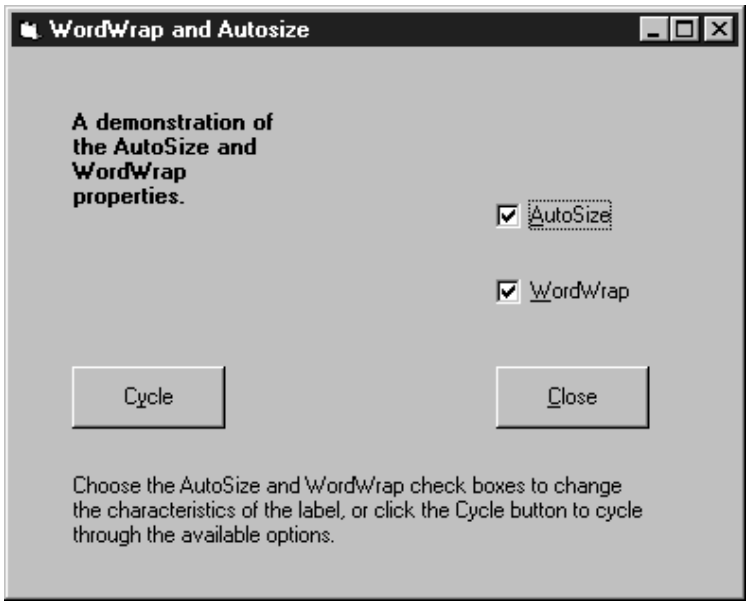
Svojstvo AutoSize određuje hoće li kontrola automatski promijeniti veličinu tako da odgovara sadržaju. Ako je postavljeno na True, natpis će postati dulji tako kako bi mogao ispisati cijeli sadržaj, kao što je prikazano na slici 3.5.

Slika 3.5 Primjer svojstva AutoSize



Svojstvo AutoWrap uzrokuje povećanje natpisa po okomici da bi cijeli tekst mogao biti prikazan, dok širina ostaje ista, kao što je prikazano na slici 3.6. Za radnu verziju ovog primjera, pogledajte Wordwrap.frm u primjeru aplikacije Controls.vbp.

Slika 3.6 Primjer svojstva WordWrap



Napomena Ako pokrenete primjer AutoSize svojstva iz aplikacije Controls.vbp, uočit ćete da obje kontrolne kućice moraju biti potvrđene kako bi radio primjer svojstva WordWrap. Razlog tome je da svojstvo AutoSize mora biti postavljeno na True kako bi svojstvo WordWrap imalo učinka. Vodoravna dimenzija natpisa će se povećati samo ako je duljina najdulje riječi veća od trenutne duljine kontrole.

Za više informacija Za dodatne informacije o svojstvima kontrole natpisa pogledajte 7. poglavlje "Korištenje standardnih kontrola Visual Basica".

Rad s okvirima s tekstom

Okviri s tekstom su svestrane kontrole koje mogu biti korištene za upis teksta od strane korisnika ili za prikaz teksta. Okviri za tekst ne bi se trebali koristiti za prikaz teksta koji korisnik ne smije mijenjati, osim ako ne postavite vrijednost svojstva Locked na True.

Tekst koji je trenutno ispisan u okviru s tekstom određen je sadržajem svojstva Tekst. On može biti određen na tri različita načina: tijekom izrade aplikacije putem prozora sa svojstvima, tijekom izvođenja aplikacije postavljanjem iz programskog koda, ili upisom od strane korisnika. Trenutni sadržaj okvira s tekstom može biti dohvaćen tijekom izvođenja aplikacije čitanjem sadržaja svojstva Text.

Višelinijski okviri s tekstom i prijelom teksta

U pravilu, okvir s tekstom prikazuje jednu liniju teksta i ne prikazuje trake za pomicanje. Ako je tekst dulji od raspoloživog prostora, bit će vidljiv samo dio teksta. Izgled i ponašanje okvira s tekstom može biti promijenjeno postavljanjem drugačijih vrijednosti dvaju svojstava, MultiLine i ScrollBars, koja su dostupna samo za vrijeme kreiranja aplikacije.

> Napomena Svojstvo ScrollBars ne smijete pomiješati s kontrolama traka za pomicanje, koje nisu dio okvira s tekstom i imaju vlastiti komplet svojstava.

Postavljanje svojstva MultiLine na True omogućuje okviru s tekstom da prihvati ili prikaže više linija teksta tijekom izvođenja aplikacije. Višelinijski okvir s tekstom automatski upravlja prijelomom teksta sve dok ne postoji vodoravna traka za pomicanje. Svojstvo ScrollBars po standardu ima vrijednost 0–None. Automatski prijelom teksta pošteđuje korisnika od problema umetanja znakova za prekid linije na kraju linija teksta. Kad je linija teksta dulja od onog što možemo prikazati u jednoj liniji, okvir s tekstom prelomit će tekst u iduću liniju.

Znakovi za prekid linije ne mogu biti upisivani u prozor sa svojstvima tijekom kreiranja aplikacije. Unutar potprograma, možete kreirati prekid linije ubacivanjem oznake kraja retka (carriage return, ANSI kod 13) zajedno s oznakom za novi red (linefeed, ANSI kod 10). Također možete upotrijebiti i konstantu vbCrLf za ubacivanje kombinacije kraj reda/novi red. Na primjer, sljedeći potprogram ispisuje dvije linije teksta u višelinijski okvir s tekstom (Text1) kod učitavanja forme:

```
Sub Form_Load ()
    Text1.Text = “Ovo su dvije linije” _
    & vbCrLf & “u okviru za tekst”
End Sub
```

Rad s tekstom u okviru za tekst

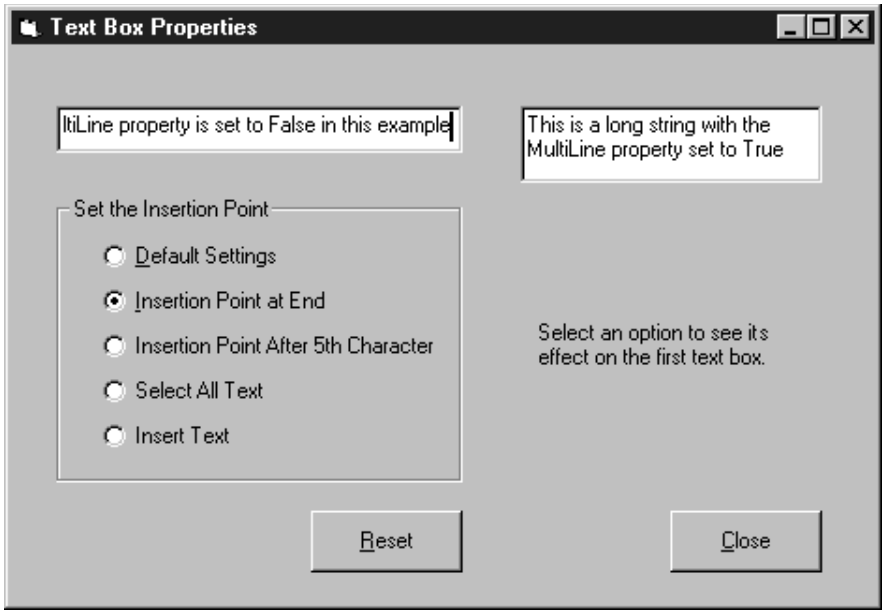
Mjesto ubacivanja teksta i ponašanje označenog teksta u okviru s tekstom možete kontrolirati svojstvima SelStart, SelLength i SelText. Ova svojstva dostupna su samo tijekom izvođenja aplikacije.

Kad okvir za tekst prvi put dobije fokus, standardno mjesto ubacivanja teksta ili mjesto kursora unutar okvira s tekstom je lijevo od postojećeg teksta. Korisnik ga može pomaknuti tipkovnicom ili mišem. Ako okvir s tekstom izgubi i zatim ponovno dobije fokus, ovo mjesto bit će tamo gdje je to odredio korisnik posljednji put.

U nekim slučajevima, ovakvo ponašanje može zbuniti korisnika. U aplikaciji za obradu teksta, korisnik može očekivati pojavljivanje novih karaktera iza postojećeg teksta. U aplikaciji za unos podataka, korisnik može očekivati da će njegov upis teksta zamijeniti postojeći upis. Svojstva SelStart i SelLength dopuštaju vam promjene ponašanja tako da odgovaraju vašim potrebama.

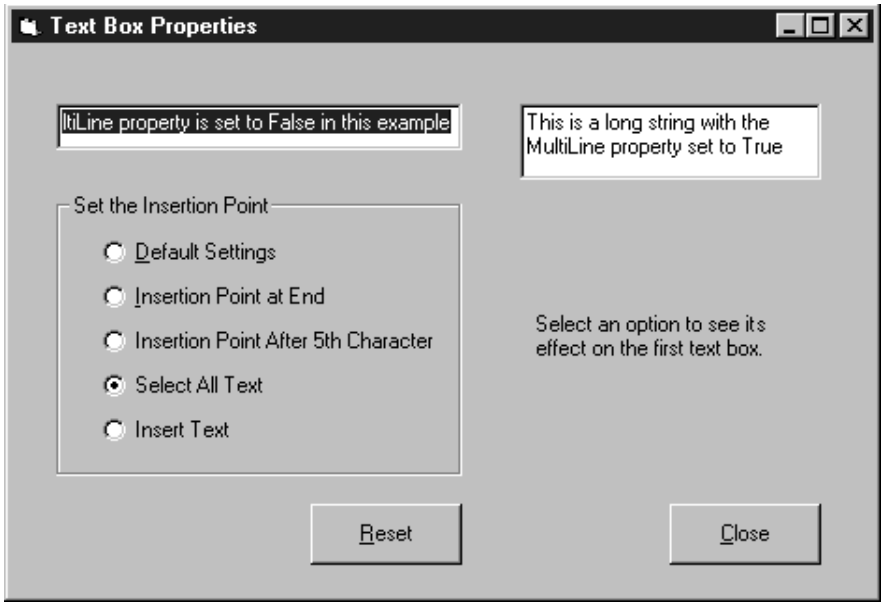
Vrijednost svojstva SelStart je broj koji ukazuje na mjesto ubacivanja teksta unutar tekstualnog niza, gdje je 0 krajnji lijevi položaj. Ako svojstvo SelStart sadrži vrijednost koja je jednaka ili veća od broja karaktera unutar okvira s tekstom, mjesto ubacivanja teksta bit će postavljeno iza posljednjeg karaktera, kako je prikazano na slici 3.7. Za radnu verziju ovog primjera, pogledajte Text.frm u aplikaciji Controls.vbp.

Slika 3.7 Primjer mjesta ubacivanja teksta



Vrijednost svojstva SelLength je broj koji određuje širinu mjesta ubacivanja. Dodjeljivanjem vrijednosti veće od 0 ovom svojstvu odgovarajući broj karaktera će biti odabran i označen, počevši od trenutnog mjesta ubacivanja. Slika 3.8 pokazuje odabir teksta.

Slika 3.8 Primjer odabira teksta



Ako korisnik počne upisivati tekst dok je dio teksta označen, taj dio bit će zamijenjen upisanim. U nekim slučajevima, možete poželjeti zamijeniti označeni tekst novim koristeći naredbu lijepljenja. Svojstvo SelText je tekstualni niz kojeg možete postaviti umjesto odabranog. Ako ne postoji odabrani tekst, SelText će zalijepiti svoj tekst na mjesto ubacivanja.

Za više informacija Za dodatne informacije o svojstvima kontrole okvira s tekstom, pogledajte 7. poglavlje "Korištenje standardnih kontrola Visual Basica".

Kontrole s izborom

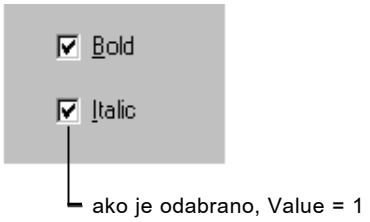
U većini aplikacija potrebno je pružiti neki izbor korisnicima, počevši od jednostavnog da/ne, do odabira s liste koja sadržava nekoliko stotina izbora. Visual Basic uključuje nekoliko standardnih kontrola koje su korisne za predstavljanje izbora. Sljedeća tabela rezimira te kontrole i njihovu odgovarajuću uporabu.

za omogućavanje ovih mogućnosti	odaberite ovu kontrolu
Manji broj izbora u kojoj korisnik može izabrati jednu ili više stavki	kontrolne kućice (check boxes)
Manji broj izbora od kojih korisnik može odabrati samo jedan	gumbi izbora (option boxes) – upotrijebite okvire (frames) za stvaranje dodatnih grupa
Pomična lista izbora	okvir s listom (list box)
Pomična lista izbora zajedno s poljem za upis teksta. Korisnik može odabrati stavku s liste ili upisati izbor u tekstualno polje.	kombinirani okvir (combo box)

Odabir pojedinih izbora kontrolnim kućicama

Kontrolna kućica pokazuje je li određeni uvjet uključen ili isključen. Kontrolne kućice možete upotrijebiti u aplikaciji za davanje točno/netočno ili da/ne izbora korisnika. Pošto su kontrolne kućice međusobno neovisne, korisnik istovremeno može odabrati željeni broj izbora. U primjeru na slici 3.9 podebljano (bold) i nakošeno (italic) mogu biti istovremeno odabrani.

Slika 3.9 Kontrolne kućice

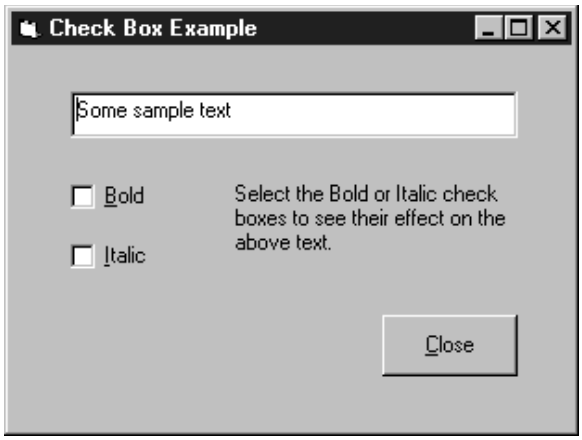


Aplikacija Check Box

Aplikacija s kontrolnim kućicama Check Box koristi ove kontrole za utvrđivanje hoće li tekst biti ispisan u normalnom, podebljanom ili nakošenom pismu. Za radnu verziju ovog primjera, pogledajte Check.frm u aplikaciji Controls.vbp.

Aplikacija sadrži okvir s tekstom, natpis, naredbeni gumb i dvije kontrolne kućice, kao što je prikazano na slici 3.10.

Slika 3.10 Primjer kontrolnih kućica



Sljedeća tabela sadrži vrijednosti svojstava za objekte u ovoj aplikaciji.

objekt	svojstvo	vrijednost
Forma	Name Caption	frmCheck Check Box Example
Okvir s tekstom	Name Text	txtDisplay Some sample text
Prva kontrolna kućica	Name Caption	chkBold &Bold
Druga kontrolna kućica	Name Caption	chkItalic &Italic
Naredbeni gumb	Name Caption	cmdClose &Close

Kad potvrdite Bold ili Italic, svojstvo kontrolne kućice Value dobiva vrijednost 1; kad je odznačena, svojstvo Value ima vrijednost 0. Standardna vrijednost je 0, pa ako ne promijenite svojstvo Value, kontrolne kućice će biti prazne prvi put kad se pojave. Možete iskoristiti konstante vbChecked i vbUnchecked kao zamjenu za vrijednosti 1 i 0.

Događaji u aplikaciji s kontrolnim kućicama

Događaj Click kontrolne kućice pojavljuje se onog trenutka kad kliknete na tu kontrolu. Potprogram ovog događaja ispituje je li kontrolna kućica potvrđena (je li Value = vbChecked). Ako je, tekst se pretvara u podebljano ili nakošeno pismo postavljanjem svojstava Bold ili Italic objekta Font dobivenog od svojstva Font okvira s tekstom.

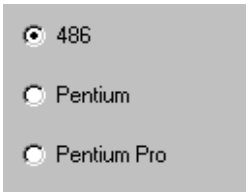
```
Private Sub chkBold_Click ()
    If ChkBold.Value = vbChecked Then            ‘ ako je potvrđen
      txtDisplay.Font.Bold = True
    Else                           ‘ ako nije potvrđen
       txtDisplay.Font.Bold = False
    End If
End Sub
Private Sub chkItalic_Click ()
    If ChkItalic.Value = vbChecked Then          ‘ ako je potvrđen
       txtDisplay.Font.Italic = True
    Else                           ‘ ako nije potvrđen
       txtDisplay.Font.Italic = False
    End If
End Sub
```

Grupiranje izbora gumbima izbora

Gumbi izbora daju korisniku skup od dva ili više izbora. Za razliku od kontrolnih kućica, međutim, gumbi izbora trebali bi uvijek biti dio grupe: odabir jednog gumba izbora trenutno poništava sve ostale izbore u grupi. Određivanje grupe s gumbima izbora kaže korisniku “Ovdje je grupa mogućnosti od kojih možeš izabrati jednu i samo jednu”.

Na primjer, u grupi s gumbima izbora prikazanoj na slici 3.11, korisnik može odabrati jednu od tri mogućnosti.

Slika 3.11 Odabir gumba izbora

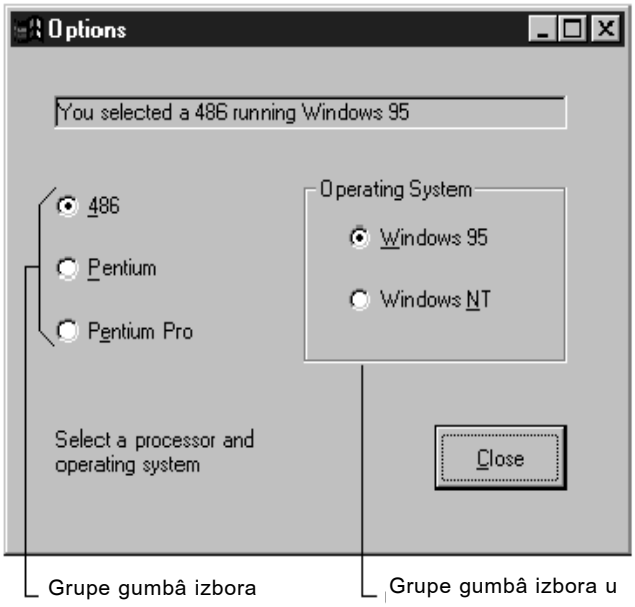


Stvaranje grupa s gumbima izbora

Svi gumbi izbora postavljeni na formu (dakle, ne u okvir ili okvir za sliku) čine jednu grupu. Ako želite kreirati dodatne grupe s gumbima izbora, neke od njih morate postaviti u okvire ili okvire za sliku.

Svi gumbi izbora unutar okvira čine zasebnu grupu, kao i svi gumbi izbora unutar okvira za sliku. Kad stvarate posebnu grupu na ovaj način, uvijek najprije kreirajte okvir ili okvir za sliku, te tek nakon toga na njemu kreirajte gumbe izbora. Slika 3.12 prikazuje formu s dvije grupe gumbi izbora.

Slika 3.12 Grupe gumbâ izbora



Korisnik može odabrati samo jedan gumb izbora u grupi kad kreirate gumbe izbora u okviru.

Kako grupirati kontrole u okviru

1. Odaberite kontrolu okvira (frame) u alatnom okviru i kreirajte okvir na formi.
2. Odaberite kontrolu gumba izbora u alatnom okviru i kreirajte je unutar okvira (frame).
3. Ponavljajte 2. korak za svaki dodatni gumb izbora koji želite dodati u okvir.

Kreiranje okvira pa tek zatim kreiranje kontrola na okviru dopušta vam pomicanje okvira zajedno s kontrolama. Ako već postojeće kontrole pomaknete na okvir, one se neće pomicati pri pomaku okvira.

Napomena Ako imate već kreirane kontrole koje želite grupirati na okviru, odaberite sve kontrole, odsijecite ih (cut) i ulijepite (paste) na kontrolu okvira ili okvira za sliku.

Spremnici kontrola

Iako su kontrole neovisni objekti, postoji stanovit odnos *roditelj–dijete* (*parent and child relationship*) između formi i kontrola. Slika 3.12 pokazuje kako gumbi izbora mogu biti sadržani unutar forme ili kontrole okvira.

Za razumijevanje pojma spremnika, trebate shvatiti da su sve kontrole djeca forme na kojoj se nalaze. Zapravo, većina kontrola podržava svojstvo Parent koje se može samo pročitati, a koje vraća formu na kojoj se nalazi kontrola. Svojstva kontrole Left i Top odnose se na roditeljsku formu, i kontrole ne mogu biti pomaknute izvan granica roditelja. Pomicanje spremnika pomiče i kontrole, a položaj kontrole koji je relativan u odnosu na svojstva spremnika Left i Top se ne mijenja jer se kontrole pomiču zajedno sa spremnikom.

Potvrđivanje ili odznačivanje gumbâ izbora

Gumb izbora može biti odabran:

- Klikom mišem tijekom izvođenja.
- Dolaskom na grupu gumbâ izbora uz pomoć tipke TAB te korištenjem kursorskih tipaka za odabir gumba unutar grupe.
- Postavljanjem svojstva Value na vrijednost True programskim kodom:

  ```
  optChoice.Value = True
  ```

- Korištenjem pristupne prečice (shortcut key) koja je naznačena u sadržaju natpisa.

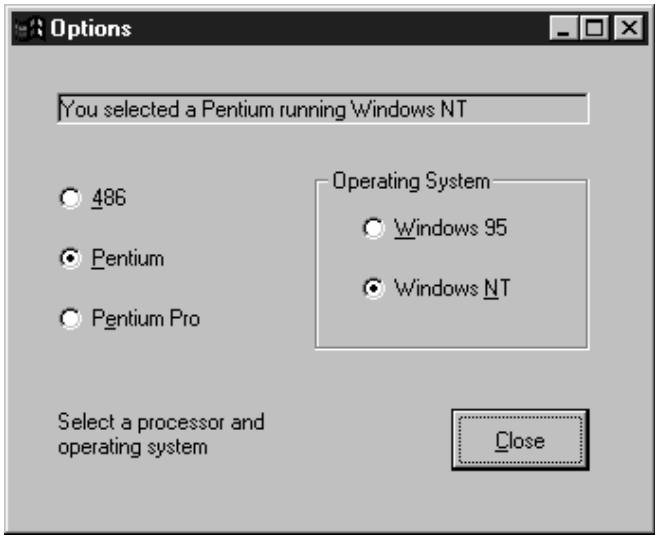
Da bi gumb izbora bio predodređen u grupi, postavite njegovo svojstvo Value na True tijekom izrade aplikacije. On će ostati odabran sve dok korisnik ne odabere neki drugi gumb ili dođe do promjene programskim kodom.

Za onemogućavanje gumba izbora, postavite njegovo svojstvo Enabled na False. Tijekom rada aplikacije on će imati zadimljen natpis, što označava njegovu nedostupnost.

Aplikacija Options

Forma prikazana na slici 3.13 koristi gumbe izbora za utvrđivanje tipa procesora i operativnog sustava za zamišljeno računalo. Kad korisnik odabere gumb izbora u jednoj ili drugoj grupi, sadržaj kontrole natpisa se mijenja tako da odražava trenutni izbor. Za radnu verziju ovog primjera, pogledajte Option.frm u aplikaciji Controls.vbp.

Slika 3.13 Primjer sa gumbima izbora



Sljedeća tablica sadrži vrijednosti svojstava za objekte ove aplikacije.

objekt	svojstvo	vrijednost
natpis (label)	Name Caption	lblDisplay (prazno)
naredbeni gumb	Name Caption	cmdClose &Close
prvi gumb izbora	Name Caption	opt486 &486
drugi gumb izbora	Name Caption Value	opt586 &Pentium True
treći gumb izbora	Name Caption	opt686 P&entium Pro
okvir (frame)	Name Caption	fraSystem &Operatimg System
četvrti gumb izbora	Name Caption	optWin95 &Windows 95
peti gumb izbora	Name Caption Value	optWinNT Windows &NT True

Događaji u aplikaciji Options

Aplikacija Options odgovara na događaje na sljedeći način:

- Događaji Click za prva tri gumba izbora dodjeljuju odgovarajući opis u tekstovnu varijablu forme, strComputer.
- Događaji Click za posljednja dva gumba izbora dodjeljuju odgovarajući opis u drugu tekstovnu varijablu forme, strSystem.

Ključ ovakvom pristupu je korištenje dvije varijable na nivou forme, strComputer i strSystem. Ove varijable sadrže različite tesktovne nizove, ovisno o tome koji je gumb izbora odabran.

Svaki put kad je odabran novi gumb izbora, programski kod u događaju Click ažurira odgovarajuću varijablu:

```
Private Sub opt586_Click()
    strComputer = “Pentium”
    Call DisplayCaption
End Sub
```

Ovaj kod poziva potprogram, nazvan DisplayCaption, koji povezuje dvije varijable i ažurira sadržaj svojstva natpisa Caption:

```
Sub DisplayCaption()
    lblDisplay.Caption = You selected, a “ & _
    strComputer & “ running “ & strSystem
End Sub
```

Ovaj potprogram se koristi jer je postupak ažuriranja svojstva Caption u biti jednak za svih pet gumbâ izbora, jedino se vrijednost varijable mijenja od slučaja do slučaja. Ovakav postupak pošteđuje vas ponavljanja istog programskog koda za svaki događaj Click.

Za više informacija O varijablama i potprograma detaljno se raspravlja u 5. poglavlju “Osnove programiranja”.

Korištenje okvira s popisom i kombiniranih okvira

Okviri s popisom (list box) i kombinirani okviri (combo box) daju listu izbora korisniku. U pravilu, stavke su ispisane okomito u jednoj koloni, iako možete napraviti i okvire sa više kolona. Ako je broj stavki veći od onog što okvir s popisom ili kombinirani okvir mogu prikazati, u kontroli će se automatski pojaviti trake za pomicanje. Korisnik se tada može kretati po listi s lijeva na desno ili gore – dolje. Slika 3.14 prikazuje jednokolonski okvir s popisom.

Slika 3.14 Jednokolonski okvir s popisom

Kombinirani okvir spaja mogućnosti okvira s tekstom i okvira s popisom. Ova kontrola dopušta korisniku da odabere stavku upisom teksta u kombinirani okvir ili biranjem stavke sa popisa. Slika 3.15 prikazuje kombinirani okvir.

Slika 3.15 Kombinirani okvir

U usporedbi s nekim drugim kontrolama koje sadrže jednostruku vrijednost, na primjer svojstvo Caption kontrole natpisa ili svojstvo Text kontrole okvira s tekstom, okviri s popisom i kombinirani okviri sadrže više vrijednosti ili zbirku vrijednosti. Ove kontrole imaju ugrađene metode za dodavanje, brisanje i pronalaženje vrijednosti iz njihovih zbirki tijekom rada aplikacije. Za dodavanje nekoliko stavki u okvir s popisom nazvan List1, programski kod bi mogao izgledati ovako:

```
List1.AddItem "Paris"
List1.AddItem "New York"
List1.AddItem "San Francisco"
```

Okviri s popisom i kombinirani okviri su djelotvoran način predstavljanja velikog broja stavki korisniku na ograničenom prostoru.

Za više informacija Za dodatne informacije o okvirima s popisom i kombiniranim okvirima, pogledajte 7. poglavlje "Korištenje standardnih kontrola Visual Basica".

Korištenje kliznih traka kao ulaznih jedinica

Iako su klizne trake često vezane uz okvire s tekstom ili prozore, ponekad ćete vidjeti i kako se koriste kao ulazne jedinice. Budući da ove kontrole prikazuju trenutni položaj na skali, one se mogu koristiti zasebno za unos podataka u aplikaciji – na primjer, za kontrolu jačine zvuka ili za prilagođavanje boja na slici. Kontrole HScrollBar (vodoravna traka) i VScrollBar (okomita traka) djeluju neovisno od ostalih kontrola i imaju vlastitu zbirku događaja, svojstava i postupaka. Kontrole kliznih traka nisu iste kao i one ugrađene u okvire s tekstom, okvire s popisom, kombinirane okvire ili MDI forme (okviri s tekstom i MDI forme imaju svojstvo ScrollBars uz pomoć kojeg možete dodati ili maknuti klizne trake ugrađene u te kontrole).

Smjernice Windows sučelja savjetuju korištenje kontrola klizača (slider) kao ulaznih jedinica umjesto kliznih traka. Primjeri kontrola klizača može se vidjeti u kontrolnom prozoru (control panel) Windowsa 95/98. Kontrola klizača takvog izgleda uključena je u Professional i Enterprise verzije Visual Basica.

Za više informacija Za dodatne informacije o kontrolama kliznih traka, pogledajte 7. poglavlje "Korištenje standardnih kontrola Visual Basica".

Kontrole za prikaz slika i grafike

Budući da Windowsi grafičko korisničko sučelje, važno je imati način prikazivanja grafike i slika u sučelju vaše aplikacije. Visual Basic uključuje četiri kontrole koje pojednostavljuju rad s grafikom; kontrolu okvira za sliku (picture box), kontrolu slike (image), kontrolu lika (shape) i kontrolu linije (line).

Kontrole slike, lika i linije ponekad se nazivaju i grafičkim kontrolama "lake kategorije". One zahtijevaju manje sistemskih izvora i zato prikazuju objekte nešto brže od kontrole okvira za sliku; sadrže dio svojstava, postupaka i događaja dostupnih u okviru za sliku. Svaka od ovih kontrola opremljena je za određenu namjenu.

za pružanje ove mogućnosti	iskoristite ovu kontrolu
spremnik drugih kontrola	okvir za sliku
postupci tiskanja ili crtanja	okvir za sliku
prikaz slike	kontrola slike ili okvir za sliku
prikaz jednostavnog grafičkog elementa	kontrola lika ili linije

Rad s kontrolom okvira za sliku

Glavna svrha kontrole okvira za sliku je prikazivanje slike korisniku. Trenutna slika koja se prikazuje određena je svojstvom Picture. Svojstvo Picture sadrži ime datoteke (i stazu, ako je potrebno) slike koju želite prikazati.

> Napomena Objekti forme također imaju svojstvo Picture koje može biti podešeno tako da prikazuje sliku direktno na podlozi forme.

Za prikaz i zamjenu slike tijekom rada aplikacije, iskoristit ćete funkciju LoadPicture za postavljanje vrijednosti svojstva Picture. Dovoljno je dati ime (i stazu, ako je potrebno) slike, a funkcija LoadPicture će se pobrinuti za detalje učitavanja i prikazivanja slike:

```
picMain.Picture = LoadPicture("VANGOGH.BMP")
```

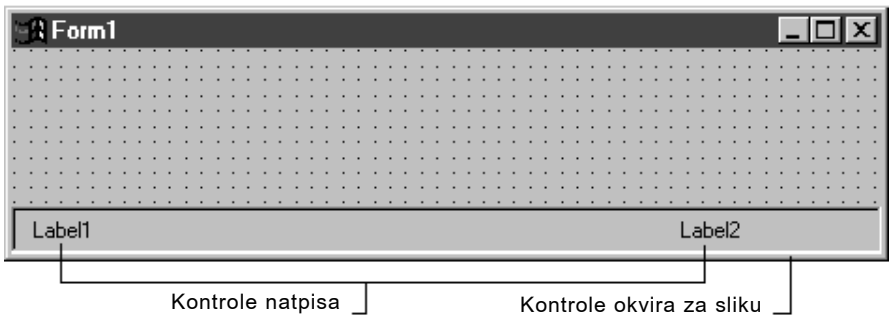
Kontrola okvira za sliku ima svojstvo AutoSize koje, ako je postavljeno na True, uzrokuje automatsku promjenu veličine okvira za sliku tako da odgovara dimenzijama sadržaja. Kreirajte svoju formu s posebnom pažnjom ako namjeravate koristiti okvir za sliku sa uključenim svojstvom AutoSize. Slika će promijeniti veličinu bez obzira na ostale kontrole na formi, vjerojatno uzrokujući neočekivane rezultate, kao što je prekrivanje drugih kontrola. Dobra je ideja isprobavanje ovog postupka učitavanjem svake od slika tijekom izrade aplikacije.

Korištenje okvira za sliku kao spremnika

Kontrola okvira za sliku može se iskoristiti i kao spremnik drugih kontrola. Kao i kod kontrole okvira, možete postaviti druge kontrole unutar okvira za sliku. Tako sadržane kontrole pomiču se sa okvirom za sliku i njihova svojstva Top i Left relativna su u odnosu na okvir za sliku, a ne na formu.

Uobičajena upotreba okvira za sliku kao spremnika je kao alatna traka ili statusna traka. Na okvir možete postaviti kontrole natpisa koje će djelovati kao gumbi, ili dodati natpise koji će prikazivati poruke o trenutnom stanju. Postavljanjem svojstva Align na Top, Bottom, Left ili Right okvir za sliku će se "zalijepiti" uz rub forme. Slika 3.16 prikazuje okvir za sliku sa svojstvom Align postavljenim na Bottom. Ovaj okvir sadržava dvije kontrole natpisa koje mogu biti iskorištene za prikaz statusnih poruka.

Slika 3.16 Okvir za sliku korišten kao statusna traka



Ostale upotrebe okvira za sliku

Okvir za sliku ima nekoliko postupaka koji ga čine korisnim u druge svrhe. Zamislite okvir za sliku kao prazno platno po kojem možete bojati, crtati ili pisati. Jedna kontrola može biti iskorištena za prikaz teksta, grafike ili čak jednostavnije animacije.

Postupak Print omogućuje vam usmjeravanje teksta prema okviru za sliku isto kao i prema pisač. Više svojstava pisma dostupno vam je za nadzor svojstava ispisa teksta postupkom Print; postupak Cls može biti iskorišten za brisanje ispisa.

Postupci Circle, Line, Point i Pset mogu biti iskorišteni za crtanje grafike u okviru za sliku. Svojstva kao što su DrawWidth, FillColor i FillStyle omogućuju vam prilagodbu izgleda grafike.

Animacija može biti kreirana korištenjem svojstva PaintPicture koje će pomicati slike unutar kontrole okvira za sliku i brzim mijenjanjem nekoliko različitih slika.

Za više informacija Za dodatne informacije o kontroli okvira za sliku, pogledajte 7. poglavlje “Korištenje standardnih kontrola Visual Basica”.

Grafičke kontrole lake kategorije

Kontrole slike, lika i linije smatraju se kontrolama lake kategorije; razlog je podrška samo dijelu svojstava, postupaka i događaja koje podržava okvir za sliku. Zbog toga, ove kontrole uobičajeno zahtijevaju manje sistemskih izvora i učitavaju slike brže od okvira za sliku.

Korištenje kontrole slike umjesto okvira za sliku

Kontrola slike slična je kontroli okvira za sliku, ali koristi se samo za prikazivanje slika. Nema sposobnost djelovanja kao spremnik drugih kontrola, i ne podržava napredne postupke okvira za sliku.

Slike se učitavaju u kontrolu slike kao i u okvir za sliku: tijekom izrade aplikacije, upišite ime slike i stazu u svojstvo Picture; tijekom rada aplikacije iskoristite funkciju LoadPicture.

Ponašanje promjene veličine kontrole slike razlikuje se od okvira za sliku. Kontrola slike ima svojstvo Stretch dok okvira za sliku ima svojstvo AutoSize. Postavljanje svojstva AutoSize na True uzrokuje kod okvira za sliku promjenu veličine prema dimenzijama slike; postavljanje na False ima za posljedicu odsijecanje slike (vidljiv je samo dio slike). Kad je postavljeno na False (po standardu), svojstvo Stretch kontrole slike ima za posljedicu promjenu veličine slike prema veličini kontrole, što može prouzročiti izobličenje slike.

Za više informacija Za dodatne informacije o kontroli slike pogledajte 7. poglavlje “Korištenje standardnih kontrola Visual Basica”.

Korištenje kontrole slike za stvaranje gumbâ

Kontrola slike također prepoznaje događaj Click, tako da je možete koristiti umjesto naredbenog gumba. To je zgodan način stvaranja gumba sa slikom umjesto s natpisom. Grupiranje više kontrola slike vodoravno pri vrhu ekrana – obično unutar okvira za sliku – omogućuje vam stvaranje alatne trake u vašoj aplikaciji.

Primjer Test Buttons pokazuje kontrolu slike koju korisnik može odabrati kao što bi odabrao naredbeni gumb. Kad se forma prvi put prikaže, kontrola slike prikazuje jednu od tri ikone semafora iz biblioteke Icon instalirane s Visual Basicom. Svaki put kad se klikne na kontrolu slike, bit će prikazana drugačija ikona (za radnu verziju ovog primjera, pogledajte Button.frm u aplikaciji Controls.vbp).

Ako pregledate formu tijekom izrade aplikacije, vidjet ćete da ona zapravo sadrži sve tri ikone koje su "složene u hrpu" jedna iznad druge. Mijenjanjem svojstva Visible najgornje ikone u False, omogućujete idućoj ikoni (sa svojstvom Visible postavljenim na True) pojavljivanje na vrhu.

Slika 3.17 prikazuje kontrolu slike s jednom od ikona semafora (Trffc10a.ico).

Slika 3.17 Kontrola slike s ikonom semafora

Za stvaranje okvira oko kontrole slike, postavite svojstvo BorderStyle na 1–Fixed Single.

> Napomena Za razliku od naredbenih gumbâ, kontrole slike ne izgledaju pritisnute kad kliknete na njih. To znači da ako ne promijenite sliku događajem MouseDown, nema vidljive oznake korisniku da je "pritisnuo" gumb.

Za više informacija Za informacije o prikazivanju slika i grafike u kontroli slike, pogledajte 7. poglavlje "Korištenje standardnih kontrola Visual Basica".

Korištenje kontrola lika i linije

Kontrole lika i linije su korisne kod crtanja grafičkih elemenata na površini forme. Ove kontrole ne podržavaju nikakve događaje; njihova namjena je isključivo ukrasna.

Postoji nekoliko svojstava za reguliranje pojavljivanja kontrole lika. Postavljanjem svojstva Shape, ova kontrola može biti prikazana kao pravokutnik, kvadrat, elipsa, krug, te pravokutnik ili kvadrat sa zaobljenim kutovima. Svojstva BorderColor i FillColor mogu odrediti boju; svojstva BorderStyle, BorderWidth, FillStyle i DrawMode određuju kako će lik biti iscrtan.

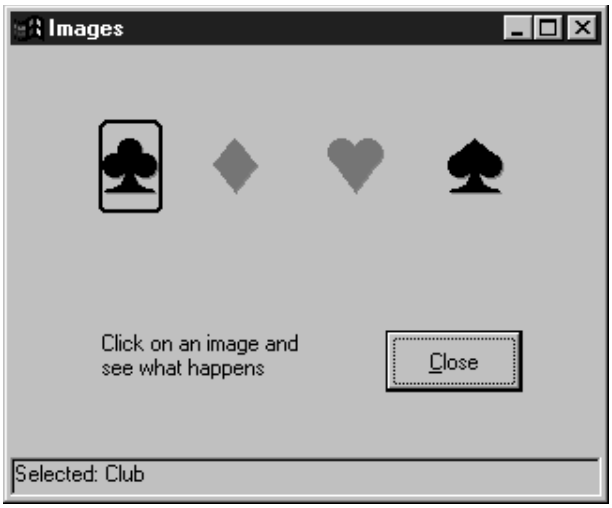
Kontrola linije je slična kontroli lika, ali se može koristiti samo za crtanje ravnih linija.

Za više informacija Za dodatne informacije o kontrolama lika i linije pogledajte 7. poglavlje "Korištenje standardnih kontrola Visual Basica".

Aplikacija Images

Forma prikazana na slici 3.18 koristi četiri kontrole slike, kontrolu lika, okvir za sliku i naredbeni gumb. Kad korisnik odabere neki od kartaških simbola, kontrola lika će označiti taj simbol, a u okviru za sliku će se pojaviti opis. Za radnu verziju ovog primjera, pogledajte Images.frm u aplikaciji Controls.vbp.

Slika 3.18 Primjer kontrola slike i lika



Sljedeća tablica daje vrijednosti svojstava za objekte ove aplikacije.

objekt	svojstvo	vrijednost
okvir za sliku	Name Align	picStatus Bottom
prva kontrola slike	Name Picture	imgClub Spade.ico
druga kontrola slike	Name Picture	imgDiamond Diamond.ico
treća kontrola slike	Name Picture	imgHeart Heart.ico
četvrta kontrola slike	Name Picture	imgSpade Spade.ico

objekt	svojstvo	vrijednost
kontrola lika	Name Shape Border Width Height Width	shpCard 4-Rounded Rectangle 2 735 495
naredbeni gumb	Name Caption	cmdClose &Close

Događaji u aplikaciji Images

Aplikacija Images odgovara na događaje kako slijedi:

- Događaj Click u svakoj od kontrola slike vrijednost svog svojstva Left dodjeljuje svojstvu Left kontrole lika, mičući kontrolu lika iznad slike.
- Poziva se postupak Cls okvira za sliku, koji briše trenutni sadržaj statusne trake.
- Poziva se postupak Print okvira za sliku, koji ispisuje novi tekst u statusnoj traci.

Programski kod u događaju Click kontrole slike izgleda ovako:

```
Private Sub imgHeart_Click()
    shpCard.Left = imgClub.Left
    picStatus.Cls
    picStatus.Print "Selected: Club"
    shpCard.Visible = True
End Sub
```

Uočite da prva linija događaja Click dodjeljuje vrijednost (svojstva Left kontrole slike) svojstvu Left kontrole lika koristeći operator =. Iduće dvije linije pozivaju postupke, tako da operator nije potreban. U trećoj liniji, vrijednost ("Selected: Club") je argument postupka Print.

Postoji još jedna linija koda u aplikaciji koja je zanimljiva; nalazi se u događaju Form Load.

```
shpCard.Visible = False
```

Postavljanjem svojstva Visible kontrole lika na False, kontrola lika će biti nevidljiva sve dok se ne klikne na neku od slika. Svojstvo Visible postavlja se na True kao posljednji korak u događaju Click kontrole slike.

Za više informacija Za dodatne informacije o svojstvima, postupcima i događajima pogledajte 5. poglavlje "Osnove programiranja".

Dodatne kontrole

U alatni okvir Visual Basica uključeno je nekoliko drugih standardnih kontrola. Neke kontrole su korisne za rad s velikim brojem podataka koji se nalaze u vanjskim bazama. Druge kontrole mogu biti iskorištene za pristup datotečnom sustavu Windowsa. Ostale kontrole ne mogu se svrstati u određenu vrstu, ali su unatoč tome korisne.

U aplikaciji Visual Basica također možete koristiti ActiveX kontrole, prethodno nazivane korisničke ili OLE kontrole, na isti način kao što koristite i standardne kontrole. Verzije Visual Basica Professional i Enterprise uključuju nekoliko ActiveX kontrola kao i mogućnost izgradnje vlastitih kontrola. Dodatne ActiveX kontrole za bilo koju moguću svrhu dostupne su kod brojnih prodavača.

Za više informacija Za dodatne informacije o korištenju ActiveX kontrola pogledajte 4. poglavlje “Upravljanje projektima”.

Kontrole pristupa podacima

U današnjem poslu, većina podataka spremljena je u jednoj ili više središnjih baza podataka. Visual Basic uključuje nekoliko kontrola za pristup podacima koje mogu pristupiti većini popularnih baza podataka, uključujući Microsoft Access i SQL poslužitelj.

- Kontrola ADO Data koristi se za povezivanje s bazom podataka. Shvatite je kao cjevovod između baze podataka i drugih kontrola na vašoj formi. Svojstva, postupci i događaji ove kontrole omogućuju vam kretanje i obradu vanjskih podataka iz vaše aplikacije.
- Kontrola DataList slična je kontroli okvira s popisom. Kad se koristi u suradnji s kontrolom ADO Data, može biti automatski popunjena popisom podataka iz polja vanjske baze podataka.
- Kontrola DataCombo sliči kombinaciji kontrole DataList i okvira s tekstom. Odabrani tekst u okviru s tekstom može biti mijenjan, a promjene će se pojaviti u bazi podataka na popisu.
- Kontrola DataGrid prikazuje podatke u matrici ili tablici. Kad se koristi u suradnji s kontrolom ADO Data, predstavlja podatke iz vanjske baze podataka u više polja koji se mogu mijenjati.
- Kontrola Microsoft Hierarchical FlexGrid je jedinstvena kontrola za prikazivanje višestrukih pregleda podataka. Shvatite je kao kombinaciju mreže i stabla. Tijekom rada aplikacije, korisnik može promijeniti raspored redova i kolona za pružanje drugačijih pogleda na podatke.

Za više informacija Za dodatne informacije o kontrolama pristupa podacima, pogledajte 7. poglavlje “Korištenje standardnih kontrola Visual Basica”. Za više informacija o radu s vanjskim podacima, potražite vodič *Visual Basic Data Access Guide*.

Kontrole datotečnog sustava

Visual Basic uključuje tri kontrole za dodavanje mogućnosti baratanja datotekama u vašu aplikaciju. Ove kontrole obično se koriste zajedno za pružanje pregleda pogona, direktorija i datoteka; imaju posebna svojstva i događaje koji ih povezuju.

- Kontrola okvira s popisom pogonskih uređaja (DriveListBox) izgleda kao kombinirani okvir. Korisnik može izabrati pogon sa spuštajućeg popisa pogona.
- Kontrola okvira s popisom direktorija (DirListBox) slična je okviru s popisom, ali ima ugrađenu mogućnost prikazivanja popisa direktorija na trenutno odabranom pogonu.
- Kontrola okvira s popisom datoteka (FileListBox) također izgleda kao okvir s popisom i sadrži popis datoteka u odabranom direktoriju.

  Napomena Ove kontrole postoje prvenstveno zbog sukladnosti s aplikacijama kreiranim u ranijim verzijama Visual Basica. Kontrola općenitog dijaloškog okvira pruža jednostavniji postupak pristupa datotekama. Za više informacija o kontroli općenitog dijaloškog okvira pogledate sljedeći dio “Raznovrsne kontrole”.

Raznovrsne kontrole

U Visual Basic je uključeno nekoliko ostalih standardnih kontrola. Svaka od njih ima jedinstvenu svrhu.

- Kontrola mjerača vremena (timer control) može biti iskorištena za stvaranje događaja u vašoj aplikaciji u određeno vrijeme. To je korisno kod izvođenja programskog koda bez potrebe za akcijom korisnika.
- Kontrola OLE spremnika (OLE container control) je jednostavan način dodavanja mogućnosti vašoj aplikaciji kao što su povezivanje i umetanje. Kroz kontrolu OLE spremnika, pružate pristup funkcionalnostima bilo koje aplikacije koja podržava OLE (object linking and embeding, povezivanje i umetanje objekata) kao što su Microsoft Excel, Word i puno drugih.
- Kontrola općeg dijaloškog okvira (common dialog control) dodaje u vašu aplikaciju dijaloške okvire za odabir datoteka, boja, pisama i funkcija ispisa.

Za više informacija Za dodatne informacije o bilo kojoj standardnoj kontroli, pogledajte 7. poglavlje “Korištenje standardnih kontrola Visual Basica”.

Razumijevanje fokusa

Fokus je mogućnost prijema informacije od korisnika putem miša ili tipkovnice. Kad objekt ima fokus, može primiti podatke od korisnika. U sučelju Microsoft Windowsa, u isto vrijeme može raditi više aplikacija, ali samo aplikacija s fokusom ima aktivnu naslovnu traku i može primati podatke od korisnika. Na formi Visual Basica s nekoliko okvira s tekstom, samo okvir s tekstom koji ima fokus će prikazati tekst upisan upotrebom tipkovnice.

Događaji GotFocus i LostFocus pojavljuju se kad objekt dobije ili izgubi fokus. Ove događaje podržavaju forme i većina kontrola.

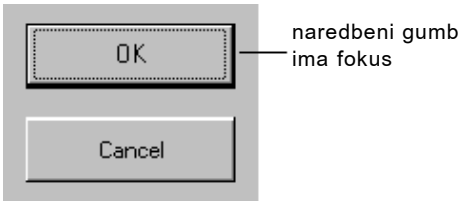
događaj	opis
GotFocus	Pojavljuje se kad objekt dobije fokus.
LostFocus	Pojavljuje se kad objekt izgubi fokus. Potprogram događaja LostFocuss u pravilu koristi za utvrđivanje i potvrdu novih podataka, ili za zamjenu ili promjenu uvjeta koji su određeni u potprogramu GotFocus istog objekta.

Objekt može dobiti fokus na sljedeće načine:

- Odabirom objekta tijekom rada aplikacije.
- Korištenjem pristupnog znaka za odabir objekta tijekom rada aplikacije.
- Korištenjem postupka SetFocus u programskom kodu.

Možete vidjeti imaju li objekti fokus. Na primjer, kad naredbeni gumbi imaju fokus pojavljuju se s označenim rubom oko natpisa (pogledajte sliku 3.19).

Slika 3.19 Naredbeni gumb s vidljivim fokusom



Objekt može dobiti fokus samo ako su svojstva Enabled i Visible postavljena na True. Svojstvo Enabled dopušta objektu da odgovara na događaje uzrokovane akcijom korisnika kao što su upotreba tipkovnice ili miša. Svojstvo Visible određuje hoće li objekt biti vidljiv na ekranu.

Napomena Forma može imati fokus samo ako ne sadrži ni jednu kontrolu koja može dobiti fokus.

Događaj Validate u kontrolama

Kontrole imaju i događaj Validate, koji se pojavljuje prije nego što kontrola izgubi fokus. Unatoč tome, ovaj događaj se pojavljuje samo ako je svojstvo CausesValidation kontrole koja će dobiti fokus postavljeno na True. U većini slučajeva, budući da se događaj Validate pojavljuje prije gubljenja fokusa, to je prikladniji način od korištenja događaja LostFocus za provjeru valjanosti gubitka fokusa. Za više informacija, pogledajte “Provjera valjanosti podataka u kontroli ograničavanjem fokusa” u 7. poglavlju “Korištenje standardnih kontrola Visual Basica”.

Kontrole koje ne mogu imati fokus

Neke kontrole, kao što su kontrole lake kategorije, ne mogu dobiti fokus. Kontrole lake kategorije uključuju:

- Kontrolu okvira (frame control)
- Kontrolu slike (image control)
- Kontrolu natpisa (label control)
- Kontrolu linije (line control)
- Kontrolu lika (shape control)

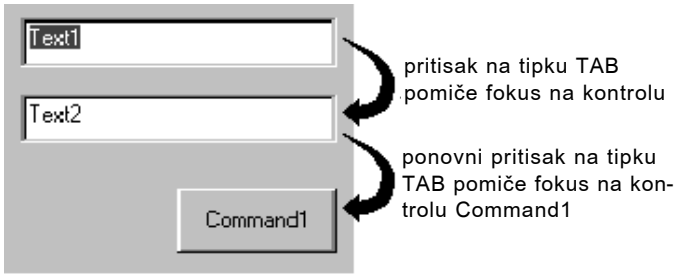
Kontrole koje su nevidljive tijekom rada aplikacije, kao kontrola mjerača vremena (timer control), također ne mogu dobiti fokus.

Postavljanje tabulatornog reda

Tabulatorni red (*tab order*) je red po kojem se korisnik može kretati od jedne do druge kontrole korištenjem tipke TAB. Svaka forma ima svoj tabulatorni red. U pravilu, tabulatorni red odgovara redoslijedu kreiranja kontrola.

Na primjer, pretpostavimo da ste kreirali dva okvira s tekstom, Text1 i Text2, te zatim naredbeni gumb, Command1. Kad se aplikacija pokrene, kontrola Text1 imat će fokus. Pritisak na tipku TAB pomaknut će fokus po kontrolama redoslijedom kojim su kreirane, kako je prikazano na slici 3.20.

Slika 3.20 Primjer tabulatornog reda



Za promjenu tabulatornog reda kontrole, podesite svojstvo TabIndex. Svojstvo TabIndex kontrole određuje gdje je postavljena u tabulatornom redu. U pravilu, prva kreirana kontrola ima vrijednost svojstva TabIndex 0, druga 1 i tako dalje. Kad promijenite redoslijed kontrole u tabulatornom redu, Visual Basic automatski mijenja mjesta ostalih kontrola u tabulatornom redu tako da odražava ubacivanja i brisanja. Na primjer, ako kontrolu Command1 postavite kao prvu u tabulatornom redu, svojstva TabIndex ostalih kontrola će automatski biti povećana za jedan, kao što je prikazano u sljedećoj tablici.

Kontrola	TabIndex prije promjene	TabIndex poslije promjene
Text1	0	1
Text2	1	2
Command1	2	0

Najveća vrijednost svojstva TabIndex je uvijek za jedan manja od broja kontrola u tabulatornom redu (jer brojanje počinje od 0). Čak i ako svojstvu TabIndex dodijelite vrijednost veću od broja kontrola, Visual Basic tu vrijednost pretvara u broj za jedan manji od broja kontrola.

Napomena Kontrole koje ne mogu dobiti fokus, kao i kontrole koje su onemogućene ili nevidljive, nemaju svojstvo TabIndex i nisu uključene u tabulatorni red. Kad korisnik pritišće tipku TAB, takve kontrole se preskaču.

Uklanjanje kontrole iz tabulatornog reda

Uobičajeno, pritisak tipke TAB tijekom rada aplikacije odabire iduću kontrolu po tabulatornom redu. Kontrolu iz tabulatornog reda možete udaljiti postavljanjem njezinog svojstva TabStop na False (0).

Kontrola čije je svojstvo TabStop postavljeno na False i dalje zadržava svoje mjesto u tabulatornom redu, iako će biti preskočena kruženjem po kontrolama tipkom TAB.

Napomena Grupa gumbâ izbora ima jedinstven odabir tipkom TAB. Potvrđeni gumb (onaj čije je svojstvo Value postavljeno na True) automatski ima svojstvo TabStop postavljeno na True, a ostali gumbi u grupi imaju svojstvo TabStop postavljeno na False.

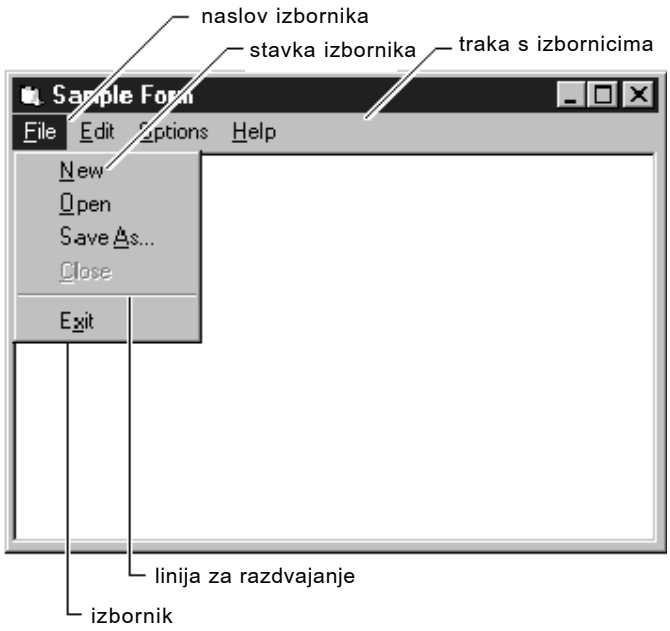
Osnove izbornika

Želite li da vaša aplikacija omogući niz naredbi korisniku, izbornici nude prikladan i dosljedan način grupiranja naredbi te jednostavan način pristupa korisnika.

Slika 3.21 pokazuje dijelove sučelja s izbornicima na formi bez imena.

Traka s izbornicima (*menu bar*) pojavljuje se odmah ispod *naslovne trake* (*title bar*) forme i sadržava jedan ili više *naslova izbornika* (*menu titles*). Kad kliknete na naslov izbornika (na primjer File) spušta se izbornik koji sadrži niz stavki. Stavke izbornika mogu sadržavati naredbe (na primjer New ili Exit), linije za razdvajanje (separator bars), i naslove s podizbornicima. Svaka stavka izbornika koju vidi korisnik odgovara kontroli izbornika koju određujete u editoru izbornika (menu editor, opisano kasnije u ovom poglavlju).

Slika 3.21 Dijelovi sučelja s izbornicima na formi Visual Basica



Za lakše korištenje vaše aplikacije, trebali bi grupirati stavke izbornika prema njihovim funkcijama. Na slici 3.21, na primjer, naredbe New, Open i Save As... vezane za datoteke nalaze se u izborniku File.

Neke stavke izbornika odmah pokreću akcije; na primjer, stavka Exit izbornika File zatvara aplikaciju. Ostale stavke izbornika prikazuju *dijaloški okvir* – prozor koji traži podatak od korisnika potreban kako bi aplikacija mogla obaviti akciju. Takve stavke izbornika trebale bi biti ispisane s točkicama u nastavku (...). Na primjer, kad odaberete stavku Save As... u izborniku File, pojavit će se dijaloški okvir Save File As.

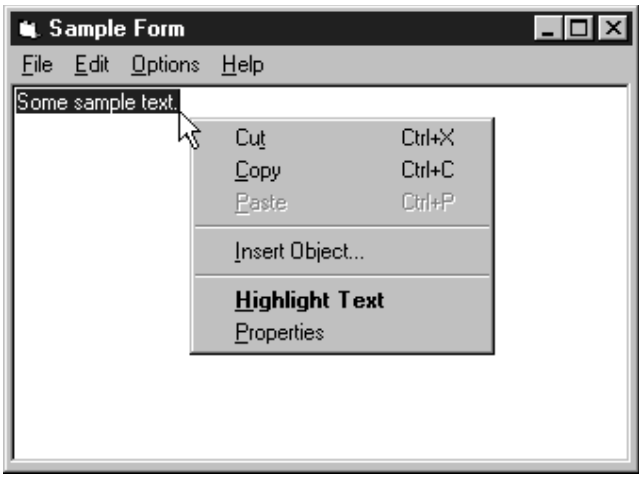
Kontrola izbornika je objekt; kao i ostali objekti ima svojstva uz pomoć kojih se mogu odrediti njezin izgled i ponašanje. Svojstva Caption, Enabled, Visible, Checked i druga, možete odrediti tijekom izrade ili rada aplikacije. Kontrole izbornika sadrže samo jedan događaj, događaj Click, koji se poziva kad je stavka izbornika odabrana mišem ili tipkovnicom.

Za više informacija Za dodatne informacije o kontrolama izbornika, pogledajte "Stvaranje izbornika editorom izbornika" u 6. poglavlju "Stvaranje korisničkog sučelja".

Izbornici prečica

Izbornik prečica (*pop–up menu*) je plivajući izbornik koji se prikazuje iznad forme, neovisno o traci s izbornicima, kao što je prikazano na slici 3.22. Stavke ispisane u pomoćnom izborniku ovise o položaju pokazivača kad je pritisnuta desna tipka miša; zbog toga se izbornici prečica često nazivaju i *kontekstni izbornici* (*context menus*). (U Windowsima 95/98, kontekstne izbornike možete aktivirati klikom desnom tipkom miša.) Pomoćne izbornike trebate koristiti kao pružanje efikasnog postupka za pristup uobičajenim, prikladnim naredbama. Na primjer, ako kliknete desnom tipkom miša na okvir s tekstom, pojavit će se kontekstni izbornik, kao što je prikazano na slici 3.22.

Slika 3.22 Izbornik prečica



Svaki izbornik koji ima barem jednu stavku može biti prikazan tijekom rada aplikacije kao izbornik prečica. Za prikaz izbornika, iskoristite postupak PopupMenu.

Za više informacija Za dodatne informacije o kreiranju pomoćnih izbornika pogledajte "Stvaranje izbornika editorom izbornika" u 6. poglavlju "Stvaranje korisničkog sučelja".

Korištenje editora izbornika

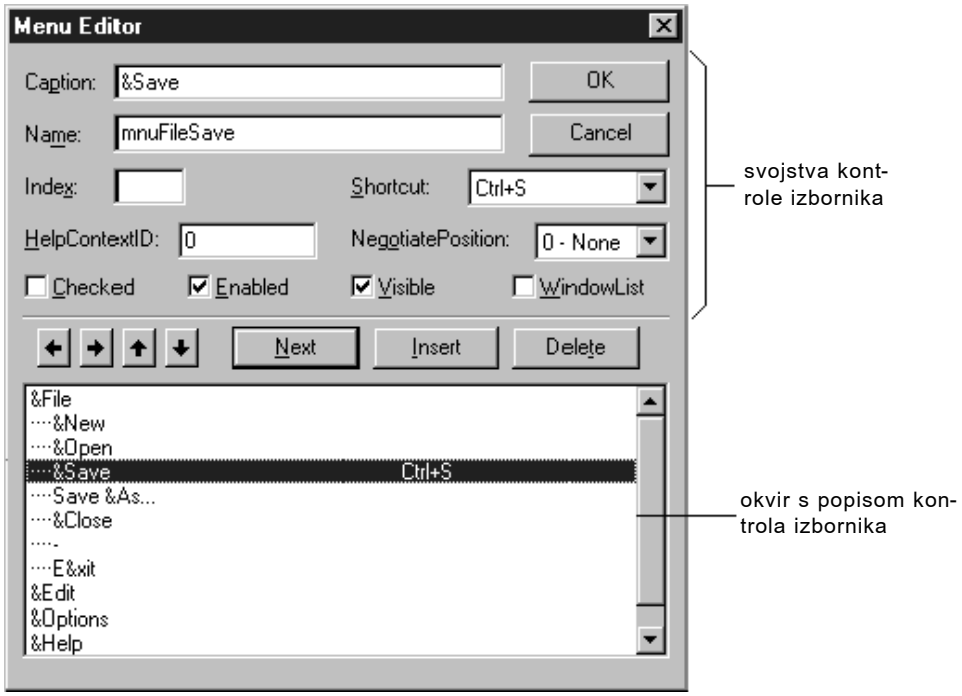
Uz pomoć editora izbornika možete dodati nove stavke u postojeće izbornike, zamijeniti postojeće naredbe vlastitim, kreirati nove izbornike i trake s izbornicima, te promijeniti i obrisati postojeće izbornike i trake izbornika. Glavna prednost editora izbornika je njegova jednostavnost upotrebe. Možete prilagoditi izbornike na potpuno interaktivan način koji uključuje vrlo malo programiranja.

Kako prikazati editor izbornika

- U izborniku **Tools** odaberite stavku **Menu Editor**.

Ovaj postupak će otvoriti editor izbornika, prikazan na slici 3.23.

Slika 3.23 Editor izbornika



Većina svojstava kontrola izbornika može biti podešena korištenjem editora izbornika; sva svojstva izbornika su također dostupna i u prozoru sa svojstvima. Uobičajeno ćete kreirati izbornik u editoru izbornika; međutim, iskoristite prozor sa svojstvima za brzu promjenu određenog svojstva.

Za više informacija Za dodatne informacije o kreiranju izbornika i radu s editorom izbornika, pogledajte "Stvaranje izbornika editorom izbornika" u 6. poglavlju "Stvaranje korisničkog sučelja".

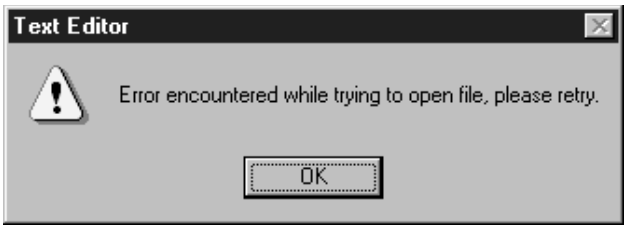
Upit korisniku dijaloškim okvirom

U aplikacijama temeljenim na Windows okruženju, dijaloški okviri se koriste za postavljanje upita korisniku o podacima potrebnim kako bi aplikacija nastavila izvođenje ili za prikaz obavijesti korisniku. Dijaloški okviri su specijalizirani oblik objekta forme koji može biti stvoren na tri načina:

- *Predefinirani* dijaloški okviri mogu biti stvoreni programskim kodom koristeći funkcije MsgBox ili InputBox.
- *Prilagođeni* dijaloški okviri mogu biti stvoreni korištenjem standardne forme ili prilagođavanjem postojećeg dijaloškog okvira.
- *Standardni* dijaloški okviri, kao što su Print i File Open, mogu biti stvoreni korištenjem kontrole općenitog dijaloškog okvira.

Slika 3.24 prikazuje primjer predefiniranog dijaloškog okvira kreiranog korištenjem funkcije MsgBox.

Slika 3.24 Predefinirani dijaloški okvir



Ovaj dijaloški okvir se prikazuje kad u programskom kodu pozovete funkciju MsgBox. Programski kod za prikaz dijaloškog okvira prikazanog na slici 3.24 izgleda ovako:

```
MsgBox "Error encountered while trying to open file," & vbCrLf & "please retry.",
➥ vbExclamation "Text Editor"
```

Funkciju MsgBox opskrbljujete s tri informacije, ili argumenta: tekstom poruke, konstantom (numeričke vrijednosti) za određivanje stila dijaloškog okvira, te naslovom. Stilovi se određuju zahvaljujući raznim kombinacijama gumbâ i ikona što olakšava kreiranje dijaloških okvira.

Budući da većina dijaloških okvira zahtijeva akciju korisnika, obično su prikazani kao *obavezni* dijaloški okviri. Obavezni dijaloški okvir mora biti zatvoren (skriven ili obrisan) prije nego što možete nastaviti raditi s ostatkom aplikacije. Na primjer, dijaloški okvir je obavezan ako od vas traži klik na OK ili Cancel prije nego što se možete prebaciti na drugu formu ili dijaloški okvir.

Neobavezni dijaloški okviri omogućuju vam promjenu fokusa između dijaloškog okvira i druge forme bez potrebe za zatvaranjem dijaloškog okvira. Možete nastaviti s radom bilo gdje u trenutnoj aplikaciji iako je dijaloški okvir otvoren. Neobavezni dijaloški okviri su rijetki; uglavnom ćete prikazati dijaloški okvir jer je potreban odgovor prije nastavka rada aplikacije. Dijaloški okvir koji dobivate odabirom stavke Find u izborniku Edit Visual Basica je primjer neobaveznog dijaloškog okvira. Upotrijebite neobavezne dijaloške okvire za prikaz često korištenih naredbi ili informacija.

Za više informacija Za dodatne informacije o stvaranju dijaloških okvira, pogledajte 6. poglavlje “Stvaranje korisničkog sučelja”.

4. POGLAVLJE

Upravljanje projektima

Tijekom kreiranja aplikacije Visual Basicom, radite s projektima. *Projekt* je kolekcija datoteka koje koristite za izgradnju aplikacije. Ovo poglavlje opisuje kako izgraditi i upravljati projektima.

Kad stvarate aplikaciju, obično najprije stvarate novu formu; također možete ponovno upotrijebiti ili promijeniti forme koje su stvorene za prethodne projekte. Isto vrijedi za ostale module ili datoteke koje možete uključiti u svoj projekt. ActiveX kontrole i objekti iz drugih aplikacija također mogu biti dijeljeni među projektima.

Nakon sastavljanja svih dijelova u projektu i pisanja programskog koda, potrebno je prevesti (compile) projekt kako bi stvorili izvršnu datoteku.

Sadržaj

- Rad s projektima
- Sastav projekta u Visual Basicu
- Stvaranje, otvaranje i spremanje projekata
- Dodavanje, micanje i spremanje datoteka
- Dodavanje kontrola u projekt
- Izrada i pokretanje izvršne datoteke
- Postavljanje mogućnosti projekta
- Korištenje čarobnjaka i dodataka

Rad s projektima

Tijekom razvoja aplikacije, radite s projektom kako bi upravljali različitim datotekama koje čine aplikaciju. Projekt se sastoji od:

- Jedne projektne datoteke koja čuva tragove svih dijelova (.vbp).
- Jedne datoteke za svaku formu (.frm).
- Jedne binarne podatkovne datoteke za svaku formu koja sadrži podatke o svojstvima kontrola na formi (.frx). Ove datoteke se ne mogu mijenjati i automatski se stvaraju za svaku .frm datoteku koja sadrži binarna svojstva, kao što su Picture ili Icon.

- Prema potrebi, po jedna datoteka za svaki modul klase (.cls).
- Prema potrebi, po jedna datoteka za svaki standardni modul (.bas).
- Prema potrebi, jedna ili više datoteka koje sadrže ActiveX kontrole (.ocx).
- Prema potrebi, jedna datoteka s izvorima (.res).

Projektna datoteka je jednostavno popis svih datoteka i objekata pridruženih projektu, kao i informacije o mogućnostima sučelja koje ste odredili. Ove informacije se ažuriraju svaki put kad snimate projekt. Sve datoteke i objekti također mogu biti dostupni i drugim projektima.

Kad ste završili s izradom svih datoteka projekta, potrebno je pretvoriti projekt u izvršnu datoteku (.exe): odaberite naredbu Make *projekt.exe* u izborniku File.

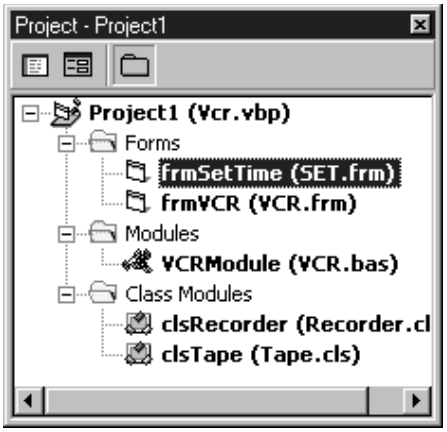
> Napomena S Professional i Enterprise verzijama Visual Basica, možete kreirati i druge vrste izvršnih datoteka kao što su .ocx i .dll datoteke. Smjernice u ovom poglavlju pretpostavljaju standardni .exe projekt. Za dodatne informacije o ostalim vrstama projekata pogledajte vodič *Microsoft Visual Basic 6.0 Component Tools Guide* iz biblioteke *Microsoft Visual Basic 6.0 Reference Library*.

Za više informacija Za dodatne detalje o stvaranju izvršnih datoteka, pogledajte "Izrada i pokretanje izvršne datoteke" kasnije u ovom poglavlju. Za informacije o binarnim i projektnim datotekama, pogledajte Dodatak A "Specifikacije, ograničenja i vrste datoteka Visual Basica".

Projektni prozor (project explorer)

Kako stvarate, dodajete ili brišete promjenljive datoteke u projektu, Visual Basic odražava vaše promjene u projektnom prozoru, koji sadrži trenutnu listu datoteka u projektu. Projektni prozor na slici 4.1 pokazuje neke od tipova datoteka koje možete uključiti u projekt Visual Basica.

Slika 4.1 Projektni prozor



Datoteka projekta

Svaki put kad snimite projekt, Visual Basic ažurira datoteku projekta (.vbp). Datoteka projekta sadrži listu datoteka koja je jednaka onoj prikazanoj u projektnom prozoru, kao i smjernice prema ActiveX kontrolama i umetnutim objektima koji se koriste u projektu.

Postojeći projekt možete otvoriti dvoklikom na njegovu ikonu, odabirom naredbe Open u izborniku File, ili povlačenjem datoteke i ispuštanjem na projektni prozor.

Za više informacija Poseban oblik podataka spremljenih u datoteci .vbp objašnjen je u Dodatku A "Specifikacije, ograničenja i vrste datoteka Visual Basica".

Sastav projekta u Visual Basicu

Sljedeći odlomci opisuju različite tipove datoteka i objekata koje možete uključiti u projekt.

Moduli forme

Moduli forme (datoteke s nastavkom .frm) mogu sadržavati tekstovne opise forme i njezinih kontrola, uključujući vrijednosti svojstava. Također mogu sadržavati i deklaracije konstanti, varijabli i vanjskih potprograma; potprograma događaja; i općih potprograma.

Za više informacija Više o kreiranju formi pisano je u 2. poglavlju "Razvijanje aplikacije u Visual Basicu" i 6. poglavlju "Stvaranje korisničkog sučelja". Za informacije o obliku i sadržaju datoteke forme, pogledajte Dodatak A "Specifikacije, ograničenja i vrste datoteka Visual Basica".

Moduli klase

Moduli klase (datoteke s nastavkom .cls) slični su modulima forme, osim što nemaju vidljivo korisničko sučelje. Module klase možete iskoristiti za stvaranje vlastitih objekata, uključujući programski kod za postupke i svojstva.

Za više informacija Za informacije o pisanju programskog koda u modulima klase, pogledajte "Stvaranje vlastitih klasa" u 9. poglavlju "Programiranje objektima".

Standardni moduli

Standardni moduli (datoteke s nastavkom .bas) mogu sadržavati javne ili modularne deklaracije tipova, konstanti, varijabli, vanjskih i javnih potprograma.

Za više informacija Za informacije o korištenju modula pogledajte 5. poglavlje "Osnove programiranja" i 9. poglavlje "Programiranje objektima".

Datoteke s izvorima

Datoteke s izvorima (datoteke s nastavkom .res) sadržavaju bitmapirane slike, tekstovne nizove, te ostale podatke koje možete mijenjati bez promjena programskog koda. Na primjer, ako želite lokalizirati svoju aplikaciju na drugi jezik, možete držati sav tekst i slike korisničkog sučelja u datoteci s izvorima, pa ih promijeniti prema potrebama umjesto mijenjanja cijele aplikacije. Projekt može sadržavati samo jednu datoteku s izvorima.

Za više informacija Za više informacija o korištenju datoteka s izvorima, pogledajte "Korištenje datoteka s izvorima za lokalizaciju" kasnije u ovom poglavlju, te 16. poglavlje "Međunarodna izdanja".

ActiveX dokumenti

ActiveX dokumenti (.dob) slični su formama, ali ih se može prikazati u Internet pretraživačima kao što je Internet Explorer. Professional i Enterprise verzije Visual Basica imaju mogućnost stvaranja ActiveX dokumenata.

Za više informacija Za dodatne informacije o ActiveX dokumentima, pogledajte "Stvaranje ActiveX sastavnih dijelova" u vodiču *Microsoft Visual Basic 6.0 Component Tools Guide*.

Korisničke kontrole i moduli sa svojstvima

Korisničke kontrole (.ctl) i moduli sa svojstvima (.pag) su također slični formama, ali se koriste za stvaranje ActiveX kontrola i pripadajućih stranica sa svojstvima za prikazivanje svojstava tijekom izrade aplikacije. Professional i Enterprise verzije Visual Basica imaju mogućnost stvaranja ActiveX kontrola.

Za više informacija Za dodatne informacije o stvaranju ActiveX kontrola, pogledajte "Stvaranje ActiveX kontrola" u "Stvaranju ActiveX sastavnih dijelova" iz vodiča *Microsoft Visual Basic 6.0 Component Tools Guide*, dijela biblioteke *Microsoft Visual Basic 6.0 Reference Library*.

Sastavni dijelovi

Nekoliko različitih vrsta sastavnih dijelova može biti dodano projektu kao dodatak formama i modulima.

ActiveX kontrole

ActiveX kontrole (datoteke s nastavkom .ocx) su alternativne kontrole koje mogu biti dodane u alatni okvir i korištene na formama. Kad instalirate Visual Basic, datoteke koje sadrže kontrole uključene u Visual Basic kopiraju se opći direktorij (poddirektorij \Windows\System kod Windowsa 95/98). Dodatne ActiveX kontrole su dostupne iz više izvora. Možete također stvarati i vlastite kontrole koristeći Professional i Enterprise verzije Visual Basica.

Za više informacija Za dodatne informacije o korištenju uključenih ActiveX kontrola, pogledajte "Korištenje ActiveX kontrola" u vodiču *Microsoft Visual Basic 6.0 Component Tools Guide*.

Dodatni objekti

Dodatni objekti, kao objekt radnog lista Microsoft Excela, su sastavni dijelovi koje možete upotrijebiti pri izradi blokova za stvaranje ugrađenih rješenja. *Ugrađeno rješenje* može sadržavati podatke različitih formata, kao što su proračunske tablice, slike i tekst, gdje su svi ti podaci kreirani u različitim aplikacijama.

Za više informacija Za dodatne informacije o korištenju objekata iz drugih aplikacija, pogledajte 10. poglavlje "Programiranje sastavnim dijelovima".

Pokazivači

Možete također dodati pokazivače prema vanjskim ActiveX dijelovima koje može koristiti vaša aplikacija. Pokazivače dodajete korištenjem dijaloškog okvira References, kojem pristupate odabirom stavke References u izborniku Project.

Za više informacija Za dodatne informacije o pokazivačima, pogledajte "Korištenje objekata drugih aplikacija" kasnije u ovom poglavlju.

ActiveX kreatori

ActiveX kreatori su alati za oblikovanje klasa iz kojih mogu biti kreirani objekti. Sučelje kreatora za forme je standardan kreator. Dodatni kreatori mogu biti dostupni iz drugih izvora.

Za više informacija Za dodatne informacije o ActiveX kreatorima, pogledajte odlomak "ActiveX kreatori" u 9. poglavlju "Programiranje objektima".

Standardne kontrole

Standardne kontrole isporučuju se sa Visual Basicom. Standardne kontrole, kao što su naredbeni gumb ili kontrola okvira, su uvijek uključene u alatni okvir, za razliku od ActiveX kontrola i dodatnih objekata, koji mogu biti maknuti ili dodani u alatni okvir.

Za više informacija Za dodatne informacije o standardnim kontrolama, pogledajte 3. poglavlje "Forme, kontrole i izbornici" i 7. poglavlje "Korištenje standardnih kontrola Visual Basica".

Stvaranje, otvaranje i spremanje projekata

Četiri naredbe u izborniku File omogućuju vam kreiranje, otvaranje i snimanje projekata.

naredba izbornika	opis
New Project	Zatvara trenutni projekt, pitajući vas želite li snimiti datoteke koje su promijenjene. Možete odabrati vrstu projekta iz dijaloškog okvira New Project. Visual Basic zatim kreira novi projekt s jednom novom datotekom.
Open Project	Zatvara trenutni projekt, pitajući vas želite li spremiti promjene. Visual Basic zatim otvara postojeći projekt, uključujući forme, module i ActiveX kontrole uključene u datoteci projekta (.vbp).
Save Project	Ažurira datoteku trenutnog projekta i svih njegovih formi, standardnih modula i modula klase.
Save Project As	Ažurira datoteku trenutnog projekta, snimajući datoteku projekta s imenom koje odredite. Visual Basic vas također pita i za snimanje svih formi i modula koji su bili promijenjeni.

Moguće je dijeljenje objekata među projektima. Jedna datoteka, kao što je forma, može biti dio više projekata. Zapamtite da će promjene napravljene na formi ili modulu u jednom projektu biti prenesene u sve projekte koji koriste taj modul.

Za više informacija Za dodatne informacije o dijeljenju datoteka pogledajte "Dodavanje, micanje i spremanje datoteka" kasnije u ovom poglavlju.

Rad s više projekata

U Professional i Enterprise verzijama Visual Basica, moguće je istovremeno imati otvoreno i više projekata. Ova mogućnost korisna je kod izgradnje i isprobavanja rješenja koja uključuju korisničke kontrole ili druge sastavne dijelove. Kad je otvoreno više projekata, naslov projektnog prozora će se promijeniti u naziv projektne grupe i bit će prikazani sastavni dijelovi svih projekata.

Kako dodati projekt u projektnu grupu

1. U izborniku **File** odaberite **Add Project**.

   Prikazat će se dijaloški okvir **Add Project**.

2. Odaberite postojeći projekt ili novu vrstu projekta, i odaberite **Open**.

Kako maknuti projekt iz projektne grupe

1. Odaberite projekt ili dio projekta u projektnom prozoru.
2. U izborniku **File** odaberite stavku **Remove Project**.

Za više informacija Da biste više naučili o radu s višestrukim projektima, pogledajte "Stvaranje ActiveX sastavnih dijelova" u vodiču *Microsoft Visual Basic 6.0 Component Tools Guide*.

Dodavanje, micanje i spremanje datoteka

Rad sa datotekama unutar projekta sličan je radu sa samim projektima.

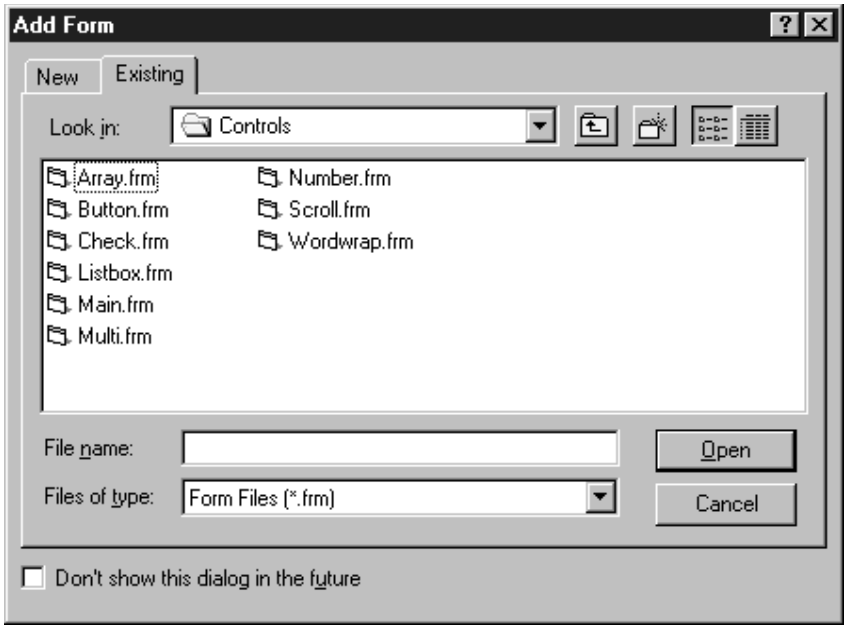
Kako projektu dodati datoteku

1. U izborniku **Project**, odaberite stavku **Add** *vrstadatoteke* (gdje je *vrstadatoteke* vrsta datoteke).

   Prikazat će se dijaloški okvir **Add** *vrstadatoteke* (slika 4.2).

2. Odaberite postojeću datoteku ili novu vrstu datoteke, i odaberite **Open**.

Slika 4.2 Dijaloški okvir Add Form



Kada projektu dodate datoteku, zapravo ste u projekt uključili pokazivač prema toj datoteci; niste dodali ili ulijepili datoteku. Zbog toga, ako napravite promjene u datoteci i snimite je, te promjene će utjecati na svaki projekt koji uključuje tu datoteku. Za promjenu datoteke bez utjecaja na druge projekte, odaberite datoteku u projektnom prozoru, odaberite Save *imedatoteke* u izborniku File, i snimite datoteku pod novim imenom.

Napomena Možete povući i ispustiti datoteke iz Windows Explorera, File Managera ili Network Neighborhood prozora u projektni prozor kako bi ih dodali u projekt. Također možete povući i ispustiti .ocx datoteke na alatni okvir za dodavanje novih kontrola.

Kako maknuti datoteku iz projekta

1. Odaberite datoteku u projektnom prozoru.
2. U izborniku **Project** odaberite stavku **Remove** *imedatoteke*.
3. Datoteka će biti uklonjena iz projekta, ali neće biti obrisana s diska.

Ako maknete datoteku iz projekta, Visual Basic će ažurirati informacije o datoteci projekta kad ga budete snimali. Međutim, ako obrišete datoteku izvan Visual Basica, Visual Basic ne može ažurirati datoteku projekta; zbog toga će, kad otvorite projekt, Visual Basic prikazati poruku pogreške upozoravajući vas da nedostaje datoteka.

Kako snimiti određenu datoteku bez snimanja cijelog projekta

1. Odaberite datoteku u projektnom prozoru.
2. U izborniku **File** odaberite **Save** *imedatoteke*.

Sjedinjavanje teksta

Možete također ubaciti postojeći tekst iz drugih datoteka u jednu cjelinu unutar modula vašeg programskog koda. Ovaj postupak je koristan kod dodavanja liste konstanti ili za dodavanje dijelova programskog koda koje želite spremiti u tekstualne datoteke.

Kako ubaciti tekst u programski kod

1. U projektnom prozoru odaberite formu ili modul u koji želite ubaciti tekst.
2. Odaberite gumb **View Code** i u kodnom prozoru pomaknite kursor na mjesto gdje želite ubaciti tekst.
3. U izborniku **Edit** odaberite **Insert File**.
4. vOdaberite ime tekstualne datoteke koju želite ubaciti, i odaberite **Open**.

Napomena Ako editirate izvorne datoteke Visual Basica koristeći neki drugi editor osim Visual Basica, pripazite da ne mijenjate vrijednosti atributa VB_PredeclareId. Promjena tog atributa osobito može uzrokovati ozbiljne probleme s klasama GlobalMultiUse i GlobalSingleUse.

Općenito, ne bi trebali mijenjati atribute ručno, jer takav postupak može dovesti module u nestabilno stanje.

Dodavanje kontrola u projekt

Komplet kontrola dostupnih u alatnom okviru može se prilagoditi svakom projektu. Svaka dana kontrola mora se nalaziti u alatnom okviru prije nego što je možete dodati na formu projekta. Osnovni komplet kontrola koji se uvijek pojavljuje u alatnom okviru opisan je u 3. poglavlju “Forme, kontrole i izbornici”.

Dodavanje ActiveX kontrole u projekt

ActiveX kontrole i dodatne objekte možete pripojiti svom projektu postavljajući ih u alatni okvir.

Kako dodati kontrole u alatni okvir projekta

1. U izborniku **Project** odaberite stavku **Components**.

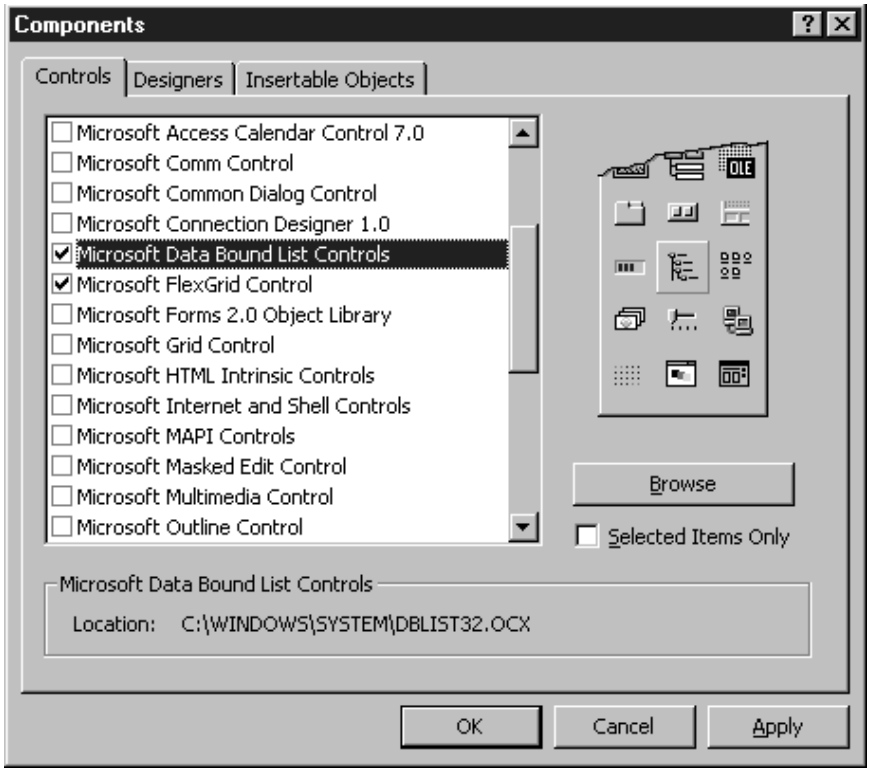
   Prikazati će se dijaloški okvir **Components**, kako je prikazano na slici 4.3. Stavke ispisane u ovom dijaloškom okviru uključuju sve registrirane ActiveX kontrole, dodatne kontrole i ActiveX kreatore.

2. Da biste dodali kontrolu (datoteke s nastavkom .ocx) ili dodatni objekt u alatni okvir, potvrdite kontrolnu kućicu lijevo od imena kontrole.

   Odaberite karticu **Controls** kako bi vidjeli kontrole s .ocx nastavkom u imenu datoteke. Za pregled dodatnih objekata, kao što je Microsoft Excel Chart, odaberite karticu **Insertable Objects**.

3. Odaberite gumb **OK** za zatvaranje dijaloškog okvira **Components**. Sve ActiveX kontrole koje ste odabrali pojavit će se u alatnom okviru.

Slika 4.3 Dijaloški okvir Components



Za dodavanje ActiveX kontrola u dijaloški okvir Components, odaberite gumb Browse, i pretražite druge direktorije kako bi pronašli datoteke s nastavkom .ocx. Kad dodate ActiveX kontrolu na listu raspoloživih kontrola, Visual Basic automatski potvrđuje kontrolnu kućicu.

Napomena Svaka ActiveX kontrola ima prateću datoteku s nastavkom .oca. Ta datoteka čuva bibliotečne informacije te ostale podatke svojstvene kontroli. Datoteke s nastavkom .oca obično su snimljene u istom direktoriju gdje su i ActiveX kontrole i stvaraju se prema potrebi (veličina datoteke i datum stvaranja mogu se mijenjati).

Micanje kontrola iz projekta

Kako maknuti kontrolu iz projekta

1. U izborniku **Project** odaberite stavku **Components**.

   Prikazat će se dijaloški okvir **Components**.

2. Odznačite kontrolnu kućicu pored svake kontrole koju želite maknuti.

   Ikone tih kontrola bit će maknute iz alatnog okvira.

   Napomena Ne možete maknuti kontrolu iz alatnog okvira u slučaju ako se ta kontrola koristi na formi u projektu.

Korištenje objekata drugih aplikacija

Možete također koristiti i objekte iz drugih aplikacija, kao što su oni uključeni u biblioteku objekata Microsoft Excela, kao kontrole u alatnom okviru ili kao programabilne objekte u programskom kodu. Za dodavanje objekata u alatni okvir pogledajte “Dodavanje kontrola u projekt”, ranije u ovom poglavlju.

Da bi objekt druge aplikacije bio pristupan vašem programskom kodu, ali ne kao kontrola, postavite pokazivač na biblioteku objekata te aplikacije.

Kako dodati pokazivač na biblioteku objekata druge aplikacije

1. U izborniku **Project** odaberite stavku **References**.

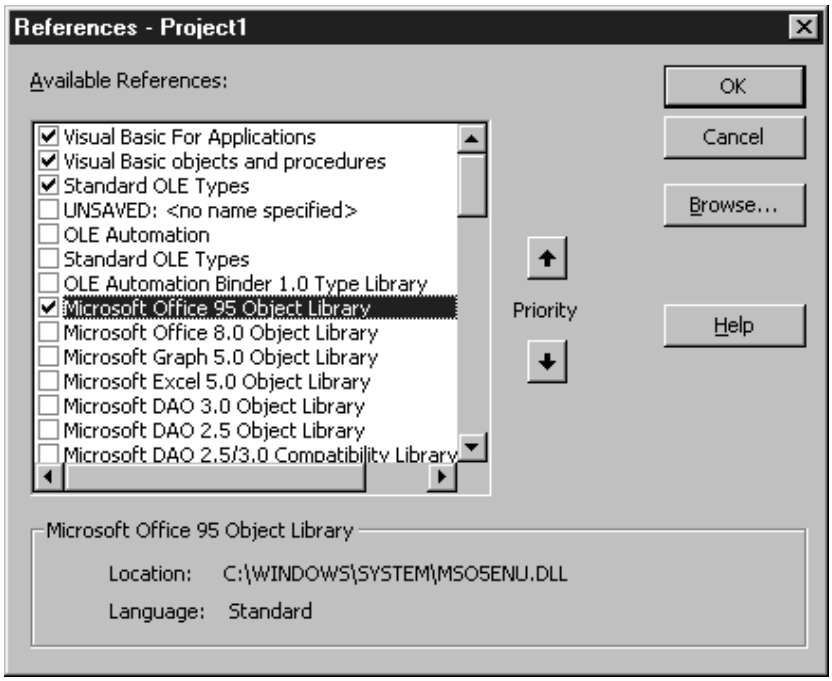
   Prikazat će se dijaloški okvir **References**, kako je prikazano na slici 4.4.

2. Potvrdite kontrolnu kućicu pored pokazivača kojeg želite dodati svom projektu.

   Za dodavanje pokazivača na aplikacije koje nisu na listi dijaloškog okvira **References**, odaberite gumb **Browse**, pa odaberite aplikaciju.

3. Odaberite gumb **OK** za dodavanje odabranog pokazivača svom projektu.

Slika 4.4 Dijaloški okvir References



Ako ne koristite neki objekt u biblioteci pokazivača, trebate odznačiti kontrolnu kućicu za taj pokazivač kako bi smanjili broj pokazivača na objekte koje Visual Basic mora riješiti, pa ćete na taj način smanjiti vrijeme koje će biti potrebno vašem projektu za prevođenje.

Jednom kad postavite pokazivače na biblioteke objekata koje želite, određeni objekt, njegove postupke i svojstva možete pronaći u pretraživaču objekata odabirom stavke Object Browser u izborniku View. U vašem programskom kodu možete iskoristiti svaki objekt koji je izlistan u pretraživaču objekata.

Za više informacija Za informacije o pretraživaču objekata pogledajte "Odgonetavanje objekata" u 9. poglavlju "Programiranje objektima".

Korištenje datoteka s izvorima

Datoteka s izvorima omogućuje vam skupljanje svih specifičnih tekstova i slika aplikacije na jednom mjestu. Ona može sadržavati deklaracije konstanti, ikone, tekst s ekrana te ostali materijal koji se može mijenjati u prilagođenim verzijama aplikacije, u naknadnim revizijama ili za određene konfiguracije.

Kako projektu dodati datoteku

1. U izborniku **Project**, odaberite stavku **Add File**.

   Prikazat će se dijaloški okvir **Add File**.

2. Odaberite postojeću datoteku s izvorima (.res) i odaberite **Open**.

Jedan projekt može imati samo jednu datoteku s izvorima; ako dodate drugu datoteku s nastavkom .res, doći će do pogreške.

Za više informacija Za dodatne informacije o sadržaju datoteke s izvorima pogledajte 16. poglavlje "Međunarodna izdanja".

Izrada i pokretanje izvršne datoteke

Izvršnu datoteku (.exe) možete napraviti iz Visual Basica koristeći sljedeći postupak.

Kako napraviti izvršnu datoteku u Visual Basicu

1. U izborniku **File** odaberite stavku **Make** *imeprojekta***.exe** gdje je *imeprojekta* ime aplikacije za projekt.
2. Upišite ime datoteke, ili pretražite direktorije i odaberite već postojeću izvršnu datoteku kako bi ju zamijenili novijom verzijom.
3. Klikom na gumb **Options** možete u dijaloškom okviru **Project Properties** odrediti i detalje vezane za verziju izvršne datoteke.
4. Ako želite promijeniti broj verzije projekta, postavite odgovarajuće **Major**, **Minor** i **Revision** vrijednosti. Odabir opcije **Auto Increment** će automatski povećavati za jedan broj **Revision** svaki put kad pokrenete naredbu **Make** *imeprojekta***.exe** za taj projekt.
5. Kako bi odredili novo ime aplikacije, u okviru **Application** upišite novo ime u okvir **Title**. Ako želite odrediti novu ikonu, odaberite jednu sa popisa.
6. U okviru **Version Information** možete upisati i napomene vezane uz aplikaciju o nizu svojstava (napomene, ime tvrtke, informacije o tvorničkom znaku i autorskim pravima i slično) odabirom predmeta u okviru s popisom i upisom podataka u okvir s tekstom.
7. Odaberite gumb **OK** za zatvaranje dijaloškog okvira **Project Properties**, pa odaberite gumb **OK** na dijaloškom okviru **Make** *imeprojekta***.exe** kako bi započelo prevođenje i povezivanje izvršne datoteke.

Izvršnu datoteku možete pokrenuti kao i bilo koju Windows aplikaciju: dvoklikom na ikonu izvršne datoteke.

Napomena Stvaranje izvršne datoteke iz naredbene linije DOS prozora može biti korisno ako želite prevesti projekt programskom naredbom. Sintaksa naredbe je sljedeća:

Vb6 /make *imeprojekta***[.vbp]** [*exeime*]

Kao *imeprojekta* upišite ime datoteke projekta. Varijablom *exeime* određujete ime izvršne datoteke koja će biti kreirana.

Uvjetno prevođenje

Uvjetno prevođenje dopušta vam djelomično prevođenje određenih dijelova aplikacije. U različitim verzijama možete uključiti specifična svojstva vaše aplikacije, kao što su filteri za promjenu datuma i valute za aplikacije na više različitih jezika.

Za više informacija Kako bi naučili više o uvjetnom prevođenju pogledajte "Korištenje uvjetnog prevođenja" u 8. poglavlju, Više o programiranju".

Postavljanje mogućnosti projekta

Visual Basic vam omogućuje prilagodbu svakog projekta postavljanjem niza svojstava. Iskoristite dijaloški okvir Project Properties, dostupan izborom naredbe Project Properties u izborniku Project. Vrijednosti svojstava čuvaju se u datoteci projekta (.vbp).

Sljedeća tabela opisuje neke od mogućnosti koje možete mijenjati.

opcija	opis
Startup Object	Prva forma koju će Visual Basic prikazati nakon pokretanja ili Sub Main ( ).
Project Name	Ime projekta, oznaka projekta u programskom kodu. Može sadržavati točke, razmake i može započeti s neslovnim karakterom. Kod imena javne klase, ime projekta i ime klase ne mogu biti dulji od ukupno 37 karaktera.
Help File	Ime datoteke s pomoći pridružene projektu.
Project Help Context ID	Sadržajni ID za određeni pojam iz pomoći koji će se prikazati kad korisnik odabere gumb "?" dok je biblioteka objekata aplikacije odabrana u pretraživaču objekata.
Project Description	Korisničko ime projekta. Prikazuje se u dijaloškom okviru References i pretraživaču objekata.

Dostupno je puno drugih opcija, uključujući one za prevođenje, sastavne dijelove i višenitni rad. Kad vam budu potrebne neke od naprednijih opcija, više informacija možete pronaći pretražujući ugrađenu pomoć.

Za više informacija Kako bi naučili više o opcijama sučelja koje utječu na sve projekte, pogledajte 2. poglavlje "Razvijanje aplikacije u Visual Basicu".

Korištenje čarobnjaka i dodataka

Visual Basic vam omogućuje odabir i upravljanje *dodacima* (*add-ins*), proširenjima Visual Basica. Ta proširenja dodaju mogućnosti razvojnom okruženju Visual Basica, na primjer, posebne mogućnosti nadzora izvornog koda.

Microsoft i ostali proizvođači stvorili su dodatke koje možete koristiti u svojim aplikacijama. Čarobnjaci su vrsta dodatka koji pojednostavljuju neke zadatke, kao što je kreiranje forme. U Visual Basic je uključeno nekoliko čarobnjaka.

Kako bi se dodatak pojavio u dijaloškom okviru Add-In Manager, proizvođač dodatka mora osigurati njegovu pravilnu instalaciju.

Korištenje upravitelja dodacima

Dodatak možete pripojiti ili maknuti iz svog projekta korištenjem upravitelja dodacima (Add-In Managera), koji je dostupan iz izbornika Add-Ins. Dijaloški okvir Add-In Manager sadrži popis raspoloživih dodataka.

Kako instalirati dodatak

1. U izborniku **Add-Ins**, odaberite stavku **Add-In Manager**.
2. Odaberite dodatak sa popisa i kliknite željeno ponašanje u okviru Load Behavior. Za micanje dodatka ili sprječavanje učitavanja, odznačite sve kontrolne kućice u okviru Load Behavior.
3. Kad ste odabrali, kliknite gumb **OK**.

   Ovisno o postavkama u okviru Load Behavior, Visual Basic povezuje odabrane dodatke ili prekida vezu s odznačenim dodacima.

Visual Basic snima vaš odabir dodataka između pokretanja.

Napomena Odabir dodatka može rezultirati dodavanjem novih stavki u izborniku Add-Ins Visual Basica.

Korištenje čarobnjaka

Čarobnjaci čine rad s Visual Basicom još jednostavnijim pružajući pomoć prikladnu zadatku. Na primjer, Application Wizard uključen u Visual Basic pomaže vam pri stvaranju kostura aplikacije pružajući niz pitanja ili izbora. On stvara forme i pripadajući programski kod ovisno o vašim izborima; sve što trebate je dodati programski kod za vaše posebne zadatke.

Professional i Enterprise verzije Visual Basica uključuju druge čarobnjake, kao što su Data Form Wizard za kreiranje formi koje će biti korištene sa bazama podataka, i ActiveX Document Wizard za pretvaranje formi koje će se koristiti u Internet aplikacijama.

Čarobnjaci se instaliraju i miču korištenjem upravitelja dodacima. Jednom instalirani, pojavljivat će se kao stavke izbornika Add-Ins. Neki od čarobnjaka pojavljuju se također i kao ikone u odgovarajućim dijaloškim okvirima; na primjer, Application Wizard može biti pokrenut i korištenjem njegove ikone u dijaloškom okviru New Project.

Kako pokrenuti Application Wizzard

- U izborniku **Add-Ins** odaberite stavku **Application Wizard**.

  - ili -

1. U izborniku **File** odaberite stavku **New Project**.
2. Odaberite ikonu Application Wizard.

5. POGLAVLJE

Osnove programiranja

Ovo poglavlje predstavlja bitne sastavne dijelove programskog jezika Visual Basic. Nakon stvaranja sučelja vaše aplikacije koristeći forme i kontrole, trebate napisati programski kod kojim ćete odrediti ponašanje aplikacije. Kao i svaki suvremeni programski jezik, Visual Basic podržava niz uobičajenih programskih konstrukcija i jezičnih elemenata.

Visual Basic je objektno orijentirani programski jezik. Puko spominjanje objekata može uzrokovati pretjeranu zabrinutost kod većine programera. Ne brinite se: uviđate li to ili ne, radili ste s objektima većinu svog života. Jednom kad shvatite nekoliko osnovnih pojmova, objekti će zapravo pomoći učiniti programiranje jednostavnijim nego ikad prije.

Ako ste programirali drugim jezicima, većina materijala pokrivenog ovim poglavljem bit će vam poznata. Iako je većina konstrukcija slična drugim jezicima, događajima upravljana priroda Visual Basica uvodi neke suptilne razlike. Pokušajte pristupiti ovom materijalu otvorena uma; jednom kad shvatite razlike možete ih iskoristiti u svoju korist.

Ako ste novi u programiranju, materijal u ovom poglavlju poslužit će kao uvod u osnovnu gradnju blokova pri pisanju programskog koda. Jednom kad shvatite osnove, bit ćete sposobni kreirati moćne aplikacije korištenjem Visual Basica.

Sadržaj

- Struktura aplikacije Visual Basica
- Prije programiranja
- Način pisanja koda
- Uvod u varijable, konstante i tipove podataka
- Uvod u potprograme
- Uvod u strukturu kontrola
- Rad s objektima

Primjer aplikacije: Vcr.vbp

Većina primjera programskog koda u ovom poglavlju uzeta je iz primjera aplikacije Vcr.vbp koja se nalazi u direktoriju Samples.

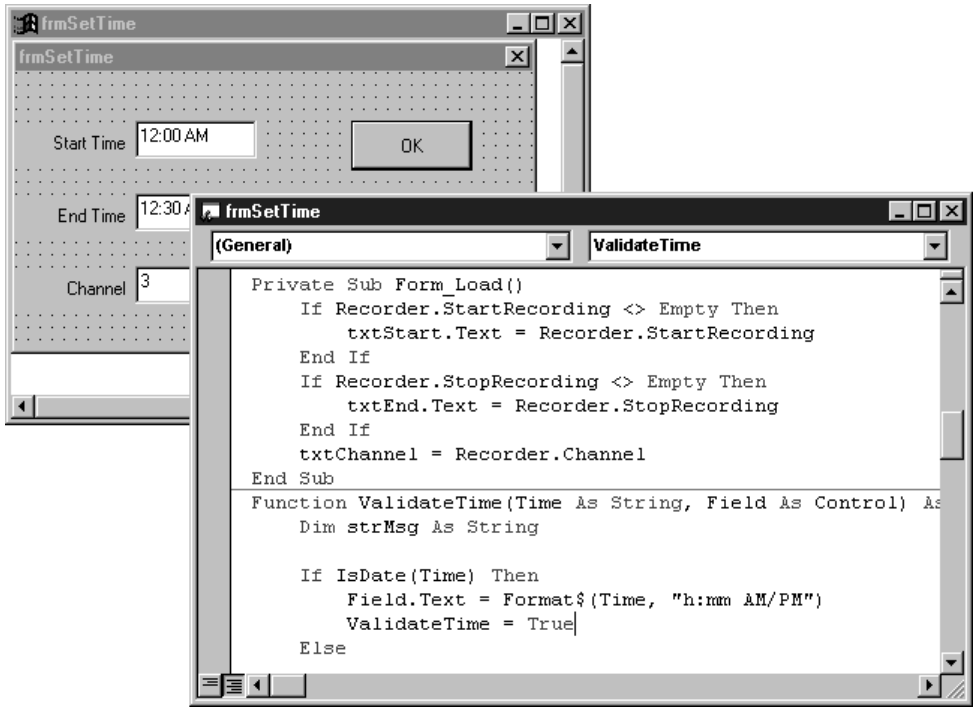
Struktura aplikacije Visual Basica

Aplikacija zapravo nije ništa više od niza instrukcija koje računalu daju upute kako obaviti zadatak ili zadatke. Struktura aplikacije je način na koji su instrukcije organizirane; dakle, gdje su instrukcije spremljene i red po kojem se instrukcije izvode.

Jednostavne aplikacije kao što je klasični primjer "Zdravo, svijete" imaju jednostavnu strukturu; organizacija i nije bitna s jednom linijom programskog koda. Kako aplikacije postaju sve složenije, potreba za organizacijom strukture postaje očita. Zamislite zbrku do koje bi moglo doći kad bi dopustiti izvođenje programskog koda aplikacije slučajnim redoslijedom. Struktura je važna programeru kao dodatak kontroliranju izvođenja aplikacije: kako jednostavno možete pronaći određenu instrukciju unutar svoje aplikacije?

Budući da je Visual Basic aplikacija temeljena na objektima, struktura njenog programskog koda podjednako odgovara fizičkom predočenju na ekranu. Po definiciji, objekti sadrže podatke i programski kod. Forma koju vidite na ekranu je predočenje svojstava koja određuju njezin izgled i stvarno ponašanje. Za svaku formu u aplikaciji, postoji pripadajući *modul forme* (*form module*, datoteka s nastavkom .frm) koji sadrži njen programski kod.

Slika 5.1 Forma i pripadajući modul forme



Svaki modul forme sadrži *događajem uvjetovane potprograme* (*event procedures*) - dijelove koda gdje postavljate instrukcije koje će se izvršiti kao odgovor na određene događaje. Forme mogu sadržavati kontrole. Za svaku kontrolu na formi, postoji odgovarajući skup potprograma događaja u modulu forme. Kao dodatak potprogramima događaja, moduli forme mogu sadržavati opće potprograme koji će biti izvršeni kao odgovor na poziv bilo kojeg potprograma događaja.

Programski kod koji nije vezan uz određenu formu ili objekt može se nalaziti u drugom tipu modula, *standardnom modulu* (*standard module*, .BAS). Potprogram koji može biti korišten kao odgovor na događaje u više različitih objekata treba biti postavljen u standardni modul, radije nego umnožavanje koda u potprogramima događaja za svaki objekt.

Modul klase (*class module*, .CLS) koristi se kod stvaranja objekata koje mogu biti pozivani od potprograma unutar vaše aplikacije. Dok standardni modul sadrži samo programski kod, modul klase sadrži i kod i podatke - možete ga shvatiti kao kontrolu koja nema vidljivi dio.

Dok 4. poglavlje “Upravljanje projektima” opisuje koje kontrole dodati aplikaciji, ovo poglavlje objašnjava kako napisati programski kod za razne dijelove koji čine aplikaciju. U pravilu, vaš projekt sadržava jedan modul forme. Po potrebi možete dodati dodatne forme, module klase i standardne module. Moduli klase su obrađeni u 9. poglavlju “Programiranje objektima”.

Kako radi aplikacija upravljana događajima

Događaj je akcija prepoznata od forme ili kontrole. Aplikacije upravljane događajima izvode programski kod Basic kao odgovor na događaje. Svaka forma i kontrola u Visual Basicu ima unaprijed određeni skup događaja. Ako se pojavi neki od tih događaja i postoji pripadajući potprogram događaja, Visual Basic poziva taj kod.

Iako objekti u Visual Basicu automatski prepoznaju unaprijed određeni skup događaja, na vama je da odlučite hoće li oni i kako odgovoriti na određeni događaj. Dio programskog koda - potprogram događaja - odgovara svakom događaju. Kad želite odgovor kontrole na neki događaj, napisat ćete odgovarajući programski kod u potprogram tog događaja.

Postoje razne vrste događaja prepoznatih od objekta, ali puno vrsta je zajedničko za većinu kontrola. Na primjer, većina kontrola prepoznaje događaj Click - ako korisnik klikne na formu, izvršit će se programski kod u potprogramu događaja Click forme; ako korisnik klikne naredbeni gumb, izvršit će programski kod u potprogramu događaja Click gumba. Stvarni kod u oba slučaja će vrlo vjerojatno biti prilično različit.

Evo tipičnog niza događaja u aplikaciji upravljanoj događajima:

1. Aplikacija počinje izvođenjem i forma se učitava i prikazuje.
2. Forma (ili kontrola na formi) prima događaj. Događaj može biti uzrokovan akcijom korisnika (na primjer, pritiskom na tipku), sistemom (na primjer, događaj mjerača vremena), ili posredno vašim programskim kodom (na primjer, događajem Load kad vaš kod učita formu).

3. Ako postoji kod u odgovarajućem potprogramu događaja, izvodi se.
4. Aplikacija čeka na idući događaj.

Napomena Većina događaja pojavljuje se u sprezi s drugim događajima. Na primjer, kad se pojavi događaj DblClick, pojavljuju se i događaji MouseDown, MouseUp i Click.

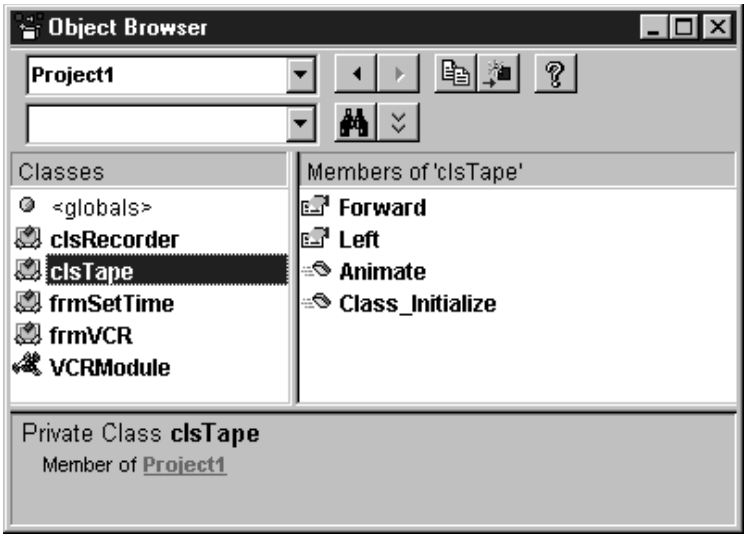
Prije programiranja

Faza oblikovanja je možda najvažniji (i često zanemareni) dio stvaranja aplikacije u Visual Basicu. Iako je očito da trebate oblikovati korisničko sučelje svoje aplikacije, možda nije tako očito da trebate oblikovati i strukturu programskog koda. Način oblikovanja vaše aplikacije može utjecati na razlike u njezinom izvođenju jednako kao i na mogućnost održavanja i iskoristivosti programskog koda.

Programski kod u Visual Basic aplikaciji je organiziran na hijerarhijski način. Tipična aplikacija sastoji se od jednog ili više modula: modula forme za svaku formu u aplikaciji, mogućih standardnih modula za zajednički kod, te mogućih modula klase. Svaki modul sadrži jedan ili više potprograma koji sadrže programski kod: potprograme događaja, potprograme Sub ili Function, te potprograme Property.

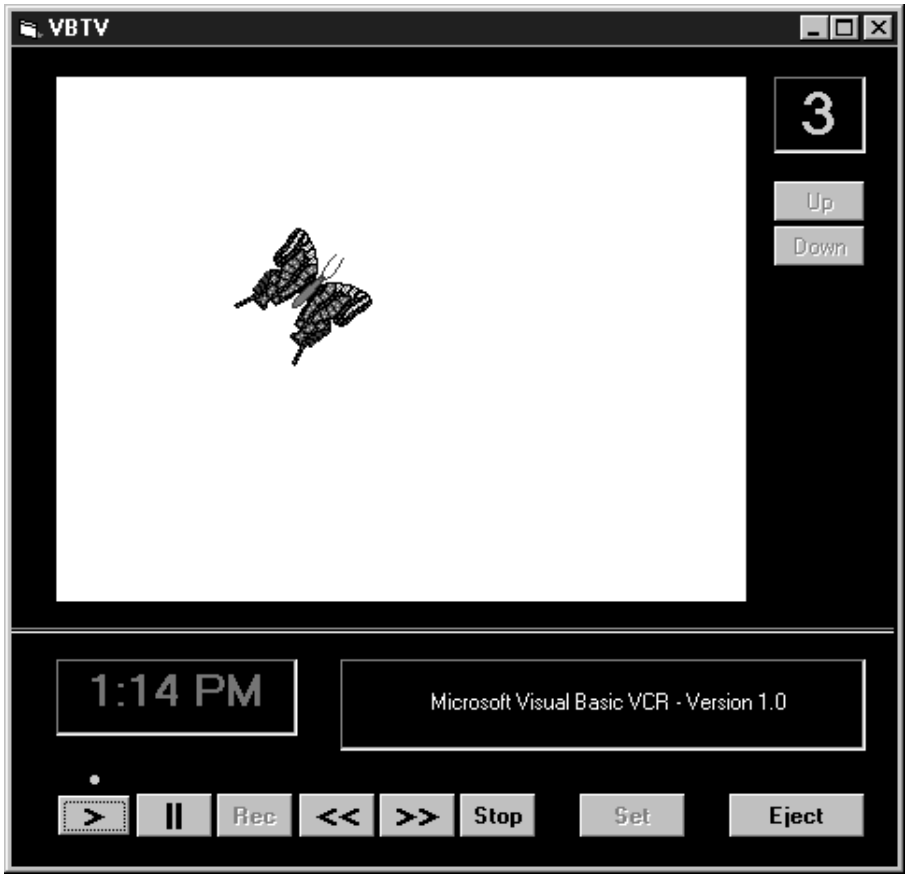
Određivanje koji potprogrami pripadaju u koji modul ponekad ovisi u vrsti aplikacije koju stvarate. Budući da je Visual Basic temeljen na objektima, korisno je razmišljati o vašoj aplikaciji kao o skupu objekata koji je čine. Oblikovanje aplikacije koja služi kao primjer za ovo poglavlje, Vcr.vbp, temeljeno je na objektima koji sadrže videokasetni snimač i televizor. Aplikacija VCR sastoji se od dva modula forme, standardnog modula i dva modula klase. Možete upotrijebiti pretraživač objekata za pregled strukture projekta (slika 5.2).

Slika 5.2 Struktura projekta VCR prikazana u pretraživaču objekata



Glavna forma aplikacije VCR (frmVCR) izgleda kao kombinacija videosnimača i televizijskog ekrana (slika 5.3). Sastavljena je od nekoliko objekata tog tipa kakvi se mogu naći i u stvarnom svijetu. Grupa naredbenih gumbâ (cmdPlay, cmdRecord i tako dalje) oponaša gumbe koji ste koriste pri radu videosnimača. Ovaj softverski videosnimač također sadrži i sat (lblTime), pokazivač kanala (lblChannel), pokazivače funkcija (shpPlay, shpRecord, i tako dalje), te "ekran" (picTV). Potprogrami događaja svih ovih objekata nalaze se u modulu forme Vcr.frm.

Slika 5.3 Glavna forma aplikacije VCR



U puno slučajeva postoje ponavljajući postupci koje dijeli više objekata. Na primjer, kada se "pritisnu" gumbi Play, Rewind ili Record, gumbi Pause i Stop trebaju biti omogućeni. Umjesto ponavljanja takvog programskog koda u potprogramu događaja Click svakog gumba, bolje je stvoriti zajednički potprogram Sub kojeg će moći pozvati svaki gumb. Ako te rutine u budućnosti trebaju biti mijenjane, sve promjene bit će napravljene na jednom mjestu. Ovaj i svi ostali zajednički potprogrami nalaze se u standardnom modulu Vcr.bas.

Neki dijelovi aplikacije stvarnog videosnimača nisu vidljivi, kao što je mehanizam za pokretanje trake ili podrška snimanju televizijskog programa. Jednako tome, neke funkcije aplikacije VCR nemaju vidljivih dijelova. One su uključene u dva modula klase: Recorder.cls i Tape.cls. Programski kod koji pokreće postupak "snimanja" sadržan je u modulu clsRecorder; programski kod koji nadzire smjer i brzinu kretanja "trake" sadržan je u modulu clcTape. Klase definirane u tim modulima nemaju direktnih veza s bilo kojim objektom na formama. Budući da su to neovisni programski moduli, lako mogu biti iskorišteni za izgradnju tonskog snimača bez ikakvih promjena.

Kao dio oblikovanja strukture vašeg programskog koda, važno je uspostaviti dogovore o dodjeli imena. U pravilu, Visual Basic daje prvoj formi u projektu ime Form1, drugoj Form2 i tako dalje. Ako u aplikaciji imate više formi, dobra je ideja dati im smislena imena kako ne bi došlo do zabune pri pisanju ili editiranju programskog koda. Neki savjeti za dodjelu imena predstavljeni su u Dodatku B "Pravila programiranja Visual Basica".

Tijekom daljnjeg učenja o objektima i pisanju programskog koda, u aplikaciji VCR možete pogledati primjere raznih tehnika programiranja.

Način pisanja koda

Prije nego što počnete, važno je razumjeti način pisanja programskog koda u Visual Basicu. Kao i svaki programski jezik, Visual Basic ima vlastita pravila za organiziranje, editiranje i oblikovanje koda.

Moduli koda

Programski kod je u Visual Basicu spremljen u modulima. Postoje tri vrste modula: forme, standardni moduli i moduli klase.

Jednostavna aplikacija može se sastojati samo od jedne forme, a sav kod aplikacije nalazi se unutar modula forme. Kako će vaše aplikacije postajati veće i naprednije, dodavat ćete dodatne forme. Naposljetku ćete doći do zajedničkog koda kojeg želite izvršavati u nekoliko formi. Ne želite umnožavati isti kod u više formi, pa ćete stvoriti odvojene module s potprogramima koji sadrže zajednički kod. Takav odvojeni modul mogao bi biti standardni modul. Nakon nekog vremena, možete napraviti biblioteku modula koji sadržavaju djeljive potprograme.

Svaki standardni modul, modul klase ili forme mogu sadržavati:

- Definicije. Na razini modula standardnih modula, modula klase ili forme možete postaviti određivanja konstanti, tipova, varijabli i dinamički povezanih biblioteka (DLL).
- Potprogrami. Sub, Function i Property potprogrami sadržavaju dijelove koda koji mogu biti izvršeni kao cjelina. Potprogrami su raspravljeni u dijelu "Uvod u potprograme" kasnije u ovom poglavlju.

Moduli forme

Moduli forme (datoteke s nastavkom .FRM) su temelj većine aplikacija u Visual Basicu. Mogu sadržavati potprograme koji će obrađivati događaje, opće potprograme, te određivanja varijabli, konstanti, tipova i vanjskih potprograma na razini forme. Ako nekim editorom teksta pogledate modul forme, vidjet ćete i opise forme i njezinih kontrola, uključujući vrijednosti svojstava. Programski kod koji upisujete u modul forme svojstven je aplikaciji kojoj pripada forma; može također ukazivati na ostale forme i objekte unutar te aplikacije.

Standardni moduli

Standardni moduli (datoteke s nastavkom .BAS) su spremnici potprograma i određivanja obično pozivanih od drugih modula unutar aplikacije. Oni mogu sadržavati opća određivanja (dostupna cijeloj aplikaciji) ili određivanja na razini modula za varijable, konstante, tipove, vanjska ili opća određivanja. Programski kod koji pišete u standardni modul nije nužno povezan s određenom aplikacijom; ako pripazite da ne pozivate forme ili kontrole njihovim imenima, standardni modul može biti korišten u više različitih aplikacija.

Moduli klase

Moduli klase (datoteke s nastavkom .CLS) su temelj objektno orijentiranog programiranja u Visual Basicu. U modulima klase možete upisati programski kod za stvaranje novih objekata. Ti novi objekti mogu uključivati vlastita prilagođena svojstva i postupke. Zapravo, forme su samo moduli klase koji mogu imati kontrole postavljene na njih i mogu prikazivati prozor forme.

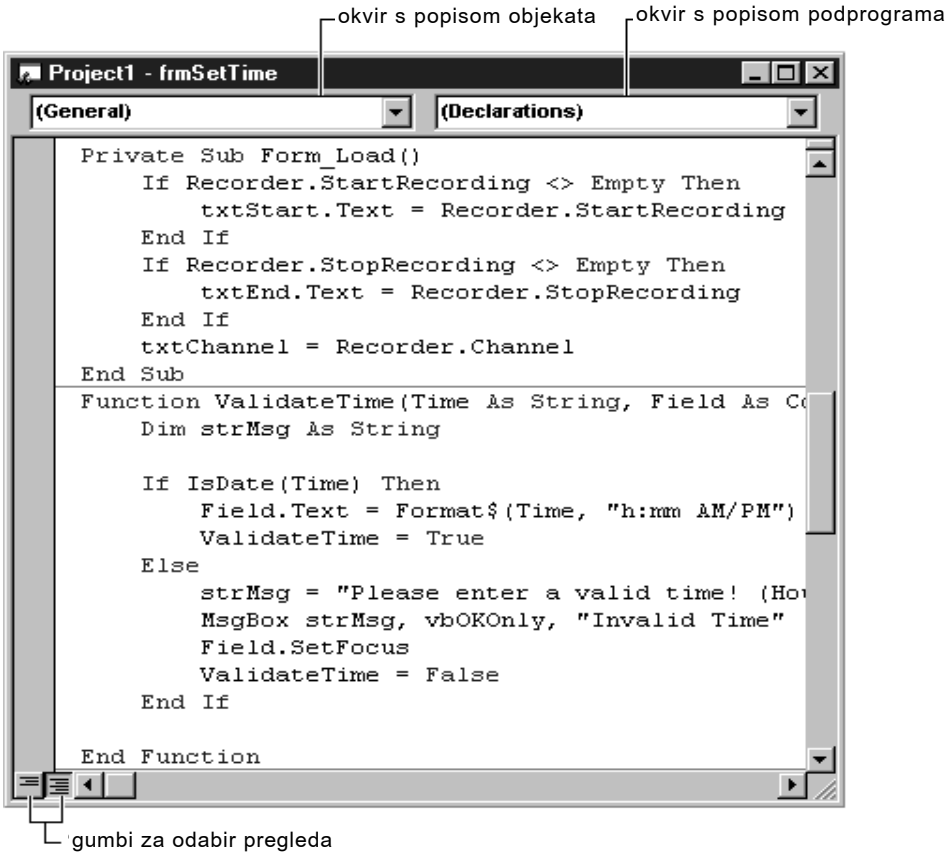
Za više informacija Za informacije o pisanju koda u modulima klase pogledajte 9. poglavlje “Programiranje objektima”.

Napomena Professional i Enterprise verzije Visual Basica također uključuju ActiveX dokumente, ActiveX kreatore te korisničke kontrole. Ove verzije uvode nove vrste modula s drugačijim imenima datoteka. Sa stajališta pisanja programskog koda, takvi moduli mogu se smatrati jednakima modulima forme.

Korištenje kodnog prozora

Kodni prozor Visual Basica je prozor u kojem ćete pisati većinu svog programskog koda. Izgleda kao visoko specijaliziran obradnik teksta s nizom svojstava koja čine pisanje Visual Basic koda daleko jednostavnijim. Kodni prozor je prikazan na slici 5.4.

Slika 5.4 Kodni prozor



Budući da s programskim kodom Visual Basica radite u modulima, otvara se poseban kodni prozor za svaki modul koji odaberete unutar projektnog prozora. Programski kod unutar svakog modula je podijeljen u odvojene odjeljke za svaki objekt unutar modula. Prebacivanje između odjeljaka obavlja se korištenjem okvira s popisom objekata. U modulu forme, popis uključuje opći odjeljak, odjeljak za samu formu, te odjeljke za svaku kontrolu koju sadržava forma. Kod modula klase, popis uključuje opći odjeljak i odjeljak klase; kod standardnih modula postaji samo opći odjeljak.

Svaki odjeljak programskog koda može sadržavati različite potprograme, kojima možete pristupiti preko okvira s popisom potprograma. Lista potprograma za modul forme sadrži odvojene odjeljke za svaki potprogram događaja forme ili kontrole. Na primjer, popis potprograma za kontrolu natpisa, između ostalih, sadržava i odjeljke za događaje Change, Click i DblClick. Moduli klase sadrže samo popis događaja za samu klasu – događaje Initialize i Terminate. Standardni moduli nemaju popis potprograma događaja, jer standardni moduli ne podržavaju događaje.

Popis potprograma za opći odjeljak modula sadrži samo jednu stavku – odjeljak Declarations, gdje postavljate varijable i konstante na razini modula, te određivanja dinamičnih biblioteka. Dodavanjem potprograma Sub ili Function modulu, ti potprogrami se dodaju i u okvir s popisom potprograma ispod odjeljka Declarations.

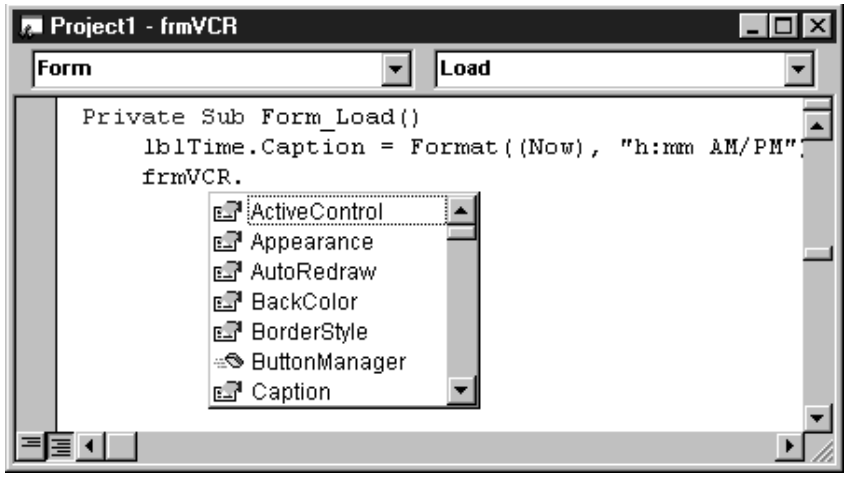
U kodnom prozoru su moguća dva načina pregleda vašeg programskog koda. Možete odabrati pregled samo jednog potprograma, ili pregled svih potprograma u modulu gdje su potprogrami međusobno odvojeni linijom (kako je prikazano na slici 5.4). Za prijelaz između ova dva načina, iskoristite gumbe za odabir pregleda (view selection buttons) u donjem lijevom kutu kodnog prozora.

Automatsko završavanje koda

Visual Basic čini pisanje programskog koda daleko lakšim mogućnostima koje mogu automatski dovršiti naredbe, svojstva i argumente umjesto vas. Dok upisujete programski kod, editor prikazuje popis prikladnih izbora, primjera naredbi ili funkcija, ili vrijednosti. Opcije za uključivanje ili isključivanje ove i sličnih mogućnosti nalaze se na kartici Editor dijaloškog okvira Options, dostupnog naredbom Options u izborniku Tools.

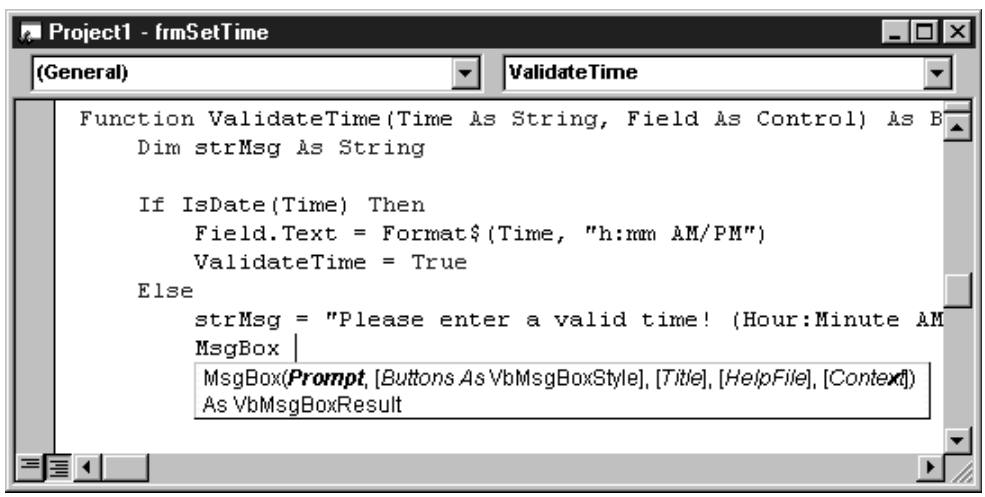
Kad u vaš programski kod upišete ime kontrole, mogućnost automatskog završavanja izraza (Auto List Members) prikazuje spuštajući popis svojstava dostupnih za tu kontrolu (slika 5.5). Upišite nekoliko prvih slova imena svojstva i njegovo ime će biti odabrano s popisa; tipka TAB će završiti upis umjesto vas. Ova mogućnost je od pomoći kad niste sigurni koja su svojstva na raspolaganju za određenu kontrolu. Čak i ako odlučite isključiti mogućnost automatskog završavanja izraza, možete joj pristupiti kombinacijom tipki CTRL+J.

Slika 5.5 Mogućnost automatskog završavanja izraza



Automatska brza pomoć (auto quick info) prikazuje sintaksu naredbi i funkcija (slika 5.6). Kad upišete ime naredbe ili funkcije Visual Basica, odmah ispod trenutnog izraza će se prikazati sintaksa, s prvim argumentom ispisanim podebljanim pismom. Nakon što upišete prvi argument, u pomoćnom okviru idući argument će biti ispisan podebljanim pismom. Automatska brza pomoć može se pozvati i kombinacijom tipki CTRL+I.

Slika 5.6 Automatska brza pomoć



Zabilješke

Zabilješke (bookmarks) mogu biti upotrijebljene za označavanje linija programskog koda u kodnom prozoru tako da im se kasnije lako možete vratiti. Naredbe za uključivanje i isključivanje zabilježavanja kao i za kretanje kroz postojeće zabilješke dostupne su odabirom stavke Bookmarks u izborniku Edit, ili preko alatne trake Edit.

Temelji kodiranja

Ovaj dio predstavlja informacije o načinima pisanja programskog koda, uključujući lomljenje i sastavljanje linija koda, dodavanje komentara vašem kodu, korištenje brojeva u kodu, te uobičajene načine nazivanja u Visual Basicu.

Lomljenje jedne linije u više linija

Dugačku naredbenu liniju u kodnom prozoru možete rastaviti u više linija korištenjem *oznake za nastavak linije* (*line-continuation character*) koja se sastoji od razmaka i podvučene linije. Korištenje ove oznake može učiniti vaš programski kod lakšim za čitanje, i tijekom programiranja i kod ispisa na pisač. Sljedeći programski kod je rastavljen u tri linije korištenjem oznaka za nastavak linije (_):

```
Data1.RecordSource = _
“SELECT * FROM Titles, Publishers” _
& “WHERE Publishers.PubId = Titles.PubID” _
& “AND Publishers.State = ‘CA’”
```

Iza oznake za nastavak linije ne smije se nalaziti komentar u istoj liniji. Također postoje i neka ograničenja gdje se oznake za nastavak linije smiju koristiti.

Spajanje naredbi u jednoj liniji

Obično se u jednoj liniji nalazi jedna naredba Visual Basica, i ne postoji oznaka za kraj naredbe. Međutim, u jednu liniju možete postaviti dvije ili više naredbi ako iskoristite dvotočku (:) da bi ih razdvojili:

```
Text1.Text = “Pozdrav” : Crveno = 255 : Text1.BackColor = _
Crveno
```

Kako bi vaš programski kod bio čitljiviji, ipak je bolje postaviti svaku naredbu u odvojenu liniju.

Za više informacija Za više informacija, pogledajte Dodatak A “Specifikacije, ograničenja i vrste datoteka Visual Basica”.

Dodavanje komentara u vaš kod

Čitajući primjere programskog koda u ovoj knjizi, često ste mogli naići na oznaku komentara (‘). Ovaj simbol govori Visual Basicu da zanemari riječi koje se nalaze iza njega. Takve riječi su komentari postavljeni u programski kod kao pomoć programeru, te ostalim programerima koji kasnije mogu istraživati taj kod. Na primjer:

```
‘ Ovo je komentar koji počinje
‘ na lijevoj strani ekrana.
Text1.Text = “Pozdrav!”          ‘ Ispis pozdrava
                                 ‘ u okvir s tekstom.
```

Komentari se mogu nalaziti iza naredbi u istoj liniji ili mogu zauzeti cijelu liniju. Oba slučaja su prikazana u prethodnom ispisu programskog koda. Zapamtite da se komentari ne mogu nalaziti u liniji iza oznake za nastavak linije.

> **Napomena** Možete postaviti ili maknuti oznake komentara za blok programskog koda tako da odaberete dvije ili više linija koda pa odaberete gumbe Comment Block ili Uncomment Block na alatnoj traci Edit.

Razumijevanje sustava brojeva

Većina brojeva u ovoj dokumentaciji su decimalni (s bazom 10). Ponekad je prikladno koristiti heksadecimalne brojeve (s bazom 16) i oktalne brojeve (s bazom 8). Visual Basic prikazuje brojeve u heksadecimalnom sustavu s prefiksom &H te u oktalnom s prefiksom &O. Sljedeća tabela pokazuje iste brojeve u decimalnom, oktalnom i heksadecimalnom sustavu.

decimalno	oktalno	heksadecimalno
9	&O11	&H9
15	&O17	&HF
16	&O20	&H10
20	&O24	&H14
255	&O377	&HFF

Općenito ne trebate sami naučiti heksadecimalni i oktalni sustav brojeva jer računalo može raditi s brojevima bilo kojeg sustava. Unatoč tome, neki brojevni sustavi posuđuju se određenim zadacima, kao što je korištenje heksadecimalnih brojeva za postavke boja ekrana i kontrola.

Uobičajeni načini nazivanja u Visual Basicu

Tijekom pisanja programskog koda Visual Basica, određujete i imenujete puno elemenata (Sub i Function potprograme, varijable, konstante i tako dalje). Imena potprograma, varijabli i konstanti koje određujete u vašem programskom kodu Visual Basica moraju slijediti ove smjernice:

- Moraju počinjati slovom.
- Ne mogu sadržavati točku ili oznaku tipa podatka (posebni znakovi za određivanje tipa podatka).
- Ne smiju biti dulji od 255 znakova. Imena kontrola, formi, klasa i modula ne smiju biti duža od 40 znakova.
- Ne smiju biti ista kao i ograničene ključne riječi.

Ograničena ključna riječ (*restricted keyword*) je riječ koju Visual Basic koristi kao dio programskog jezika. Ovo uključuje predefinirane naredbe (kao If i Loop), funkcije (kao Len i Abs), te operatore (kao Or i Mod).

Za više informacija Za kompletan popis ključnih riječi, pogledajte dio *Microsoft Visual Basic 6.0 Language Reference* biblioteke *Microsoft Visual Basic 6.0 Reference Library*.

Vaše forme i kontrole mogu imati isto ime kao i ključna riječ. Na primjer, možete imati kontrolu imena Loop. Unatoč tome, u vašem programskom kodu ne možete se pozivati na tu kontrolu uobičajenim načinom, jer Visual Basic pretpostavlja da pozivate ključnu riječ Loop. Na primjer, sljedeći kod će izazvati pogrešku:

```
Loop.Visible = True                ‘ Izaziva pogrešku.
```

Za poziv na kontrolu koja ima isto ime kao i ključna riječ, morate ju označiti vrstom ili upisati u uglate zagrade: [ ]. Na primjer, sljedeći programski kod neće izazvati pogrešku:

```
MyForm.Loop.Visible = True         ‘ Označeno imenom forme.
[Loop].Visible = True              ‘ Djeluju i uglate zagrade.
```

Uglate zagrade možete koristiti na ovaj način kad pozivate forme i kontrole, ali ne kad deklarirate varijable ili definirate potprograme istog imena kao i ključna riječ. Uglate zagrade također se mogu koristiti za prisiljavanje Visual Basica u prihvaćanju imena koja pružaju biblioteke drugih vrsta, a ta imena se sukobljavaju s ključnim riječima.

Napomena Budući da pisanje uglatih zagrada može biti zamorno, možda se poželite uzdržati od korištenja ključnih riječi kao naziva formi i kontrola. Unatoč tome, možete koristiti ovu tehniku ako sljedeća verzija Visual Basica odredi novu ključnu riječ koja se sukobljava s imenom postojeće forme ili kontrole kod prilagodbe vašeg koda radu s novom verzijom.

Uvod u varijable, konstante i tipove podataka

Kod izvođenja računanja Visual Basicom često trebate privremeno spremiti neke vrijednosti. Na primjer, želite proračunati nekoliko vrijednosti, usporediti ih, i ovisno o rezultatu usporedbe, izvesti nad njima različite radnje. Morate zadržati vrijednosti ako ih želite usporediti, ali ih ne želite sačuvati za nastavak rada.

Visual Basic, kao i većina programskih jezika, koristi *varijable* za čuvanje vrijednosti. Varijable imaju ime (riječ koju koristite za poziv na vrijednost koju sadrži varijabla) i *tipu podatka* (koji određuje tip podatka koji varijabla može spremiti). *Matrice* mogu biti korištene za spremanje indeksiranih skupova vezanih varijabli.

Konstante također čuvaju vrijednosti, ali kako im to ime pokazuje, te vrijednosti ostaju stalne tijekom rada aplikacije. Korištenje konstanti može učiniti vaš programski kod čitljivijim koristeći smislena imena umjesto brojeva. Postoji niz konstanti ugrađenih u Visual Basic, ali i vi možete stvoriti vlastite.

Tip podatka nadzire unutarnje skladište podataka u Visual Basicu. U pravilu, Visual Basic koristi tip podatka Variant. Postoji niz drugih raspoloživih tipova podataka koji vam omogućuju prilagodbu vašeg programskog koda brzini i veličini kad vam nije potrebna fleksibilnost koju pruža tip Variant.

Varijable

U Visual Basicu koristite varijable za privremenu pohranu podataka tijekom rada aplikacije. Varijable imaju ime (riječ koju koristite za poziv na vrijednost koju sadrži varijabla) i tip podatka (koja određuje tip podatka koji varijabla može spremiti).

Varijablu možete shvatiti kao oznaku mjesta u memoriji za nepoznatu vrijednost. Na primjer, zamislite da stvarate aplikaciju za trgovinu voćem koji će pratiti prodaju jabuka. Ne znate cijenu jabuka ili prodanu količinu sve dok do prodaje stvarno ne dođe. Možete koristiti dvije varijable za čuvanje nepoznatih vrijednosti – nazovimo ih JabukeCijena i JabukeProdano. Svaki put kad se pokrene aplikacija, korisnik daje vrijednosti za ove dvije varijable. Za izračun konačne cijene i njegov prikaz u okviru s tekstom nazvanom txtSales, vaš programski kod bi mogao izgledati ovako:

```
txtSales.text = JabukeCijena + JabukeProdano
```

Ovaj izraz vraća svaki put drugačiji rezultat, ovisno o vrijednostima koje da korisnik. Varijable vam omogućuju izradu proračuna bez potrebe za poznavanjem unaprijed koliki će biti stvarni unosi.

U ovom primjeru, tip podatka za varijablu JabukeCijena je Currency (valuta); tip podatka za JabukeProdano je cijeli broj. Varijable mogu sadržavati i puno drugačijih vrijednosti: tekstove, datume, različite brojčane tipove, čak i objekte.

Spremanje i čitanje podataka iz varijabli

Za izvođenje proračuna i dodjelu vrijednosti varijablama koristite naredbe s dodjeljivanjem:

```
JabukeProdano = 10        ‘ Vrijednost 10 se
                          ‘ dodjeljuje varijabli.
JabukeProdano = JabukeProdano + 1        ‘ Varijabla je povećana.
```

Uočite da je u ovom primjeru znak jednakosti operator za dodjelu vrijednosti, a ne operator izjednačavanja; vrijednost (10) se dodjeljuje varijabli (JabukeProdano).

Određivanje varijabli

Odrediti varijablu znači unaprijed reći aplikaciji o njoj. Varijable određujete naredbom Dim, pridodajući joj ime varijable:

Dim *imevarijable* [**As** *tip*]

Varijable određene naredbom Dim unutar potprograma postoje samo dok se izvodi taj potprogram. Kad se potprogram završi, vrijednost varijable nestaje. Slično tome, vrijednost varijable u potprogramu je *lokalna* za taj potprogram – to znači da ne možete pročitati varijablu jednog potprograma iz drugog potprograma. Ova svojstva omogućuju vam korištenje varijabli istog imena u različitim potprogramima bez brige u sukobima ili slučajnim promjenama.

Ime varijable:

- Mora počinjati slovom.
- Ne smije sadržavati točku ni oznaku tipa podatka.
- Ne smije biti dulje od 255 znakova.
- Mora biti jedinstveno u svom *opsegu*, području u kojem varijabla može biti pozivana – potprogramu, formi ili sličnom.

Neobavezan uvjet As *tip* u naredbi Dim dopušta vam definiranje tipa podatka ili objekta za varijablu koju određujete. Tipovi podataka određuju vrstu informacije koju će varijabla čuvati. Neki primjeri tipova podataka su String, Integer i Currency. Varijable također mogu sadržavati objekte Visual Basica i drugih aplikacija. Primjeri objekata i klasa Visual Basica su Object, Form i TextBox.

Za više informacija Za dodatne informacije pogledajte 9. poglavlje “Programiranje objektima” i 10. poglavlje “Programiranje sastavnim dijelovima”. Tipovi podataka su detaljnije razrađeni u odlomku “Tipovi podataka”, kasnije u ovom poglavlju.

Postoje i drugi načini određivanja varijabli:

- Određivanje varijable u odjeljku Declarations forme, standardnog modula ili modula klase, radije nego u potprogramu, čini varijablu dostupnom svim potprogramima u modulu.
- Određivanje varijable korištenjem ključne riječi Public čini je dostupnom cijeloj aplikaciji.
- Određivanje lokalne varijable korištenjem ključne riječi Static čuva vrijednost varijable kad potprogram završi.

Podrazumijevano određivanje

Ne morate odrediti varijable prije korištenja. Na primjer, možete napisati funkciju u kojoj ne morate odrediti varijablu TempVal prije korištenja:

```
Function SafeSqr(num)
    TempVal = Abs(num)
    SafeSqr = Sqr(TempVal)
End Function
```

Visual Basic automatski kreira varijablu tog imena, i možete je koristiti kao da ste je izričito odredili. Iako je ovo zgodan način, može dovesti do tajanstvenih pogrešaka u vašem programskom kodu ako pogrešno napišete ime varijable. Na primjer, pretpostavimo da je ovo funkcija koju ste napisali:

```
Function SafeSqr(num)
    TempVal = Abs(num)
    SafeSqr = Sqr(TemVal)
End Function
```

Na prvi pogled, izgleda isto. Međutim, zbog pogrešno napisanog imena varijable `TemVal` u pretposljednjoj liniji, ova funkcija će uvijek kao rezultat vraćati nulu. Kad Visual Basic otkrije novo ime, ne može utvrditi jeste li zapravo mislili odrediti novu varijablu u hodu ili ste pogrešno napisali ime već postojeće varijable, pa kreira novu varijablu takvog imena.

Izričito određivanje

Kako bi izbjegli problem pogrešnog pisanja imena varijabli, možete odrediti da vas Visual Basic uvijek upozori kad otkrije ime koje izričito nije određeno kao varijabla.

Kako izričito odrediti varijable

- Postavite sljedeći izraz u odjeljak Declarations klase, forme ili standardnog modula:

  ```
  Option Explicit
  ```

  - ili -

  U izborniku **Tools** odaberite stavku **Options**, kliknite karticu **Editor** i potvrdite kontrolnu kućicu za izbor **Require Variable Declaration**. Ovaj postupak će automatski postaviti izraz Option Explicit u svaki novi modul, ali neće u već kreirane module; zbog toga morate ručno dodati izraz Option Explicit u svaki postojeći modul projekta.

Da je ovaj izraz bio aktivan za formu ili standardni modul koji sadrži funkciju SafeSqr, Visual Basic bi prepoznao `TempVal` i `TemVal` kao nedefinirane varijable i prijavio bi pogrešku za obje. Nakon toga bi mogli izričito odrediti `TempVal`:

```
Function SafeSqr(num)
    Dim TempVal
    TempVal = Abs(num)
    SafeSqr = Sqr(TemVal)
End Function
```

Sad bi odmah shvatili problem jer bi Visual Basic prikazao poruku pogreške za nepravilno napisanu varijablu TemVal. Budući da vam izraz Option Explicit pomaže u hvatanju ovakvih vrsta pogrešaka, dobra je ideja koristiti ga u svakom programskom kodu.

Napomena Izraz Option Statement radi na temelju modula; mora biti postavljen u odjeljak Declarations svake forme, standardnog ili klasnog modula za koji želite da Visual Basic traži izričito određivanje varijabli. Ako potvrdite izbor Require Variable Declaration, Visual Basic ubacuje izraz Option Explicit u sve sljedeće forme, standardne i klasne module, ali ga ne dodaje u već postojeći programski kod. Izraz Option Explicit morate ručno dodati u svaki postojeći modul unutar projekta.

Razumijevanje područja varijabli

Područje varijable određuje koji dijelovi vašeg programskog koda će biti svjesni njenog postojanja. Kad odredite varijablu unutar potprograma, samo programski kod unutar tog potprograma može pristupiti varijabli ili joj promijeniti vrijednost; varijabla ima lokalno područje u tom potprogramu. Ponekad se, međutim, pojavljuje potreba za varijablama sa širim područjem, kao što su one čija je vrijednost dostupna svim potprogramima unutar istog modula, ili čak svim potprogramima cijele aplikacije. Visual Basic vam omogućuje označavanje područja varijable kad je---- određujete.

Područje varijabli

Ovisno o načinu određivanja, varijable mogu vrijediti na području potprograma (lokalne varijable) ili modula.

područje	privatne (private)	javne (public)
potprogram	Varijable su privatne za potprogram u kojem se pojavljuju.	Neprimjenjivo. Ne možete odrediti javnu varijablu unutar potprograma.
modul	Varijable su privatne za modul u kojem se pojavljuju.	Varijable su dostupne svim modulima.

Varijable unutar potprograma

Varijable na razini potprograma prepoznaju se samo u potprogramu u kojem su određene. Ove varijable nazivaju se i lokalne varijable. Određujete ih naredbama Dim ili Static. Na primjer:

```
Dim intTemp As Integer
```

- ili -

```
Static intPermanent As Integer
```

Vrijednosti u lokalnim varijablama određenima naredbom Static postoje cijelo vrijeme rada vaše aplikacije dok varijable određene naredbom Dim postoje samo dok se izvršava potprogram.

Lokalne varijable su dobar izbor za bilo koju vrstu privremenog proračuna. Na primjer, možete kreirati desetak različitih potprograma koji sadrže varijablu nazvanu intTemp. Sve dok je svaka varijabla intTemp određena kao lokalna, svaki potprogram prepoznaje samo vlastitu varijantu varijable intTemp. Bilo koji potprogram može promijeniti vrijednost svoje lokalne varijable intTemp bez utjecaja na varijable intTemp u ostalim potprogramima.

Varijable unutar modula

U pravilu, varijable na razini modula dostupne su svim potprogramima u tom modulu, ali ne i programskom kodu u drugim modulima. Varijablu na razini modula možete stvoriti određivanjem naredbom Private u odjeljku Declarations na vrhu programskog koda modula. Na primjer:

```
Private intTemp As Integer
```

Na razini modula ne postoji razlika između naredbi Private i Dim, ali preporučuje se Private jer je u stvarnoj suprotnosti s naredbom Public i čini vaš kod razumljivijim.

Varijable za sve module

Upotrijebite naredbu Public za određivanje varijable, kako bi varijable na razini modula učinili dostupnima ostalim modulima. Vrijednosti u takvim javnim varijablama su dostupne svim potprogramima u vašoj aplikaciji. Kao i sve varijable na razini modula, javne varijable se određuju u odjeljku Declarations na vrhu programskog koda modula. Na primjer:

```
Public intTemp As Integer
```

> **Napomena** Ne možete odrediti javne varijable unutar potprograma, već samo unutar odjeljka Declarations modula.

Za više informacija Za dodatne informacije o varijablama, pogledajte sljedeći odlomak "Napredne teme o varijablama".

Napredne teme o varijablama

Korištenje više varijabli s istim imenom

Ako više javnih varijabli u različitim modulima dijeli isto ime, moguće je razlikovati ih u programskom kodu pristupajući im i po imenu modula i po imenu varijable. Na primjer, ako postoji javna varijabla imena intX i tipa Integer određena u modulima Form1 i Module1, možete je pozivati kao Module1.intX i Form1.intX kako bi dobili ispravne vrijednosti.

Da bi vidjeli kako ovo radi, ubacite dva standardna modula u novi projekt i kreirajte tri naredbena gumba na formi.

Jedna varijabla, intX, određena je u prvom standardnom modulu, Module1. Potprogram Test postavlja njezinu vrijednost:

```
Public intX As Integer            ‘ Određivanje varijable intx u Module1.
Sub Test()
    ‘ Postavljanje vrijednosti varijable intX u Module1.
    intX = 1
End Sub
```

Druga varijabla, koja ima isto ime, intX, određuje se u drugom standardnom modulu, Module2. Ponovno, potprogram Test postavlja njezinu vrijednost:

```
Public intX As Integer            ‘ Određivanje varijable intx u Module2.
Sub Test()
    ‘ Postavljanje vrijednosti varijable intX u Module2.
    intX = 2
End Sub
```

Treća varijabla `intX` je određena u modulu forme. I ponovno, potprogram imena Test postavlja njenu vrijednost.

```
Public intX As Integer            ‘ Određivanje varijable intx u formi.
Sub Test()
    ‘ Postavljanje vrijednosti varijable intX u formi.
    intX = 3
End Sub
```

Svaki od događajem pokretanih potprograma Click tri naredbena gumba poziva pripadajući potprogram Test i koristi okvir za poruke MsgBox za prikaz vrijednosti tri varijable.

```
Private Sub Command1_Click()
    Module1.Test                  ‘ Poziva Test u Module1.
    MsgBox Module1.intX           ‘ Prikazuje intX iz Module1.
End Sub

Private Sub Command2_Click()
    Module2.Test                  ‘ Poziva Test u Module2.
    MsgBox Module2.intX           ‘ Prikazuje intX iz Module2.
End Sub

Private Sub Command3_Click()
    Test                          ‘ Poziva Test u Form1.
    MsgBox intX                   ‘ Prikazuje intX iz Form1.
End Sub
```

Pokrenite aplikaciju i kliknite svaki od tri naredbena gumba. Vidjet ćete odvojene poruke s pozivanjem tri javne varijable. Uočite da u potprogramu događaja Click naredbenog gumba Command3 nije potrebno navesti `Form1`.Test za pozivanje potprograma Test forme, ni `Form1.intX` za čitanje vrijednosti varijable intX forme. Ako postoji više potprograma i varijabli istog imena, Visual Basic će uzeti vrijednost od lokalnije varijable, što je ovom slučaju varijabla forme Form1.

Javne protiv lokalnih varijabli

Možete imati i varijable istog imena u različitim područjima. Na primjer, možete imati javnu varijablu imena `Temp`, a zatim, unutar potprograma, odrediti lokalnu varijablu imena `Temp`. Pozivanje varijable `Temp` unutar potprograma će pristupiti lokalnoj varijabli; pozivanje varijable `Temp` izvan potprograma će pristupiti javnoj varijabli. Varijabla na razini modula može biti pozvana iz potprograma određivanjem te varijable s imenom modula.

```
Public Temp As Integer
Sub Test()
    Dim Temp As Integer
    Temp = 2              ‘ Varijabla Temp dobiva vrijednost 2.
    MsgBox Form1.Temp     ‘ Ispis Form1.Temp varijable koja
                            ima vrijednost 1.
End Sub
```

```
Private Sub Form_Load()
    Temp = 1              ‘ Varijabla Form1.Temp dobiva vrijednost 1.
End Sub
Private Sub Command1_Click()
    Test
End Sub
```

Općenito, kad varijable imaju isto ime, ali različito područje djelovanja, lokalnija varijabla uvijek *zasjenjuje* (prije joj se pristupa) manje lokalnu varijablu. Ako, dakle, imate varijablu na razini potprograma, nazvanu Temp, ona će zasjeniti javnu varijablu Temp unutar tog modula.

Zasjenjivanje svojstava forme i kontrola

Svojstva forme, kontrole, konstante i potprogrami se, zbog efekta zasjenjivanja, tretiraju kao varijable na razini modula u modulu forme. Nije pravilno imati svojstvo forme ili kontrolu istog imena kao i varijabla na razini forme, konstanta, korisnički tip varijable ili potprogram jer oba podatka pokrivaju isto područje.

Unutar modula forme, lokalne varijable istog imena kao i kontrole na formi zasjenjuju te kontrole. Morate odrediti kontrolu pozivom na formu ili ključnom riječi Me kako bi postavili ili pročitali njezinu vrijednost ili neko od njezinih svojstava. Na primjer:

```
Private Sub Form_Click()
Dim Text1.BackColor
‘ Pretpostavimo da postoji i kontrola
‘ na formi nazvana Text1.
    Text1 = “Variable”     ‘ Varijabla zasjenjuje kontrolu.
    Me.Text1 = “Control”   ‘ Morate označiti s Me za
                           ‘ pristup kontroli.
    Text1.Top = 0          ‘ Ovo će izazvati pogrešku!
    Me.Text1.Top = 0       ‘ Morate označiti s Me za
                           ‘ pristup kontroli
    BackColor = 0          ‘ Varijabla zasjenjuje kontrolu.
    Me.BackColor = 0       ‘ Morate označiti s Me za
                           ‘ pristup svojstvu kontrole.
End Sub
```

Korištenje varijabli i potprograma istog imena

Imena vaših privatnih i javnih varijabli na razini modula mogu također doći u sukob s imenima vaših potprograma. Varijabla u modulu ne može imati isto ime kao i bilo koji potprogram ili tip postavljen u modulu. Može, unatoč tome, imati isto ime kao i javni potprogram, tip ili varijabla određeni u drugim modulima. U takvom slučaju, kad se pristupa varijabli iz drugog modula, ona mora biti označena imenom modula.

Iako pravila zasjenjivanja opisna malo prije nisu komplicirana, zasjenjivanje često može biti zbunjujuće i zna dovesti do podmuklih pogrešaka u vašem kodu; dobro je programersko iskustvo razlikovati međusobno imena vaših varijabli. U modulima forme pokušajte koristiti imena varijabli koja će biti drugačija od imena kontrola na toj formi.

Statičke varijable

Kao dodatak području djelovanja, varijable imaju *vrijeme trajanja*, vrijeme tijekom kojeg zadržavaju svoju vrijednost. Vrijednosti u javnim i varijablama na razini modula se čuvaju cijelo vrijeme trajanja vaše aplikacije. Nasuprot tome, lokalne varijable određene naredbom Dim postoje smo dok se izvodi potprogram u kojem su određene. Obično, kad potprogram završi izvođenje, vrijednosti njegovih lokalnih varijabli se ne čuvaju i memorija korištena za lokalne varijable se vraća sustavu. Kad se potprogram idući put pokrene, ponovno se pokreću sve njegove lokalne varijable.

Unatoč tome, možete sčuvati vrijednosti lokalnih varijabli čineći varijablu *statičkom*. Upotrijebite naredbu Static za određivanje jedne ili više varijabli unutar potprograma, točno kako bi to napravili i s naredbom Dim:

```
Static Dubina
```

Na primjer, sljedeća funkcija računa ukupan zbroj dodavanjem nove vrijednosti na zbroj prethodnih vrijednosti sčuvan u statičkoj varijabli JabukeProdano:

```
Function UkupanZbroj(num)
    Static JabukeProdano
    JabukeProdano = JabukeProdano + num
    UkupanZbroj = JabukeProdano
End Function
```

Da je varijabla `JabukeProdano` bila određena naredbom Dim umjesto Static, prethodno dodane vrijednosti ne bi bile sčuvane kroz pozive funkcije, i funkcija bi jednostavno vratila istu vrijednost kojom je i pozvana.

Isti rezultat možete dobiti određivanjem varijable `JabukeProdano` u odjeljku Declarations modula, određujući je varijablom na razini modula. Međutim, jednom kad promijenite područje varijable na ovaj način, potprogram više nema isključivo pravo na pristup toj varijabli. Budući da ostali potprogrami mogu pristupiti varijabli i promijeniti joj vrijednost, ukupan zbroj dobiven funkcijom može biti nepouzdan, i bit će teže održavati funkcionalnost programskog koda.

Određivanje svih lokalnih varijabli kao statičkih

Kako bi sve lokalne varijable u potprogramu učinili statičkim, postavite naredbu Static na početak poglavlja potprograma. Na primjer:

```
Static Function UkupanZbroj(num)
```

To će odrediti sve lokalne varijable u potprogramu kao statičke, neovisno o tome hoće li one biti određene podrazumijevano ili naredbama Static, Dim i Private. Naredbu Static možete postaviti na početak svakog potprograma vrste Sub ili Function, uključujući potprograme događaja te one određene kao Private.

Konstante

Često ćete vidjeti da vaš programski kod sdržava nepromjenljive vrijednosti koje se stalno pojavljuju. Možete i vidjeti da programski kod ovisi o određenim brojevima koje je teško zapamtiti – brojevima koji smi po sebi nemaju smisleno značenje.

U takvim slučajevima, znatno možete poboljšati čitljivost svog koda – i učiniti ga lakšim za održavanje – koristeći konstante. *Konstanta* je smisleno ime koje zauzima mjesto broja ili teksta koji se neće mijenjati. Iako konstanta ponekad sliči varijabli, konstantu ne možete promijeniti dodjeljujući joj novu vrijednost kao što to možete s varijablom. Postoje dva izvora za konstante:

- *Ugrađene* ili *sistemski određene* konstante dane od aplikacija i kontrola. Konstante Visual Basica popisne su u bibliotekama objekata Visual Basica (VB) i Visual Basica za aplikacije (VBA) u pretraživaču objekata. Ostale aplikacije koje imaju biblioteke objekata, kao što su Microsoft Excel i Microsoft Project, također daju popis konstanti koje možete iskoristiti s njihovim objektima, postupcima i svojstvima. Konstante su također određene u biblioteci objekata za svaku ActiveX kontrolu. Za detalje o korištenju pretraživača objekata, pogledajte 9. poglavlje “Programiranje objektima”.
- *Simbolične* ili *korisnički određene* konstante se postavljaju korištenjem naredbe Const. Korisnički određene konstante su opisne u sljedećem odlomku “Stvaranje vlastitih konstanti”.

Imena konstanti u Visual Basicu sstoje se od kombiniranih malih i velikih slova, s prefiksom koji ukazuje na biblioteku objekata koja određuje konstantu. Imena konstanti iz Visual Basica i Visual Basica za aplikacije počinju prefiksom “vb” – na primjer, vbTileHorizontal.

Svrha prefiks je sprečavanje slučajnih sukoba u slučajevima gdje konstante imaju isto ime i sdrže različite vrijednosti. Čak i s prefiksima, još uvijek je moguće da dvije biblioteke objekata mogu sdržavati istoimene konstante različitih vrijednosti. Koja konstanta će u kojem slučaju biti pozvana ovisi o tome koja biblioteka objekata ima veću prednost.

Da biste bili potpuno sigurni u zaobilaženju sukoba imena konstanti, pozive konstanti možete odrediti sljedećom sintaksom:

[*imebiblioteke*.] [*imemodula*.] *imekonstante*

Imebiblioteke je obično ime kontrole ili biblioteke. *Imemodula* je ime modula koji određuje konstantu. *Imekonstante* je ime konstante. Svaki od ovih elemenata je određen u biblioteci objekata i može se vidjeti u pretraživaču objekata.

Stvaranje vlastitih konstanti

Sintaks određivanja konstante je sljedeća:

[**Public**|**Private**] **Const** *imekonstante* [**As** *tip*] = *izraz*

Argument *imekonstante* je ispravno simboličnо ime (pravila su ista kao i kod stvaranja imena varijabli), a *izraz* se sstoji od broja ili teksta te operatora; unatoč tome, u *izrazu* ne možete koristiti pozive funkcija.

Naredba Const može određivati matematički izraz ili izraz datum/vrijeme:

```
Const conPi = 3.14159265358979
Public Const conMaxPlaneta As Integer = 9
Const conDatumObjave = #1/1/95#
```

Naredba Const može također biti iskorištena za određivanje tekstualnih konstanti:

```
Public Const conVerzija = “07.10.A”
Const conImeKoda = “Enigma”
```

U jednu liniju možete postaviti više od jednog određivanja konstante ako ih odvojite zarezima:

```
Public Const conPi = 3.14, conMaxPlaneta = 9, _
conPopSvijeta = 6E+09
```

Izraz na desnoj strani znaka jednakosti (=) je često broj ili tekstualni niz (string), ali može također biti i izraz rezultat kojeg će biti broj ili tekst (iako taj izraz ne može sdržavati pozive funkcija). Možete čak i odrediti konstante pozivom prethodno određenih konstanti:

```
Const conPi2 = conPi * 2
```

Jednom kad odredite konstante, možete ih postaviti u svoj programski kod kako bi učinili čitljivijim. Na primjer:

```
Static Sun~evSustav(1 To conMaxPlaneta)
If numLjudi > conPopSvijeta Then Exit Sub
```

Područje korisnički određenih konstanti

Naredba Const ima područje djelovanja kao i određivanje varijable, te se pojavljuju ista pravila:

- Za kreiranje konstante koja postoji smo unutar potprograma, odredite ju unutar tog potprograma.
- Za kreiranje konstante dostupne svim potprogramima unutar modula, ali ne i programskom kodu izvan tog modula, odredite konstantu unutar odjeljka Declarations modula.
- Za kreiranje konstante dostupne cijeloj aplikaciji, odredite konstantu unutar odjeljka Declarations standardnog modula, i postavite naredbu Public prije naredbe Const. Javne varijable ne mogu biti određene u formi ili modulu klase.

Za više informacija Za dodatne informacije o području, pogledajte "Razumijevanje područja varijabli" ranije u ovom poglavlju.

Zaobilaženje kružnog odnos

Budući da konstante mogu biti određene koristeći druge konstante, morate paziti da ne uspostavite *krug*, kružni odnos između dvije ili više konstanti. Krug se pojavljuje kad imate dvije ili više javnih konstanti, a svaka od njih je određena pozivom na drugu.

Na primjer:

```
‘ U modulu 1:
Public Const conA = conB * 2      ‘ Dostupna cijeloj
                                  ‘ aplikaciji.
‘ U modulu 2:
Public Const conB = conA / 2      ‘ Dostupna cijeloj
                                  ‘ aplikaciji.
```

Ako se pojavi krug, Visual Basic stvara pogrešku kad pokušate pokrenuti takvu aplikaciju. Ne možete pokrenuti svoj programski kod sve dok ne riješite kružni odnos. Kako bi zaobišli mogućnost stvaranja kruga, ograničite sve svoje javne konstante na jedan modul ili, barem, na manji broj modula.

Tipovi podataka

Varijable su oznake mjesta koje se koriste za čuvanje vrijednosti; imaju imena i tip podataka. Tip podatka varijable određuje kako će u računalnoj memoriji biti posloženi bitovi koji sdrže podatak. Kad odredite varijablu, možete joj također dodati i tip podatka. Sve varijable imaju tip podatka koji određuje kakav podatak može biti pohranjen u nju.

U pravilu, ako ne odredite tip podatka, varijabla će biti određena tipom podatka Variant. Tip podatka Variant je kao kameleon – može sdržavati puno različitih vrsta podataka u različitim situacijama. Ne morate pretvarati različite tipove podataka u određeni tip kad ih dodjeljujete varijabli Variant; Visual Basic automatski obavlja sve potrebne pretvorbe.

Ako znate da će varijabla uvijek sdržavati podatke određenog tipa, Visual Basic može efikasnije obrađivati te podatke ako varijabli odredite tip podatka. Na primjer, varijabla koja treba čuvati ime neke osobe se najbolje obrađuje kao tekstualni tip podatka, jer je njezin sdržaj uvijek sstavljen od znakova.

Tipovi podatka pripadaju i drugim stvarima osim varijablama. Kad dodijelite vrijednost svojstvu, ta vrijednost ima tip podatka; argumenti funkcija također imaju tipove podataka. Zapravo, gotovo sve u Visual Basicu što uključuje podatke također uključuje i tip podatka.

Možete odrediti i matrice bilo kojeg osnovnog tipa.

Za više informacija Za dodatne informacije, pogledajte odlomak "Matrice" kasnije u ovom poglavlju. Odabir tipa podataka za poboljšanje izvođenja aplikacije je raspravljen u 15. poglavlju "Oblikovanje u korist izvođenja i sukladnosti".

Određivanje varijable s tipom podatka

Prije nego što odredite varijablu koja nije promjenljivog tipa Variant, morate upotrijebiti naredbe Private, Public, Dim ili Static za izraz As *tip*. Na primjer, sljedeći izrazi određuju redom varijable tipa Integer, Double, String i Currency:

```
Private I As Integer
Dim Amt As Double
Static VašeIme As String
Public Plaćeno As Currency
```

Linija odjeljka Declarations može sadržavati više određivanja, kao u ovim linijama:

```
Private I As Integer, Amt As Double
Private VašeIme As String, Plaćeno As Currency
Private Test, Iznos, J As Integer
```

Napomena Ako ne odredite tip podatka, varijabla će imati predodređeni tip. U prethodnom primjeru, varijable Test i Iznos bit će tipa Variant. To vas može iznenaditi ako na temelju vašeg iskustva s drugim programskim jezicima očekujete da će sve varijable u naredbi određivanja imati isti određeni tip (u ovom slučaju, Integer).

Brojčani tip podatka

Visual Basic ima nekoliko brojčanih tipova podataka – Integer (cijeli broj), Long (dugi cijeli broj), Single (jednostruka preciznost pomičnog zareza), Double (dvostruka preciznost pomičnog zareza) i Currency (valuta). Korištenje brojčanog tipa podatka općenito zahtjeva manje mjesta za pamćenje nego varijabla općeg tipa.

Ako znate da će varijabla uvijek sdržavati cijele brojeve (na primjer 12), a ne brojeve s decimalnim vrijednostima (na primjer 3.57) odredite ju kao tip Integer ili Long. Računanja s cijelim brojevima su brža, i ovakvi tipovi varijabli zauzimaju manje memorije od ostalih tipova podataka. One su posebno iskoristive kao varijable brojača u petljama For...Next.

Za više informacija Kako bi doznali više o strukturama kontrole, pročitajte odlomak "Uvod u strukturu kontrola" kasnije u ovom poglavlju.

Ako varijabla sdrži decimalni dio, odredite ju kao varijablu Single, Double ili Currency. Tip podatka Currency podržava do četiri znamenke desno od decimalnog zareza i petnaest znamenki lijevo; to je precizan tip podatka s nepomičnim zarezom prikladan za proračune s valutama. Brojevi s pomičnim zarezom (Single i Double) imaju daleko veći opseg od tipa Currency, ali mogu biti predmet manjih pogrešaka kod zaokruživanja.

Napomena Vrijednosti s pomičnim zarezom mogu biti izražene kao *mmm***E***eee* ili *mmm***D***eee*, gdje *mmm* predstavlja mantisu, a *eee* eksponent (potenciju broja 10). Najveća moguća vrijednost tipa Single je 3.402823E+38 ili 3.4 puta 10 na 38-u potenciju; najveća moguća vrijednost tipa Double je 1.79769313486232D+308, ili oko 1.8 puta 10 na 308-u potenciju. Korištenje slova **D** za razdvajanje mantise i eksponenta u brojčanoj vrijednosti uzrokuje obradu varijable kao da je tipa Double. Slično tome, korištenje slova **E** na isti način obrađuje varijablu kao da je tipa Single.

Tip podatka Byte

Ako varijabla sdrži binarne podatke, odredite je kao matricu tipa Byte (o matricama se raspravlja u odlomku "Matrice" kasnije u ovom poglavlju). Korištenje varijabli tipa Byte za spremanje binarnih podataka čuva podatke kod pretvaranja oblika. Kad se varijable tipa String pretvaraju između ANSI i Unicode oblika, pokvarit će se svaki binarni podatak u varijabli. Visual Basic može automatski vršiti pretvorbu između ANSI i Unicode standarda kad:

- čita iz datoteka.
- piše u datoteke.
- poziva dinamičke biblioteke (DLL)
- poziva postupke i svojstva objekata.

Svi operatori koji rade s cijelim brojevima radit će i s podacima tipa Byte osim u slučajevima oduzimanja. Budući da je tip Byte nedostupan izvan opsega 0-255, ne može imati negativnu vrijednost. Zbog toga, kod oduzimanja, Visual Basic prisiljava varijablu tipa Byte da zadrži pozitivnu vrijednost.

Sve brojčane varijable mogu biti dodjeljivane međusobno kao i varijablama tipa Variant. Visual Basic će radije zaokružiti nego odbaciti decimalni dio brojeva s pomičnim zarezom prije nego što ih dodijeli cjelobrojnoj vrijednosti.

Za više informacija Za detalje o pretvaranjima po standardima Unicode i ANSI, pogledajte 16. poglavlje "Međunarodna izdanja".

Tip podatka String

Ako imate varijablu koja će uvijek sdržavati tekst, a nikad brojčanu vrijednost, možete je odrediti tipom String:

```
Private S As String
```

Nakon toga možete varijabli dodijeliti stringove (slovne nizove) i upravljati njima koristeći funkcije stringa:

```
S = "Database"
S = Left(S, 4)
```

U pravilu, tekstualna varijabla ili argument je *string promjenjive duljine*; on raste ili se smanjuje kako mu dodjeljujete nove podatke. Također možete odrediti i stringove koji imaju određenu duljinu. *String stalne duljine* možete odrediti sljedećom sintaksom:

String * *veličina*

Na primjer, za određivanje stringa koji će uvijek biti dug 50 karaktera, upotrijebite programski kod sličan ovom:

```
Dim ProIme As String * 50
```

Ako varijabli ProIme dodijelite string s manje od 50 karaktera, ona će se popuniti potrebnim brojem razmaka tako da bude duljine 50 karaktera. Ako dodijelite string koji je predugačak za string stalne duljine, Visual Basic će jednostavno odbaciti višak karaktera.

Budući da se stringovi stalne duljine popunjavaju razmacima, funkcije Trim i RTrim, koje miču razmake, mogle bi vam biti korisne pri radu s takvim stringovima.

Nizovi stalne duljine u standardnim modulima mogu biti određeni kao Public ili Private. U formama i modulima klase, stringovi stalne duljine moraju biti određeni kao Private.

Za više informacija Pogledajte "Funkcije LTrim, RTrim i Trim" u djelu *Microsoft Visual Basic 6.0 Language Reference* biblioteke *Microsoft Visual Basic 6.0 Reference Library*.

Razmjena stringova i brojeva

Možete dodijeliti string brojčanoj vrijednosti ako string sdrži brojčanu vrijednost. Također je moguće dodijeliti brojčanu vrijednost varijabli stringa. Na primjer, postavite na formu naredbeni gumb, okvir s tekstom i okvir s popisom. Upišite sljedeći programski kod u događaj Click naredbenog gumba. Pokrenite aplikaciju i kliknite na naredbeni gumb.

```
Private Sub Command1_Click()
    Dim intX As Integer
    Dim strY As String
    strY = "100.23"
    intX = strY            ' Prosljeđivanje stringa
                           ' brojčanoj varijabli.
    List1.AddItem Cos(strY)        ' Ispisuje kosinus broja u
                                   ' stringu u okvir s popisom.
    strY = Cos(strY)               ' Prosljeđuje kosinus
                                   ' u varijablu stringa.
    Text1.Text = strY              ' Varijabla stringa se
                                   ' ispisuje u okviru s tekstom.
End Sub
```

Visual Basic će automatski prilagoditi varijable prikladnom tipu podatka. Potrebno je pripaziti kod razmjene stringova i brojeva; prosljeđivanje vrijednosti koja nije brojčana u string uzrokovat će pojavu pogreške tijekom izvođenja.

Tip podatka Boolean

Ako imate varijablu koja će sdržavati jednostavnu točno/netočno, da/ne ili uključeno/isključeno informaciju, možete odrediti da bude tipa Boolean. Podrazumijevana vrijednost varijable tipa Boolean je False. U sljedećem primjeru, blnKretanje je varijabla tipa Boolean koja sdrži jednostavnu vrijednost da/ne.

```
Dim blnKretanje As Boolean
    ‘ Provjera kreće li se traka
    If Recorder.Direction = 1 Then
    blnKretanje = True
End If
```

Tip podatka Date

Vrijednosti datuma i vremena mogu biti sdržane kako u posebnom tipu podatka Date tako i u varijablama tipa Variant. Kod oba tipa pojavljuju se iste općenite osobine.

Za više informacija Pogledajte odlomak “Vrijednosti datuma i vremena spremljene u tipu Variant”, u dijelu “Napredne teme o tipu Variant”, kasnije u ovom poglavlju.

Kad se ostali brojčani tipovi podataka pretvaraju u tip Date, vrijednosti lijevo od decimalne točke predstavljaju podatak datuma, dok vrijednosti s desne strane decimalne točke predstavljaju vrijeme. Ponoć je 0, a podne 0.5. Negativni cijeli brojevi predstavljaju datume prije 30. prosinca 1899.

Tip podatka Object

Varijable tipa Object spremaju se kao 32-bitne (4 bajta) adrese koje označuju objekte unutar vaše ili neke druge aplikacije. Varijabla određena kao Object je ona koja kasnije može biti određena (koristeći naredbu Set) tako da pokazuje na bilo koji postojeći objekt prepoznat od aplikacije.

```
Dim objDb As Object
Set objDb = OpenDatabase(“c:\Vb6\Biblio.mdb”)
```

Kad određujete varijable tipa Object, pokušajte koristiti određene klase (kao TextBox umjesto Control ili, u prethodnom slučaju, Database umjesto Object) umjesto općenitog Object. Visual Basic može riješiti pokazivače na svojstva i postupke objekata određenog tipa prije pokretanja aplikacije. To omogućuje aplikaciji da brže radi tijekom vremena izvođenja. Specifične klase su ispisne u pretraživaču objekata.

Kad radite s objektima drugih aplikacija, umjesto korištenja tipa Variant ili općenitog tipa Object, odredite objekte onako kako su ispisni u popisu klas unutar pretraživača objekata. Na ovaj način Visual Basic će s sigurnošću prepoznati određeni tip objekta na koji ukazujete, omogućujući pokazivaču da bude obrađen tijekom rada aplikacije.

Za više informacija Za dodatne informacije o stvaranju i dodjeljivanju objekata i objektnih varijabli pogledajte “Kreiranje objekata” kasnije u ovom poglavlju.

Pretvaranje tipova podataka

Visual Basic pruža nekoliko funkcija pretvaranja koje možete koristiti za pretvaranje vrijednosti u određeni tip podatka. Za pretvaranje vrijednosti u tip Currency, na primjer, koristit ćete funkciju CCur.

```
PlaćaZaTjedan = CCur(sti * stnica)
```

funkcija pretvaranja	pretvara izraz u tip
CBool	Boolean
CByte	Byte
CCur	Currency
CDate	Date
CDbl	Double
CInt	Integer
CLng	Long
CSng	Single
CStr	String
CVar	Variant
CVErr	Error

Napomena Vrijednosti koje prosljeđujete funkcijama za pretvorbu moraju biti ispravne za tip podatka u koji će biti pretvorene ili će se pojaviti pogreška. Na primjer, ako pokušate pretvoriti tip Long u tip Integer, Long mora biti unutar opsega za tip podatka Integer.

Za više informacija Pogledajte *Microsoft Visual Basic 6.0 Language Reference* za specifične funkcije pretvaranja.

Tip podatka Variant

Varijabla tipa Variant je sposobna spremiti sve sistemski određene tipove podataka. Ne morate pretvoriti između tih tipova podataka ako ih dodjeljujete varijabli tipa Variant; Visual Basic automatski izvodi svaku potrebnu pretvorbu. Na primjer:

```
Dim NekaVrijednost                ‘ podrazumijevano Variant
NekaVrijednost = “17”             ‘ NekaVrijednost sadržava “17”
                                  ‘ (dvoslovni string)
NekaVrijednost = NekaVrijednost - 15     ‘ NekaVrijednost sd sadrži
                                         ‘ brojčanu vrijednost 2.
NekaVrijednost = “U” & NekaVrijednost    ‘ NekaVrijednost sd sadrži
                                         ‘ “U2” (dvoslovni string)
```

Iako možete izvoditi ovakve radnje na varijablama tipa Variant bez puno brige o tipu podatka koji sdrže, postoje neke zamke koje bi trebali zaobići.

- Ako izvodite aritmetičke operacije ili funkcije na varijabli tipa Variant, ona mora sdržavati nekakvu brojčanu vrijednost. Za detalje pogledajte dio "Brojčane vrijednosti spremljene u tipu Variant" u odlomku "Napredne teme o tipu Variant", kasnije u ovom poglavlju.
- Ako spajate stringove, upotrijebite operator & umjesto operatora +. Za detalje pogledajte dio "Nizovi spremljeni u tipu Variant", u odlomku "Napredne teme o tipu Variant", kasnije u ovom poglavlju.

Kao dodatak mogućnostima da djeluje kao ostali uobičajeni tipovi podataka, tip Variant može sdržavati i tri posebne vrijednosti: Empty, Null i Error.

Vrijednost Empty

Ponekad trebate znati je li kreiranoj varijabli ikad dodana neka vrijednost. Varijabla tipa Variant ima vrijednost Empty prije nego što joj se neka vrijednost doda. Vrijednost Empty je posebna vrijednost različita od nule (0), praznog stringa ("") ili vrijednosti Null. Funkcijom IsEmpty možete ispitati ima li neka varijabla vrijednost Empty:

```
If IsEmpty(Z) Then Z = 0
```

Kad varijabla tipa Variant sdrži vrijednost Empty, možete je koristiti u izrazima; obrađuje se kao 0 ili prazni string, ovisno o izrazu.

Vrijednost Empty nestaje čim je neka vrijednost (uključujući 0, prazni string, ili Null) dodijeljena varijabli tipa Variant. Varijabli tipa Variant možete opet dodijeliti vrijednost Empty dodjeljivanjem ključne riječi Empty varijabli Variant.

Vrijednost Null

Podatak tipa Variant može sdržavati još jednu posebnu vrijednost: Null. Vrijednost Null se obično koristi aplikacijama baza podataka za ukazivanje na nepoznat ili nepostojeći podatak. Zbog načina na koji se koristi u bazama podataka, vrijednost Null ima neke jedinstvene osobine:

- Izrazi koji uključuju vrijednost Null uvijek će kao rezultat dati vrijednost Null. Za ovu vrijednost se kaže da se "širi" kroz izraze; ako bilo koji dio izraza dobije vrijednost Null, cijeli izraz će rezultirati vrijednošću Null.
- Prosljeđivanje vrijednosti Null, varijable tipa Variant s ovom vrijednošću, ili izraza koji rezultira ovom vrijednošću, kao argumenata većine funkcija uzrokovat će rezultatom funkcije s vrijednošću Null.
- Vrijednosti Null šire se kroz ugrađene funkcije koje vraćaju podatke tipa Variant.

Nekoj varijabli možete dodijeliti vrijednost Null uz pomoć ključne riječi Null:

```
Z = Null
```

Funkcijom IsNull možete ispitati sdrži li neka varijabla tipa Variant vrijednost Null:

```
If IsNull(X) And IsNull(Y) Then
   Z = Null
Else
   Z = 0
End If
```

Ako varijabli bilo kojeg tipa osim tipa Variant dodijelite vrijednost Null, doći će do uhvatljive pogreške. Dodjela vrijednosti Null varijabli tipa Variant neće uzrokovati pogrešku, a vrijednost Null će se širiti kroz izraze utječući na varijable tipa Variant (iako se vrijednost Null ne širi kroz neke funkcije). Kao rezultat bilo kojeg potprograma Function možete dobiti vrijednost Null ako je rezultat spremljen u varijabli tipa Variant.

Varijable nemaju vrijednost Null osim ako im je izričito ne dodijelite, pa ako u vašoj aplikaciji ne koristite vrijednost Null, ne morate pisti programski kod koji će ispitivati i obrađivati tu vrijednost.

Za više informacija Za informacije o korištenju vrijednosti Null u izrazima, pogledajte odlomak "Null" u biblioteci *Microsoft Visual Basic 6.0 Language Reference*.

Vrijednost Error

U varijabli tipa Variant, vrijednost Error je posebna vrijednost koja se koristi za ukazivanje na stanje pogreške koja se pojavila u potprogramu. Unatoč tome, za razliku od svih ostalih vrsta pogrešaka, neće se pojaviti uobičajena obrada pogreške na razini aplikacije. To vam dozvoljava, ili smoj aplikaciji, poduzimanje drugih postupaka temeljenih na vrijednosti pogreške. Vrijednosti pogrešaka stvaraju se pretvaranjem stvarnih brojeva u vrijednosti pogreške korištenjem funkcije CVErr.

Za više informacija Za informacije o korištenju vrijednosti Error u izrazima, pogledajte dio "Funkcija CVErr" u biblioteci *Microsoft Visual Basic 6.0 Language Reference*. Za informacije o obradi pogreške, pogledajte 13. poglavlje "Traženje i obrada pogrešaka". Za dodatne informacije o podatku tipa Variant, pogledajte sljedeći odlomak "Napredne teme o tipu Variant".

Napredne teme o tipu Variant

Unutarnje predstavljanje vrijednosti u tipu Variant

Varijable tipa Variant sdrže unutarnji prikaz vrijednosti koje spremaju. Taj prikaz određuje kako će Visual Basic tretirati te vrijednosti pri izvođenju uspoređivanja i drugih operacija. Kad varijabli tipa Variant dodijelite vrijednost, Visual Basic koristi najsažetiji mogući prikaz koji točno pamti vrijednost. Kasnije radnje mogu biti povod Visual Basicu za promjenu prikaza koji koristi za određenu varijablu (varijabla tipa Variant nije varijabla bez tipa; prije je to varijabla koja može slobodno mijenjati tip). Ovakvi unutarnji prikazi odgovaraju jasno određenim tipovima podataka raspravljenim u odlomku "Tipovi podataka" ranije u ovom poglavlju.

Napomena Varijabla tipa Variant uvijek zauzima 16 bajtova, neovisno o podatku koji čuva. Objekti, stringovi i matrice nisu stvarno spremljeni u varijablu tipa Variant; kod takvih slučajeva, koriste se četiri bajta varijable za čuvanje poziva objekta, pokazivača na string ili matricu. Stvarni podaci su spremljeni na drugom mjestu.

Uglavnom ne trebate pridavati važnost unutarnjem prikazu koji Visual Basic koristi za određenu varijablu; Visual Basic barata pretvorbama automatski. Ako unatoč tome želite sznati koji tip vrijednosti Visual Basic koristi, možete upotrijebiti funkciju VarType.

Na primjer, ako u varijablu tipa Variant spremite brojčanu vrijednost s decimalnim dijelom, Visual Basic će uvijek koristiti unutarnji prikaz tipa Double. Ako znate da vaša aplikacija ne treba visoku točnost (i manju brzinu) koju pruža vrijednost tipa Double, možete ubrzati svoje proračune pretvorbom te vrijednosti u tip Single, ili čak u tip Currency:

```
If VarType(X) = 5 Then X = CSng(X)       ‘ Pretvorba u tip Single.
```

Kod matričnih varijabli, vrijednost izraza VarType je sžetak povratnih vrijednosti tipa matrice i podataka. Na primjer, sljedeća matrica sdrži vrijednosti tipa Double:

```
Private Sub Form_Click()
    Dim dblPrimjer(2) As Double
    MsgBox VarType(dblPrimjer)
End Sub
```

Sljedeće verzije Visual Basica možda će imati dodatne prikaze tipa Variant, pa bi sv programski kod koji napišete za stvaranje izbora ovisnih o povratnoj vrijednosti funkcije VarType trebao bez problema obraditi povratne vrijednosti koje trenutno nisu određene.

Za više informacija Za informacije o funkciji VarType, pogledajte odlomak “Funkcija VarType” u biblioteci *Microsoft Visual Basic 6.0 Language Reference*. Da biste sznali više o matricama, pogledajte odlomak “Matrice” kasnije u ovom poglavlju. Za detalje o pretvorbi tipova podataka, pogledajte odlomak “Tipovi podataka” ranije u ovom poglavlju.

Brojčane vrijednosti spremljene u tipu Variant

Kad u varijable tipa Variant spremite cijeli broj, Visual Basic koristi najsžetiji mogući prikaz. Na primjer, ako spremite mali broj bez decimalnog dijela, tip Variant koristi za tu vrijednost prikaz Integer. Ako zatim varijabli dodijelite veći broj, Visual Basic će upotrijebiti prikaz Long, ili, ako je riječ o vrlo velikom broju s decimalnim dijelom, prikaz Double.

Ponekad ćete poželjeti upotrijebiti određeni prikaz za neki broj Na primjer, možda poželite da varijabla tipa Variant sčuva brojčanu vrijednost s prikazom Currency kako bi zaobišli pogreške zaokruživanja u kasnijim proračunima. Visual Basic pruža nekoliko funkcija pretvorbe koje možete iskoristiti za pretvaranje vrijednosti u određeni tip (pogledajte odlomak “Pretvaranje tipova podataka” ranije u ovom poglavlju). Za pretvorbu vrijednosti u tip Currency, na primjer, možete upotrijebiti funkciju CCur:

```
Pla□aZaTjedan = CCur(sti * stnica)
```

Pojavit će se pogreška ako pokušate izvesti matematičku operaciju ili funkciju na varijabli tipa Variant ako ona ne sdrži broj ili nešto što bi se moglo protumačiti kao broj. Na primjer, ne možete izvesti nikakvu aritmetičku operaciju s vrijednošću U2 iako ona sdržava brojčani karakter, budući da cijela vrijednost nije valjan broj. Slično tome, ne možete izvesti nikakvu operaciju s vrijednošću 1040EZ; međutim, možete izvoditi proračune s vrijednostima +10 ili -1.7E6 jer su to valjani brojevi. Zbog ovog razloga, često ćete željeti odrediti da varijabla tipa Variant sdrži vrijednost koja može biti iskorištena kao broj. Takav zadatak izvodi funkcija IsNumeric:

```
Do
   nekiBroj = InputBox(“Upišite broj”)
Loop Until IsNumeric(nekiBroj)
MsgBox “Korijen je: “ & Sqr(nekiBroj)
```

Kad Visual Basic pretvara prikaz koji nije brojčan (kao string koji sdrži broj) u brojčanu vrijednost, koristi lokalne postavke (određene u kontrolnom panelu Windows) za ispis separatora tisućica, decimalnog separatora te simbola valute.

Prema tome, ako su lokalne postavke u kontrolnom panelu Windows postavljene za Sjedinjene Američke Države, Kanadu ili Australiju, sljedeća dva izraza će biti točna:

```
IsNumeric(“$100”)
IsNumeric(“1,560.50”)
```

Dok će ova dva izraza biti netočna:

```
IsNumeric(“DM100”)
IsNumeric(“1.560,00”)
```

Međutim, slučaj bi bio obrnut – prva dva izraza bi bila netočna, a druga dva točna – ako su regionalne postavke u kontrolnom panelu Windows postavljene za Njemačku.

Ako varijablu tipa Variant koja sdrži broj dodijelite varijabli stringa ili svojstvu, Visual Basic automatski pretvara prikaz broja u tekst. Ako izričito želite pretvoriti broj u tekst, upotrijebite funkciju CStr. Možete također iskoristiti i funkciju Format za pretvaranje broja u tekst koji uključuje oblikovanja kao što su simboli valute, separatora tisućica i decimalnog separatora. Funkcija Format automatski koristi prikladne simbole određene u dijaloškom okviru Regional Settings Properties kontrolnog panela Windows.

Za više informacija Pogledajte odlomak “Funkcija Format” i teme o funkcijama pretvaranja u biblioteci *Microsoft Visual Basic 6.0 Language Reference*. Za informacije o pisnju programskog koda za aplikacije koje će biti korištene u drugim zemljama, pogledajte 16. poglavlje “Međunarodna izdanja”.

Stringovi spremljeni u tipu Variant

Općenito, čuvanje i korištenje stringova u varijablama tipa Variant postavlja nekoliko problema. Kao što je prije spomenuto, rezultat operatora + ipak ponekad može biti nepouzdan ako se koristi s dvije varijable tipa Variant. Ako obje varijable tipa Variant sdrže brojeve, operator + obavit će zbrajanje. Ako obje varijable tipa Variant sdrže stringove, tada će operator + spojiti stringove u niz. Međutim, ako je jedna vrijednost predstavljena kao broj, a druga kao string, situacija postaje zamršenija. Visual Basic prvo pokušava pretvoriti string u broj. Ako je pretvorba uspješna, operator + zbraja dvije vrijednosti; ako nije, doći će do pojave pogreške Type Mismatch.

Kako bi bili sigurni da je došlo do povezivanja vrijednosti, upotrijebite operator &, neovisno o prikazu vrijednosti u varijabli. Na primjer, sljedeći programski kod:

```
Sub Form_Click()
    Dim X, Y
    X = “6”
    Y = “7”
    Print X + Y, X & Y
    X = 6
    Print X + Y, X & Y
End Sub
```

predočit će ovaj rezultat na formi:

```
67    67
13    67
```

Napomena Visual Basic interno pamti stringove u standardu Unicode. Za više informacija o standardu Unicode, pogledajte 16. poglavlje “Međunarodna izdanja”.

Vrijednosti datuma i vremena spremljene u tipu Variant

Varijable tipa Variant također mogu sdržavati vrijednosti tipa datum/vrijeme. Nekoliko funkcija vraća vrijednosti takvog tipa. Na primjer, uz pomoć funkcije DateSerial možemo izračunati broj preostalih dana u godini:

```
Private Sub Form_Click()
    Dim upravosd, ostalodana, ostalosti, ostalominuta
    upravosd = Now         ‘ Now vraća trenutni datum i vrijeme
    ostalodana = Int(DateSerial(Year(upravosd) _
    + 1, 1, 1) - upravosd)
    ostalosti = 24 - Hour(upravosd)
    ostalominuta = 60 - Minute(upravosd)
    Print ostalodana & “ dana do kraja godine.”
    Print ostalosti & “ sti do kraja dana.”
    Print ostalominuta & “ minuta do kraja sta.”
End Sub
```

Na vrijednostima tipa datum/vrijeme možete obavljati i matematičke operacije. Dodavanje ili oduzimanje cijelih brojeva dodat će ili oduzeti dane; dodavanje ili oduzimanje razlomaka dodat će ili oduzeti vrijeme. Prema tome, dodajući 20 dodat ćete 20 dana, a ako oduzmete 1/24 oduzet ćete jedan st.

Opseg datuma spremljenih u varijabli tipa Variant je od 1. siječnja 0100. godine do 31. prosinca 9999. godine. Proračuni s datumima, međutim, ne uzimaju u obzir prerade kalendara prije prelaska na gregorijanski kalendar, pa će proračun čiji rezultat daje datum prije godine u kojoj je gregorijanski kalendar usvojen (1752. godina u Velikoj Britaniji; prije ili poslije u drugim zemljama) biti neispravan.

U vašem programskom kodu možete koristiti vrijednosti datum/vrijeme u tekstualnom obliku ako ih zatvorite simbolom ogradice (#), na isti način kako tekstualne stringove zatvarate navodnicima (“”). Na primjer, možete usporediti podatak tipa Variant koji sdrži vrijednost datum/vrijeme s tekstualnim datumom:

```
If NekiDatum > #3/6/93# Then
```

Slično tome, možete usporediti vrijednost datum/vrijeme s potpunim tekstualnim datumom/vremenom:

```
If NekiDatum > #3/6/93 1:20pm# Then
```

Ako u tekstualni izraz datum/vrijeme ne uključite vrijeme, Visual Basic će vremenski dio vrijednosti postaviti na ponoć (početak dana). Ako ne uključite datum u tekstualni izraz datum/vrijeme, Visual Basic će kao datumski dio vrijednosti odrediti 30. prosinca 1899.

Visual Basic prihvaća veliku raznolikost formata datuma i vremena u tekstualnim izrazima. Sve su ovo ispravne vrijednosti datuma/vremena:

```
NekiDatum = #3-6-93 13:20#
NekiDatum = #March 27, 1993 1:20am#
NekiDatum = #Apr-2-93#
NekiDatum = #4 April 1993#
```

Za više informacija Za informacije o obradi datuma u međunarodnim formatima, pogledajte 16. poglavlje “Međunarodna izdanja”.

Na isti način kako koristite funkciju IsNumeric za utvrđivanje sdržava li varijabla tipa Variant vrijednost koja može biti smatrana ispravnom brojčanom vrijednošću, možete koristiti funkciju IsDate za utvrđivanje sdrži li varijabla tipa Variant vrijednost koja može biti smatrana ispravnom vrijednošću datum/vrijeme. Nakon toga možete upotrijebiti funkciju CDate za pretvaranje u vrijednost datum/vrijeme.

Na primjer, sljedeći programski kod funkcijom IsDate ispituje sdržaj svojstva Text okvira s tekstom. Ako to svojstvo sdrži tekst koji se može smatrati ispravnim datumom, Visual Basic pretvara tekst u datum i izračunava dane preostale do kraja godine:

```
Dim NekiDatum, ostalodana
If IsDate(Text1.Text) Then
    NekiDatum = CDate(Text1.Text)
    ostalodana = DateSerial(Year(NekiDatum) + _
    1, 1, 1) - NekiDatum
    Text2.Text = ostalodana & " dana do kraja godine."
Else
    MsgBox Text1.Text & " nije ispravan datum."
End If
```

Za više informacija Za informacije o raznim funkcijama datuma/vremena, pogledajte odlomak "Funkcije datuma" u biblioteci *Microsoft Visual Basic 6.0 Language Reference*.

Objekti spremljeni u tipu Variant

U varijable tipa Variant mogu se spremati objekti. Ovo može biti korisno kad na skladan način morate obraditi razne tipove podataka, uključujući objekte. Na primjer, svi elementi matrice moraju imati isti tip podatka. Postavljanje tipa podataka u matrici na Variant dopušta vam da u matricu spremite objekte zajedno s ostalim tipovima podataka.

Matrice

Ako ste programirali u drugim programskim jezicima, vjerojatno ste upoznati s pojmom matrica. Matrice vam omogućuju poziv na nizove varijabli koje imaju isto ime, ali koriste broj (indeks) koji ih razlikuje. U puno situacija to vam pomaže u stvaranju manjeg i jednostavnijeg programskog koda, jer možete napraviti petlje koje će djelotvorno baratati bilo kojim brojem podataka korištenjem indeksnog broja. Matrice imaju gornje i donje granice, a elementi matrice se dodiruju unutar tih granica. Budući da Visual Basic svakom indeksnom broju dodjeljuje određeni prostor, izbjegavajte određivanje većih matrica nego što je potrebno.

> **Napomena** Matrice raspravljene u ovom poglavlju su matrice varijabli, određene u programskom kodu. One se razlikuju od matrica kontrola koje određujete postavljanjem svojstva Index kontrole tijekom izrade aplikacije. Matrice varijabli su uvijek jedna cjelina; za razliku od matrica kontrola, ne možete ubacivati ni vaditi elemente iz sredine matrice.

Svi elementi matrice imaju isti tip podatka. Naravno, kad je tip podatka Variant, pojedini elementi mogu sdržavati različite vrste podataka (objekte, stringove, brojeve i tako dalje). Možete odrediti matricu bilo kojeg osnovnog tipa podatka, uključujući i korisnički definirane tipove (opisne u odlomku "Stvaranje vlastitih tipova podataka" u 8. poglavlju, Više o programiranju") i varijable objekata (opisne u 9. poglavlju "Programiranje objektima").

U Visual Basicu postoje dvije vrste matrica; *matrice stalne veličine* koje uvijek ostaju iste veličine, i *dinamičke matrice* čija se veličina može mijenjati tijekom rada aplikacije. Dinamičke matrice su detaljnije raspravljene u odlomku "Dinamičke matrice" kasnije u ovom poglavlju.

Određivanje matrica stalne veličine

Postoje tri načina za određivanje matrice stalne veličine, ovisno o području na kojem želite imati matricu:

- Za stvaranje *javne matrice* upotrijebite naredbu Public kod određivanja matrice u odjeljku Declarations modula.
- Za stvaranje *matrice na području modula* upotrijebite naredbu Private kod određivanja matrice u odjeljku Declarations modula.
- Za stvaranje *lokalne matrice* upotrijebite naredbu Private kod određivanja matrice u potprogramu.

Određivanje gornje i donje granice

Kad određujete matricu, iza njezinog imena u zagradama trebate napisti i gornju granicu. Gornja granica ne smije biti izvan opsega tipa podatka Long (od -2,147,483,648 do 2,147,483,647). Na primjer, sljedeća određivanja matrica mogu se pojaviti u odjeljku Declarations modula:

```
Dim Brojači(14) As Integer        ‘ 15 elemenata.
Dim Zbrojevi(20) As Double        ‘ 21 element.
```

Za stvaranje javne matrice jednostavno postavite Public umjesto Dim:

```
Public Brojači(14) As Integer
Public Zbrojevi(20) As Double
```

Ista određivanja unutar potprograma koriste Dim:

```
Dim Brojači(14) As Integer
Dim Zbrojevi(20) As Double
```

Prvo određivanje stvara matricu s 15 elemenata, s indeksnim brojevima od 0 do 14. Drugo određivanje stvara matricu s 21 elementom, s indeksnim brojevima od 0 do 20. Standardna donja granica je 0.

Da biste odredili donju granicu, morate je izričito navesti (kao tip podatka Long) koristeći naredbu To:

```
Dim Brojači(1 To 15) As Integer
Dim Zbrojevi(100 To 120) As String
```

U prethodnim određivanjima, indeksni brojevi matrice Brojači nalaze se u opsegu od 1 do 15, a indeksni brojevi matrice Zbrojevi u opsegu od 100 do 120.

Matrice koje sdrže druge matrice

Moguće je stvoriti matricu tipa Variant i napuniti je drugim matricama drugačijeg tipa podataka. Sljedeći programski kod stvara dvije matrice, jednu koja sdržava cijele brojeve i drugu s stringovima. Nakon toga određuje se treća matrica tipa Variant i popunjava s prethodne dvije matrice.

```
Private Sub Command1_Click()
    Dim intX As Integer           ‘ Određivanje varijable brojača.
    ‘ Određivanje i popunjavanje matrice s cijelim brojevima.
Dim brojačiA(5) As Integer
    For intX = 0 To 4
            brojačiA(intX) = 5
    Next intX
‘ Određivanje i popunjavanje matrice s stringovima.
Dim brojačiB(5) As String
    For intX = 0 To 4
            brojačiB(intX) = “pozdrav”
    Next intX
Dim arrX(2) As Variant            ‘ Određivanje nove matrice
                                  ‘ s dva elementa.
    arrX(1) = brojačiA()          ‘ Popunjavanje ove matrice
    arrX(2) = brojačiB()          ‘ s prethodne dvije matrice.
    MsgBox arrX(1)(2)             ‘ Prikaz člana svake matrice
    MsgBox arrX(2)(3)
End Sub
```

Višedimenzionalne matrice

Ponekad trebate pratiti srodne informacije u matrici. Na primjer, kako bi u matrici sčuvali oznaku svakog piksela na ekranu svog računala, trebate pokazivati na njegove X i Y koordinate. Možete to učiniti korištenjem višedimenzionalne matrice koja će čuvati vrijednosti.

U Visual Basicu možete odrediti matrice s više dimenzija. Na primjer, sljedeći izraz određuje dvodimenzionalnu matricu veličine 10 puta 10 unutar potprograma:

```
Static MatricaA(9, 9) As Double
```

Jedna ili obje dimenzije mogu biti određene s izričitim donjim granicama:

```
Static MatricaA(1 To 10, 1 To 10) As Double
```

Možete to proširiti na više od dvije dimenzije. Na primjer:

```
Dim MultiD(3, 1 To 10, 1 To 15)
```

Ovo određivanje stvara matricu koja ima tri dimenzije te veličinu 4 puta 10 puta 15. Ukupni broj elemenata je umnožak te tri dimenzije i iznosi 600.

> **Napomena** Kad matrici počnete dodavati dimenzije, dramatično se povećava ukupna memorija potrebna za matricu, pa upotrebljavajte višedimenzionalne matrice s oprezom. Budite posebno oprezni s matricama tipa Variant, jer su one veće od ostalih tipova podataka.

Korištenje petlji za upravljanje matricama

Višedimenzionalne matrice možete efikasno obrađivati korištenjem ugniježđenih petlji For. Na primjer, sljedeći izrazi dodjeljuju svakom elementu matrice MatricaA vrijednost temeljenu na položaju u matrici:

```
  Dim I As Integer, J As Integer
  Static MatricaA(1 To 10, 1 To 10) As Double
  For I = 1 To 10
   For J = 1 To 10
     MatricaA(I, J) = I * 10 + J
   Next J
Next I
```

Za više informacija Za informacije o petljama pogledajte “Strukture petlji”, kasnije u ovom poglavlju.

Dinamičke matrice

Ponekad nećete točno znati koliko velika treba biti matrica. Možda poželite imati sposobnost promjene veličine matrice tijekom rada aplikacije.

Veličina dinamičke matrice se može mijenjati u bilo kojem trenutku. Dinamičke matrice spadaju u najelastičnije i najprikladnije osobine Visual Basica, i pomažu vam da djelotvorno iskoristite memoriju. Na primjer, možete upotrijebiti veliku matricu u kratkom vremenu pa zatim osloboditi memoriju sustava smanjivanjem matrice kad je više ne koristite.

Alternativa tome je određivanje matrice s najvećom mogućom veličinom te zanemarivanje elemenata matrice koje ne trebate. Međutim, ovakav pristup, ako pretjerate, može uzrokovati gubitak memorije dostupne radnoj okolini.

Kako stvoriti dinamičku matricu

1. Odredite matricu naredbom Public (ako želite da matrica bude javna), naredbom Dim na razini modula (ako želite matricu za područje modula) ili naredbama Static ili Dim unutar potprograma (ako želite lokalnu matricu). Matricu određujete kao dinamičku dajući joj praznu listu dimenzija.

   ```
   Dim DinMatrica()
   ```

2. Postavite potreban broj elemenata naredbom ReDim.

   ```
   ReDim DinMatrica(X + 1)
   ```

Naredba ReDim može se pojaviti smo unutar potprograma. Za razliku od naredbi Dim i Static, naredba ReDim je izvršna – tjera aplikaciju da napravi neku akciju tijekom rada.

Naredba ReDim podržava istu sintaksu kao i matrice stalne veličine. Svaka naredba ReDim može mijenjati broj elemenata, kao i donje i gornje granice, svakoj dimenziji. Međutim, broj dimenzija matrice ne može biti promijenjen.

```
ReDim DinMatrica(4 To 12)
```

Na primjer, dinamička matrica Matrica1 prvo je stvorena određivanjem na razini modula:

```
Dim Matrica1() As Integer
```

Potprogram zatim dodjeljuje prostor za matricu:

```
Sub IzračunajNešto()
    .
    .
    .
    ReDim Matrica1(19, 29)
End Sub
```

Naredba ReDim ovdje prikazana dodjeljuje matricu od 20 puta 30 cijelih brojeva (ukupno 600 elemenata). Alternativno, granice dinamičke matrice mogu biti određene koristeći varijable:

```
ReDim Matrica1(X, Y)
```

> **Napomena** Možete dodijeliti stringove matricama bajtova promjenjive veličine. Matrica bajtova također može biti dodijeljena stringu promjenjive duljine. Budite svjesni da se broj bajtova u stringu mijenja ovisno o platformi. Na platformama koje koriste standard Unicode string sdržava dvostruko više bajtova nego na drugim platformama.

Čuvanje sdržaja dinamičkih matrica

Svaki put kad izvršite naredbu ReDim, sve vrijednosti trenutno spremljene u matrici su izgubljene. Visual Basic postavlja sve vrijednosti na Empty (u matrice tipa Variant), nulu (u brojčane matrice), prazni string (u matrice stringa) ili Nothing (u matrice objekata).

Ovo je korisno kad želite pripremiti neku matricu za nove podatke, ili kad želite smanjiti veličinu matrice kako bi zauzimala minimalnu memoriju. Ponekad ćete željeti promijeniti veličinu matrice bez gubljenja podataka u njoj. To možete napraviti korištenjem naredbe ReDim zajedno s naredbom Preserve. Na primjer, možete povećati matricu za jedan element bez gubljenja vrijednosti postojećih elemenata koristeći funkciju UBound za čitanje vrijednosti gornje granice:

```
ReDim Preserve DinMatrica(UBound(DinMatrica) + 1)
```

Korištenjem naredbe Preserve možete promijeniti smo gornju granicu posljednje dimenzije u višedimenzionalnoj matrici; ako pokušate promijeniti bilo koju drugu dimenziju, ili donju granicu posljednje dimenzije, pojavit će se pogreška tijekom izvođenja. Prema tome, možete koristiti programski kod kao ovaj:

```
ReDim Preserve Matrica(10, UBound(Matrica, 2) + 1)
```

Ali ne možete koristiti ovakav kod:

```
ReDim Preserve Matrica(UBound(Matrica, 1) + 1, 10)
```

Za više informacija Za informacije o dinamičkim matricama, pogledajte "Naredba Statement" u biblioteci *Microsoft Visual Basic 6.0 Language Reference*. Kako bi naučili više o matricama objekata, pogledajte 9. poglavlje "Programiranje objektima".

Uvod u potprograme

Programske zadatke možete pojednostaviti razbijanjem aplikacije u manje logične dijelove. Ti dijelovi – nazvani *potprogrami* – mogu tada postati blokovi za izgradnju koji će vam dopustiti poboljšanje i proširivanje Visual Basica.

Potprogrami su korisni za sžimanje zadataka koji se ponavljaju ili dijele, kao što su često korišteni proračuni, baratanje tekstovima ili objektima, i operacije u bazama podataka.

Postoje dvije velike prednosti programiranja s potprogramima:

- Potprogrami vam dopuštaju razbijanje vaših aplikacija u zasebne logične cjeline, a svaku od njih možete lakše ispraviti nego cijelu aplikaciju bez potprograma.
- Potprogrami korišteni u jednoj aplikaciji mogu se upotrijebiti kao blokovi za izgradnju drugih aplikacija, uz neznatne promjene ili bez promjena.

Postoji nekoliko vrsta potprograma korištenih u Visual Basicu:

- Potprogrami tipa Sub ne vraćaju vrijednost.
- Potprogrami funkcija vraćaju vrijednost.
- Potprogrami svojstava mogu vratiti i dodjeljivati vrijednosti, i postavljati pokazivače prema objektima.

Za više informacija Potprogrami svojstava obrađeni su u 9. poglavlju "Programiranje objektima".

Potprogrami tipa Sub

Potprogram tipa Sub je blok programskog koda koji se izvršava kao odgovor na događaj. Razbijanjem koda u modulu na Sub potprograme, postaje daleko lakše pronaći i mijenjati kod u vašoj aplikaciji.

Sintaks potprograma tipa Sub je:

[**Private** | **Public**] [**Static**] **Sub** *imepotprograma* (*argumenti*)
izrazi

End Sub

Svaki put kad se pozove potprogram, izvršavaju se *izrazi* između naredbi Sub i End Sub. Potprogrami tipa Sub mogu biti postavljeni u standardnim modulima, modulima klase i modulima forme. Sub potprogrami su u pravilu tipa Public u svim modulima, što znači da mogu biti pozivani bilo gdje iz aplikacije.

Argumenti potprograma slični su određivanju varijable, određujući vrijednosti koje su proslijeđene iz poziva potprograma.

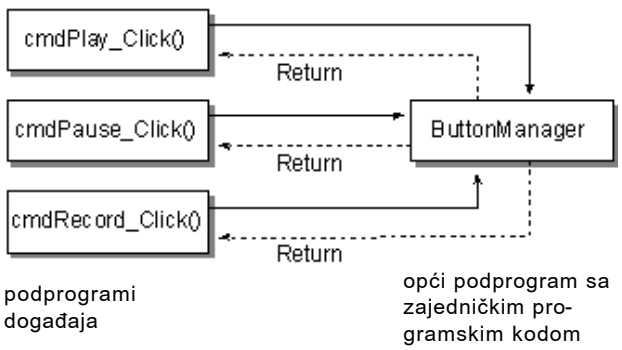
Korisno je u Visual Basicu razlikovati dva tipa Sub potprograma, *opće potprograme* i *potprograme događaja*.

Opći potprogrami

Opći potprogram kazuje aplikaciji kako izvesti određeni zadatak. Kad se jednom odredi opći potprogram, njega mora pozvati aplikacija. Suprotno tome, potprogram događaja ostaje besposlen sve dok ne bude pozvan da odgovori na događaje koje je uzrokovao korisnik ili pokrenuo sustav.

Zašto kreirati opće potprograme? Jedan razlog je moguća potreba izvođenja iste akcije u različitim potprogramima događaja. Dobra strategija programiranja je postavljanje zajedničkih izraza u odvojeni potprogram (opći potprogram) te omogućavanje potprogramima događaja da ih pozivaju. Takav postupak uklanja potrebu za umnožavanjem koda te čini aplikaciju lakšom za održavanje. Na primjer, aplikacija VCR koristi opći potprogram pozivan događajima klikanja od nekoliko različitih gumbâ za pomicanje. Slika 5.7 prikazuje korištenje općeg potprograma. Programski kod u događajima Click poziva Sub potprogram ButtonManager, koji izvodi vlastiti kod, te zatim vraća upravljanje potprogramu događaja Click.

Slika 5.7 Kako potprogrami događaja pozivaju opće potprograme



Potprogrami događaja

Kad objekt u Visual Basicu prepozna pojavljivanje događaja, automatski poziva potprogram događaja s imenom jednakim događaju. Budući da ime uspostavlja udruživanje objekta i programskog koda, za potprograme događaja se kaže da su pridruženi formama i kontrolama.

- Ime potprograma događaja kontrole sstoji se od stvarnog imena kontrole (određenog u svojstvu Name), podvlake (_) i imena događaja. Na primjer, ako želite da naredbeni gumb imena cmdPlay, kad se klikne na njega, pozove potprogram događaja, primijenite potprogram cmdPlay_Click.
- Ime potprograma forme sstoji se od riječi "Form", podvlake i imena događaja. Ako želite da forma, kad se klikne na nju, pozove potprogram događaja, upotrijebite potprogram Form_Click. Slično kontrolama, i forme imaju jedinstvena imena, ali se ona ne koriste u imenima potprograma događaja. Ako koristite MDI formu, potprogram događaja sstoji se od riječi "MDIForm", podvlake, te imena događaja, kao MDIForm_Load.

Svi potprogrami događaja koriste istu opću sintaksu.

sintaks događaja kontrole	sintaks događaja forme
Private Sub *imekontrole_imedogađaja* (*argumenti*) *izrazi* End Sub	Private Sub Form_*imedogađaja* (*argumenti*) *izrazi* End Sub

Iako potprograme događaja možete improvizirati, jednostavnije je iskoristiti potprograme pripravljene od Visual Basica, koji automatski uključuju ispravna imena potprograma. Predložak možete odabrati u kodnom prozoru odabirom objekta u okviru s popisom objekata te zatim odabirom potprograma u okviru s popisom pripadajućih potprograma.

Također je dobra ideja odrediti svojstvo Name vaših kontrola prije nego što počnete pisti njihove potprograme događaja. Ako kontroli promijenite ime nakon što ste joj dodali potprogram, morate promijeniti i ime potprograma tako da odgovara novom imenu kontrole. Inače, Visual Basic neće biti sposoban združiti kontrolu s potprogramom. Kad ime potprograma ne odgovara imenu kontrole, takav potprogram postaje opći.

Za više informacija Visual Basic prepoznaje mnoštvo potprograma za svaku vrstu forme ili kontrole. Za objašnjenja svih potprograma, pogledajte biblioteku *Microsoft Visual Basic 6.0 Langugage Reference*.

Potprogrami tipa Function

Visual Basic sdrži ugrađene ili stvarne funkcije, kao što su Sqr, Cos ili Chr. Kao dodatak, možete upotrijebiti naredbu Function za pisnje vlastitog potprograma tipa Function.

Sintaks potprograma tipa Function je:

[**Private** | **Public**] [**Static**] **Function** *imepotprograma* (*argumenti*) [**As** *tip*]
izrazi

End Function

Kao i potprogrami tipa Sub, potprogrami tipa Function su odvojeni potprogrami koji mogu prihvatiti argumente, izvesti niz izraza i promijeniti vrijednosti njegovih argumenata. Za razliku od potprograma tipa Sub, potprogrami tipa Function mogu vratiti neku vrijednost potprogramu iz kojeg su pozvani. Postoje tri razlike između potprograma tipa Sub i potprograma tipa Function:

- Općenito, funkciju pozivate postavljanjem imena potprograma funkcije s argumentima na desnu stranu veće naredbe ili izraza (*povratnavrjiednost = funkcija()*).
- Funkcijski potprogrami imaju tipove podataka, kao i varijable. Tako je određen i tip povratne vrijednosti (ako izostavite klauzulu As, tip će biti standardni tip Variant).
- Vrijednost vraćate dodjeljujući je smom imenu potprograma (*imepotprograma*). Kad potprogram tipa Function vrati vrijednost, ta vrijednost može zatim postati dio većeg izraza.

Na primjer, mogli bi napisti funkciju koja računa treću stranicu, hipotenuzu, pravokutnog trokuta, dajući joj vrijednosti za ostale dvije stranice:

```
Function Hipotenuza (A As Integer, B As Integer) As String
   Hipotenuza = Sqr(A ^ 2 + B ^ 2)
End Function
```

Potprogram tipa Function pozivate na isti način kao i bilo koju ugrađenu funkciju Visual Basica:

```
Label1.Caption = Hipotenuza(CInt(Text1.Text), _
CInt(Text2.Text))
strX = Hipotenuza(širina, visina)
```

Za više informacija Za dodatne detalje o potprogramima tipa Function, pogledajte "Naredba Function" u *Microsoft Visual Basic 6.0 Language Reference*. Tehnike pozivanja svih tipova potprograma raspravljene su u odlomku "Pozivanje potprograma", kasnije u ovom poglavlju.

Rad s potprogramima

Stvaranje novih potprograma

Kako kreirati novi opći potprogram

- Napišite naslov potprograma u kodnom prozoru i pritisnite ENTER. Naslov potprograma može biti jednostavan kao Sub ili Function s imenom. Na primjer, možete upisti bilo što od ovog:

  ```
  Sub AžurirajFormu()
  Function UzmiKoord()
  ```

  Visual Basic će na to odgovoriti ispunjavanjem predloška za novi potprogram.

Odabir postojećih potprograma

Kako vidjeti potprogram u trenutnom modulu

- Kako bi vidjeli postojeći opći potprogram, odaberite "(General)" u okviru s popisom objekata, pa odaberite potprogram u okviru s popisom potprograma.

  - ili -

  Kako bi vidjeli potprogram događaja, odaberite odgovarajući objekt u okviru s popisom objekata na vrhu kodnog prozora, pa odaberite potprogram u okviru s popisom potprograma.

Kako vidjeti potprogram u drugom modulu

1. U izborniku **View** odaberite stavku **Object Browser**.
2. Odaberite projekt u okviru **Project/Library**.
3. Odaberite modul u popisu **Classes**, pa potprogram u popisu **Members of**.
4. Odaberite **View Definition**.

Pozivanje potprograma

Tehnike pozivanja potprograma su različite, ovisno o tipu potprograma, gdje se nalazi, te kako se koristi u vašoj aplikaciji. Sljedeći odlomci opisuju kako pozvati potprograme tipa Sub i Function.

Pozivanje potprograma tipa Sub

Potprogram tipa Sub razlikuje se od potprograma tipa Function po tome što potprogram tip Sub ne možete pozvati korištenjem njegovog imena unutar izraza. Poziv Sub potprograma je smostalni izraz. Također, Sub potprogram ne vraća vrijednost u svom imenu kao što to čini funkcija. Međutim, kao i funkcije, Sub potprogrami mogu mijenjati vrijednosti bilo koje varijable koja im je proslijeđena.

Postoje dva načina pozivanja potprograma tipa Sub:

```
‘ Oba ova izraza pozivaju Sub imena MojPodpr.
Call MojPodpr (PrviArgument, DrugiArgument)
MojPodpr PrviArgument, DrugiArgument
```

Uočite da kod korištenja sintakse s naredbom Call argumenti moraju biti zatvoreni u zagrade. Ako izostavite naredbu Call, morate izostaviti i zagrade oko argumenata.

Pozivanje potprograma tipa Function

Uobičajeno, funkcijski potprogram koji ste smi napisli pozivate na isti način kako pozivate ugrađenu funkciju Visual Basica kao Abs; znači, korištenjem njegovog imena u izrazu:

```
‘ Svi sljedeći izrazi će pozvati funkciju imena ToDec
Print 10 * ToDec
X = ToDec
If ToDec = 10 Then Debug.Print “Izvan opsega”
X = DrugaFunkcija(10 * ToDec)
```

Također je moguće pozvati funkciju na isti način kao i potprogram tipa Sub. Oba sljedeća izraza će pozvati istu funkciju:

```
Call Year(Now)
Year Now
```

Kad pozivate funkciju na ovakav način, Visual Basic će odbaciti povratnu vrijednost.

Pozivanje potprograma u drugim modulima

Javni potprogrami u drugim modulima mogu biti pozvani odasvud u projektu. Možda ćete trebati odrediti modul koji sdrži pozvani potprogram. Tehnike pozivanja mogu biti različite, ovisno o tome je li potprogram postavljen u formi, klasnom ili standardnom modulu.

Potprogrami u formama

Svi pozivi izvan modula forme moraju pokazivati na modul forme u kojem se nalazi potprogram. Ako se potprogram imena NekiSub nalazi u modulu forme imena Forma1, možete taj potprogram pozvati sljedećim izrazom:

```
Call Forma1.NekiSub(argumenti)
```

Potprogrami u modulima klase

Kao pozivanje potprograma u formi, pozivanje potprograma u modulu klase zahtijeva da poziv potprograma bude označen varijablom koja pokazuje na pojavu klase. Na primjer, DemoKlas je pojava klase nazvane Klas1:

```
Dim DemoKlas As New Klas1
DemoKlas.NekiSub
```

Međutim, za razliku od forme, ime klase ne smije biti korišteno kao pokazivač kod poziva na pojavu klase. Pojava klase mora biti prvo određena kao varijabla objekta (u ovom slučaju, DemoKlas) pa onda pozvana imenom varijable.

Za više informacija Detalje o varijablama objekata i modulima klase možete pronaći u 9. poglavlju “Programiranje objektima”.

Potprogrami u standardnim modulima

Ako je ime potprograma jedinstveno, u poziv ne morate uključiti i ime modula. Poziv unutar ili izvan modula će uvijek pokazivati na taj jedinstven potprogram. Potprogram je jedinstven ako se pojavljuje smo na jednom mjestu.

Ako dva ili više modula sdrže potprograme istog imena, trebat ćete ih označiti s imenom modula. Poziv običnog potprograma iz istog modula pokrenut će potprogram u tom modulu. Na primjer, s potprogramom imena ObičnoIme u modulima Modul1 i Modul2, poziv potprograma ObičnoIme iz modula Modul2 će pokrenuti ObičnoIme u modulu Modul2, a ne ObičnoIme u modulu Modul1.

Poziv običnog potprograma iz drugog modula mora imenovati i modul u kojem se potprogram nalazi. Na primjer, ako iz modula Modul1 želite pozvati potprogram ObičnoIme u modulu Modul2, upotrijebite sljedeću sintaksu:

```
Modul2.ObičnoIme(argumenti)
```

Prosljeđivanje argumenata potprogramima

Programski kod u potprogramu obično treba neku informaciju o stanju aplikacije kako bi obavio svoj posao. Takva informacija se obično sstoji od varijabli koje se prosljeđuju potprogramu kad ga se poziva. Kad se varijabla prosljeđuje potprogramu, ona se naziva *argument*.

Tipovi podataka argumenta

Argumenti za potprograme koje pišete u pravilu imaju podatak tipa Variant. Međutim, za argumente možete odrediti i druge tipove podataka. Na primjer, sljedeća funkcija prihvaća string i broj:

```
Function ŠtoJeZaRučak(Dan As String, St As Integer) As String
    ‘ Vraća jelovnik temeljen na danu i vremenu.
    If Dan = “Petak” Then
      ŠtoJeZaRučak = “Riba”
    Else
      ŠtoJeZaRučak = “Piletina”
    End If
    If St > 4 Then ŠtoJeZaRučak = “Prekasno”
End Function
```

Za više informacija Detalji o tipovima podataka Visual Basica su dani ranije u ovom poglavlju. Također možete pogledati u biblioteku *Microsoft Visual Basic 6.0 Language Reference* za pomoć o određenom tipu podatka.

Prosljeđivanje argumenta s vrijednošću

Kad se argument proslijedi s vrijednošću prosljeđuje se smo kopija varijable. Ako potprogram promijeni vrijednost, promjena utječe smo na kopiju, a ne na smu varijablu. Iskoristite naredbu ByVal kako bi ukazali na argument kojem je dodijeljena vrijednost.

Na primjer:

```
Sub PoštanskiRačun(ByVal intBrojRačuna As Integer)
    .
    . ‘ Ovdje postavite izraze.
    .
End Sub
```

Prosljeđivanje argumenta upućivanjem

Prosljeđivanje argumenta upućivanjem daje potprogramu pristup stvarnom sadržaju varijable na položaju njezine memorijske adrese. Kao rezultat, vrijednost varijable može biti promijenjena za stalno od potprograma koje je proslijeđena. Prosljeđivanje upućivanjem je standardni način u Visual Basicu.

Ako odredite tip podatka argumenta proslijeđenog upućivanjem, u argument morate staviti vrijednost takvog tipa. Ovo načelo možete zaobići prosljeđivanjem izraza kao argumenta, radije nego podatka. Visual Basic procjenjuje izraz i prosljeđuje ga kao traženi tip ako može.

Najjednostavniji način pretvaranja varijable u izraz je zatvoriti je u zagrade. Na prim-

jer, kako bi potprogramu koji očekuje broj kao argument proslijedili varijablu određenu kao cijeli broj, možete uraditi sljedeće:

```
Sub PozivPotprograma()
    Dim intX As Integer
    intX = 12 * 3
    Napiši(intX)
End Sub
Sub Napiši(Bar As String)
    MsgBox Traka          ‘ Vrijednost varijable Traka je string “36”.
End Sub
```

Korištenje neobaveznih argumenata

Možete odrediti argumente potprograma kao neobavezne postavljanjem naredbe Optional u popisu argumenata. Ako odredite neobavezan argument, sljedeći argumenti u popisu argumenata također moraju biti neobavezni i određeni naredbom Optional. Dva primjera programskog koda koji slijede, pretpostavljaju da postoji forma s naredbenim gumbom i okvirom s popisom.

Na primjer, ovaj programski kod sdrži smo neobavezne argumente:

```
Dim strIme As String
Dim strAdres As String

Sub ListTekst(Optional x As String, Optional y As String)
    List1.AddItem x
    List1.AddItem y
End Sub

Private Sub Command1_Click()
    strIme = “vaše ime”
    strAdres = 12345      ‘ Pribavljena su oba argumenta.
    Call ListTekst(strIme, strAdres)
End Sub
```

Sljedeći programski kod, međutim, ne pribavlja sve neobavezne argumente:

```
Dim strIme As String
Dim strAdres As Variant
Sub ListTekst(x As String, Optional y As Variant)
    List1.AddItem x
    If Not IsMissing(y) Then
     List1.AddItem y
    End If
End Sub
```

```
Private Sub Command1_Click()
    strIme = "vaše ime"    ' Drugi argument nije
                           ' pribavljen.
    Call ListTekst(strIme)
End Sub
```

U slučaju kad neobavezni argument nije pribavljen, kao argument se postavlja varijabla tipa Variant s vrijednošću Empty. Prethodni primjer pokazuje kako provjeriti nedostaje li neki neobavezni argument korištenjem funkcije IsMissing.

Pribavljanje standarda za neobavezni argument

Također je moguće odrediti standardnu vrijednost za neobavezni argument. Sljedeći primjer vraća standardnu vrijednost ako neobavezni argument nije proslijeđen potprogramu funkcije:

```
Sub ListTekst(x As String, Optional y As Integer = 12345)
    List1.AddItem x
    List1.AddItem y
End Sub

Private Sub Command1_Click()
    strIme = "vaše ime"    ' Drugi argument nije
                           ' pribavljen.
    Call ListTekst(strIme)         ' Dodaje "vaše ime" i
                                   ' "12345".
End Sub
```

Korištenje neodređenog broja argumenata

Općenito, broj argumenata u pozivu funkcije mora biti isti kao i u određivanju potprograma. Korištenje naredbe ParamArray omogućuje vam da odredite kako će potprogram prihvatiti neodređen broj argumenata. To vam dopušta pisnje funkcija kao što je Zbroj u ovom primjeru:

```
Dim x As Integer
Dim y As Integer
Dim intZbroj As Integer

Sub Zbroj(ParamArray intBrojevi())
    For Each x In intBrojevi
    y = y + x
    Next x
    intZbroj = y
End Sub
```

```
Private Sub Command1_Click()
    Zbroj 1, 3, 5, 7, 8
    List1.Additem intZbroj
End Sub
```

Stvaranje jednostavnijih izraza s imenovanim argumentima

Za većinu ugrađenih funkcija, izraza i postupaka Visual Basic pruža mogućnost korištenja *imenovanih argumenata* kao prečica za pisnje vrijednosti argumenta. S imenovanim argumentima, možete pribaviti bilo koji ili sve argumente, bilo kojim redom, dodjelom vrijednosti imenovanom argumentu. To ćete napraviti pisnjem imena argumenta s dvotočkom i znakom jednakosti te vrijednošću (MojArgument:= ”Neka vrijednost”) te postavljanjem takvih dodjeljivanja bilo kojim redom, razdvojenih zarezima. Uočite da su u sljedećem primjeru argumenti obrnutog redoslijeda u odnosu na očekivane argumente:

```
Function ListTekst(strIme As String, Optional strAdres As String)
    List1.AddItem strIme
    List2.Additem strAdres
End Sub

Private Sub Command1_Click()
    ListTekst strAdres:=”12345”, strIme:=”vaše ime”
End Sub
```

Ovo je posebno korisno ako vaši potprogrami imaju nekoliko neobaveznih argumenata koje ne morate uvijek odrediti.

Određivanje podrške za imenovane argumente

Kako bi utvrdili koje funkcije, izrazi i postupci podržavaju imenovane argumente, iskoristite svojstvo AutoQuickInfo u kodnom prozoru, provjerite u pretraživaču objekata, ili pogledajte biblioteku *Microsoft Visual Basic 6.0 Language Reference*. Razmišljajte o sljedećem kad radite s imenovanim objektima:

- Imenovani argumenti ne podržavaju postupci na objektima iz objektne biblioteke Visual Basica (VB). Podržani su od svih ključnih riječi jezika u objektnoj biblioteci Visual Basica za aplikacije (VBA), te od postupaka u objektnoj biblioteci pristupa podacima (DAO).
- Imenovani argumenti se u sintaksi ispisuju podebljanim i nakošenim pismom. Svi ostali argumenti su ispisni smo nakošenim pismom.

  Važno Ne možete upotrijebiti imenovane argumente za zaobilaženje unos traženih argumenata. Možete izostaviti smo neobavezne argumente. Pretraživač objekata zatvara neobavezne argumente u uglate zagrade [ ] kod pregleda objektnih biblioteka Visual Basica (VB) i Visual Basica za aplikacije (VBA).

Uvod u strukturu kontrola

Struktura kontrola dopušta vam kontrolu toka izvođenja vaše aplikacije. Ako je ne odredite izrazima za kontrolu toka, izvođenje aplikacije će teći kroz izraze s lijeva na desno, te odozgo prema dolje. Iako neke vrlo jednostavne aplikacije mogu biti napisne smo s takvim neodređenim tokom, te iako neki tokovi mogu biti kontrolirani korištenjem operatora koji će određivati prednost operacija, većina snage i iskoristivosti svakog programskog jezika dolazi iz njegove sposobnosti promjene redoslijeda izraza korištenjem struktura i petlji.

Strukture odluke

Potprogrami Visual Basica mogu ispitivati uvjete te, ovisno o rezultatima ispitivanja, izvesti različite operacije. Strukture odluke koje Visual Basic podržava uključuju:

- If...Then
- If...Then...Else
- Select Case

If...Then

Upotrijebite strukturu If...Then za uvjetno izvođenje jednog ili više izraza. Možete upotrijebiti ili sintaksu s jednom linijom ili sintaksu višelinijskog bloka:

If *uvjet* **Then** *izraz*

If *uvjet* **Then**
    izrazi

End If

Uvjet je obično usporedba, ali može biti bilo koji izraz koji se ocjenjuje kao brojčana vrijednost. Visual Basic tumači tu vrijednost kao True ili False; brojčana vrijednost nule jednaka je False, a svaka brojčana vrijednost različita od nule smatra se kao True. Ako je *uvjet* jednak True, Visual Basic izvršava sve *izraze* koji slijede naredbu Then. Možete upotrijebiti jednolinijsku ili višelinijsku sintaksu za uvjetno izvođenje smo jednog izraza (sljedeća dva primjera su jednaka):

```
If nekiDatum < Now Then nekiDatum = Now
```

```
If nekiDatum < Now Then
   nekiDatum = Now
End If
```

Uočite da jednolinijski oblik If...Then ne koristi naredbu End If. Ako želite izvršiti više od jedne linije programskog koda kad je *uvjet* jednak True, morate upotrijebiti sintaksu višelinijskog bloka If...Then...End If.

```
If nekiDatum < Now Then
   nekiDatum = Now
   Timer1.Enabled = False    ‘ Isključivanje kontrole mjerača vremena.
End If
```

If...Then...Else

Upotrijebite blok If...Then...Else za određivanje nekoliko blokova izraza, od kojih će se izvršiti jedan:

If *uvjet1* **Then**
    [*blokizraza-1*]
[**ElseIf** *uvjet2* **Then**
    [*blokizraza-2*]]...
[**Else**
    [*blokizraza-n*]]
End If

Visual Basic prvo ispituje *uvjet1*. Ako je rezultat False, Visual Basic nastavlja ispitivati *uvjet2*, i tako dalje, sve dok ne pronađe uvjet rezultata True. Kad pronađe uvjet rezultata True, Visual Basic izvršava pripadajući blok izraza te zatim izvodi kod koji slijedi naredbu End If. Kao mogućnost, možete uključiti blok izraza koji počinje naredbom Else, kojeg će Visual Basic izvršiti ako ni jedan od uvjeta ne da rezultat True.

If...Then...ElseIf je zapravo smo poseban slučaj If...Then...Else. Zapamtite da možete imati bilo koji broj uvjete ElseIf, ili ni jedan. Uvjet Else možete uključiti neovisno o tome imate li uvjete ElseIf.

Na primjer, vaša aplikacija mogla bi izvesti različite akcije ovisno o tome koja je kontrola kliknuta u području izbornika s kontrolama:

```
Private Sub mnuOdreži_Click (Index As Integer)
   If Index = 0 Then             ‘ Naredba odsijecanja.
      KopirajAktivnuKontrolu     ‘ Poziv općih potprograma.
      ObrišiAktivnuKontrolu
   ElseIf Index = 1 Then         ‘ Naredba kopiranja.
      KopirajAktivnuKontrolu
   ElseIf Index = 2 Then         ‘ Naredba brisnja.
      ObrišiAktivnuKontrolu
   Else                          ‘ Naredba lijepljenja.
      UlijepiAktivnuKontrolu
   End If
End Sub
```

Uočite da uvijek možete pripojiti dodatne dijelove ElseIf u vašu strukturu If...Then. Međutim, ovakva sintaks može postati zamorna ako svaki blok ElseIf uspoređuje isti izraz s drugačijom vrijednošću. Za takve situacije, možete upotrijebiti strukturu odluke Select Case.

Za više informacija Pogledajte dio "Naredba If...Then...Else" u biblioteci *Microsoft Visual Basic 6.0 Language Reference*.

Select Case

Visual Basic pruža strukturu Select Case kao alternativu strukturi If...Then...Else za izvođenje izabranog bloka izraza između višestrukih blokova izraza. Izraz Select Case pruža mogućnosti slične izrazu If...Then...Else, ali čini programski kod čitljivijim kad postoji više izbora.

Struktura Select Case radi s jednim tekstualnim izrazom koji se jednom procjenjuje, na vrhu strukture. Visual Basic zatim uspoređuje rezultat tog izraza s vrijednostima svakog izraza Case unutar strukture. Ako postoji slaganje, izvodi se blok izraza pridružen tom izrazu Case:

Select Case *tekstualniizraz*
    [**Case** *listaizraza1*
        [*blokizraza-1*]]
    [**Case** *listaizraza2*
        [*blokizraza-2*]]
        .
        .
        .
    [**Case Else**
        [*blokizraza-n*]]

End Select

Svaka *listaizraza* je lista jedne ili više vrijednosti. Ako postoji više vrijednosti u jednoj listi, te vrijednosti su razdvojene zarezima. Svaki *blokizraza* sdrži nulu ili više izraza. Ako se više od jednog izraza Case slaže s tekstualnim izrazom, izvršit će se smo blok izraza pridružen prvom podudarajućem izrazu Case. Visual Basic izvršava izraze u bloku Case Else (koji je neobavezan) ako ni jedna od vrijednosti u listama izraza ne odgovara tesktualnom izrazu.

Na primjer, pretpostavimo da ste dodali još jednu naredbu izborniku Edit u primjeru danom u prethodnom If...Then...Else odlomku. Možete dodati još jedan uvjet ElseIf, ili možete napisti funkciju koristeći Select Case strukturu:

```
Private Sub mnuOdreži_Click (Index As Integer)
   Select Case Index
      Case 0                          ‘ Naredba odsijecanja.
         KopirajAktivnuKontrolu       ‘ Poziv općih potprograma.
         ObrišiAktivnuKontrolu
      Case 1                          ‘ Naredba kopiranja.
         KopirajAktivnuKontrolu
      Case 2                          ‘ Naredba brisnja.
         ObrišiAktivnuKontrolu
```

```
        Case 3                      ‘ Naredba lijepljenja.
            UlijepiAktivnuKontrolu
        Case Else
            frmNa|i.Show            ‘ Prikaz dijaloškog okvira Find.
    End Select
End Sub
```

Uočite da izraz Select Case ocjenjuje izraz jednom na početku strukture. Suprotno tome, struktura If...Then...Else može ocijeniti različit izraz u svakom izrazu ElseIf. Strukturu If...Then...Else možete zamijeniti strukturom Select Case smo ako naredba If te svaki izraz ElseIf ocjenjuju isti izraz.

Strukture petlje

Strukture petlje omogućuju vam ponavljajuće izvođenje jedne ili više linija programskog koda. Strukture petlje koje Visual Basic podržava uključuju:

- Do...Loop
- For...Next
- For Each...Next

Do...Loop

Upotrijebite petlju tipa Do za izvođenje bloka izraze neodređeni broj puta. Postoji nekoliko varijacija izraza Do...Loop, ali svaka ocjenjuje brojčani uvjet kako bi odlučila hoće li nastaviti izvođenje. Kao i kod strukture If...Then, *uvjet* mora biti vrijednost ili izraz koji se može ocijeniti kao False (nula) ili True (različit od nule).

U sljedećem predlošku strukture Do...Loop, *izrazi* će se izvršavati sve dok *uvjet* ima vrijednost True:

Do While *uvjet*
    izrazi

Loop

Kad Visual Basic izvršava ovu petlju, prvo ispituje *uvjet*. Ako *uvjet* ima vrijednost False (nula), preskaču se svi izrazi. Ako uvjet ima vrijednost True (različit od nule), Visual Basic izvršava sve izraze i zatim se vraća na naredbu Do While te ponovno provjerava uvjet.

Shodno tome, petlja se može izvoditi neograničeno vrijeme, sve dok *uvjet* ne bude imao vrijednost različitu od nule ili vrijednost True. Naredbe se nikad ne izvode ako uvjet ima početnu vrijednost False. Na primjer, sljedeći potprogram broji pojavljivanja ciljnog stringa unutar drugog stringa vrteći se u petlji sve dok ciljni string ne bude pronađen:

```
Function BrojiStringove (dugistring, cilj)
    Dim položaj, brojač
    položaj = 1
```

```
    Do While InStr(položaj, dugistring, cilj)
       polo`aj = InStr(položaj, dugistring, cilj)
        + 1
       brojač = brojač + 1
    Loop
    BrojiStringove = brojač
End Function
```

Ako ciljni string ne postoji u drugom stringu, funkcija InStr će imati vrijednost 0, i petlja se neće izvršiti.

Druga varijacija petlje tipa Do...Loop prvo izvodi izraze pa ispituje *uvjet* nakon svakog izvođenja. Ovakva varijacija garantira barem jedno izvođenje *izraza*:

Do

izrazi

Loop While *uvjet*

Dvije druge varijacije su slične prethodnim dvjema, osim što se petlja vrti sve dok je vrijednost *uvjeta* False, a ne True.

petlja nikad ili više puta	petlja barem jednom
Do Until *uvjet* *izrazi* Loop	Do *izrazi* Loop Until *uvjet*

For...Next

Petlje tipa Do dobro rade kad ne znate koliko ćete puta trebati izvesti izraze unutar petlje. Međutim, kad znate da morate izvesti izraze određen broj puta, petlja For...Next je bolji izbor. Za razliku od petlje tipa Do...Loop, Petlja tipa For koristi varijablu nazvanu brojač kojoj se vrijednost povećava ili smanjuje kod svakog ponavljanja petlje. Sintaks je:

For *brojač* = *početak* **To** *kraj* [**Step** *korak*]
izrazi

Next [*brojač*]

Svi argumenti *brojač*, *početak*, *kraj* i *korak* su brojčani.

Napomena Argument *korak* može biti pozitivan ili negativan. Ako je *korak* pozitivan, *početak* mora biti manji ili jednak *kraju* ili se izrazi unutar petlje neće izvršiti. Ako je *korak* negativan, *početak* mora biti veći ili jednak *kraju* kako bi se izveo glavni dio petlje. Ako Step nije zadan, vrijednost *koraka* je 1.

Izvodeći petlju tipa For Visual Basic:

1. Postavlja *brojač* jednak *početku*.
2. Ispituje je li *brojač* veći od *kraja*. Ako je, Visual Basic izlazi iz petlje.

   (Ako je *korak* negativan, Visual Basic ispituje je li *brojač* manji od *kraja*)
3. Izvršava *izraze*.
4. Povećava *brojač* za 1 ili za vrijednost *koraka*, ako je određen.
5. Ponavlja stavke od 2 do 4.

Sljedeći programski kod ispisuje imena svih raspoloživih pisma funkcije Screen:

```
Private Sub Form_Click()
    Dim I As Integer
    For I = 0 To Screen.FontCount
       Print Screen.Fonts(I)
    Next
End Sub
```

U aplikaciji VCR, potprogram HighlightButton koristi petlju tipa For...Next za prolaz kroz zbirku kontrola forme VCR i prikaz prigodne kontrole Shape:

```
Sub HightlightButton(MyControl As Variant)
    Dim i As Integer
    For i = 0 To frmVCR.Controls.Count - 1
       If TypeOf frmVCR.Controls(i) Is Shape Then
            If frmVCR.Controls(i).Name = MyControl Then
               frmVCR.Controls(i).Visible = True
            Else
               frmVCR.Controls(i).Visible = False
            End If
       End If
    Next
End Sub
```

For Each...Next

Petlja tipa For Each...Next je slična petlji tipa For...Next, ali ponavlja grupu izraza za svaki element u kolekciji elemenata ili u matrici umjesto ponavljanja izraza određen broj puta. Ovaj način je posebno koristan ako ne znate koliko se elemenata nalazi u kolekciji.

Ovo je sintaks za petlju tipa For Each...Next:

For Each *element* **In** *grupa*
    izrazi

Next *element*

Na primjer, sljedeći potprogram tipa Sub otvara bazu podataka Biblio.mdb i dodaje ime svake tabele u okvir s popisom.

```
Sub ListTableDefs()
    Dim objDb As Database
    Dim MyTableDef As TableDef
    Set objDb = OpenDatabase("c:\vb\biblio.mdb", True, False)
    For Each MyTableDef In objDb.TableDefs()
       List1.AddItem MyTableDef.Name
    Next MyTableDef
End Sub
```

Kod korištenja petlji tipa For Each...Next imajte na umu sljedeća ograničenja:

- Za kolekcije, *element* može biti smo varijabla tipa Variant, opća varijabla tipa Object ili objekt ispisn u pretraživaču objekata.
- Za matrice, *element* može biti smo varijabla tipa Variant.
- Ne možete koristiti petlju tipa For Each...Next s matricama korisničkog tipa jer varijabla tipa Variant ne može sdržavati korisnički tip podatka.

Rad s strukturama kontrola

Ugniježđene strukture kontrola

Možete postaviti strukture kontrola unutar drugih struktura kontrola (kao blok If...Then unutar petlje tipa For...Next). Za strukturu kontrole koja je postavljena unutar druge strukture kontrole kaže se da je *ugniježđena*.

Strukture kontrola u Visual Basicu mogu biti ugniježđene u proizvoljnom broju razina. Uobičajena je praks uvući tijelo strukture ili petlje kako bi ugniježđene strukture odluka ili petlje bile čitljivije.

Na primjer, ovaj potprogram ispisuje imena svih pisma koja su zajednička ispisima na pisač i ekran:

```
Private Sub Form_Click()
    Dim EPismo, TPismo
    For Each EPismo In Screen.Fonts()
       For Each TPismo In Printer.Fonts()
          If EPismo = TPismo Then
             Print EPismo
          End If
       Next TPismo
    Next EPismo
End Sub
```

Uočite da prva naredba Next zatvara unutarnju petlju tipa For a posljednja naredba Next zatvara vanjsku petlju tipa For. Slično tome, u ugniježđenim petljama tipa If, naredba End If automatski se dodjeljuje najbližoj prethodnoj naredbi If. Ugniježđene strukture tipa Do...Loop rade na sličan način, najunutarnjija naredba Loop odgovara najunutarnjijoj naredbi Do.

Izlaz iz strukturne kontrole

Naredba Exit omogućuje vam direktan izlaz iz petlji tipa For i Do, te potprograma tipa Sub i Function. Sintaks naredbe Exit je jednostavna; izraz Exit For može se pojaviti koliko je potrebno puta unutar petlje tipa For, a izraz Exit Do može se pojaviti koliko je potrebno puta unutar petlje tipa Do:

For *brojač* = *početak* **To** *kraj* [**Step** *korak*]
    [*blokizraza*]
    [**Exit For**]
    [*blokizraza*]

Next [*brojač*[, *brojač*] [,...]]

Do [{**While** | **Until**} *uvjet*]
    [*blokizraza*]
    [**Exit Do**]
    [*blokizraza*]

Loop

Naredba Exit Do radi s svim varijacijama petlji tipa Do.

Naredbe Exit For i Exit Do su korisne jer je ponekad potrebno odmah izaći iz petlje, bez izvođenja bilo kojeg idućeg ponavljanja ili izraza unutar petlje. Na primjer, u prethodnom primjeru koji je ispisivao pisma zajednička ispisu na ekran i pisač, programski kod nastavlja uspoređivati pisma pisača s pismima ekrana čak i kad je pronađeno pismo zajedničko prethodnom pismu pisača. Efikasnija verzija dane funkcije će izaći iz petlje čim se pronađe prvo zajedničko pismo:

```
Private Sub Form_Click()
    Dim EPismo, TPismo
    For Each EPismo In Screen.Fonts()
       For Each TPismo In Printer.Fonts()
         If EPismo = TPismo Then
            Print EPismo
            Exit For              ‘ Izlaz iz unutarnje petlje.
         End If
       Next TPismo
    Next EPismo
End Sub
```

Kao što pokazuje ovaj primjer, naredba Exit gotovo se uvijek pojavljuje unutar petlji tipa If ili Select Case ugniježđnih unutar druge petlje.

Kad koristite naredbu Break za prekid petlje, vrijednost varijable brojača je različita, ovisno o načinu izlaska iz petlje:

- Kad završite petlju, varijabla brojača sdrži zbroj vrijednosti gornje granice i koraka.
- Kad izađete iz petlje prije kraja, varijabla brojača sdržava svoju vrijednost ovisnu o uobičajenim pravilima područja.
- Kad izađete iz petlje određene kolekcijom prije kraja, varijabla brojača sdrži Nothing ako je tipa Object, ili Empty ako je tipa Variant.

Izlaz iz potprograma tipa Sub ili Function

Iz potprograma možete izaći i iz unutrašnjosti strukture kontrole. Sintaks izraza Exit Sub i Exit Function je slična sintaksi izraza Exit For i Exit Do iz prethodnog odlomka "Izlaz iz strukturne kontrole". Izraz Exit Sub se može pojaviti potreban broj puta, bilo gdje unutar tijela potprograma tipa Sub. Izraz Exit Function se može pojaviti potreban broj puta, bilo gdje unutar tijela potprograma tipa Function.

Izrazi Exit Sub i Exit Function su korisni kad je potprogram napravio sve što je trebao i može se trenutno vratiti. Na primjer, ako želite promijeniti prethodni primjer tako da ispisuje samo prvo pismo zajedničko pisaču i ekranu koje nađe, možete upotrijebiti izraz Exit Sub:

```
Private Sub Form_Click()
    Dim EPismo, TPismo
    For Each EPismo In Screen.Fonts()
       For Each TPismo In Printer.Fonts()
         If EPismo = TPismo Then
            Print EPismo
            Exit Sub                    ‘ Izlaz iz potprograma.
         End If
       Next TPismo
    Next EPismo
End Sub
```

Rad s objektima

Kad stvarate aplikaciju u Visual Basicu, radite s objektima. Možete koristiti objekte koje pruža Visual Basic – kao što su kontrole, forme i objekti za pristup podacima. Možete također kontrolirati i objekte drugih aplikacija iz vaše Visual Basic aplikacije. Možete čak i stvoriti vlastite objekte, i odrediti im pripadajuća svojstva i postupke.

Što je objekt?

Objekt je kombinacija programskog koda i podataka koji mogu biti obrađivani kao cjelina. Objekt može biti dio aplikacije, kao kontrola ili forma. Cijela aplikacija također može biti objekt. Sljedeća tabela opisuje primjere tipova objekata koje možete koristiti u Visual Basicu.

primjer	opis
Naredbeni gumb	Kontrole na formi, kao naredbeni gumb i okviri, su objekti.
Forma	Svaka forma u projektu Visual Basica je zaseban objekt.
Baza podataka	Baze podataka su objekti, i sdrže druge objekte, kao polja i indekse.
Dijagram	Dijagram u Microsoft Excelu je objekt.

Otkud dolaze objekti?

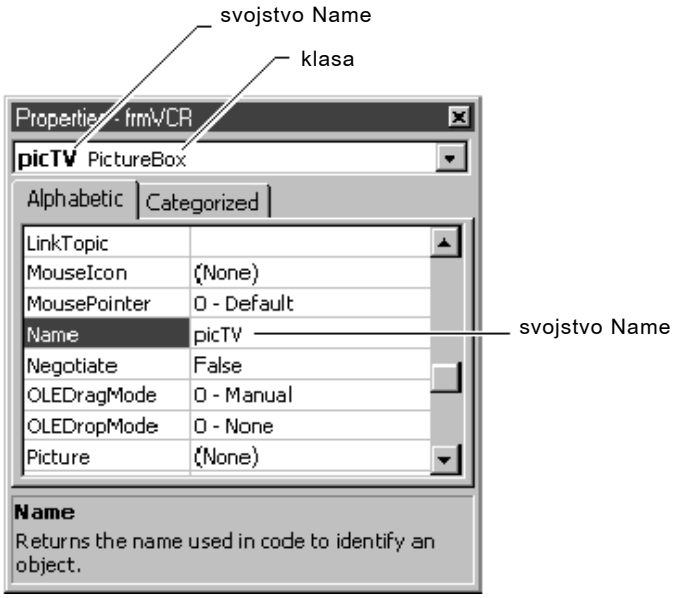
Svaki objekt u Visual Basicu je određen *klasom*. Kako bi razumjeli odnos između objekta i njegove klase, razmislite o rezaču kolačića i kolačićima. Rezač kolačića je klas. On određuje osobine svakog kolačića – na primjer, veličinu i oblik. Klas se koristi za stvaranje objekata. Objekti su kolačići.

Ovo bi mogla pojasniti dva primjera odnos između klas i objekata u Visual Basicu.

- Kontrole u alatnom okviru Visual Basica predstavljaju klase. Objekt poznat kao kontrola ne postoji dok ga ne stvorite na formi. Kad stvorite kontrolu, stvorili ste kopiju *primjera* klase te kontrole. Taj primjer klase je objekt kojeg pozivate u svojoj aplikaciji.
- Forma s kojom radite za vrijeme stvaranja aplikacije je klas. Kad se aplikacija izvodi, Visual Basic stvara primjer klase forme.

Prozor s svojstvima prikazuje klasu i svojstvo imena objekata u vašoj Visual Basic aplikaciji, kao što je prikazano na slici 5.8.

Slika 5.8 Imena klase i objekta prikazana u prozoru s svojstvima



Svi objekti stvaraju se kao jednake kopije njihove klase. Jednom kad postoje kao smostalni objekti, njihova se svojstva mogu mijenjati. Na primjer, ako na formi kreirate tri naredbena gumba, svaki naredbeni gumb je primjer klase naredbenog gumba. Svaki objekt dijeli zajednički komplet svojstava i mogućnosti (svojstva, postupke i događaje), određenih klasom. Međutim, svaki od njih ima svoje ime, može odvojeno biti omogućen ili onemogućen, postavljen na različit položaj na form, i tako dalje.

Zbog jednostavnosti, većina materijala izvan ovog poglavlja neće se pozivati na klasu objekta. Smo zapamtite da izraz “kontrola okvira s popisom”, na primjer, znači “primjer klase okvira s popisom”.

Što možete s objektima?

Objekt pruža programski kod koji ne morate pisti. Na primjer, možete kreirati vlastite dijaloške okvire za otvaranje i snimanje datoteka, ali ne morate. Umjesto toga, možete iskoristiti opću kontrolu dijaloškog okvira (objekt) koju pruža Visual Basic. Mogli bi napisti vlastiti programski kod za rukovanje sstancima i prihodima, ali ne morate. Umjesto toga, možete upotrijebiti objekte kalendara, prihoda i zadaća koje pruža Microsoft Project.

Visual Basic može kombinirati objekte iz drugih izvora

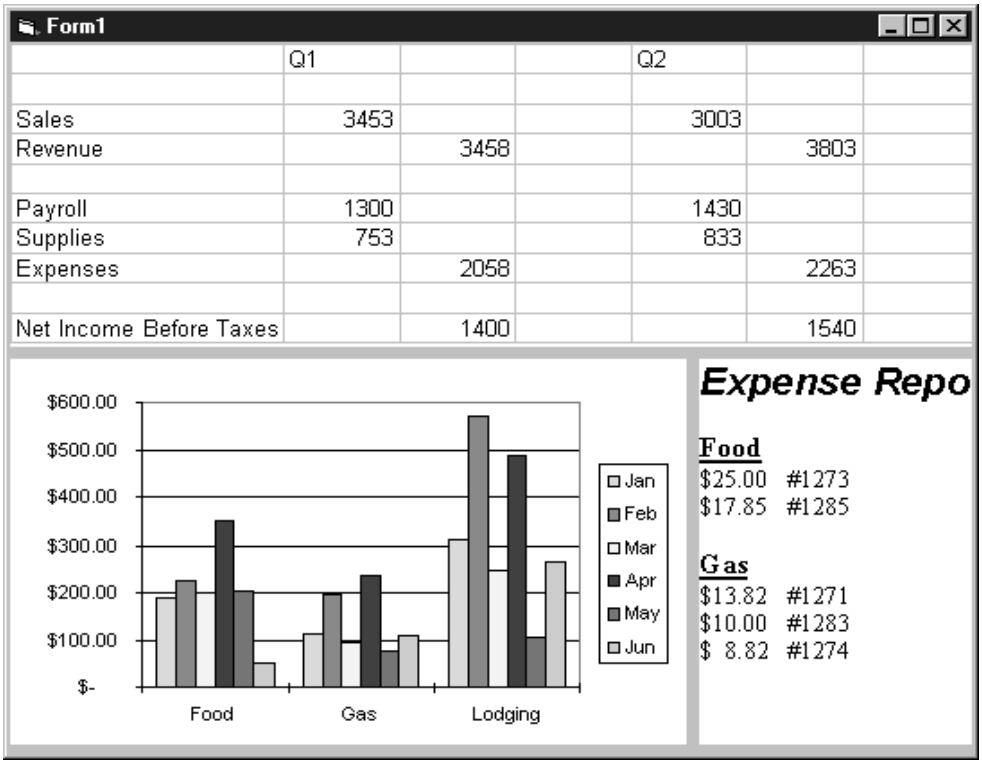
Visual Basic pruža alate koji vam omogućuju kombiniranje objekata iz različitih izvora. Sd možete graditi korisnička rješenja kombinirajući najmoćnije osobine Visual Basica i aplikacija koje podržavaju automatizaciju (dosd znanu kao OLE automatizaciju). *Automatizacija* je osobina *komponentnog objektnog modela* (*Component Object Model*, COM), industrijskog standarda kojeg koristi aplikacija za izlaganje objekata razvojnim alatima i drugim aplikacijama.

Aplikacije možete izgraditi povezivanjem ugrađenih kontrola Visual Basica, a možete i koristiti objekte koje pružaju druge aplikacije. Razmislite o postavljanju sljedećih objekata na formu Visual Basica:

- Objekt grafikona Microsoft Excela (Chart)
- Objekt radnog lista Microsoft Excela (Worksheet)
- Objekt dokumenta Microsoft Worda

Ove objekte možete upotrijebiti za stvaranje aplikacije koja će raditi kao knjiga računa, kao što je ona prikazana na slici 5.9. To će vam uštedjeti vrijeme jer ne morate pisti programski kod za pružanje upotrebljivosti kakvu daju objekti Microsoft Excela i Microsoft Worda.

Slika 5.9 Korištenje objekata iz drugih aplikacija



Osnove rada s objektima

Objekti Visual Basica podržavaju svojstva, postupke i događaje. U Visual Basicu, podaci objekta (postavke ili atributi) se nazivaju svojstva, dok se različiti potprogrami koji mogu utjecati na objekt nazivaju postupcima. Događaj je akcija prepoznata od objekta, kao što je klik mišem ili pritisak neke tipke, i možete napisti programski kod koji će odgovoriti na taj događaj.

Karakteristike objekta možete promijeniti mijenjanjem svojstava. Uzmite za primjer radio: jedno svojstvo radija je njegova glasnoća. U Visual Basicu, mogli bi reći da radio ima svojstvo "Glasnoća" kojeg možete prilagoditi mijenjajući mu vrijednost. Pretpostavimo da glasnoću radija možete namjestiti na vrijednost od 0 do 10. Kad bi mogli upravljati radiom s Visual Basicom, napisli bi sljedeći programski kod u potprogram koji mijenja vrijednost svojstva "Glasnoće" s 3 na 5 kako bi radio glasnije svirao:

```
Radio.Glasnoća = 5
```

Objekti imaju i postupke, kao dodatak svojstvima. Postupci su dijelovi objekta kao i svojstva. Općenito, postupci su akcije koje želite izvesti, dok su svojstva osobine koje postavljate ili pronalazite. Na primjer, birate broj na telefonskom aparatu kako bi nazvali nekoga. Možete reći da telefon ima postupak "Zovi", i mogli bi upotrijebiti sljedeću sintaksu za poziv sedmeroznamenkastog broja 5551111:

```
Telefon.Zovi 5551111
```

Objekti također imaju i događaje. Događaji su pokrenuti kad se promijeni neki dio objekta. Na primjer, radio bi mogao imati događaj "PromjenaGlasnoće". Telefon bi morao imati događaj "Zvoni".

Kontroliranje objekata njihovim svojstvima

Pojedina svojstva razlikuju se po tome kad možete postaviti ili čitati njihove vrijednosti. Neka svojstva mogu se odrediti tijekom stvaranja aplikacije. Možete iskoristiti prozor s svojstvima kako bi takvim svojstvima postavili vrijednosti bez potrebe za pisnjem koda. Neka svojstva nisu dostupna tijekom stvaranja, pa zbog toga morate napisti programski kod koji će ih odrediti tijekom rada aplikacije.

Svojstva koja možete postavljati i čitati tijekom rada aplikacije zovu se *svojstva za čitanje i pisnje* (*read-write*). Svojstva koja možete smo čitati tijekom izvođenja zovu se *svojstva smo za čitanje* (*read-only*).

Postavljanje vrijednosti svojstava

Vrijednosti svojstava postavljate kad želite promijeniti izgled ili ponašanje objekta. Na primjer, svojstvo Text okvira s tekstom mijenjate kako bi promijenili sdržaj ispisnog teksta u okviru.

Za postavljanje vrijednosti u svojstvo, upotrijebite sljedeću sintaksu:

```
objekt.svojstvo = izraz
```

Sljedeći izrazi pokazuju kako postavljate svojstva:

```
Text1.Top = 200    ‘ Postavljanje svojstva Top na 200 twips.
Text1.Visible = True      ‘ Prikaz okvira s tekstom.
Text1.Text = “pozdrav”    ‘ Ispis “pozdrav” u okviru s tekstom.
```

Čitanje vrijednosti svojstava

Vrijednost svojstva čitate kad želite sznati stanje objekta prije nego što vaš programski kod izvrši određenu akciju (kao što je dodjela vrijednosti drugom objektu). Na primjer, možete pročitati vrijednost svojstva Text kontrole okvira s tekstom kako bi odredili sdržaj teksta prije pokretanja koda koji bi tu vrijednost mogao promijeniti.

U većini slučajeva, koristite sljedeću sintaksu za čitanje vrijednosti svojstva:

varijabla = objekt.svojstvo

Vrijednost svojstva možete pročitati i u sklopu složenijeg izraza, bez dodjeljivanja vrijednosti svojstva varijabli. U sljedećem primjeru programskog koda, svojstvo Top novog člana područja kontrola proračunava se kao svojstvo Top prethodnog člana, kojem je dodano 400:

```
Private Sub cmdDodaj_Click()
    ‘ [izrazi]
    optGumb(n).Top = optGumb(n - 1).Top + 400
    ‘ [izrazi]
End Sub
```

Svjet Ako vrijednost svojstva namjeravate koristiti više puta, vaš programski kod će se brže izvršavati ako tu vrijednost spremite u varijablu.

Izvođenje akcija postupcima

Postupci mogu utjecati na vrijednosti svojstava. Na primjer, u radijskoj usporedbi, postupak OdrediGlasnoću mijenja svojstvo Glasnoća. Slično tome, u Visual Basicu, okviri s listom imaju svojstvo List, koje može biti promijenjeno postupcima Clear i AddItem.

Korištenje postupaka u kodu

Kad u programskom kodu koristite postupak, način pisnja izraza ovisi o tome koliko argumenata postupak zahtijeva, te hoće li postupak vratiti vrijednost. Kad postupak ne treba argumente, napist ćete programski kod koristeći sljedeću sintaksu:

objekt.postupak

U sljedećem primjeru, postupak Refresh ponovno iscrtava sdržaj okvira za sliku:

```
Picture1.Refresh   ‘ Forsira iscrtavanje kontrole.
```

Neki postupci, kao što je postupak Refresh, nemaju argumente i ne vraćaju vrijednosti.

Ako postupak traži više od jednog argumenta, argumente ćete razdvojiti zarezima. Na primjer, postupak Circle koristi argumente za određivanje položaja, polumjera i boje kruga na formi:

```
‘ Crtanje plavog kruga s polumjerom od 1200 twips.
Forma1.Circle (1600, 1800), 1200, vbBlue
```

Ako želite sčuvati vrijednost postupka, morate zatvoriti argumente u zagrade. Na primjer, postupak GetData vraća sliku iz odlagališta:

```
Slika = Clipboard.GetData (vbCFBitmap)
```

Ako ne postoji povratna vrijednost, argumenti se zadaju bez zagrada. Na primjer, postupak AddItem ne vraća vrijednost:

```
List1.AddItem = “va e ime”        ‘ Dodaje tekst “vaše ime”
                                  ‘ u okvir s popisom.
```

Za više informacija Pogledajte biblioteku *Microsoft Visual Basic 6.0 Language Reference* za sintakse i argumente svih postupaka koje pruža Visual Basic.

Kako su objekti međusobno povezani?

Kad na formu postavite dva naredbena gumba, oni su odvojeni objekti s različitim vrijednostima svojstva Name (*Command1* i *Command2*), ali dijele zajedničku klasu – klasu CommandButton.

Njihova zajednička osobina je i položaj na istoj formi. Vidjeli ste ranije u ovom poglavlju da je kontrola na formi također sdržana u formi. Ovaj način postavlja kontrole u hijerarhiju. Kako bi mogli pozivati kontrolu morate prvo pozvati formu, na isti način kako morate nazvati pozivni broj države ili županije prije poziva određenog telefonskog broja.

Zajednička osobina dva naredbena gumba je i da su oba kontrole. Sve kontrole imaju opće osobine koje ih razlikuju od formi i drugih kontrola sučelja Visual Basica. Sljedeći odlomci objašnjavaju kako Visual Basic koristi zbirke za grupiranje objekata koji su povezani.

Hijerarhija objekata

Hijerarhija objekata pruža organizaciju koja određuje kako su objekti međusobno povezani, te kako im možete pristupiti. U većini slučajeva, ne morate se brinuti o hijerarhiji objekata Visual Basica. Ipak:

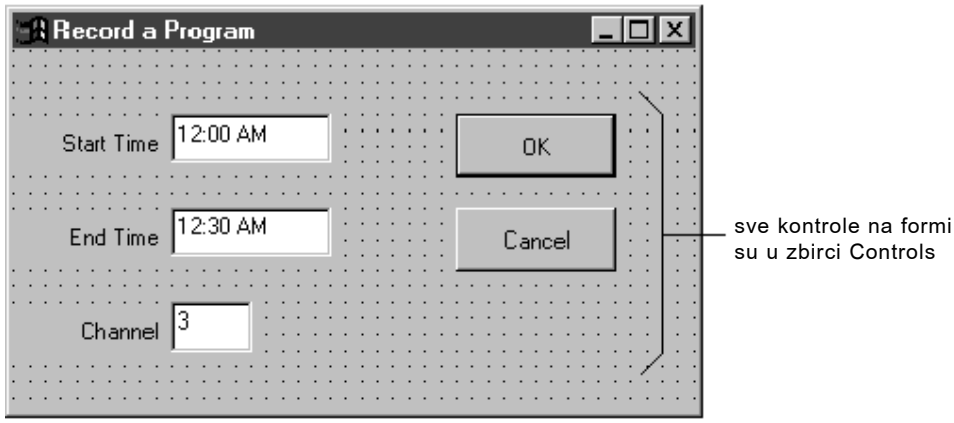
- Kad upravljate objektima drugih aplikacija, morate biti upoznati s hijerarhijom objekata te aplikacije. Za informacije o upravljanju hijerarhijom objekata, pogledajte 10. poglavlje “Programiranje sstavnim dijelovima”.
- Kad radite s objektima za pristup podacima, morate biti upoznati s hijerarhijom tih objekata.

U Visual Basicu postoje neki zajednički slučajevi hijerarhije kad jedan objekt sdržava druge. Takvi slučajevi su opisni u sljedećim odlomcima.

Rad s zbirkama objekata

Objekti zbirke imaju svoja vlastita svojstva i postupke. Objektima u objektu zbirke pristupa se kao *članovima* zbirke. Svaki član zbirke je redom obrojčan počevši od 0; to je *redni broj* člana. Na primjer, zbirka kontrola Controls sdrži sve kontrole na određenoj formi, kao što je prikazano na slici 5.10. Zbirke možete koristiti za pojednostavljivanje programskog koda ako trebate obaviti jednu operaciju na svim objektima u zbirci.

Slika 5.10 Zbirka kontrola



Na primjer, sljedeći programski kod prolazi kroz zbirku kontrola Controls i ispisuje ime svakog člana u okviru s popisom.

```
Dim MyControl As Control
For Each MyControl In Form1.Controls
    ‘ Dodavanje imena svake kontrole u okvir s popisom.
    List1.AddItem.MyControl.Name
Next MyControl
```

Dodavanje svojstava i postupaka članovima zbirke

Postoje dvije općenite tehnike koje možete koristiti za adresiranje člana zbirke:

- Odredite ime člana. Sljedeći izrazi su jednaki:

```
  Controls("List1")
Controls!List1
```

- Upotrijebite redni broj člana:

```
  Controls(3)
```

Kad jednom budete sposobni adresirati sve članove zbirno, ili pojedine članove osobno, možete im dodavati svojstva i postupke koristeći jedan od sljedeća dva pristupa:

```
‘ Postavljanje svojstva Top kontrole okvira s popisom na 200.
Controls!List1.Top = 200
```

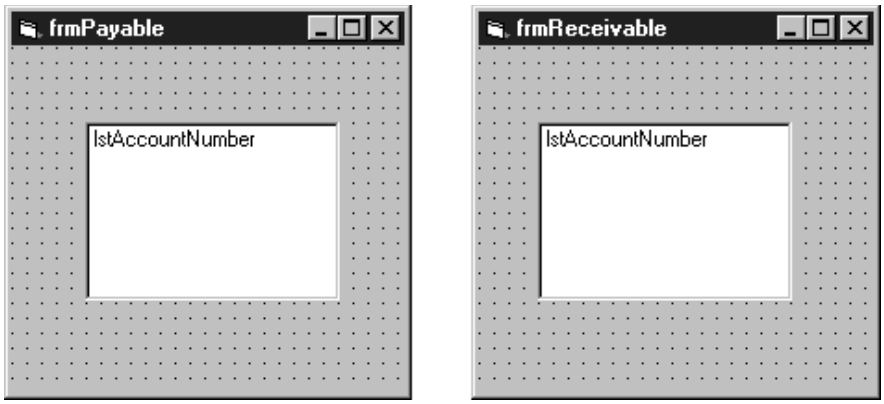
- ili -

```
Dim MyControl As Control
For Each MyControl In Form1.Controls()
    ‘ Postavljanje svojstva Top svakog člana na 200.
    MyControl.Top = 200
Next MyControl
```

Objekti koji mogu sdržavati druge objekte

Neki objekti u Visual Basicu sdržavaju druge objekte. Na primjer, forma obično sdržava jednu ili više kontrola. Prednost posjedovanja objekata kao spremnika za druge objekte je u tome da u vašem kodu možete pozvati spremnik kako bi razjasnili koji objekt želite koristiti. Na primjer, slika 5.11 prikazuje dvije različite forme koje bi mogli imati u aplikaciji – jednu za upis prihoda i drugu za upis rashoda.

Slika 5.11 Dvije različite forme mogu sdržavati kontrole istog imena



Obje forme mogu imati okvir s popisom nazvan lstAcctNo. Možete točno odrediti koji okvir želite koristiti pozivanjem forme koja sdrži taj okvir s popisom:

```
frmReceivable.lstAcctNo.AddItem 1201
```

- ili -

```
frmPayable.lstAcctNo.AddItem 1201
```

Opće zbirke u Visual Basicu

U Visual Basicu postoje neki opći slučajevi gdje jedan objekt sdrži druge objekte. Sljedeća tablica ukratko opisuje najčešće opće zbirke u Visual Basicu.

zbirka	opis
Forms	Sdrži učitane forme.
Controls	Sdrži kontrole na formi.
Printers	Sdrži dostupne objekte tipa Printer.

I vi također možete izvršiti povezivanje objekata u Visual Basicu.

Za više informacija Za informacije o povezivanju objekata, pogledajte "Korištenje zbirki umjesto matrica" u 8. poglavlju "Više o programiranju". Za informacije o zbirci Printers, pogledajte 12. poglavlje "Rad s tekstom i grafikom". Za detalje o zbirkama formi i kontrola, pogledajte biblioteku *Microsoft Visual Basic 6.0 Language Reference* te dio *Microsoft Visual Basic 6.0 Controls Reference* biblioteke *Microsoft Visual Basic 6.0 Reference Library*.

Svojstvo Container

Svojstvo Container možete upotrijebiti za promjenu spremnika objekta unutar forme. Sljedeće kontrole mogu sdržavati druge kontrole:

- Kontrola okvira (frame)
- Kontrola okvira za sliku (picture box)
- Kontrola alatne trake (toolbar, smo Professional i Enterprise verzije)

Ovaj primjer prikazuje pomicanje naredbenog gumba naokolo po formi iz spremnika u spremnik. Otvorite novi projekt, i kreirajte na formi kontrolu okvira, kontrolu okvira za sliku te naredbeni gumb.

Sljedeći ispis programskog koda iz događaja Click forme povećava varijablu brojača, te koristi petlju tipa Select Case za kruženje naredbenog gumba iz spremnika u spremnik.

```
Private Sub Form_Click()
Static intX As Integer
    Select Case intX
     Case 0
     Set Command1.Container = Picture1
     Command1.Top = 0
     Command1.Left = 0

     Case 1
     Set Command1.Container = Frame1
     Command1.Top = 0
     Command1.Left = 0

     Case 2
     Set Command1.Container = Form1
     Command1.Top = 0
     Command1.Left = 0

    End Select
    intX = intX + 1
End Sub
```

Za više informacija Pogledajte odlomak "Svojstvo Container" u biblioteci *Microsoft Visual Basic 6.0 Language Reference*.

Komunikacija među objektima

Kao dodatak stvaranju i korištenju objekata unutar Visual Basica, iz vaše aplikacije možete i komunicirati s drugim aplikacijama i upravljati njihovim objektima. Mogućnost dijeljenja podataka među aplikacijama je jedna od glavnih osobina operativnog sustava Microsoft Windows. S Visual Basicom, imate veliku fleksibilnost načina komuniciranja s drugim aplikacijama.

Za više informacija Za detalje o korištenju objekata i komuniciranju s objektima drugih aplikacija, pogledajte 10. poglavlje "Programiranje sstavnim dijelovima".

Stvaranje objekata

Najlakši način stvaranja objekta je dvoklik na kontrolu u alatnom okviru. Međutim, za ostvarivanje potpunog iskorištenja svih objekata dostupnih u Visual Basicu te iz drugih aplikacija, možete iskoristiti mogućnosti programibilnosti Visual Basica za stvaranje objekata tijekom rada aplikacije.

- Možete stvoriti pozive objekta uz pomoć varijabli objekata.
- Možete stvoriti vlastite objekte "iz dijelova" s modulima klase.
- Možete stvoriti vlastite zbirke korištenjem objekta Collection.

Za više informacija Ostala poglavlja pokazuju vam kako pristupati objektima. Funkcije CreatObject i GetObject, na primjer, raspravljene su u 10. poglavlju "Programiranje sstavnim dijelovima".

Korištenje varijabli objekata

Kao dodatak spremanju vrijednosti, varijable mogu upućivati na objekt. Objekt dodjeljujete varijabli s istim razlozima zbog kojih bilo koju vrijednost dodjeljujete varijabli:

- Imena varijabli su često kraća i lakša za pamćenje od vrijednosti koje sdrže (ili, u ovom slučaju, od objekata na koje upućuju).
- Varijable mogu biti promijenjene tijekom rada aplikacije kako bi upućivale na druge objekte.
- Upućivanje na varijablu koja sdrži objekt je djelotvornije neko ponavljano upućivanje na sm objekt.

Korištenje varijable objekta je slično korištenju uobičajene varijable, ali s jednim dodatnim korakom – dodjeljivanjem objekta varijabli:

- Prvo morate odrediti varijablu:

  Dim *varijabla* As *klas*

- Zatim joj možete dodijeliti objekt:

  Set *varijabla* = *objekt*

Određivanje varijabli objekata

Varijablu objekta određujete na isti način kao i ostale varijable, naredbama Dim, ReDim, Static, Private ili Public. Jedina razlika je neobavezna ključna riječ New i argument *klase*; oboje će biti raspravljeno kasnije u ovom poglavlju. Sintaks je:

{**Dim** | **ReDim** | **Static** | **Private** | **Public**} *varijabla* **As** [**New**] *klas*

Na primjer, možete odrediti varijablu objekta koja upućuje na formu u aplikaciji nazvanu frmGlavna:

```
Dim FormVar As New frmGlavna      ‘ Određivanje varijable objekta
                                  ‘ tipa frmGlavna.
```

Također možete odrediti varijablu objekta koja može upućivati na bilo koju formu u aplikaciji:

```
Dim nekaForma As Form             ‘ Opća varijabla forme.
```

Slično tome, možete odrediti varijablu objekta koja može upućivati na bilo koji okvir s tekstom u vašoj aplikaciji:

```
Dim nekiTekst As TextBox  ‘ Može upućivati na bilo koji okvir s
                          ‘ tekstom (ali smo okvir s tekstom).
```

Možete također odrediti varijablu objekta koja može upućivati na kontrolu bilo kojeg tipa:

```
Dim nekaKontrola As Control       ‘ opća varijabla kontrole.
```

Uočite da možete odrediti varijablu forme koja upućuje na određenu formu u aplikaciji, ali ne možete odrediti varijablu kontrole koja će upućivati na određenu kontrolu. Možete odrediti varijablu kontrole koja će upućivati na određen tip kontrole (kao TextBox ili ListBox), ali ne na određenu kontrolu tog tipa (kao txtUpis ili Lista1). Unatoč tome, određenu kontrolu možete dodijeliti varijabli tog tipa. Na primjer, za formu koja sdrži okvir s popisom imena lstPrimjer, mogli bi napisti:

```
Dim objDemo As ListBox
Set objDemo = lstPrimjer
```

Dodjela varijabli objekata

Objekt dodjeljujete varijabli objekta naredbom Set:

Set *varijabla = objekt*

Upotrijebite naredbu Set kad god želite da varijabla objekta upućuje na objekt.

Ponekad ćete koristiti varijable objekata, te posebno varijable kontrola, jednostavno zbog skraćivanja programskog koda koji morate napisti. Na primjer, mogli bi imati kod poput ovog:

```
If frmPrikazRačuna!txtStanjeRačuna.Text < 0 Then
    frmPrikazRačuna!txtStanjeRačuna.BackColor = 0
    frmPrikazRačuna!txtStanjeRačuna.ForeColor = 255
End If
```

Takav programski kod možete značajno skratiti korištenjem varijable kontrole:

```
Dim Bal As TextBox
Set Bal = frmPrikazRačuna!txtStanjeRačuna
If Bal.Text < 0 Then
    Bal.BackColor = 0
    Bal.ForeColor = 255
End If
```

Specifični i opći tipovi objekata

Varijable specifičnih objekata moraju upućivati na jedan određen tip objekta ili klase. Varijabla specifične forme može upućivati smo na jednu formu u aplikaciji (iako može upućivati i na jedan od puno primjera te forme). Slično tome, varijabla specifične kontrole može upućivati na smo jedan određeni tip kontrole u vašoj aplikaciji, kao što je okvir s tekstom ili okvir s popisom. Kako bi vidjeli jedan primjer, otvorite novi projekt i postavite okvir s tekstom na formu. Dodajte sljedeći kod formi:

```
Private Sub Form_Click()
    Dim nekiTekst As TextBox
    Set nekiTekst = Text1
    nekiTekst.Text = "pozdrav"
End Sub
```

Pokrenite aplikaciju i kliknite na formu. Svojstvo Text okvira s tekstom će se promijeniti u "pozdrav".

Varijable općih objekata mogu upućivati na jedan od puno specifičnih tipova objekata. Varijabla opće forme, na primjer, može upućivati na bilo koju formu u aplikaciji; varijabla opće kontrole može upućivati na bilo koju kontrolu na bilo kojoj formi u aplikaciji. Kako bi vidjeli primjer, otvorite novi objekt i postavite nekoliko kontrola okvira, natpis i naredbenog gumba na formu, bilo kojim redom. Formi dodajte sljedeći kod:

```
Private Sub Form_Click()
    Dim nekaKontrola As Control
    Set nekaKontrola = Form1.Controls(3)
    nekaKontrola.Caption = "pozdrav"
End Sub
```

Pokrenite aplikaciju i kliknite na formu. Natpis na kontroli koju ste postavili na formu kao treću po redu bit će promijenjen u "pozdrav".

U Visual Basicu postoje četiri tipa općeg objekta:

tip općeg objekta	upućivanje na objekt
Form	Bilo koja forma u aplikaciji (uključujući MDI djecu i MDI formu).
Control	Bilo koja kontrola u vašoj aplikaciji.
MDIForm	MDI forma vaše aplikacije (ako je aplikacija ima).
Object	Bilo koji objekt.

Varijable općih objekata korisne su kad ne znate specifičan tip objekta na koji će varijabla upućivati tijekom izvođenja aplikacije. Na primjer, ako želite napisti programski kod koji će raditi s bilo kojom formom u aplikaciji, morate koristiti varijablu opće forme.

Napomena Budući da u aplikaciji može postojati smo jedna MDI forma, nema potrebe za korištenjem tipa općeg objekta MDIForm. Umjesto toga, možete koristiti specifičan tip (MDIForm1, ili što ste odredili za svojstvo Name MDI forme) kad god trebate odrediti varijablu forme koja će upućivati na MDI formu. Zapravo, pošto Visual Basic može riješiti pokazivače na svojstva i postupke specifičnog tipa forme prije početka rada aplikacije, uvijek bi trebali koristiti specifičan tip za MDI Formu.

Opći tip MDIForm postoji smo zbog cjelovitosti; mogao bi postati koristan ako sljedeće verzije Visual Basica omoguće više MDI formi u jednoj aplikaciji.

Forme kao objekti

Forme se najčešće koriste za izradu sučelja aplikacije, ali su one i objekti koje mogu pozivati ostali moduli u vašoj aplikaciji. Forme su blisko povezane s modulima klase. Glavna razlika među njima je što forme mogu biti vidljivi objekti, dok moduli klase nemaju vidljivo sučelje.

Dodavanje korisničkih postupaka i svojstava

Formama možete dodati korisničke postupke i svojstva te im pristupiti iz drugih modula vaše aplikacije. Za stvaranje novog postupka na formi, dodajte potprogram određen korištenjem naredbe Public.

```
‘ Korisnički postupak na Form1
Public Sub BrojačNovihPoslova()
    .
    . ‘ <izrazi>
    .
End Sub
```

Potprogram BrojačNovihPoslova možete pozvati iz drugog modula korištenjem ovog izraza:

```
Form1.BrojačNovihPoslova
```

Stvaranje novog svojstva forme može biti jednostavno kao određivanje javne varijable u modulu forme:

```
Public IDBroj As Integer
```

Vrijednost svojstva IDBroj možete postaviti i pročitati iz drugog modula korištenjem ova dva izraza:

```
Form1.IDBroj = 3
Text1.Text = Form1.IDBroj
```

Za dodavanje korisničkih svojstava formi također možete upotrijebiti i potprograme

tipa Property.

Za više informacija Detalji o potprogramima tipa Property dani su u 9. poglavlju "Programiranje objektima".

> **Napomena** Možete pozvati varijablu ili korisnički postupak, te odrediti korisničko svojstvo na formi bez učitavanja forme. To vam omogućuje izvođenje programskog koda na formi bez učitavanja u memoriju. Također, upućivanje na kontrolu bez upućivanja na neko od njezinih svojstava ili postupaka neće učitati formu.

Korištenje ključne riječi New

Upotrijebite ključnu riječ New za stvaranje novog objekta određenog njegovom klasom. Riječ New može biti upotrijebljena za stvaranje primjera forme, klas određenih u modulima klase, te zbirki.

Korištenje ključne riječi New s formama

Svaka forma koju stvorite tijekom izrade aplikacije je klas. Ključna riječ New može se upotrijebiti za stvaranje novih primjera iz te klase. Da bi vidjeli kako to djeluje, stvorite naredbeni gumb i nekoliko drugih kontrola na formi. U prozoru s svojstvima dodijelite svojstvu forme Name sdržaj Primjer. Dodajte sljedeći programski kod potprogramu događaja Click vašeg naredbenog gumba:

```
Dim x As New Smple
x.Show
```

Pokrenite aplikaciju i kliknite na naredbeni gumb nekoliko puta. Pomaknite najgornju formu na stranu. Budući da je forma klas s vidljivim sučeljem, možete vidjeti dodane kopije. Svaka forma ima iste kontrole, na istim položajima kao i forma tijekom izrade aplikacije.

> **Napomena** Kako bi varijabla forme i primjer učitane forme bili stalni, upotrijebite varijable tipa Static ili Public umjesto lokalne varijable.

Ključnu riječ New može koristiti i zajedno s naredbom Set. Isprobajte sljedeći programski kod u potprogramu događaja Click naredbenog gumba:

```
Dim f As Form1
Set f= New Form1
f.Caption = "pozdrav"
f.Show
```

Korištenje riječi New s naredbom Set je brži i preporučljiviji postupak.

Korištenje ključne riječi New s ostalim objektima

Ključna riječ New može se upotrijebiti za stvaranje zbirki i objekata iz klas koje ste odredili u modulima klase. Isprobajte sljedeći primjer da vidite kako to radi.

Ovaj primjer pokazuje kako ključna riječ New stvara primjere klase. Otvorite novi projekt i kreirajte naredbeni gumb na formi Form1. U izborniku Project, odaberite stavku Add Class Module kako bi projektu dodali modul klase. Postavite svojstvo Name modula klase na PokažiMi.

Sljedeći programski kod u modulu forme Form1 stvara novi primjer klase PokažiMi i poziva potprogram sdržan u modulu klase.

```
Public clsNovo As PokažiMi
    Private Sub Command1_Click()
    Set clsNovo = New PokažiMi
    clsNovo.PokažiFormu
End Sub
```

Potprogram PokažiFormu u modulu klase stvara novi primjer klase Form1, prikazuje formu, i zatim je smanjuje.

```
Sub Poka~iFormu()
    Dim frmNovo As Form1
    frmNovo.Show
    frmNovo.WindowState = 1
End Sub
```

Za korištenje ovog primjera, pokrenite aplikaciju, i nekoliko puta kliknite na naredbeni gumb. Vidjet ćete kako se na vašoj radnoj površini pojavljuje ikona smanjene forme svaki put kad klas PokažiMi stvori primjer nove forme.

Za više informacija Za informacije o korištenju ključne riječi New pri stvaranju objekata, pogledajte 10. poglavlje “Programiranje sstavnim dijelovima”.

Ograničenja ključne riječi New

Sljedeća tablica opisuje što ne možete kreirati uz pomoć ključne riječi New.

ne možete upotrijebiti New za stvaranje	primjer koda koji nije dopušten
Varijable osnovnih tipova podataka.	`Dim X As New Integer`
Varijablu općeg tipa objekta.	`Dim X As New Control`
Varijablu bilo kojeg specifičnog tipa kontrole.	`Dim X As New ListBox`
Varijablu bilo koje specifične kontrole.	`Dim X As New lstImena`

Oslobađanje pokazivača objekata

Svaki objekt koristi memoriju i sistemske izvore. Dobra je praks programiranja osloboditi te izvore kad više ne koristite objekt.

- Upotrijebite naredbu Unload za izbacivanje forme ili kontrole iz memorije.
- Upotrijebite ključnu riječ Nothing za oslobađanje izvora korištenih varijablom objekta. Dodijelite vrijednost Nothing varijabli objekta naredbom Statement.

Za više informacija Pogledajte "Događaj Unload" i "Nothing" u biblioteci *Microsoft Visual Basic 6.0 Language Reference*.

Prosljeđivanje objekata potprogramima

Možete proslijediti objekte potprogramima u Visual Basicu. U sljedećem primjeru programskog koda, pretpostavljeno je da na formi postoji naredbeni gumb:

```
Private Sub Command1_Click()
    ‘ Poziva potprogram Demo i prosljeđuje mu formu.
   Demo Form1
End Sub

Private Sub Demo(x As Form1)
    ‘ Centrira formu na ekranu.
   x.Left = (Screen.Width - x.Width) / 2
End Sub
```

Također je moguće pozivanjem proslijediti objekt argumentu, te zatim, unutar potprograma, dodijeliti argument novom objektu. Da biste vidjeli kako to radi, otvorite projekt, i dodajte drugu formu. Postavite kontrolu okvira za sliku na svaku formu. Sljedeća tabela pokazuje vrijednosti svojstava koja se trebaju promijeniti:

objekt	svojstvo	vrijednost
Okvir za sliku na drugoj formi Form2	Name	Picture2
	Picture	c:\vb\icons\arrows\arw01dn.ico

Potprogram događaja Form1_Click poziva potprogram UzmiSliku u drugoj formi Form2, i prosljeđuje mu prazni okvir za sliku.

```
Private Sub Form_Click()
Form2.UzmiSliku Picture1
End Sub
```

Potprogram UzmiSliku u formi Form2 dodjeljuje prazan okvir za sliku s forme Form1 svojstvu Picture okvira za sliku na formi Form2.

```
Private objX As PictureBox
Public Sub UzmiSliku(x As PictureBox)
    ‘ Dodjeljuje varijabli objekta
    ‘ proslijeđen okvir za sliku.
    Set objX = x
    ‘ Dodjeljuje okviru za sliku na Form1
    ‘ vrijednost svojstva Picture
    objX.Picture = Picture2.Picture
End Sub
```

Za korištenje ovog primjera, pokrenite aplikaciju, i kliknite na formu Form1. Vidjet ćete kako se ikona forme Form2 pojavljuje u okviru za sliku forme Form1.

Za više informacija Svrha prethodnih tema bila je uvod u objekte. Da biste više naučili, pogledajte 9. poglavlje “Programiranje objektima” i 10. poglavlje “Programiranje sstavnim dijelovima”.

6. POGLAVLJE

Stvaranje korisničkog sučelja

Korisničko sučelje je možda najvažniji dio aplikacije; svakako je najvidljiviji dio. Za korisnike, sučelje je aplikacija; oni vjerojatno nisu svjesni programskog koda koji se izvodi iza scene. Bez obzira na to koliko vremena i truda potrošite na pisanje i optimiziranje svog programskog koda, iskoristivost vaše aplikacije ovisi o sučelju.

Kad stvarate aplikaciju, potrebno je donijeti puno odluka što se tiče sučelja. Trebate li koristiti stil s jednim dokumentom ili s više dokumenata? Koliko ćete različitih formi trebati? Koje naredbe trebaju sadržavati vaši izbornici i hoćete li koristiti alatne trake za dupliciranje funkcija u izbornicima? Što je s dijaloškim okvirima koji međusobno djeluju s korisnikom? Koliko pomoći trebate pružiti?

Prije nego što započnete oblikovati korisničko sučelje, trebate razmisliti o namjeni aplikacije. Oblikovanje glavne aplikacije koja će se stalno koristiti trebalo bi biti drugačije od onog za aplikaciju koja će se koristiti samo povremeno u kraćem vremenskom trajanju. Aplikacija s osnovnom namjenom prikazivanja informacija ima drugačije potrebe od one koja se koristi za pribavljanje informacija.

Ciljna publika trebala bi također utjecati na vaše oblikovanje. Aplikacija namijenjena početnicima traži jednostavnost u izgledu, dok ona za iskusne korisnike može biti složenija. Ostale aplikacije koje ciljna publika koristi mogu utjecati na njihova očekivanja ponašanja aplikacije. Ako namjeravate imati međunarodnu distribuciju, jezik i kultura moraju se uzeti u obzir kao dio oblikovanja.

Oblikovanju korisničkog sučelja je najbolje pristupiti kao ponavljajućem postupku – rijetko ćete u prvom pokušaju imati savršen dizajn. Ovo poglavlje uvodi vas u postupak oblikovanja sučelja u Visual Basicu, pružajući vam informacije o alatima koje trebate pri stvaranju krasne aplikacije za vaše korisnike.

Sadržaj

- Stilovi sučelja
- Aplikacije s više dokumenata (MDI)
- Više o formama
- Korištenje izbornika u vašoj aplikaciji
- Alatne trake

- Dijaloški okviri
- Oblikovanje za različite tipove prikaza
- Oblikovanje s korisnikom na pameti

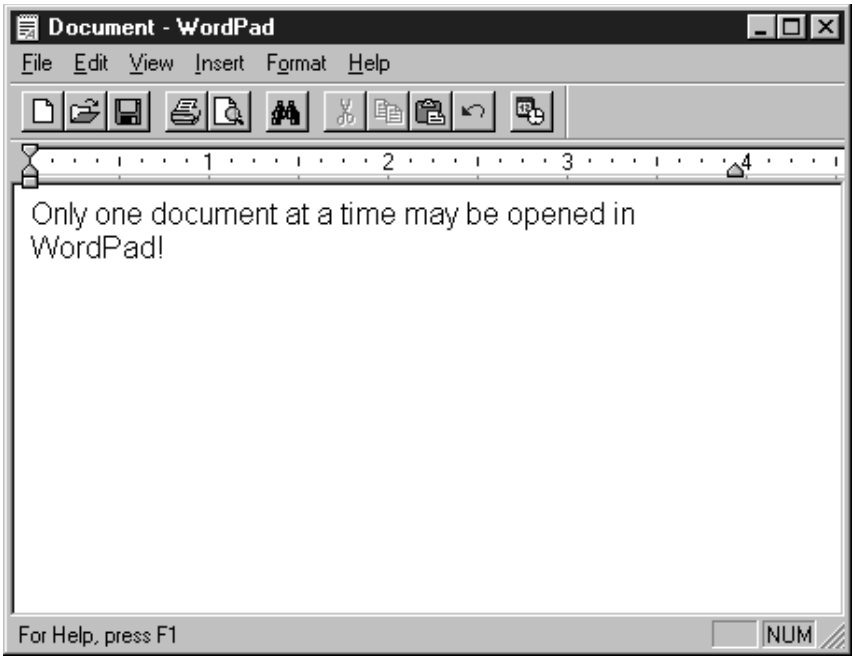
Primjeri aplikacija: Mdinote.vbp, Sdinote.vbp

Većina primjera programskog koda u ovom poglavlju uzeta je iz aplikacija Mdinote.vbp i Sdinote.vbp koje se nalaze u direktoriju Samples.

Stilovi sučelja

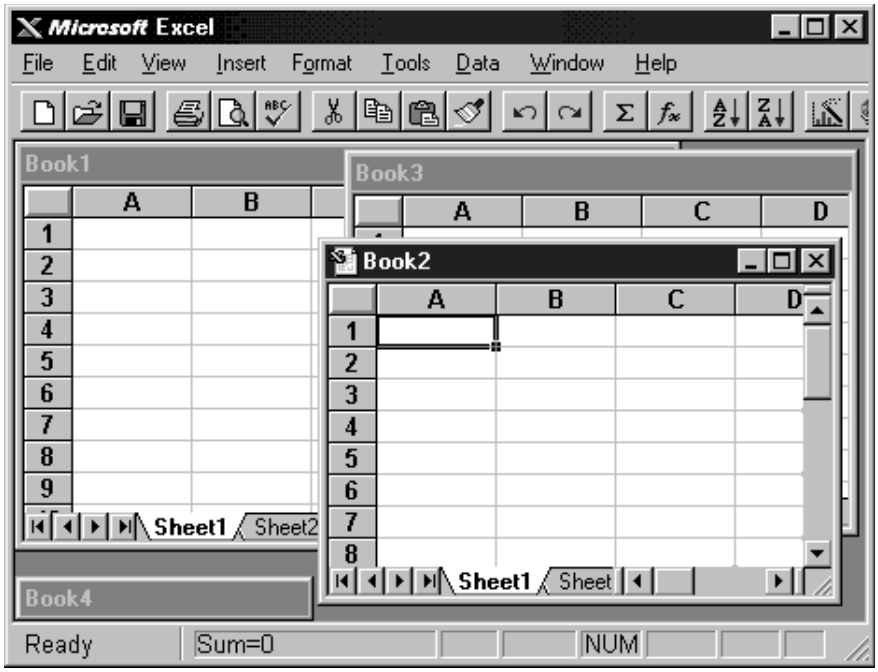
Ako ste neko vrijeme koristili aplikacije temeljene na operativnom sustavu Windows, vjerojatno ste uočili kako sva korisnička sučelja ne izgledaju isto i ne ponašaju se na isti način. Postoje dva glavna stila korisničkog sučelja: *sučelje s jednim dokumentom* (*single document interface*, SDI) i *sučelje s više dokumenata* (*multiple document interface*, MDI). Primjer SDI sučelja je aplikacija WordPad uključena u operativni sustav Microsoft Windows (slika 6.1). U WordPadu, otvoren može biti samo jedan dokument, morate zatvoriti dokument kako bi mogli otvoriti drugi.

Slika 6.1 WordPad, sučelje aplikacije s jednim dokumentom (SDI)



Aplikacije poput Microsoft Excela i Microsoft Worda za Windowse imaju MDI sučelje; ono im dopušta prikazivanje više dokumenata u isto vrijeme, s svakim dokumentom prikazanim u svom prozoru (slika 6.2). MDI aplikaciju možete reorganizirati uključivanjem izbornika Windows s podizbornicima za prebacivanje između prozora ili dokumenata.

Slika 6.2 Microsoft Excel, sučelje aplikacije s više dokumenata (MDI)

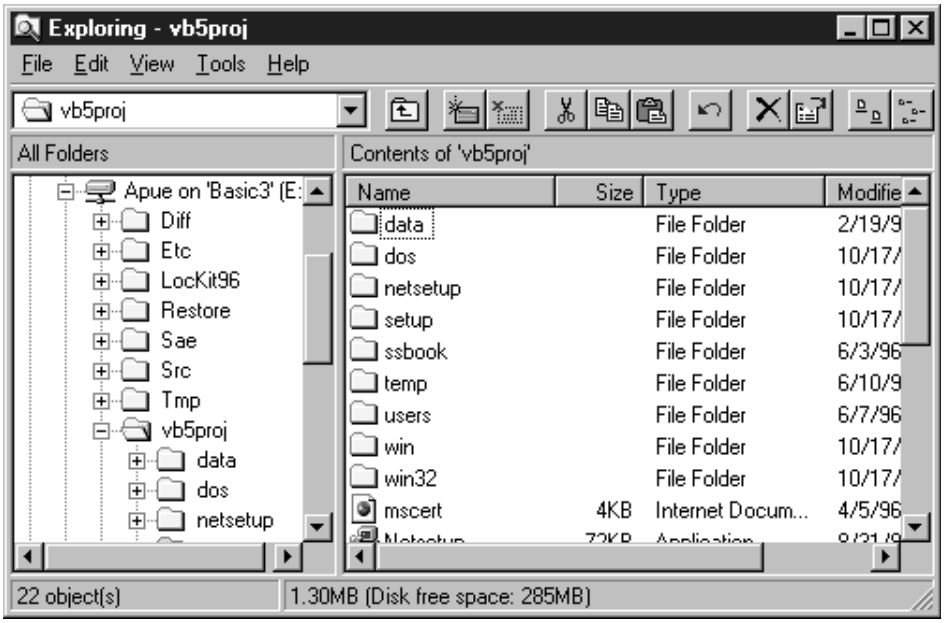


Kako bi odredili koji je stil sučelja bolji, trebate pogledati namjenu aplikacije. Apli–kacija za obradu zahtjeva za osiguranje mogla bi biti prikladna za MDI stil – činovnik će vjerojatno raditi istovremeno s više od jednog zahtjeva ili će trebati usporediti dva zahtjeva. S druge strane, aplikaciji kalendara najbolje pristaje SDI stil – nije vjerojatno da ćete istovremeno trebati otvoriti više od jednog lista kalendara; u rijetkim slučajevima kad ćete to trebati, možete otvoriti idući primjer iste SDI aplikacije.

SDI stil je uobičajeniji; većina primjera u *Microsoft Visual Basic 6.0 programerskom vodiču* pretpostavlja SDI aplikaciju. Postoji niz razmišljanja i tehnika jedinstvenih za stvaranje MDI aplikacija, koje su razmatrana u odlomku “Aplikacije s više dokumenata (MDI)” kasnije u ovom poglavlju.

Kao dodatak za dva najuobičajenija stila sučelja, SDI i MDI, treći stil sučelja postaje sve popularniji: *sučelje stila istraživača* (*explorer-style interface*, slika 6.3). Sučelje s stilom istraživača je jedan prozor koji sadrži dva *okna* ili područja, obično sastavljena od stabla ili hijerarhijskog pogleda na lijevoj i područja prikazivanja na desnoj strani, kao u Microsoft Windows Exploreru. Ovaj stil sučelja služi za kretanje ili pretraživanje većeg broja dokumenata, slika ili datoteka.

Slika 6.3 Windows Explorer, sučelje stila istraživača



Kao dodatak primjerima MDI i SDI aplikacija koje prate ovo poglavlje, čarobnjak aplikacija (application wizard) pruža dobar način za usporedbu različitih stilova sučelja. Uz pomoć čarobnjaka možete stvoriti okvir za svaki stil te vidjeti forme i programski kod koje on stvara.

Za više informacija Da biste naučili više o MDI aplikacijama, pogledajte odlomak "Aplikacije s više dokumenata (MDI)". Osnove rada s formama obrađene su u 3. poglavlju "Forme, kontrole i izbornici". Za informacije o pristupanju čarobnjaku aplikacija, pogledajte "Korištenje čarobnjaka i dodataka" u 4. poglavlju "Upravljanje projektima".

Aplikacije s više dokumenata (MDI)

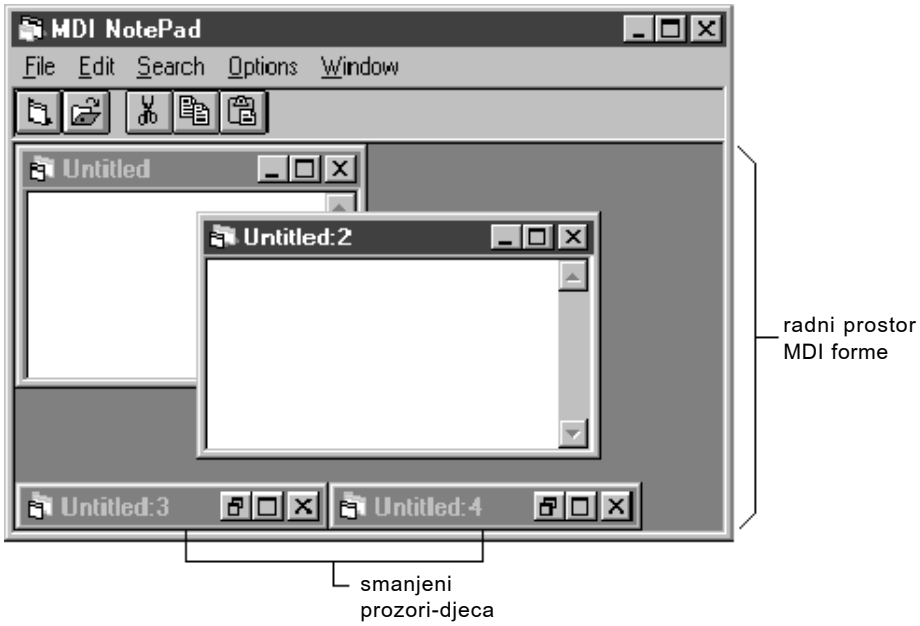
Sučelje s više dokumenata (MDI) omogućuje vam stvaranje aplikacije koja podržava više formi unutar jedne forme kao spremnika. Aplikacije poput Microsoft Excela i Microsoft Worda za Windowse imaju sučelja s više dokumenata.

MDI aplikacija omogućuje korisniku prikaz više dokumenata istovremeno, s svakim dokumentom prikazanim u vlastitom prozoru. Dokumenti ili *prozori-djeca* sadržani su u *prozoru-roditelju*, koji daje radni prostor za sve prozore-djecu u aplikaciji. Na primjer, Microsoft Excel vam dopušta stvaranje i prikazivanje više prozora s dokumentima različitog tipa. Svaki pojedinačni prozor je ograničen na područje roditeljskog prozora Excela. Kad smanjite prozor Excela, svi prozori dokumenata su također smanjeni; na traci s zadaćama se pojavljuje samo ikona prozora-roditelja.

Forma-dijete je normalna forma kojoj je svojstvo MDIChild postavljeno na True. Vaša aplikacija može uključivati puno MDI formi-djece sličnog ili različitog tipa.

Tijekom izvođenja aplikacije, forme-djeca se prikazuju unutar *radnog prostora* MDI forme-roditelja (područja unutar ruba forme te ispod naslovne trake i trake s izbornicima). Kad se smanji forma-dijete, njezina ikona će se pojaviti unutar radnog prostora MDI forme umjesto na traci s zadaćama, kao što je prikazano na slici 6.4.

Slika 6.4 Forme-djeca prikazane unutar radnog prostora MDI forme



Napomena Vaša aplikacija može uključivati i standardne, ne-MDI forme, koje nisu sadržane unutar MDI forme. Tipična primjena standardne forme u MDI aplikaciji je prikazivanje načinskog dijaloškog okvira.

MDI forma slična je uobičajenoj formi uz jedno ograničenje. Ne možete postaviti kontrolu direktno na MDI formu osim ako ta kontrola ima svojstvo Align (kao kontrola okvira za sliku) ili ako nema vidljivi dio (kao kontrola mjerača vremena).

Stvaranje MDI aplikacije

Upotrijebite sljedeći postupak za stvaranje MDI forme i njenih formi-djece.

Kako stvoriti MDI aplikaciju

1. Kreirajte MDI formu.

   U izborniku **Project**, odaberite stavku **Add MDI Form**.

   Napomena Aplikacija može imati samo jednu MDI formu. Ako projekt već ima MDI formu, naredba Add MDI Form u izborniku Project je nedostupna.

2. Kreirajte forme-djecu aplikacije.

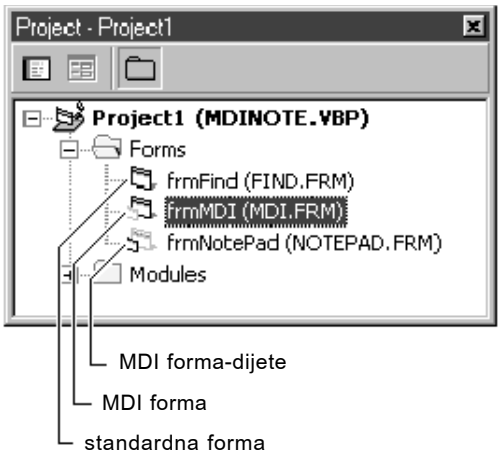
   Za kreiranje MDI forme-djeteta, kreirajte novu formu (ili učitajte postojeću) i postavite joj svojstvo MDIChild na True.

Rad s MDI formama-djecom tijekom izrade aplikacije

Tijekom izrade aplikacije, forme-djeca nisu ograničene na područje unutar MDI forme. Možete im dodavati kontrole, pisati programski kod i oblikovati svojstva kao što bi to mogli i s bilo kojom formom Visual Basica.

Možete utvrditi je li neka forma MDI dijete pogledom na njezino svojstvo MDIChild, ili ispitivanjem projektnog prozora. Ako je svojstvo MDIChild te forme postavljeno na True, to je forma-dijete. Visual Basic prikazuje posebne ikone za MDI forme i MDI forme-djecu u projektnom prozoru, kako je prikazano na slici 6.5

Slika 6.5 Ikone u projektnom prozoru označuju MDI dijete, standardnu i MDI formu

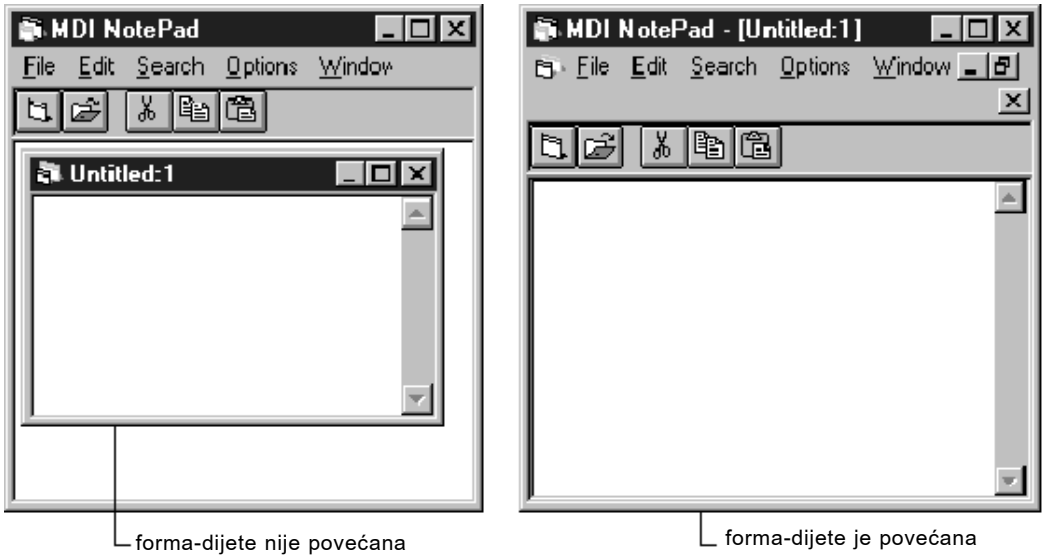


Osobine MDI formi tijekom izvođenja

Tijekom izvođenja aplikacije, MDI forme i sve njezine forme-djeca imaju specijalne karakteristike:

- Sve forme-djeca se prikazuju unutar radnog prostora MDI forme. Korisnik može micati formu-dijete i mijenjati joj veličinu kao i svakoj drugoj formi; međutim, ona je ograničena na taj radni prostor.
- Kad je forma-dijete smanjena, njezina ikona se pojavljuje unutar MDI forme umjesto na traci s zadaćama. Kad je MDI forma smanjena, MDI forma i sva njezina djeca su predstavljene jednom ikonom. Kad obnovite MDI formu, MDI forma i sve njene forme-djeca će biti prikazane u istom stanju u kakvom su bile prije smanjivanja.
- Kad se forma-dijete poveća, njezin naslov se spaja s naslovom MDI forme i prikazuje u naslovnoj traci MDI forme (pogledajte sliku 6.6).
- Određivanjem svojstva AutoShowChildren, možete automatski prikazati forme-djecu kad se učitaju (True), ili učitati forme-djecu kao skrivene (False).
- Izbornici aktivne forme-djeteta (ako ih ima) prikazani su traci s izbornicima MDI forme, a ne na formi-djetetu.

Slika 6.6 Naslov forme-djeteta spojen s naslovom MDI forme



Aplikacija MDI NotePad

Aplikacija MDI NotePad dana kao primjer je jednostavan obradnik teksta sličan aplikaciji NotePad sadržanoj u Microsoft Windowsima. Aplikacija MDI NotePad, međutim, koristi sučelje s više dokumenata (MDI). Tijekom rada aplikacije, kad korisnik zatraži novi dokument (ostvariv naredbom New u izborniku aplikacije File), aplikacija će kreirati novi primjer forme-djeteta. To omogućuje korisniku stvaranje potrebnog broja formi-djece ili dokumenata.

Za stvaranje aplikacije Visual Basicu usmjerene na dokumente, trebate barem dvije forme – MDI formu i formu-dijete. Tijekom izrade aplikacije, stvorit ćete MDI formu koja će sadržavati aplikaciju, te jednu formu-dijete koja će poslužiti kao predložak za dokument aplikacije.

Kako stvoriti vlastitu aplikaciju MDI NotePad

1. U izborniku **File** odaberite stavku **New Project**.
2. U izborniku **Project** odaberite stavku **Add MDI Form** za stvaranje forme spremnika.

   Projekt bi sad trebao sadržavati MDI formu (MDIForm1) i standardnu formu (Form1).
3. Kreirajte okvir s tekstom (Text1) na formi Form1.
4. Odredite svojstva ove dvije forme i okvira s tekstom kako slijedi.

objekt	svojstvo	vrijednost
MDIForm1	Caption	MDI NotePad
Form1	Caption	Untitled
	MDIChild	True
Text1	MultiLine	True
	Text	(prazno)
	Left	0
	Top	0

5. Upotrijebite editor izbornika odabirom stavke **Menu Editor** u izborniku **Tools** za stvaranje izbornika File na formi MDIForm1.

natpis (caption)	ime (name)	uvučen (indented)
&File	mnuFile	No
&New	mnuFileNew	Yes

6. Dodajte sljedeći programski kod u potprogram mnuFileNew_Click:

```
Private Sub mnuFileNew_Click()
  ‘ Stvaranje novog primjera forme Form1, nazvanog NewDoc.
  Dim NewDoc As New Form1
  ‘ Prikaz nove forme.
  NewDoc.Show
End Sub
```

   Ovaj potprogram stvara i zatim prikazuje novi primjer (ili kopiju) forme Form1, nazvanu NewDoc. Svaki put kad korisnik odabere stavku New u izborniku File, stvara se identičan duplikat (primjer) forme Form1, uključujući sve kontrole i programski kod koji ona sadržava.

7. Dodajte sljedeći programski kod u potprogram Form_Resize forme Form1:

```
Private Sub Form_Resize()
  ‘ Raširuje okvir s tekstom tako da popuni trenutnu formu-dijete.
  Text1.Height = ScaleHeight
  Text1.Width = ScaleWidth
End Sub
```

Programski kod potprograma događaja Form_Resize, kao i sav ostali kod forme Form1, dijeli se s svakim primjerom forme Form1. Kad je prikazano nekoliko kopija forme, svaka forma prepoznaje svoje događaje. Kad se pojavi događaj, poziva se programski kod za potprogram tog događaja. Budući da se isti kod dijeli među svim primjerima, vjerojatno se čudite kako se uputiti na formu koja je pozvala kod – posebno kad svaki primjer ima isto ime (Form1). To je raspravljeno u odlomku “Rad s MDI formama i formama-djecom”, kasnije u ovom poglavlju.

8. Pritisnite F5 za pokretanje aplikacije.

   Savjet Aplikacija Mdinote.vbp sadrži primjere mnogih MDI tehnika osim ovih spomenutih u ovom poglavlju. Uzmite vremena i prođite kroz programski kod te aplikacije kako bi otkrili te tehnike. Aplikacija Sdinote.vbp je izvedba iste aplikacije pretvorena u SDI stil; usporedite ta dva primjera kako bi naučili razlike između MDI i SDI tehnika.

Rad s MDI formama i formama-djecom

Kad korisnici vaše MDI aplikacije učitavaju, spremaju i zatvaraju više formi-djece tijekom jednog perioda rada, trebali bi imati mogućnost pozivanja aktivne forme i održavanja informacija o stanju formi-djece. Ovaj odlomak opisuje tehnike programiranja koje možete koristiti za određivanje aktivne forme-djeteta ili kontrole, učitavanje ili izbacivanje MDI formi i formi-djece, te održavanje informacija o stanju formi-djece.

Određivanje aktivne forme-djeteta ili kontrole

Ponekad želite omogućiti naredbu koja će djelovati na kontrolu s fokusom ili na trenutno aktivnu formu-dijete. Na primjer, pretpostavimo da želite kopirati odabrani tekst iz okvira s tekstom forme-djeteta u odlagalište. U aplikaciji Mdinote.vbp, događaj Click stavke Copy iz izbornika Edit poziva EditCopyProc, potprogram koji kopira selektirani tekst u odlagalište.

Budući da aplikacija ima puno primjera iste forme-djeteta, EditCopyProc treba znati koju će formu koristiti. Kako bi to odredili, upotrijebite svojstvo ActiveForm glavne MDI forme, koje će vratiti oznaku forme-djeteta koja ima fokus ili koja je posljednja bila aktivna.

> **Napomena** Kad pristupate svojstvu ActiveForm, barem jedna MDI forma-dijete mora biti učitana i vidljiva, ili će će se pojaviti pogreška.

Kad na formi imate nekoliko kontrola, također trebate odrediti i koja je kontrola aktivna. Slično svojstvu ActiveForm, svojstvo ActiveControl vraća kontrolu koja ima fokus na aktivnoj formi-djetetu. Ovdje je primjer rutine kopiranja koja može biti pozvana iz izbornika forme-djeteta, izbornika MDI forme ili gumba na alatnoj traci:

```
Private Sub EditCopyProc()
  ‘ Kopiranje selektiranog teksta u odlagalište.
  ClipBoard.SetText frmMDI.ActiveForm.ActiveControl.SelText
End Sub
```

Ako pišete programski kod koji će biti pozivan od više primjera forme, dobra je ideja *ne* koristiti identifikator forme kad pristupate kontrolama ili svojstvima forme. Na primjer, pristupite visini okvira s tekstom na formi Form1 kao Text1.Height umjesto Form1.Text1.Height. Na taj način, programski kod će uvijek djelovati na trenutnu formu.

Drugi način određivanja trenutne forme u programskom kodu je korištenje ključne riječi Me. Upotrijebite Me za poziv forme čiji se kod trenutno izvodi. Ova ključna riječ korisna je kad trebate proslijediti oznaku primjera trenutne forme kao argument za funkciju.

Za više informacija Za informacije o stvaranju višestrukih primjera forme korištenjem ključne riječi Me uz naredbu Dim, pogledajte “Uvod u varijable, konstante i tipove podataka” u 5. poglavlju “Osnove programiranja”, te “Naredba Dim” u djelu *Microsoft Visual Basic 6.0 Language Reference* biblioteke *Microsoft Visual Basic 6.0 Reference Library*.

Učitavanje MDI formi i formi-djece

Kad učitate formu-dijete, roditeljska forma (MDI forma) se automatski učitava i prikazuje. Međutim, kad učitate MDI formu, njezina djeca se ne prikazuju automatski.

U primjeru MDI NotePad, forma-dijete je određena kao početna forma, pa će se učitati i forma-dijete i MDI forma kad aplikacija počne izvođenje. Ako kao početnu formu u aplikaciji MDI NotePad postavite formu frmMDI (na kartici General u dijaloškom okviru Project Properties) te zatim pokrenete aplikaciju, učitat će se samo MDI forma. Prva forma-dijete će se učitati kad odaberete stavku New u izborniku File.

Možete upotrijebiti svojstvo AutoShowChildren kako bi učitali prozore MDI djece kao skrivene, i ostavili ih skrivenima sve dok ih ne prikažete korištenjem postupka Show. To vam omogućuje ažuriranje raznih detalja kao što su natpisi, položaji i izbornici, prije nego što forma-dijete postane vidljiva.

Ne možete načinski prikazati MDI formu-dijete i MDI formu (korištenjem postupka Show s argumentom vbModal). Ako želite koristiti načinski dijaloški okvir u MDI aplikaciji, upotrijebite formu kojoj je svojstvo MDIChild postavljeno na False.

Određivanje veličine i položaja forme-djeteta

Kad MDI forma-dijete ima okvir promjenjive veličine (BorderStyle = 2), Microsoft Windowsi određuju njenu početnu visinu, širinu i položaj kad je učitana. Početna veličina i položaj forme-djeteta s promjenjivom veličinom ovise o veličini MDI forme, a ne o veličini forme-djeteta za vrijeme izrade. Kad okvir MDI forme-djeteta nije promjenjiv (BorderStyle = 0, 1 ili 3), ona se učitava korištenjem vrijednosti svojstava Height i Width postavljenih za vrijeme izrade aplikacije.

Ako svojstvo AutoShowChildren postavite na False, možete promijeniti položaj MDI forme-djeteta nakon što ju učitate, ali prije nego što ju prikažete.

Za više informacija Pogledajte “Svojstvo AutoShowChildren” i “Postupak Show” u biblioteci *Microsoft Visual Basic 6.0 Language Reference*.

Održavanje informacija o stanju forme-djeteta

Kad korisnik odluči završiti rad MDI aplikacije mora imati mogućnost spremanja posla koji je napravio. Da bi to bilo moguće, aplikacija mora biti sposobna utvrditi, u bilo koje vrijeme, jesu li podaci u formi-djetetu promijenjeni nakon što su posljednji put snimljeni.

To možete napraviti određivanjem javne varijable za svaku formu-dijete. Na primjer, možete odrediti varijablu u odjeljku Declarations forme-djeteta:

```
Public boolPromjena As Boolean
```

Svaki put kad se promijeni tekst u okviru s tekstom Text1, događaj Change okvira s tekstom forme-djeteta postavlja varijablu boolPromjena na True. Možete dodati ovaj programski kod koji će ukazati da je sadržaj okvira s tekstom Text1 mijenjan nakon što je posljednji put snimljen:

```
Private Sub Text1_Change()
  boolPromjena = True
End Sub
```

Suprotno tome, svaki put kad korisnik snimi sadržaj forme-djeteta, događaj Change okvira s tekstom postavlja varijablu boolPromjena na False kako bi označio da sadržaj okvira s tekstom Text1 više ne treba biti snimljen. U sljedećem kodu, pretpostavljeno je da postoji naredba izbornika nazvana Save (mnuFileSave) i potprogram nazvan FileSave koji snima sadržaj okvira s tekstom:

```
Sub mnuFileSave_Click()
  ‘ Snima sadržaj okvira s tekstom Text1.
  FileSave
  ‘ Postavlja varijablu stanja.
  boolPromjena = False
End Sub
```

Izbacivanje MDI formi događajem QueryUnload

Zastavica boolPromjena postaje korisna kad korisnik odluči napustiti aplikaciju. To se može dogoditi kad korisnik odabere stavku Close u kontrolnom izborniku MDI forme, ili putem stavke izbornika koju ćete napraviti, kao što je stavka Exit izbornika File. Ako korisnik zatvori aplikaciju korištenjem kontrolnog izbornika MDI forme, Visual Basic će pokušati izbaciti formu.

Kad je MDI forma izbačena, poziva se događaj QueryUnload najprije za MDI formu te zatim za svaku formu-dijete koja je otvorena. Ako ni jedan dio koda u potprogramima događaja QueryUnload ne poništava događaj Unload, izbacuje se svaka forma-dijete, te se na kraju izbacuje i MDI forma.

Budući da se događaj QueryUnload poziva prije izbacivanja forme, možete dati korisniku mogućnost snimanja forme prije izbacivanja. Sljedeći programski kod koristi zastavicu boolPromjena kao bi utvrdio treba li pitati korisnika za snimanje forme-djeteta prije nego što bude izbačena. Uočite da vrijednosti javne varijable možete pristupiti bilo gdje u projektu. Ovaj kod pretpostavlja da postoji potprogram, FileSave, koji snima sadržaj okvira s tekstom Text1 u datoteku.

```
Private Sub mnuFExit_Click()
  ‘ Kad korisnik odabere File Exit u MDI aplikaciji,
  ‘ izbacuje MDI formu, pozivajući QueryUnload
  ‘ za svaku otvorenu formu-dijete.
  Unload frmMDI
  End
End Sub

Private Sub Form_QueryUnload(Cancel As Integer, _
UnloadMode As Integer)
  If boolPromjena Then
    ‘ Poziva rutinu za upit korisniku
    ‘ i snimanje ako je potrebno.
    FileSave
  End If
End Sub
```

Za više informacija Pogledajte "Događaj QueryUnload" u biblioteci *Microsoft Visual Basic 6.0 Language Reference*.

Više o formama

Kao dodatak temeljima oblikovanja formi, trebate razmisliti o početku i kraju vaše aplikacije. Postoji nekoliko tehnika dostupnih za određivanje što će korisnik vidjeti kad se pokrene vaša aplikacija. Također je važno biti svjestan procesa koji se pojavljuju kad aplikacija završava s radom.

Određivanje početne forme

U pravilu, prva forma vaše aplikacije određena je kao *početna forma*. Kad vaša aplikacija počne s izvođenjem, prikazat će se ta forma (pa je i prvi programski kod koji će se izvesti kod koji se nalazi u događaju Form_Initialize te forme). Ako želite da se prikaže druga forma se pokrene aplikacija, morate promijeniti početnu formu.

Kako promijeniti početnu formu

1. U izborniku **Project**, odaberite **Project Properties**.
2. Odaberite karticu **General**.
3. U okviru s popisom **Startup Object**, odaberite formu koju želite kao novu početnu formu.
4. Odaberite **OK**.

Pokretanje bez početne forme

Ponekad možete željeti pokretanje vaše aplikacije bez učitavanja bilo koje forme. Na primjer, možete trebati izvođenje programskog koda koji učitava datoteku s podacima te zatim prikazuje neku od različitih formi ovisno o tipu podataka. To možete napraviti stvaranjem potprograma tipa Sub nazvanog Main u standardnom modulu, kao u sljedećem primjeru:

```
Sub Main()
  Dim intStatus As Integer
  ‘ poziv funkcijskog potprograma za provjeru statusa korisnika.
  intStatus = UzmiStatusKorisnika
  ‘ Prikaz početne forme ovisno o statusu.
  If intStatus = 1 Then
    frmMain.Show
  Else
    frmLozinka.Show
  End If
End Sub
```

Ovaj potprogram mora biti tipa Sub, i ne smije se nalaziti u modulu forme. Za postavljanje potprograma Sub Main kao početnog objekta, u izborniku Project odaberite Project Properties, odaberite karticu General i u okviru Startup Object odaberite Sub Main.

Prikaz uvodne slike pri pokretanju

Ako pri pokretanju trebate izvršiti duži potprogram, kao što je učitavanje veće količine podataka iz baze podataka ili učitavanje nekoliko velikih slika, možete prikazati uvodnu sliku pri pokretanju. Uvodna slika (splash screen) je forma koja obično prikazuje informacije poput imena aplikacije, informacija o autorskim pravima i jednostavnije slike. Slika koja se prikazuje kad pokrenete Visual Basic je uvodna slika.

Za prikaz uvodne slike upotrijebite potprogram Sub Main kao početni objekt te uz pomoć postupka Show prikažite formu:

```
Private Sub Main()
  ‘ Prikaz uvodne slike.
  frmUvod.Show
  ‘ Ovdje možete dodati početne potprograme.
  ...
  ‘ Prikaz glavne forme i izbacivanje uvodne slike.
  frmGlavna.Show
  Unload frmUvod
End Sub
```

Uvodna slika zauzima pažnju korisnika dok se izvršavaju vaše početne rutine, stvarajući privid da se aplikacija brže učitava. Kad su početne rutine završene, može učitati vašu prvu formu i izbaciti uvodnu sliku.

Pri oblikovanju uvodne slike dobra je ideja održati je jednostavnom. Ako upotrijebite veće slike i puno kontrola, sama uvodna slika mogla bi se sporo učitati.

Završetak aplikacije

Aplikacija upravljana događajima prestaje raditi kad su sve forme zatvorene i nema programskog koda koji se izvodi. Ako i dalje postoji skrivena forma iako je posljednja vidljiva forma zatvorena, izgledat će kao da je aplikacija završila rad (jer nema vidljivih formi), ali će zapravo i dalje raditi dok ne budu zatvorene sve skrivene forme. Ovakva situacija se može pojaviti jer svaki pristup svojstvima ili kontrolama neučitane forme bezuvjetno učitava tu formu bez prikazivanja.

Najbolji način izbjegavanja ovog problema kod zatvaranja vaše aplikacije je osigurati izbacivanje svih formi. Ako imate više od jedne forme, možete upotrijebiti zbirku Forms i naredbu Unload. Na primjer, vaša glavna forma može sadržavati naredbeni gumb imena cmdKraj koji dopušta korisniku izlaz iz aplikacije. Ako vaša aplikacija ima samo jednu formu, potprogram događaja Click mogao bi biti jednostavan poput ovog:

```
Private Sub cmdKraj_Click()
  Unload Me
End Sub
```

Ako vaša aplikacija koristi više formi, možete izbaciti forme postavljanjem koda u potprogram događaja Unload vaše glavne forme. Možete upotrijebiti zbirku Forms kako bi bili sigurni da ste pronašli i zatvorili sve svoje forme. Sljedeći programski kod koristi zbirku formi za izbacivanje svih formi:

```
Private Sub Form_Unload (Cancel As Integer)
  Dim i As Integer
  ‘ Petlja za prolaz kroz zbirku formi
  ‘ i izbacivanje svake forme.
  For i = Forms.Count -1 To 0 Step -1
    Unload Forms(i)
  Next
End Sub
```

U nekim slučajevima trebat ćete završiti rad aplikacije bez obzira na stanje postojećih formi ili objekata. Visual Basic pruža naredbu End za takvu svrhu.

Naredba End trenutno završava rad aplikacije: nakon naredbe End ne izvodi se nikakav programski kod, i ne pojavljuju se daljnji događaji. Točnije, Visual Basic neće izvršiti potprograme događaja QueryUnload, Unload ili Terminate nijedne forme. Pokazivači objekata biti će oslobođeni, ali ako ste odredili vlastite klase, Visual Basic neće izvesti događaje Terminate objekata stvorenih iz vaših klasa.

Kao dodatak naredbi End, naredba Stop zaustavlja aplikaciju. Međutim, trebali bi koristiti naredbu Stop samo kad tražite pokreše, jer ona ne oslobađa pokazivače prema objektima.

Za više informacija Za informacije o naredbi Stop, pogledajte “Korištenje moda prekida” u 13. poglavlju “Traženje i obrada grešaka”, te odlomak “Naredba Stop” u *Microsoft Visual Basic 6.0 Language Reference*. Za informacije o zbirkama formi i oslobađanju pokazivača objekata, pogledajte 9. poglavlje “Programiranje objektima”.

Korištenje izbornika u vašoj aplikaciji

Većina jednostavnih aplikacija sastoji se od jedne forme i nekoliko kontrola, ali možete poboljšati vaše Visual Basic aplikacije dodavanjem izbornika. Ovaj dio pokazuje vam kako kreirati izbornike te kako ih koristiti u aplikaciji.

Stvaranje izbornika editorom izbornika

Editor izbornika (menu editor) možete upotrijebiti za stvaranje novih izbornika i traka s izbornicima, dodavanje novih naredbi u postojeće izbornike, zamjenu postojećih naredbi u izbornicima vašim naredbama, te izmjenu i brisanje postojećih izbornika i traka s izbornicima.

Kako prikazati editor izbornika

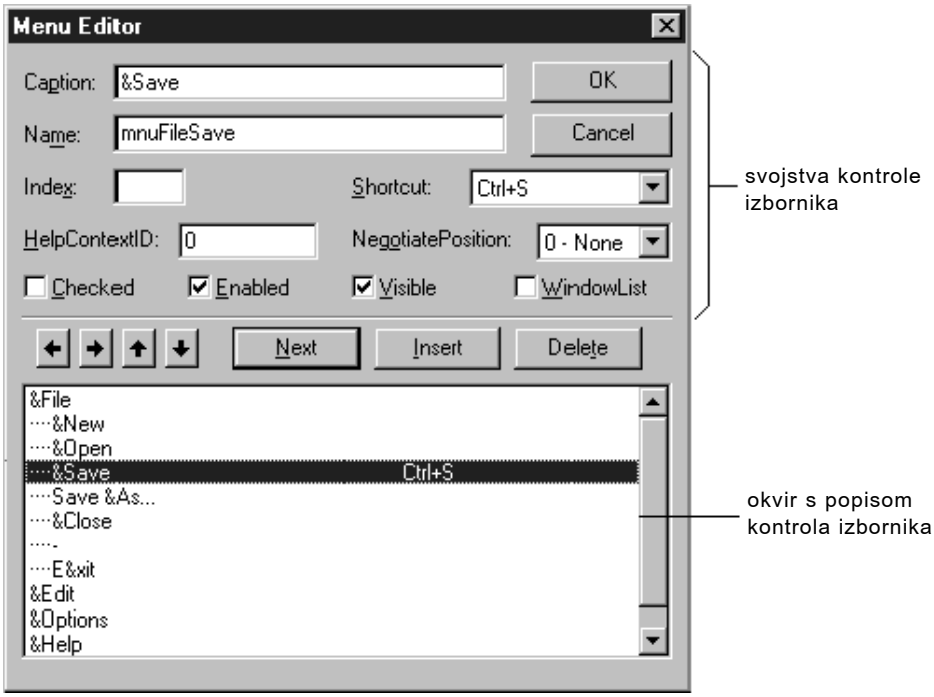
- U izborniku **Menu**, odaberite **Menu Editor**.

  - ili -

  Kliknite gumb **Menu Editor** na alatnoj traci.

Ovaj postupak će otvoriti editor izbornika, prikazan na slici 6.7.

Slika 6.7 Editor izbornika



Dok se većina svojstava kontrola u izbornicima može podesiti korištenjem editora izbornika, sva svojstva izbornika su dostupna u prozoru s svojstvima. Dva najvažnija svojstva za kontrole izbornika su:

- Ime (name) – To je ime koje ćete koristiti za poziv kontrole izbornika iz koda.
- Natpis (caption) – To je tekst koji će biti ispisan na kontroli.

Ostala svojstva u editoru izbornika, uključujući Index, Checked i NegotiatePosition, opisana su kasnije u ovom poglavlju.

Korištenje okvira s popisom u editoru izbornika

Okvir s popisom kontrola izbornika (donji dio editora izbornika) ispisuje sve kontrole izbornika za trenutnu formu. Kad u okviru s tekstom Caption upišete stavku izbornika, ta stavka će se također pojaviti i u okviru s popisom kontrola izbornika. Odabir postojeće kontrole izbornika u okviru s popisom omogućuje vam promjenu svojstava te kontrole.

Na primjer, slika 6.7 pokazuje kontrole izbornika File u tipičnoj aplikaciji. Položaj kontrola u okviru s popisom kontrola izbornika određuje je li kontrola naslov izbornika, stavka, naslov podizbornika ili stavka podizbornika:

- Kontrola koja se pojavljuje uz lijevu stranu okvira s popisom prikazuje se u traci izbornika kao naslov izbornika.
- Kontrola koja je jednom uvučena u okviru s popisom prikazuje se kao stavka u prethodnom izborniku kad korisnik klikne na njegov naslov.
- Uvučena kontrola izbornika koju slijede još uvučenije kontrole izbornika postaje naslov podizbornika. Kontrole uvučene ispod naslova podizbornika postaju stavke tog podizbornika.
- Kontrola izbornika koja kao svojstvo Caption ima crticu (-) pojavljuje se kao traka razdvajanja. *Traka razdvajanja* (*separator bar*) dijeli stavke izbornika u logičke cjeline.

  Napomena Kontrola izbornika ne može biti traka razdvajanja ako je to naslov izbornika, ako ima stavke podizbornika, ako je potvrđena ili onemogućena, ili ima prečicu kombinacijom tipki.

Kako stvoriti kontrole izbornika editorom izbornika

1. Odaberite formu.
2. U izborniku **Tools** odaberite **Menu Editor**.

   - ili -

   Kliknite gumb **Menu Editor** na alatnoj traci.
3. U okviru s tekstom **Caption**, upišite tekst naslova prvog izbornika kojeg želite na traci s izbornicima. Također, postavite znak & ispred slova za koje želite da bude pristupno slovo za taj izbornik. To slovo će u izborniku automatski biti podvučeno.

   Tekst naslova izbornika će se pojaviti u okviru s popisom kontrola izbornika.
4. U okviru s tekstom **Name**, upišite ime koje ćete koristiti za pozivanje kontrole izbornika iz programskog koda. Pogledajte “Smjernice za naslove izbornika i dodjelu imena” kasnije u ovom poglavlju.
5. Kliknite na strelicu prema lijevo ili desno kako bi promijenili stupanj uvlačenja kontrole.
6. Ako želite, odredite ostala svojstva kontrole. Možete to napraviti u editoru izbornika ili kasnije, u prozoru s svojstvima.

7. Odaberite **Next** za stvaranje iduće kontrole izbornika.

   - ili -

   Kliknite **Insert** i dodajte kontrolu izbornika između već postojećih kontrola.

   Možete također kliknuti i na strelice prema gore ili dolje kako bi pomaknuli kontrolu između postojećih kontrola izbornika.

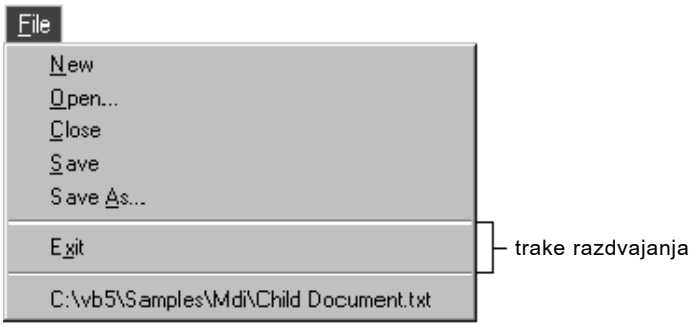
8. Odaberite **OK** za zatvaranje editora izbornika kad ste stvorili sve kontrole izbornika na toj formi.

   Naslovi izbornika koje ste kreirali prikazat će se na formi. Tijekom izrade aplikacije, klikom na naslov izbornika otvorit ćete padajući izbornik s pripadajućim stavkama.

Odvajanje stavki izbornika

Traka razdvajanja je prikazana kao vodoravna linija između stavki u izborniku. U izborniku s puno stavki, možete upotrijebiti traku razdvajanja za podjelu stavki u logičke grupe. Na primjer, izbornik File u Visual Basicu koristi trake razdvajanja za podjelu svojih stavki izbornika u tri grupe, kao što je prikazano na slici 6.8.

Slika 6.8 Trake razdvajanja



Kako stvoriti traku razdvajanja u editoru izbornika

1. Ako dodajete traku razdvajanja u postojeći izbornik, odaberite **Insert** za umetanje kontrole izbornika između stavki koje želite razdvojiti.
2. Ako je potrebno, kliknite na gumb s strelicom prema desno za uvlačenje nove stavke izbornika na istu razinu na kojoj su i stavke koje će nova kontrola razdvojiti.
3. Upišite crticu (-) u okvir s tekstom **Caption**.
4. Odredite svojstvo **Name**.
5. Kliknite na **OK** za zatvaranje editora izbornika.

   Napomena Iako se trake razdvajanja stvaraju kao kontrole izbornika, ne odgovaraju na događaj Click, i korisnici ih ne mogu odabrati.

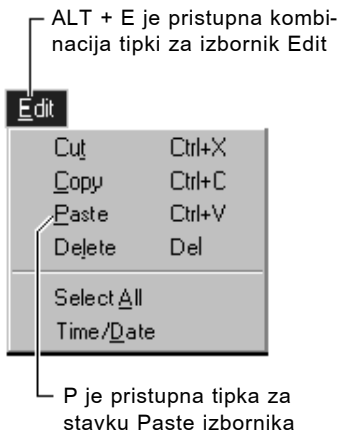
Dodjeljivanje pristupnih tipki i prečica kombinacijom tipki

Možete poboljšati pristup naredbama izbornika uz pomoć tipkovnice određivanjem pristupnih tipki i prečica kombinacijom tipki.

Pristupne tipke

Pristupne tipke (*accesss keys*) omogućuju korisniku otvaranje izbornika držanjem pritisnute tipke ALT i pritiskom na označeno slovo. Jednom kad je izbornik otvoren, korisnik može odabrati kontrolu pritiskom slova (pristupne tipke) koje joj je dodijeljeno. Na primjer, ALT+E može otvoriti izbornik Edit, a P može odabrati stavku izbornika Paste. Zadana pristupna tipka pojavljuje se kao podvučeno slovo u natpisu kontrole izbornika, kao što je prikazano na slici 6.9.

Slika 6.9 Pristupne tipke



Kako dodijeliti pristupnu tipku kontroli izbornika u editoru izbornika

1. Odaberite stavku izbornika kojoj želite dodijeliti pristupnu tipku.
2. U okviru **Caption**, upišite znak & neposredno ispred slova za koje želite da bude pristupna tipka.

Na primjer, ako je otvoren izbornik Edit prikazan na slici 6.9, sljedeće postavke svojstva Caption odgovaraju na pripadajuće tipke.

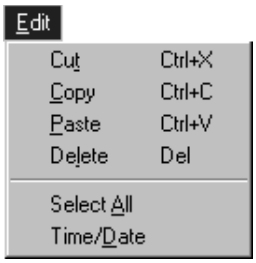
natpis kontrole	svojstvo Caption	pristupna tipka
Cut	Cu&t	t
Copy	&Copy	c
Paste	&Paste	p

natpis kontrole	svojstvo Caption	pristupna tipka
Delete	De&lete	l
Select All	Select &All	a
Time/Date	Time/&Date	d

Napomena Nemojte koristiti duplicirane pristupne tipke u izbornicima. Ako upotrijebite istu pristupnu tipku za više od jedne stavke izbornika, tipka neće raditi. Na primjer, ako je C pristupna tipka za stavke Cut i Copy, kad odaberete izbornik Edit i pritisnete C, selektirana će biti naredba Copy, ali aplikacija neće izvršiti naredbu sve dok korisnik ne pritisne ENTER. Naredba Cut neće uopće biti odabrana.

Prečice kombinacijom tipki

Prečice kombinacijom tipki (*shortcut keys*) pokreću stavku izbornika istog trenutka kad su pritisnute. Često korištenim stavkama izbornika može biti dodijeljena prečica kombinacijom tipki, koja omogućuje metodu pristupa tipkovnicom u jednom potezu, bolju od tropotezne metode pritiska tipke ALT, pristupnog znaka za naslov izbornika, pa onda pristupnog znaka za stavku izbornika. Dodjele prečica kombinacijom tipki uključuju kombinacije funkcijskih i kontrolnih tipki, kao CTRL+F1 ili CTRL+A. Ove prečice pojavljuju se u izborniku na desnoj strani pripadajuće stavke izbornika, kao što je prikazano na slici 6.10.



Slika 6.10 Prečice kombinacijom tipki

Kako dodijeliti prečicu kombinacijom tipki stavci izbornika

1. Otvorite **Menu Editor**.
2. Odaberite stavku izbornika.
3. Odaberite funkcijsku tipku ili kombinaciju tipki u kombiniranom okviru **Shortcut**.

   Za brisanje dodjele prečice, odaberite "(none)" na vrhu liste.

   Napomena Prečice kombinacijom tipki automatski se pojavljuju u izborniku; zbog toga ne morate upisati CTRL+*tipka* u okviru Caption editora izbornika.

Smjernice za naslove izbornika i dodjelu imena

Kako bi održali dosljednost s ostalim aplikacijama, dobra je ideja slijediti postojeće smjernice kod stvaranja izbornika.

Određivanje svojstva Caption

Kad stavci izbornika dodjeljujete natpis, pokušajte slijediti ove smjernice:

- Imena stavki trebaju biti jedinstvena unutar izbornika, ali se mogu ponavljati u različitim izbornicima za predstavljanje sličnih akcija.
- Imena stavki mogu se sastojati od složenica, jedne ili više riječi.
- Svako ime stavke trebalo bi imati jedinstven i lako pamtljiv znak pristupa za korisnike koji biraju naredbe tipkovnicom. Znak pristupa trebalo bi biti prvo slovo naslova izbornika, osim ako drugo slovo ne omogućuje lakše pamćenje; dva naslova izbornika ne bi smjela koristiti isti znak pristupa. Za više informacija o dodjeljivanju pristupnih tipki i prečica kombinacijom tipki, pogledajte “Stvaranje izbornika editorom izbornika” ranije u ovom poglavlju.
- Oznake nastavka izraza (...) trebaju se nalaziti iza imena naredbi koje traže više podataka prije nego što mogu biti ispunjene, kao što su naredbe za prikaz dijaloških okvira (Save As..., Preferences...).
- Neka imena stavki budu kratka. Ako ćete lokalizirati vašu aplikaciju, moglo bi biti potrebno više prostora za ispis prevedenih natpisa, a možda nećete imati dovoljno prostora za odgovarajući ispis svi vaših stavki izbornika. Za više detalja o lokaliziranju vaših aplikacija, pogledajte 16. poglavlje “Međunarodna izdanja”.

Dogovori o imenima izbornika

Kako bi vaš programski kod bio čitljiviji i lakši za održavanje, dobra je ideja slijediti postojeće smjernice imenovanja kod određivanja svojstva Name u editoru izbornika. Većina tih smjernica savjetuje prefiks za označavanje objekta (na primjer. mnu za kontrolu izbornika) iza kojeg slijedi ime glavnog izbornika (na primjer, File). Kod podizbornika, to bi trebalo biti nastavljeno natpisom podizbornika (na primjer, mnuFileOpen).

Za više informacija Za primjere predloženih dogovora imenovanja, pogledajte Dodatak B “Pravila programiranja Visual Basica”.

Stvaranje podizbornika

Svaki izbornik kojeg stvorite može sadržavati do pet razina podizbornika. *Podizbornik* se grana iz drugog izbornika kako bi prikazao svoje stavke. Podizbornik možete poželjeti upotrijebiti u sljedećim slučajevima:

- Traka s izbornicima je puna.
- Određena kontrola izbornika se rijetko koristi.
- Želite naglasiti vezu jedne kontrole izbornika s drugom.

Međutim, ako ima mjesta u traci s izbornicima, bolje je stvoriti dodatni naslov izbornika umjesto podizbornika. Na taj način, kad se otvori padajući izbornik, sve kontrole će biti vidljive korisniku. Dobra je programerska praksa i ograničeno korištenje podizbornika tako da se korisnik ne izgubi pokušavajući se kretati kroz izborničko sučelje vaše aplikacije (većina aplikacija koristi samo jednu razinu podizbornika).

U editoru izbornika, svaka kontrola izbornika koja je uvučena ispod kontrole koja *nije* naslov izbornika je *kontrola podizbornika*. Općenito, kontrole podizbornika mogu uključivati stavke podizbornika, trake razdvajanja i naslove podizbornika.

Kako stvoriti podizbornik

1. Kreirajte stavku izbornika za koju želite da bude naslov podizbornika.
2. Kreirajte stavke koje će se pojavljivati u novom podizborniku i uvucite ih klikom na gumb s strelicom prema desno.

Za svaku razinu uvlačenja ispisuju se četiri točke (....) u editoru izbornika. Za uklanjanje jedne razine uvlačenja kliknite na gumb s strelicom prema lijevo.

Napomena Ako razmišljate o korištenju više od jedne razine podizbornika, razmislite o korištenju dijaloškog okvira umjesto toga. Dijaloški okviri omogućuju korisnicima određivanje nekoliko izbora na jednom mjestu. Za informacije o korištenju dijaloških okvira, pogledajte odlomak “Dijaloški okviri” kasnije u ovom poglavlju.

Stvaranje matrice kontrola izbornika

Matrica kontrola izbornika je skup stavki izbornika u istom izborniku koje dijele isto ime i potprograme događaja. Upotrijebite matricu izborničkih kontrola za:

- Stvaranje nove stavke izbornika tijekom rada aplikacije kad ona mora biti član matrice kontrola. Aplikacija MDI Notepad, na primjer, koristi matricu kontrola izbornika za spremanje popisa nedavno otvaranih datoteka.
- Pojednostavljivanje koda, jer zajednički blokovi programskog koda mogu biti korišteni za sve stavke izbornika.

Svaki element matrice kontrola izbornika označen je jedinstvenom vrijednošću indeksa, naznačenoj u okviru svojstva Index u editoru izbornika. Kad član matrice kontrola prepozna događaj, Visual Basic prosljeđuje vrijednost svojstva Index potprogramu događaja kao dodatni argument. Vaš potprogram događaja mora sadržavati kod koji će provjeriti vrijednost svojstva Index, tako da možete odrediti koja je kontrola korištena.

Za više informacija Za dodatne informacije o matricama kontrola, pogledajte "Rad s matricama kontrola" u 7. poglavlju "Korištenje standardnih kontrola Visual Basica".

Kako stvoriti matricu kontrola izbornika u editoru izbornika

1. Odaberite formu.
2. U izborniku **Tools**, odaberite **Menu Editor**.

   - ili -

   Kliknite gumb **Menu Editor** na alatnoj traci.
3. U okviru s tekstom **Caption**, upišite tekst za naslov prvog izbornika kojeg želite na traci s izbornicima.

   Tekst naslova izbornika će biti ispisan u okviru s popisom kontrola.
4. U okviru s tekstom **Name**, upišite tekst koji ćete koristiti za pozivanje na kontrolu izbornika iz programskog koda. Okvir **Index** ostavite prazan.
5. Na sljedećoj razini uvlačenja, stvorite stavku izbornika koja će postati prvi element u matrici i odredite joj svojstva **Caption** i **Name**.
6. Postavite **Index** prvog elementa u matrici na 0.
7. Stvorite drugu stavku izbornika na istoj razini uvlačenja kao i prvu.
8. Odredite **Name** drugog elementa jednako prvom elementu i postavite **Index** na 1.
9. Ponovite korake 5-8 za sljedeće elemente matrice.

   Važno Elementi matrice kontrola izbornika moraju biti jedan do drugoga u okviru s popisom kontrola izbornika i moraju imati istu razinu uvlačenja. Kad stvarate matrice kontrola izbornika, nemojte uključiti trake razdvajanja u tom izborniku.

Stvaranje i mijenjanje izbornika tijekom izvođenja

Izbornici koje ste stvorili tijekom izrade aplikacije mogu također dinamički odgovoriti na uvjete tijekom izvođenja. Na primjer, ako akcija stavke izbornika postane neprikladna u nekom trenutku, možete spriječiti korisnika da odabere tu stavku izbornika tako da ju *onemogućite*. U aplikaciji MDI Notepad, na primjer, ako odlagalište ne sadrži nikakav tekst, stavka Paste u izborniku Edit će biti zasjenjena, i korisnik ju neće moći odabrati.

Možete također dinamički i dodati stavke izbornika, ako imate matricu kontrola izbornika. To je opisano u odlomku "Dodavanje kontrola izbornika tijekom izvođenja", kasnije u ovom poglavlju.

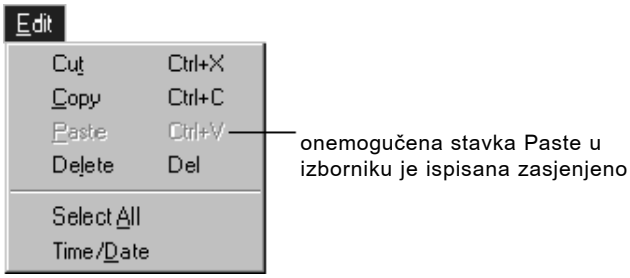
Također možete programirati vašu aplikaciju tako da koristi kvačicu kao oznaku koja je od nekoliko kontrola odabrala posljednju. Na primjer, stavka Toolbar u izborniku Options aplikacije MDI Notepad ima kvačicu ako je prikazana alatna traka. Ostale

osobine kontrola izbornika opisane u ovom dijelu uključuju programski kod koji čini stavke izbornika vidljivima ili nevidljivima te kod koji dodaje ili briše stavke izbornika.

Uključivanje i isključivanje naredbi izbornika

Sve kontrole izbornika imaju svojstvo Enabled, i kad je to svojstvo postavljeno na False, izbornik je isključen i ne odgovara na akciju korisnika. Pristup korištenjem prečice kombinacijom tipki je također onemogućen ako je svojstvo Enabled postavljeno na False. Isključena kontrola izbornika je ispisana zasjenjeno, kao stavka Paste na slici 6.11.

Slika 6.11 Onemogućena stavka izbornika



Na primjer, ovaj izraz onemogućuje stavku Paste izbornika Edit u aplikaciji MDI Notepad:

```
mnuEditPaste.Enabled = False
```

Isključivanje naslova izbornika zapravo isključuje cijeli izbornik, Budući da korisnik ne može pristupiti ni jednoj stavci, a da prvo ne klikne na naslov izbornika. Na primjer, sljedeći kod će onemogućiti izbornik Edit aplikacije MDI Notepad:

```
mnuEdit.Enabled = False
```

Prikaz kvačice uz kontrolu izbornika

Korištenjem svojstva Checked, možete postaviti kvačicu u izbornik kako bi:

- Pokazali korisniku stanje uvjeta uključeno/isključeno. Odabir naredbe izbornika naizmjenično dodaje i miče kvačicu.
- Pokazali koji je od nekoliko modova na snazi. Izbornik Options aplikacije MDI Notepad koristi kvačicu za pokazivanje stanja alatne trake, kako je prikazano na slici 6.12.

Slika 6.12 Stavka izbornika s kvačicom

Kvačice u Visual Basicu možete kreirati svojstvom Checked. Odredite početnu vrijednost svojstva Checked u editoru izbornika odabirom kontrolne kućice natpisa Checked. Za dodavanje ili brisanje kvačice uz kontrolu izbornika tijekom izvođenja aplikacije, odredite svojstvo Checked iz programskog koda. Na primjer:

```
Private Sub mnuOptions_Click()
  ‘ određuje stanje kvačice ovisno o stanju svojstva Visible.
  mnuOptionsToolbar.Checked = picToolbar.Visible
End Sub
```

Nevidljive kontrole izbornika

U editoru izbornika, određujete početnu vrijednost svojstva Visible za kontrolu izbornika odabirom kontrolne kućice natpisa Visible. Da biste kontrolu izbornika učinili vidljivom ili nevidljivom tijekom rada aplikacije, postavite njezino svojstvo Visible iz koda. Na primjer:

```
mnuFileArray(0).Visible = True ‘ kontrola postaje vidljiva.
mnuFileArray(0).Visible = False ‘ kontrola postaje nevidljiva.
```

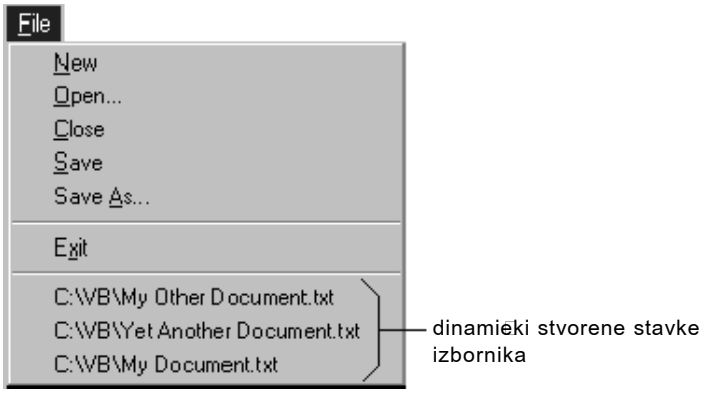
Kad je kontrola izbornika nevidljiva, ostatak kontrola u izborniku pomiče se prema gore da popuni prazan prostor. Ako je kontrola na traci s izbornicima, ostatak kontrola na toj traci pomaknut će se prema lijevo da popuni prazninu.

> **Napomena** Činjenje kontrole izbornika nevidljivom zapravo je isključuje, budući da će ta kontrola biti nedostupna kao izbornik ili uz pomoć pristupnih tipki i prečica kombinacijom tipki. Ako je nevidljiv naslov izbornika, sve kontrole tog izbornika bit će nedostupne.

Dodavanje kontrola izbornika tijekom izvođenja

Izbornik može rasti tijekom izvođenja aplikacije. Na slici 6.13, na primjer, kako se u SDI Notepad aplikaciji otvaraju datoteke, dinamički se kreiraju stavke izbornika koje prikazuju staze i imena posljednjih otvorenih datoteka.

Slika 6.13 Elementi matrice kontrola izbornika stvoreni i prikazani



tijekom izvođenja

Morate koristiti matricu kontrola za stvaranje kontrole tijekom izvođenja. Budući da je kontroli izbornika mnuRecentFile dodijeljena vrijednost svojstva Index tijekom izrade aplikacije, ona automatski postaje element matrice kontrola – čak i ako još nisu kreirani drugi elementi.

Kad stvorite mnuRecentFile(0), zapravo ste kreirali traku razdvajanja koja je nevidljiva tijekom izvođenja. Kad prvi put tijekom rada korisnik snimi datoteku, traka razdvajanja postaje vidljiva, a prvo ime datoteke se dodaje u izbornik. Svaki put kad snimite datoteku tijekom rada, u matricu se ubacuju dodatne kontrole izbornika, povećavajući izbornik.

Kontrole stvorene tijekom rada aplikacije mogu biti skrivene korištenjem postupka Hide ili postavljanjem svojstva kontrole Visible na False. Ako želite maknuti kontrolu u matrici kontrola iz memorije, upotrijebite naredbu Unload.

Pisanje koda za kontrole izbornika

Kad korisnik odabere kontrolu izbornika, pojavljuje se događaj Click. U programskom kodu trebate napisati potprogram događaja Click za svaku kontrolu. Sve kontrole izbornika osim traka razdvajanja (i isključenih ili nevidljivih kontrola) prepoznaju događaj Click.

Programski kod koji pišete u potprogramu događaja za izbornike ne razlikuje se od koda koji bi napisali u potprogramu događaja neke druge kontrole. Na primjer, kod u događaju Click stavke Close izbornika File mogao bi izgledati ovako:

```
Sub mnuFileClose_Click()
  Unload Me
End Sub
```

Visual Basic automatski prikazuje izbornik kad je odabran naslov izbornika; zbog toga nije potrebno pisati programski kod za potprogram događaja Click naslova izbornika

osim ako ne želite izvesti drugačiju akciju, kao što je onemogućavanje nekih stavki izbornika svaki put kad se izbornik prikaže.

Napomena Tijekom izrade aplikacije, izbornici koje ste kreirali pojavljuju se na formi kad zatvorite editor izbornika. Odabir stavke izbornika na formi prikazat će potprogram događaja Click za tu kontrolu izbornika.

Prikazivanje izbornika prečica

Izbornik prečica (*pop-up menu*) je plutajući izbornik koji se prikazuje iznad forme, neovisno o traci s izbornicima. Stavke prikazane u izborniku prečica ovise o tome gdje se nalazio pokazivač kad je pritisnuta desna tipka miša; zbog toga se ovi izbornici nazivaju i *kontekstni izbornici*. U Microsoft Windowsima 95/98, kontekstne izbornike aktivirate klikom desnom tipkom miša.

Svaki izbornik koji ima barem jednu stavku može tijekom izvođenja biti prikazan kao izbornik prečica. Za prikaz pomoćnog izbornika upotrijebite postupak PopupMenu. Ovaj postupak koristi sljedeću sintaksu:

[*objekt.*] **PopupMenu** *imeizbornika* [, *zastavice* [, *x* [, *y* [, *podebljananaredba*]]]]

Na primjer, sljedeći programski kod prikazuje izbornik imena mnuFile kad korisnik klikne na formu desnom tipkom miša. Možete upotrijebiti događaje MouseUp ili MouseDown za utvrđivanje kad je korisnik kliknuo desnom tipkom miša, iako se standardno koristi događaj MouseUp:

```
Private Sub Form_MouseUp (Button As Integer, Shift As _
Integer, X As Single, Y As Single)
  If Button = 2 Then      ‘ Provjera je li pritisnuta
                          ‘ desna tipka miša.
    PopupMenu mnuFile     ‘ Prikaz izbornika mnuFile
                          ‘ kao pomoćnog izbornika.
  End If
End Sub
```

Svaki programski kod koji slijedi poziv postupka PopupMenu neće biti pokrenut sve dok korisnik ne odabere stavku u izborniku ili ne odustane od izbornika.

Napomena Istovremeno može biti prikazan samo jedan izbornik prečica. Dok je izbornik prečica prikazan, zanemaruju se pozivi postupka PopupMenu. Pozivi postupka PopupMenu se također zanemaruju kad je aktivna kontrola izbornika.

Često trebate izbornik prečica za pristup izborima koji obično nisu dostupni na traci s izbornicima. Kako bi stvorili izbornik koji se neće prikazivati na traci s izbornicima, odredite glavni izbornik nevidljivim tijekom izrade aplikacije (budite sigurni da kontrolna kućica Visible u editoru izbornika nije potvrđena). Kad Visual Basic prikazuje izbornik prečica, svojstvo Visible odgovarajućeg glavnog izbornika se zanemaruje.

Argument Flags (zastavice)

Argument *Flags* koristite u postupku PopupMenu za daljnje određivanje položaja i ponašanja pomoćnog izbornika. Sljedeća tablica sadrži popis zastavica dostupnih za opis položaja pomoćnog izbornika.

konstante položaja	opis
vbPopupMenuLeftAlign	Standardno. Zadani položaj *x* određuje lijevi rub izbornika prečica.
vbPopupMenuCenterAlign	Izbornik prečica je centriran oko zadanog položaja *x*.
vbPopupMenuEightAlign	Zadani položaj *x* određuje desni rub pomoćnog izbornika.

Sljedeća tablica ispisuje zastavice dostupne za opis ponašanja izbornika prečica.

konstante ponašanja	opis
vbPopupMenuLeftButton	Standardno. Izbornik prečica se prikazuje kad korisnik klikne lijevom tipkom miša.
vbPopupMenuRightButton	Izbornik prečica se prikazuje kad korisnik klikne ili desnom ili lijevom tipkom miša.

Za određivanje zastavice, možete kombinirati po jednu konstantu iz svake grupe korištenjem operatora Or. Sljedeći programski kod prikazuje izbornik prečica s gornjim rubom centriranim na formi kad korisnik klikne naredbeni gumb. Izbornik prečica pokreće događaje Click za stavke izbornika koje su kliknute ili desnom ili lijevom tipkom miša.

```
Private Sub Command1_Click()
  ‘ Dimenzioniranje varijabli X i Y.
  Dim xpol, ypol
  ‘ Određivanje varijabli X i Y za centriranje forme.
  xpol = ScaleWidth / 2
  ypol = ScaleHeight / 2
  ‘ Prikaz izbornika prečica.
  PopupMenu mnuEdit, vbPopupMenuCenterAlign Or _
  vbPopupMenuRightButton, xpol, ypol
End Sub
```

Argument Boldcommand (podebljana naredba)

Argument *podebljana naredba* koristite za određivanje imena kontrole izbornika u prikazanom izborniku prečica koju želite ispisati podebljanim pismom. Samo jedna kontrola izbornika u izborniku prečica može biti podebljana.

Izbornici u MDI aplikacijama

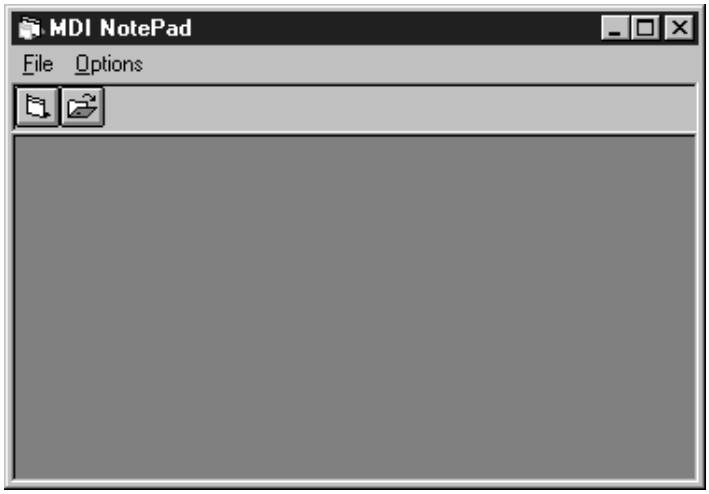
U MDI aplikaciji, izbornici za svaku formu-dijete prikazuju se na MDI formi, umjesto na samim formama-djecom. Kad forma-dijete ima fokus, izbornici te forme (ako ih ima) zamjenjuju izbornike MDI forme na traci s izbornicima. Ako nema vidljive forme-djeteta, ili forma-dijete s fokusom nema izbornike, prikazani su izbornici MDI forme (pogledajte slike 6.14 i 6.15).

Kod MDI aplikacija uobičajeno je korištenje nekoliko skupova izbornika. Kad korisnik otvori dokument, aplikacija prikazuje izbornike povezane s tim tipom dokumenta. Obično se prikazuju drugačiji izbornici ako nema vidljive forme-djeteta. Na primjer, kad nema otvorenih datoteka, Microsoft Excel prikazuje samo izbornike File i Help. Kad korisnik otvori datoteku, prikazuju se ostali izbornici (File, Edit, View, Insert, Format, Tools, Data, Window i tako dalje).

Stvaranje izbornika za MDI aplikacije

Izbornike za vašu Visual Basic aplikaciju možete kreirati dodavanjem kontrola izbornika MDI formi i formama-djecom. Jedan način upravljanja izbornicima u vašoj MDI aplikaciji je postavljanje kontrola izbornika koje želite prikazivati cijelo vrijeme, čak i kad nema vidljivih formi-djece, na MDI formu. Kad pokrenete aplikaciju, izbornici MDI forme će automatski biti prikazani kad nema vidljive forme-djeteta, kao što je prikazano na slici 6.14.

Slika 6.14 Izbornici MDI forme su prikazani kad nema vidljivih formi-djece



Kontrole izbornika koje odgovaraju formi-djetetu postavite na formu-dijete. Za vrijeme izvođenja, sve dok je vidljiva bar jedna forma-dijete, ti naslovi izbornika će biti prikazani u traci s izbornicima MDI forme.

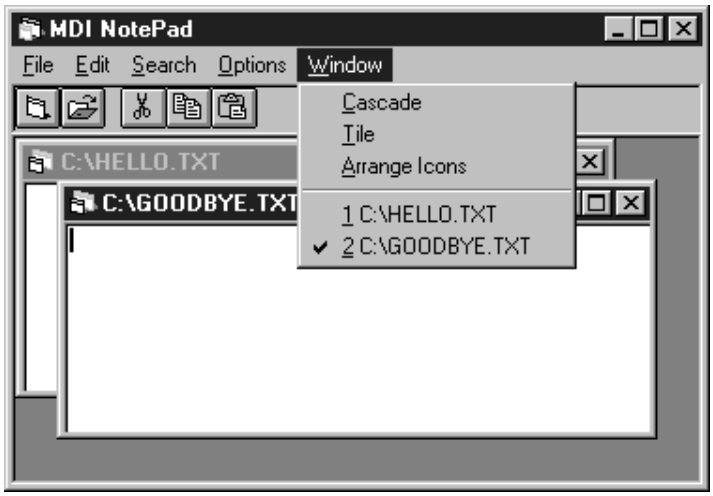
Neke aplikacije podržavaju više od jednog tipa dokumenta. Na primjer, u Microsoft Accessu, možete otvoriti tablice, upite, forme i ostale tipove dokumenata. Za stvaranje aplikacije poput takve u Visual Basicu, upotrijebite dvije forme-djeteta. Kreirajte jednu formu-dijete s izbornicima za obavljanje zadataka proračunske tablice, te drugu s izbornicima za zadatke grafikona.

Tijekom rada aplikacije, kad primjer forme proračunske tablice ima fokus, bit će prikazan izbornik za proračunsku tablicu, a kad korisnik odabere grafikon, bit će prikazani izbornici te forme. Ako su zatvorene sve proračunske tablice i grafikoni, bit će prikazani izbornici MDI forme. Za više informacija o stvaranju izbornika, pogledajte odlomak “Korištenje izbornika u vašoj aplikaciji”, ranije u ovom poglavlju.

Stvaranje izbornika Window

Većina MDI aplikacija (na primjer, Microsoft Word za Windowse i Microsoft Excel) uključuju izbornik Window. To je poseban izbornik koji prikazuje naslove svih otvorenih formi-djece, kao što je prikazano na slici 6.15. Kao dodatak, neke aplikacije u taj izbornik postavljaju i naredbe za rukovanje prozorima formi-djece, kao što su Cascade, Tile i Arrange Icons.

Slika 6.15 Izbornik Window prikazuje ime svake otvorene forme-djeteta



Svaka kontrola izbornika na MDI formi ili MDI formi-djetetu može biti upotrijebljena za prikaz popisa otvorenih formi-djece postavljenjem svojstva WindowList za tu kontrolu izbornika na True. Tijekom izvođenja aplikacije, Visual Basic automatski obrađuje i prikazuje popis naslova i prikazuje kvačicu pored onog koji je posljednji imao fokus. Kao dodatak tome, iznad popisa prozora automatski se postavlja i traka razdvajanja.

Kako postaviti svojstvo WindowList

1. Odaberite formu na kojoj želite prikazati izbornik, i u izborniku **Tools**, odaberite **Menu Editor**.

   Napomena Svojstvo WindowList odgovara samo MDI formama i MDI forma-ma-djeci. Ono nema učinka na standardne (ne-MDI) forme.

2. U okviru s popisom editora izbornika, odaberite izbornik u kojem želite prikazati popis otvorenih formi-djece.

3. Potvrdite kontrolnu kućicu **WindowList**.

Tijekom izvođenja aplikacije, taj izbornik prikazuje popis otvorenih formi-djece. Kao dodatak, svojstvo WindowList te kontrole izbornika vraća vrijednost True.

Za više informacija Pogledajte "Svojstvo WindowList" u biblioteci *Microsoft Visual Basic 6.0 Language Reference*.

Uređivanje formi-djece

Kao što je prije spomenuto, neke aplikacije u izborniku ispisuju akcije poput Tile, Cascade i Arrange Icons, zajedno s popisom otvorenih formi-djece. Upotrijebite postupak Arrange za uređivanje formi-djece unutar MDI forme. Forme-djecu možete prikazati kao kaskadno raspoređene ili vodoravno popločene prozore te kao ikone formi-djece posložene u donjem dijelu MDI forme. Sljedeći primjer pokazuje potprograme događaja Click za kontrole izbornika Cascade, Tile i Arrange Icons.

```
Private Sub mnuWCascade_Click()
  ‘ Kaskadno raspoređene forme-djeca.
  frmMDI.Arrange vbCascade
End Sub

Private Sub mnuWTile.Click()
  ‘ Vodoravno popločavanje formi-djece.
  frmMDI.Arrange vbTileHorizontal
End Sub

Private Sub mnuWArrange_Click()
  ‘ Ure|ivanje ikona svih formi-djece.
  frmMDI.Arrange vbArrangeIcons
End Sub
```

Napomena Ugrađene konstante vbCascade, vbTileHorizontal i vbArrangeIcons postoje u biblioteci objekata Visual Basica (VB) u pretraživaču objekata.

Kad kaskadno ili popločano uredite forme-djecu koje imaju nepromjenjiv stil okvira, svaka forma-dijete će biti postavljena kao da ima promjenjivi okvir. To može uzrokovati preklapanje formi-djece.

Alatne trake

Sučelje izbornika vaše aplikacije možete još više poboljšati alatnim trakama. Alatne trake sadržavaju gumbe, koji omogućavaju brz pristup najčešće korištenim naredbama u aplikaciji. Na primjer, alatna traka Visual Basica sadrži gumbe za izvođenje obično korištenih naredbi, kao što su otvaranje postojećih projekata ili snimanje postojećeg projekta.

Stvaranje alatne trake

Alatna traka (toolbar, također nazvana i vrpca ili kontrolna traka) postala je standardna osobina u većini Windows-temeljenih aplikacija. Alatna traka pruža brz pristup najčešće korištenim naredbama izbornika u aplikaciji. Stvaranje alatne trake je lako i prikladno korištenjem kontrole alatne trake, koja je dostupna u Professional i Enterprise verzijama Visual Basica. Ako koristite Learning verziju Visual Basica, alatnu traku možete stvoriti ručno kao što je opisano u odlomku “Pregovaranje o pojavljivanju izbornika i alatne trake” kasnije u ovom poglavlju.

Sljedeći primjer pokazuje stvaranje alatne trake za MDI aplikaciju; postupak stvaranja alatne trake na standardnoj formi je u osnovi isti.

Kako ručno stvoriti alatnu traku

1. Postavite okvir za sliku na MDI formu.

   Širina okvira za sliku automatski se rasteže kako bi popunila širinu radnog prostora MDI forme. Radni prostor je područje unutar okvira forme, koje ne uključuje naslovnu traku, traku s izbornicima, te nijednu alatnu traku, statusnu traku ili trake za pomicanje koje mogu biti na formi.

   > Napomena Direktno na MDI formu možete postaviti samo one kontrole koje podržavaju svojstvo Align (okvir za sliku je jedina standardna kontrola koja podržava to svojstvo).

2. U okvir za sliku postavite sve kontrole koje želite prikazati na alatnoj traci.

   U pravilu, stvarate gumbe za alatnu traku korištenjem naredbenih gumbi ili kontrola slike. Slika 6.16 prikazuje alatnu traku koja sadrži kontrole slike.

   Kako bi dodali kontrolu u okvir za sliku, kliknite gumb kontrole u alatnom okviru, i kreirajte kontrolu unutar okvira za sliku.

   > Napomena Kad MDI forma sadrži okvir za sliku, unutarnje područje MDI forme ne uključuje područje okvira za sliku. Na primjer, svojstvo ScaleHeight MDI forme vraća unutarnju visinu MDI forme, koja ne uključuje visinu okvira za sliku.

Slika 6.16 Gumbe alatne trake možete stvoriti korištenjem kontrola

slike

3. Postavite svojstva tijekom izrade.

   Jedna od prednosti alatne trake je što korisniku možete predstaviti grafički izgled naredbe. Kontrola slike je dobar izbor za gumb alatne trake jer je možete upotrijebiti za prikazivanje slike. Odredite njezino svojstvo Picture tijekom izrade aplikacije tako da prikazuje sliku; to pruža korisniku vizualan ključ naredbe koja će biti izvršena kad se klikne gumb. Možete također koristiti i *podsjetnike* (*tooltips*), koji mogu prikazivati ime gumba alatne trake kad korisnik ostavi pokazivač u mirovanju neko vrijeme iznad gumba, određivanjem svojstva ToolTipText za taj gumb.

4. Napišite programski kod.

   Budući da se gumbi alatne trake često koriste za pružanje lakog pristupa ostalim naredbama, većinu vremena će pozivati ostale potprograme, kao što je odgovarajuća naredba izbornika, iz događaja Click svakog gumba.

   Savjet Kontrole koje su nevidljive tijekom rada aplikacije (kao kontrola mjerača vremena) možete koristiti s MDI formom bez prikazivanja alatne trake. Da biste to napravili, postavite okvir za sliku na MDI formu, postavite kontrolu u okvir za sliku, i postavite svojstvo Visible okvira za sliku na False.

Pisanje koda za alatne trake

Alatne trake služe kako bi korisniku pružile brz način pristupa nekim naredbama aplikacije. Na primjer, prvi gumb alatne trake na slici 6.16 je prečica za naredbu File New. U aplikaciji MDI Notepad sad postoje tri mjesta na kojima korisnik može zatražiti novu datoteku:

- Na MDI formi (stavka New u izborniku File MDI forme).
- Na formi-djetetu (stavka New u izborniku File forme-djeteta).
- Na alatnoj traci (gumb File New)

Bolji način od kopiranja tog koda tri puta je uzimanje originalnog koda iz događaja mnuFileNew_Click forme-djeteta i postavljanje u javni potprogram forme-djeteta. Taj potprogram možete pozivati iz bilo kojeg prethodno spomenutog potprograma. Evo primjera:

```
‘ Ovaj potprogram je javni potprogram.
Public Sub FileNew()
  Dim frmNewPad As New frmNotePad
  frmNewPad.Show
End Sub

‘ Korisnik je odabrao New u izborniku File forme-djeteta.
Private Sub mnuchildFileNew_Click()
  FileNew
End Sub

‘ Korisnik je odabrao New u izborniku File MDI forme.
Private Sub mnumdiFileNew_Click()
  frmNotePad.FileNew
End Sub

‘ Korisnik je kliknuo na gumb File New alatne trake.
Private Sub btnFileNew_Click()
  frmNotePad.FileNew
End Sub
```

Pregovaranje o pojavljivanju izbornika i alatne trake

Kad se na formi aktivira objekt dobavljen od druge aplikacije, postoji više načina za prikazivanje izbornika i alatnih traka tog objekta na formi koja će ga sadržavati; zbog toga trebate odrediti kako će oni biti prikazani. Ovaj postupak se naziva *pregovaranje o korisničkom sučelju* (*user-interface negotiation*) jer se Visual Basic i objekti koje ste povezali ili umetnuli moraju dogovoriti o prostoru u formi spremniku.

Kontroliranje pojavljivanja izbornika

Postavljanjem svojstva NegotiateMenus forme možete odrediti hoće li se na formi spremniku pojaviti izbornik povezanih ili umetnutih objekata. Ako je svojstvo NegotiateMenus forme-djeteta postavljeno na True (standardno) i spremnik ima definiranu traku izbornika, izbornici objekta će biti postavljeni na traku izbornika spremnika kad se objekt aktivira. Ako spremnik nema traku s izbornicima, ili ako je svojstvo NegotiateMenus postavljeno na False, izbornici objekta se neće pojaviti kad je objekt aktiviran.

Napomena Svojstvo NegotiateMenus ne primjenjuje se za MDI forme.

Kontroliranje pojavljivanja alatne trake

Svojstvo NegotiateToolbars MDI forme određuje hoće li alatne trake povezanih ili umetnutih objekata biti plutajući okviri, ili će biti postavljene na formu-roditelja. Ovo ponašanje ne zahtijeva prisutnost alatnih traka na roditeljskoj MDI formi. Ako je svojstvo NegotiateToolbars MDI forme postavljeno na True, alatna traka objekta pojavljuje se na MDI formi. Ako je svojstvo NegotiateToolbars postavljeno na False, alatna traka objekta bit će plutajući okvir.

Napomena Svojstvo NegotiateToolbars vrijedi *samo* za MDI forme.

Ako MDI forma sadrži alatnu traku, ona je obično sadržana u kontroli okvira za sliku na roditeljskoj formi. Svojstvo Negotiate okvira za sliku određuje hoće li alatna traka spremnika i dalje biti prikazana ili će biti zamijenjena alatnom trakom objekta kad je aktiviran. Ako je svojstvo Negotiate postavljeno na True, alatna traka objekta se prikazuje kao dodatak alatnoj traci spremnika. Ako je svojstvo Negotiate postavljeno na False, alatna traka objekta zamjenjuje alatnu traku spremnika.

Napomena Pregovaranje izbornika i alatne trake pojavljuje se samo kod umetnutih objekata koji podržavaju zamjensko aktiviranje. Za više informacija o zamjenskom aktiviranju pogledajte 10. poglavlje “Programiranje sastavnim dijelovima”.

Možete vidjeti kako se ova tri svojstva isprepleću korištenjem sljedećeg postupka.

Kako izvesti pregovaranje izbornika i alatne trake

1. Dodajte alatnu traku na MDI formu. To je opisano u odlomku “Stvaranje alatne trake” prije u ovom poglavlju.
2. Postavite umetnuti objekt na formu-dijete.
3. Odredite svojstva **NegotiateMenus**, **NegotiateToolbars** i **Negotiate**.
4. Pokrenite aplikaciju i dvokliknite objekt.

Dijaloški okviri

U aplikacijama temeljenim na Windowsima, dijaloški okviri se koriste za:

- Upit korisniku za podatke potrebne nastavku rada aplikacije.
- Prikaz informacija korisniku.

U Visual Basicu, na primjer, koristite dijaloški okvir File Open za prikaz postojećih projekata. Dijaloški okvir About u Visual Basicu je također primjer kako upotrijebiti dijaloški okvir za prikaz informacija. Kad korisnik klikne stavku About Visual Basic u izborniku Help na traci s izbornicima, prikazuje se dijaloški okvir About.

Modalni i nemodalni dijaloški okviri

Dijaloški okviri su modalni ili nemodalni. *Modalni* dijaloški okvir mora biti zatvoren (skriven ili izbačen) prije nego što možete nastaviti rad s ostatkom aplikacije. Na primjer, dijaloški okvir je modalni ako od vas traži klik na gumbe OK ili Cancel prije nego što se možete prebaciti na drugu formu ili dijaloški okvir.

Dijaloški okvir About u Visual Basicu je modalni. Dijaloški okviri koji prikazuju važne informacije bi uvijek trebali biti modalni – znači, korisnik bi uvijek trebao zatvoriti dijaloški okvir ili odgovoriti na njegovu poruku prije nastavka rada.

Nemodalni dijaloški okviri dopuštaju vam prebacivanje fokusa između dijaloškog okvira i druge forme bez potrebe za zatvaranjem dijaloškog okvira. Možete nastaviti raditi bilo gdje u trenutnoj aplikaciji dok je prikazan dijaloški okvir. Nemodalni dijaloški okviri su rijetki. Odabirom stavke Find u izborniku Edit dobit ćete dijaloški okvir Find koji je primjer ovakvog dijaloškog okvira. Nemodalne dijaloške okvire koristite za prikaz često korištenih naredbi ili informacija.

Kako prikazati formu kao modalni dijaloški okvir

- Upotrijebite postupak Show s vrijednošću vbModal argumenta *stil* (konstanta za vrijednost 1).

  Na primjer:

  ```
  ‘ Prikazuje frmAbout kao modalni dijaloški okvir.
  frmAbout.Show vbModal
  ```

Kako prikazati formu kao nemodalni dijaloški okvir

- Upotrijebite postupak Show bez argumenta *stil*.

  Na primjer:

  ```
  ‘ Prikazuje frmAbout kao nemodalni dijaloški okvir.
  frmAbout.Show
  ```

  Napomena Ako se forma prikazuje kao modalna, programski kod koji slijedi postupak Show neće biti izvršen sve dok se ne zatvori dijaloški okvir. Međutim, kad je forma prikazana kao nemodalna, programski kod iza postupka Show bit će izvršen odmah nakon prikazivanja forme.

Postupak Show ima još jedan neobavezan argument, *vlasnik*, koji može biti upotrijebljen za određivanje odnosa roditelj-dijete za formu. Tim argumentom možete proslijediti ime forme kako bi tu formu učinili vlasnikom nove forme.

Kako prikazati formu kao dijete druge forme

- Upotrijebite postupak Show s argumentima *stil* i *vlasnik*.

  Na primjer:

```
‘ Prikazuje frmAbout kao nemodalno dijete forme frmMain.
frmAbout.Show vbModeless, frmMain
```

Korištenje argumenta *vlasnik* uz postupak Show osigurava smanjivanje dijaloškog okvira kad se smanji njegov roditelj, ili izbacivanje ako roditeljska forma treba biti zatvorena.

Korištenje predefiniranih dijaloških okvira

Najlakši način dodavanja dijaloškog okvira vašoj aplikaciji je korištenje predefiniranog dijaloga, jer se ne morate brinuti o oblikovanju, učitavanju ili prikazivanju dijaloškog okvira. Međutim, vaše kontroliranje njegovog izgleda je ograničeno. Predefinirani dijaloški okviri su uvijek modalni.

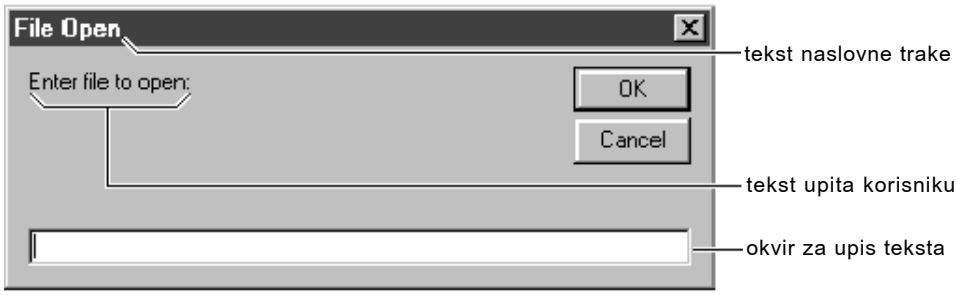
Sljedeća tablica sadrži funkcije koje možete upotrijebiti za dodavanje predefiniranih dijaloških okvira vašoj aplikaciji u Visual Basicu.

upotrijebite ovu funkciju	kako bi uradili ovo
funkcija InputBox	Prikaz naredbenog upita u dijaloškom okviru, te vraćanje onog što je korisnik upisao.
funkcija MsgBox	Prikaz poruke u dijaloškom okviru, te vraćanje vrijednosti koja ukazuje na naredbeni gumb koji je korisnik kliknuo.

Upit unosa funkcijom InputBox

Upotrijebite funkciju InputBox za prihvat podataka od korisnika. Ova funkcija prikazuje modalni dijaloški okvir koji od korisnika traži upis nekog podatka. Okvir za upis podataka prikazan na slici 6.17 traži od korisnika upis imena datoteke koju treba otvoriti.

Slika 6.17 Dijaloški okvir korištenjem funkcije InputBox



Sljedeći programski kod prikazuje okvir za upis podataka prikazan na slici 6.17:

```
FileName = InputBox("Enter file to open:" "File Open")
```

Napomena Zapamtite da kod korištenja funkcije InputBox imate malu kontrolu nad dijelovima okvira za upis podataka. Možete promijeniti samo tekst u naslovnoj traci, upit koji je ispisan korisniku, položaj okvira za upis podataka na ekranu, te hoće li prikazivati ili ne gumb Help.

Za više informacija Pogledajte "Funkcija InputBox" u biblioteci *Microsoft Visual Basic 6.0 Language Reference*.

Prikaz informacija funkcijom MsgBox

Upotrijebite funkciju MsgBox za dobivanje odgovora tipa da ili ne od korisnika, prikaz kratkih poruka kao što su pogreške, upozorenja ili uzbuna, u dijaloškom okviru. Nakon čitanja poruke, korisnik bira naredbeni gumb za zatvaranje dijaloškog okvira.

Aplikacija imena Text Editor može prikazati dijaloški okvir poruke prikazan na slici 6.18 ako datoteka ne može biti otvorena.

Slika 6.18 Dijaloški okvir s porukom pogreške stvoren funkcijom

MsgBox

Sljedeći programski kod prikazuje okvir s porukom prikazan na slici 6.18:

```
MsgBox "Error encountered while trying to open file, _
please retry.", vbExclamation "Text Editor"
```

Napomena Modalnost može biti ograničena na aplikaciju ili na sustav. Ako je modalnost okvira s porukom ograničena na aplikaciju (standardno), korisnik se ne može prebaciti na drugi dio aplikacije sve dok dijaloški okvir nije otpušten, ali se može prebaciti na drugu aplikaciju. Okvir s porukom modalni na razini sustava ne dopušta korisniku prebacivanje na drugu aplikaciju sve dok se okvir s porukom ne zatvori.

Za više informacija Pogledajte "Funkcija MsgBox" u biblioteci *Microsoft Visual Basic 6.0 Language Reference*.

Korištenje formi kao korisničkih dijaloških okvira

Korisnički dijaloški okvir je vaša forma s kontrolama – uključujući naredbene gumbe, gumbe izbora i okvire s tekstom – koja omogućuje korisniku da dostavi informacije aplikaciji. Izgled forme možete prilagoditi određivanjem vrijednosti njenih svojstava. Trebate također napisati kod za prikaz dijaloškog okvira tijekom rada aplikacije.

Za stvaranje korisničkog dijaloškog okvira, možete započeti novom formom ili prilagođivanjem postojećeg dijaloškog okvira. Tijekom vremena, moći ćete napraviti zbirku dijaloških okvira koji mogu biti korišteni u puno aplikacija.

Kako prilagoditi postojeći dijaloški okvir

1. U izborniku **Project**, odaberite **Add Form** za dodavanje postojeće forme u vaš projekt.
2. U izborniku **File**, odaberite **Save** *imedatoteke* **As** i upišite novo ime (tako sprečavate izradu promjena na postojećoj verziji forme).
3. Prilagodite izgled forme prema potrebama.
4. Prilagodite potprograme događaja u kodnom prozoru.

Kako stvoriti novi dijaloški okvir

1. U izborniku **Project**, odaberite **Add Form**.

   - ili -

   Kliknite gumb **Form** na alatnoj traci za stvaranje nove forme.
2. Prilagodite izgled forme prema potrebama.
3. Prilagodite potprograme događaja u kodnom prozoru.

Imate znatnu slobodu određivanja izgleda korisničkog dijaloškog okvira. No može biti nepomičan ili pomičan, modalni ili nemodalni. Može sadržavati različite tipove kontrola; ipak, dijaloški okviri obično ne uključuju trake s izbornicima, trake za pomicanje, gumbe za povećavanje i smanjivanje, statusne trake i rubove promjenjive veličine. Ostatak ove teme raspravlja načine stvaranja tipičnih stilova dijaloških okvira.

Dodavanje naslova

Dijaloški okvir bi uvijek trebao imati naslov koji ga označava. Za stvaranje naslova, postavite u svojstvo Caption forme tekst koji će se ispisati u naslovnoj traci. Uobičajeno je to napraviti tijekom izrade aplikacije korištenjem prozora s svojstvima, ali možete to uraditi i iz programskog koda. Na primjer:

```
frmAbout.Caption = "About"
```

Savjet Ako potpuno želite ukloniti naslovnu traku, postavite svojstva forme ControlBox, MinButton i MaxButton na False; postavite svojstvo BorderStyle na vrijednost koja ne dopušta promjenu veličine forme (0, 1 ili 3); sadržaj svojstva Caption neka bude prazni string ("").

Određivanje svojstava standardnih dijaloških okvira

Općenito, korisnik odgovara na dijaloški okvir pružanjem informacije te zatvaranjem dijaloškog okvira naredbenim gumbima OK ili Cancel. Budući da je dijaloški okvir privremen, korisnik ga obično ne treba micati, povećavati, smanjivati ili mu mijenjati veličinu. Kao posljedica toga, rubovi promjenjive veličine, kontrolni izbornik, te gumbi za smanjivanje i povećavanje koji dolaze s novim formama nisu neophodni na većini dijaloških okvira.

Ove dijelove možete maknuti određivanjem svojstava BorderStyle, ControlBox, MaxButton i MinButton. Na primjer, dijaloški okvir About mogao bi koristiti sljedeće postavke svojstava.

svojstvo	vrijednost	učinak
BorderStyle	1	Mijenja stil ruba u jednostruki nepromjenjivi, sprečavajući promjenu veličine dijaloškog okvira tijekom izvođenja.
ControlBox	False	Uklanja kontrolni izbornik.
MaxButton	False	Uklanja gumb za povećavanje, sprečavajući povećavanje dijaloškog okvira tijekom izvođenja.
MinButton	False	Uklanja gumb za smanjivanje, sprečavajući povećavanje dijaloškog okvira tijekom izvođenja.

Zapamtite da ako maknete kontrolni izbornik (ControlBox = False), morate korisniku omogućiti drugi način izlaza iz dijaloškog okvira. To se uobičajeno radi dodavanjem naredbenog gumba OK, Cancel ili Exit dijaloškom okviru te dodavanjem programskog koda u događaj Click tog gumba koji će sakriti ili izbaciti dijalog.

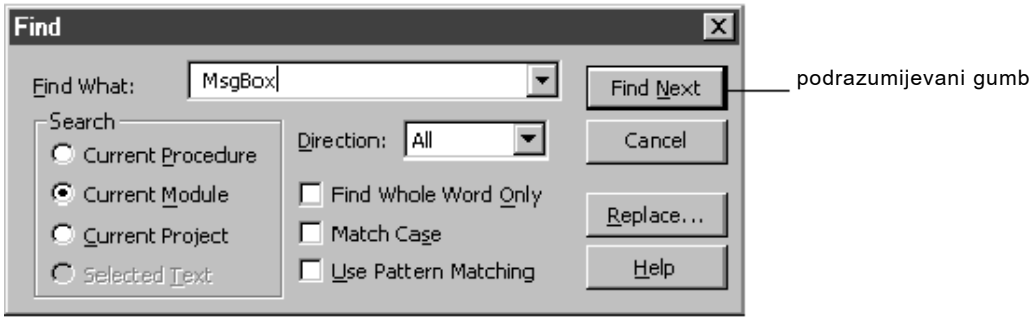
Dodavanje i postavljanje naredbenih gumbâ

Modalni dijaloški okviri moraju sadržavati bar jedan naredbeni gumb za izlaz iz dijaloškog okvira. Obično se koriste dva naredbena gumba: jedan gumb za omogućavanje korisniku da pokrene neku akciju, i jedan gumb za zatvaranje dijaloškog okvira bez pravljenja ikakvih promjena. Obično su vrijednosti svojstava Caption za te gumbe OK i Cancel. U ovakvom primjeru, naredbeni gumb OK ima svojstvo Default postavljeno na True, i gumb Cancel ima svojstvo Cancel postavljeno na True. Iako su OK i Cancel najčešće korišteni gumbi, svaka druga kombinacija natpisa radit će jednako dobro.

Dijaloški okviri koji prikazuju poruke obično koriste kontrolu natpisa za prikaz poruke o grešci ili upita korisniku, te jedan ili dva naredbena gumba za izvođenje akcije. Na primjer, svojstvu Caption kontrole natpisa možete dodijeliti poruku pogreške ili upit korisniku, i Da ili Ne svojstvima Caption dva naredbena gumba. Kad korisnik odabere Da, izvodi se jedna akcija; kad odabere Ne, pojavljuje se druga akcija.

Naredbeni gumbi na takvom tipu dijaloga obično se nalaze u donjem ili desnom dijelu dijaloškog okvira, a predodređeni gumb je gornji ili lijevi, kao što je prikazano na slici 6.19.

Slika 6.19 Položaj naredbenih gumbâ u dijaloškim okvirima



Određivanje svojstava Default, Cancel i Focus

Kontrole naredbenih gumbâ omogućuju sljedeća svojstva:

- Default
- Cancel
- TabIndex
- TabStop

Gumb s svojstvom Default bit će odabran kad korisnik pritisne ENTER. Samo jedan naredbeni gumb na formi može imati svojstvo Default s vrijednošću True. Pritisak tipke ENTER poziva događaj Click podrazumijevanog naredbenog gumba. Ova osobina djeluje povezano s kontrolom editiranja, kao što je okvir s tekstom. Na primjer, korisnik može upisati podatak u okvir s tekstom te zatim pritisnuti ENTER za izazivanje događaja Click umjesto odabira gumba OK.

Gumb s svojstvom Cancel bit će odabran kad korisnik pritisne ESC. Samo jedan naredbeni gumb na formi može imati svojstvo Cancel s vrijednošću True. Pritisak tipke ESC poziva događaj Click naredbenog gumba za odustajanje. Gumb za odustajanje može također biti podrazumijevani naredbeni gumb. Za određivanje gumba za odustajanje u dijaloškom okviru, postavite njegovo svojstvo Cancel na True.

> **Savjet** Općenito, gumb koji ukazuje na najvjerojatniju ili najsigurniju akciju trebao bi biti predodređeni gumb. Na primjer, u dijaloškom okviru za zamjenu teksta, predodređeni gumb bi trebao biti gumb za odustajanje (Cancel), a ne gumb za kompletnu zamjenu (Replace All).

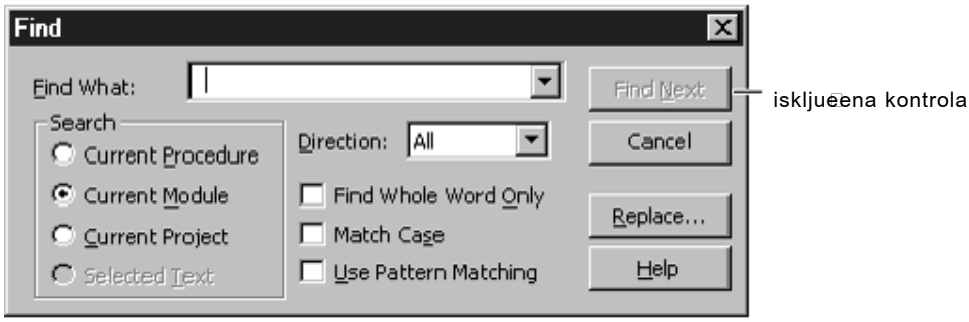
Možete također odrediti i gumb koji će imati fokus kad se prikaže dijaloški okvir. Kontrola s najmanjom vrijednosti svojstva TabIndex imat će fokus kad se prikaže forma. Pritisak na tipku ENTER pozvat će događaj Click predodređenog naredbenog gumba ili naredbenog gumba koji ima fokus. Kako bi naredbenom gumbu dali fokus kad se prikaže forma, postavite njegovo svojstvo TabIndex na 0 te svojstvo TabStop na True. Možete također upotrijebiti i postupak SetFocus za davanje fokusa određenoj kontroli kad se prikaže forma.

Za više informacija Pogledajte “Svojstvo TabIndex” i “Svojstvo TabStop” u biblioteci *Microsoft Visual Basic 6.0 Langugage Reference*.

Isključivanje kontrola na dijaloškom okviru

Ponekad kontrole trebaju biti isključene jer bi u trenutnom smislu njihove akcije bile neprikladne. Na primjer, kad se dijaloški okvir Find Visual Basica prikaže prvi put, gumb Find Next je onemogućen, kao što je prikazano na slici 6.20. Kontrolu u dijaloškom okviru možete isključiti postavljanjem njezinog svojstva Enabled na False.

Slika 6.20 Isključene kontrole na dijaloškom okviru



Kako isključiti kontrolu na dijaloškom okviru

- Postavite svojstvo Enabled te kontrole na False. Na primjer:

```
cmdFindNext. Enabled = False
cmdReplace.Enabled = False
```

Prikaz korisničkog dijaloškog okvira

Dijaloški okvir prikazujete na isti način na koji prikazujete bilo koju formu u aplikaciji. Početna forma automatski se učitava kod pokretanja aplikacije. Kad želite prikazati iduću formu ili dijaloški okvir u aplikaciji, napisat ćete programski kod koji će je učitati i prikazati. Slično tome, kad želite da forma ili dijaloški okvir nestanu, napisat ćete kod koji će ih izbaciti ili sakriti.

Sljedeći programski kod prikazuje dijaloški okvir About kad korisnik odabere stavku About u izborniku Help:

```
Private Sub mnuHelpAbout_Click()
   ‘ Koristi se postupak Show s stilom = vbModal
   ‘ za prikaz načinskog dijaloškog okvira.
   frmAbout.Show vbModal
End Sub
```

Izbori prikazivanja

Programski kod koji pišete određuje kako će se dijaloški okvir učitati u memoriju i prikazati. Sljedeća tablica opisuje razne zadatke prikazivanja formi te ključne riječi koje se koriste za njihovo izvođenje.

zadatak	ključna riječ
Učitaj formu u memoriju, ali je nemoj prikazati.	Upotrijebite naredbu Load, ili pozovite svojstvo ili kontrolu na formi.
Učitaj i prikaži nenačinsku formu.	Upotrijebite postupak Show.
Učitaj i prikaži načinsku formu.	Upotrijebite postupak Show s stilom = vbModal.
Prikaži učitanu formu.	Postavite njezino svojstvo Visible na True, ili upotrijebite postupak Show.
Sakrij formu.	Postavite njezino svojstvo Visible na False, ili upotrijebite postupak Hide.
Sakrij formu i izbaci je iz memorije.	Upotrijebite naredbu Unload.

Postupak Show učitava formu i postavlja njezino svojstvo Visible na True. Argument proslijeđen postupku Show ukazuje stil dijaloškog okvira. Ako je argument *stil* ispušten ili ima vrijednost vbModeless ili 0 (standardno), dijaloški okvir je nemodalni; ako ima vrijednost vbModal ili 1, dijaloški okvir je modalni.

Za izlaz iz dijaloškog okvira kad korisnik odabere OK ili Cancel, upotrijebite naredbu Unload ili postupak Hide. Na primjer:

```
Unload frmAbout
```

- ili -

```
frmAbout.Hide
```

Naredba Unload briše dijaloški okvir iz memorije, dok postupak Hide samo miče dijaloški okvir iz pogleda postavljajući njegovo svojstvo Visible na False. Kad izbacite formu, sama forma i sve njezzine kontrole izbacuju se iz memorije (uključujući sve kontrole koje su bile učitane tijekom rada). Kad skrijete formu, forma i njezine kontrole ostaju u memoriji.

Kad trebate sčuvati mjesto u memoriji, bolje je izbaciti formu, jer izbacivanje forme oslobađa memoriju. Ako često koristite dijaloške okvire, možete odabrati sakriti formu. Skrivanje forme čuva sve podatke koji su joj dodani, uključujući vrijednosti svojstava, izlaz ispisa, te dinamički stvorene kontrole. Skrivanjem forme, možete nastaviti s ukazivanjem na svojstva i kontrole skrivene forme u programskom kodu.

Oblikovanje za različite tipove prikaza

Microsoft Windowsi nisu ovisni o uređaju – aplikacije temeljene na Windowsima mogu se izvoditi na puno različitih računala s različitim razlučivostima i dubinama boje. Aplikacije koje napišete u Visual Basicu će također vjerojatno morati raditi na različitim tipovima zaslona; morate biti svjesni toga kad oblikujete aplikaciju.

Oblikovanje formi neovisnih o razlučivosti

U pravilu, Microsoft Visual Basic ne mijenja veličinu vaših formi i kontrola kad promijenite razlučivost ekrana. To znači da će forma koju ste oblikovali na razlučivosti 1024 s 768 izaći van rubova ekrana kad bude prikazana na razlučivosti 640 s 480. Ako želite stvoriti forme i kontrole koje će imati iste omjere bez obzira koju razlučivost ekrana koristili, morat ćete oblikovati forme na najmanjoj razlučivosti, ili ćete svojoj aplikaciji morati dodati programski kod koji mijenja forme.

Najjednostavniji način izbjegavanja problema s omjerima je oblikovanje vaših formi na razlučivosti 640 s 480. Ako više volite rad na većoj razlučivosti, i dalje trebate biti svjesni kako će vaša forma izgledati na manjoj razlučivosti. Jedan način kontrole je korištenje prozora s položajem forme u kojem možete pregledati veličinu i položaj vaše forme. Upotrijebite vodilice kako bi vidjeli koji će dio ekrana biti vidljiv pri manjim razlučivostima. Za uključivanje i isključivanje vodilica, kliknite desnom tipkom miša unutar prozora s položajem forme i odaberite stavku Resolution Guides u izborniku prečica.

Visual Basic također postavlja vašu formu tijekom izvođenja ovisno o njezinom položaju tijekom izrade aplikacije. Ako tijekom izrade radite na razlučivosti 1024 s 768 i postavite formu u donji desni kut ekrana, ona možda neće biti vidljiva kod izvođenja na manjoj razlučivosti. Kako bi to izbjegli, odredite početni položaj vaše forme tijekom izrade odabirom izbornika Startup Position u izborniku prečica prozora s položajem forme.

Alternativno, položaj forme možete odrediti tijekom izvođenja programskim kodom u događaju Form Load:

```
Private Sub Form_Load()
  Me.Move 0, 0
End Sub
```

Ovo ima isti učinak kao postavljanje svojstava forme Top i Left na 0, ali postupak Move to obavlja u jednom koraku.

Visual Basic koristi jedinicu mjere koja je neovisna o uređajima, *twip*, za izračunavanje veličine i položaja. Dva svojstva objekta Screen, TwipsPerPixelX i TwipsPerPixelY, mogu biti upotrijebljena za utvrđivanje veličine prikaza tijekom izvođenja aplikacije. Korištenjem ovih svojstava, možete napisati kod za prilagodbu veličine i položaja vaših formi i kontrola:

```
Private Sub PostaviKontrole()
  Dim X As Integer
  Dim Y As Integer
  X = Screen.TwipsPerPixelX
  Y = Screen.TwipsPerPixelY
  Select Case X, Y
    Case 15, 15
          ‘ Promjena veličine i položaja kontrola.
          txtIme.Height = 200
          txtIme.Width = 500
          txtIme.Move 200, 200
    ‘ Dodajte kod za ostale razlučivosti.
    ...
End Sub
```

Trebate također biti svjesni položaja samih prozora Visual Basica tijekom izrade aplikacije. Ako postavite projektni prozor uz desni rub ekrana na većoj razlučivosti, on možda više neće biti dostupan kad otvorite vaš projekt na manjoj razlučivosti.

Oblikovanje za različite dubine boja

Pri oblikovanju aplikacije, trebate također uzeti u obzir i mogućnosti prikazivanja boja na računalima koja bi mogla izvoditi vašu aplikaciju. Neka računala mogu prikazivati 256 ili više boja, druga su ograničena na 16 boja. Ako oblikujete formu koristeći paletu s 256 boja, posljedica *diteriranja* (*dithering,* proces kojim se simuliraju boje koje nisu dostupne) može biti nevidljivost nekih elemenata forme kad se prikazuju s 16 boja.

Da biste izbjegli tu situaciju, najbolje je ograničiti korištenje boja u vašoj aplikaciji na 16 standardnih boja Windowsa. One su predstavljene konstantama boja Visual Basica (vbBlack, vbBlue, vbCyan, i tako dalje). Ako je nužno koristiti više od 16 boja u vašoj aplikaciji, trebali bi zadržati standardne boje za tekst, gumbe i ostale elemente sučelja.

Oblikovanje s korisnikom na pameti

Osim ako ne stvarate Visual Basic aplikacije strogo za vlastite potrebe, o vrijednosti vaših kreacija presudit će drugi. Korisničko sučelje vaše aplikacije ima najveći utjecaj na mišljenje korisnika – bez obzira na to kako je tehnički briljantan ili dobro optimiziran vaš programski kod, ako korisnik uvidi da je vaša aplikacija teška za korištenje, to neće biti primljeno dobro.

Kao programer, bez sumnje ste prisni s tehnološkim aspektima računala. Lako je zaboraviti da većina korisnika ne razumije (i vjerojatno ih nije briga) tehnologiju koja se nalazi iza aplikacije. Oni vide aplikaciju kao sredstvo dolaska do kraja: način obavljanja nekog zadatka, idealno efikasniji nego što bi bio bez pomoći računala. Dobro oblikovano korisničko sučelje odjeljuje korisnika od pozadinske tehnologije, čineći jednostavnim obavljanje namjeravanog posla.

Pri oblikovanju korisničkog sučelja vaše aplikacije, trebate imati korisnika na pameti. Kako jednostavno korisnik može otkriti razne osobine vaše aplikacije bez uputa? Kako će vaša aplikacija odgovoriti kad se pojavi pogreška? Što ćete pružiti kao pomoć ili korisničku podršku? Je li dizajn estetski ugodan korisniku? Odgovori na ova i druga pitanja vezana uz dizajn usmjeren na korisnika obrađena su u ovom dijelu.

Osnove oblikovanja sučelja

Ne morate biti umjetnik da biste stvorili krasno korisničko sučelje – većina načela u oblikovanju korisničkog sučelja jednaka je temeljnim načelima oblikovanja koja se podučavaju na bilo kojem početnom tečaju umjetnosti. Početna načela oblikovanja sastava, boje i ostalog odgovaraju jednako dobro računalnom ekranu kao i listu papira ili platnu.

Iako Visual Basic čini lakim stvaranje korisničkog sučelja jednostavnim povlačenjem kontrola na formu, malo planiranja unaprijed može učiniti veliku razliku u upotrebljivosti vaše aplikacije. Mogli bi razmisliti najprije o crtanju dizanja vaše forme na papiru, utvrđujući koje su kontrole potrebne, relativnu važnost različitih elemenata, te odnose među kontrolama.

Sastav: izgled i osjećaj aplikacije

Sastav izgleda vaše forme ne utječe samo na estetski izgled, on također ima i ogroman utjecaj na upotrebljivost vaše aplikacije. Sastav uključuje čimbenike kao što su postavljanje kontrola, dosljednost elemenata, prihvatljivost, korištenje praznog prostora, te jednostavnost dizajna.

Postavljanje kontrola

U većini oblikovanja sučelja, nisu svi elementi jednake važnosti. Nužno je pažljivo oblikovanje koje će osigurati da važniji elementi odmah budu očiti korisniku. Važnim ili često pristupanim elementima treba dati istaknuti položaj; manje važni elementi trebali bi biti postavljeni na manje istaknuta mjesta.

U većini jezika, naučeni smo čitati s lijeva na desno, od vrha prema dnu stranice. Isto vrijedi i za računalni ekran – oči većine korisnika najprije će vidjeti gornji lijevi dio ekrana, pa bi najvažniji elementi trebali biti postavljeni tamo. Na primjer, ako se informacija na formi odnosi na kupca, polje s imenom trebalo bi biti prikazano tamo gdje će odmah biti uočeno. Gumbi, kao OK ili Next, trebaju biti postavljeni u donjem desnom dijelu ekrana; korisnik im uobičajeno neće pristupiti sve dok ne završe rad s formom.

Grupiranje elemenata i kontrola je također važno. Pokušajte logički grupirati informacije prema funkcijama ili odnosu. Budući da su njihove funkcije povezane, gumbi za kretanje kroz bazu podataka trebali bi biti vizualno grupirani zajedno radije nego da su rasijani po cijeloj formi. Isto vrijedi i za informacije; polja za ime i adresu se redovito grupiraju zajedno, jer su blisko povezani. U većini slučajeva, možete upotrijebiti kontrole okvira koje će vam pomoći u učvršćivanju odnosa među kontrolama.

Dosljednost elemenata sučelja

Dosljednost je vrlina oblikovanja korisničkog sučelja. Dosljedan izgled i osjećaj stvaraju sklad u aplikaciji – sve izgleda međusobno prilagođeno. Manjak dosljednosti u vašem sučelju može biti zbunjujuć, te može izgled aplikacije učiniti kaotičnim, neorganiziranim i banalnim, vjerojatno uzrokujući i sumnju korisnika u pouzdanost aplikacije.

Za vidljivu dosljednost, ustanovite strategiju oblikovanja i stilska pravila prije nego što počnete razvoj. Elementi dizajna kao što su tipovi kontrola, standardi za veličinu i grupiranje kontrola, te izbor pisama mogu unaprijed biti određeni. Možete stvoriti prototipove mogućih dizajna koji će vam pomoći u donošenju odluka o konačnom izgledu.

Široka raznolikost kontrola dostupnih za korištenje u Visual Basicu mami na korištenje svih kontrola. Izbjegavajte to iskušenje; odaberite manji dio kontrola koje najbolje odgovaraju vašoj određenoj aplikaciji. Iako za predstavljanje popisa informacija mogu biti upotrijebljeni okvir s popisom, kombinirani okvir, te kontrole mreže i stabla, najbolje je zadržati se na jednom stilu gdje je to moguće.

Također, pokušajte koristiti prikladne kontrole; iako se kontrola okvira s tekstom može postaviti kao samo za čitanje i upotrijebiti za prikazivanje teksta, kontrola natpisa je obično prikladnija za takvu namjenu. Zadržite dosljednost u postavljanju svojstava vaših kontrola – ako koristite bijelu boju pozadine za tekst na jednom mjestu, nemojte koristiti sivu na drugom osim ako za to ne postoji dobar razlog.

Dosljednost među različitim formama u vašoj aplikaciji važna je za upotrebljivost. Ako koristite sivu podlogu i trodimenzionalne efekte na jednoj formi, a bijelu podlogu na drugoj, te forme će izgledati nepovezane. Odaberite stil i zadržite ga u cijeloj aplikaciji, čak i ako to za posljedicu ima reoblikovanje nekih osobina.

Prihvatljivost: obrazac slijedi funkciju

Prihvatljivost je vidljivi ključ funkcije objekta. Iako ovaj izraz može biti neuobičajen, primjeri prihvatljivosti su svuda oko vas. Ručka na biciklu ima utore u koje postavljate svoje prste, to je prihvatljivost gdje je očigledno da je namijenjena hvatanju. Prekidači i gumbi za pritiskanje imaju prihvatljivost – samo pogledom na njih možete razabrati njihovu namjenu.

Korisničko sučelje također upotrebljava prihvatljivost. Na primjer, trodimenzionalni efekti korišteni na naredbenim gumbima daju im izgled kao da ih trebate pritisnuti. Ako ste oblikovali naredbeni gumb s ravnim rubom, izgubit ćete njegovu prihvatljivost i korisniku neće biti jasno da je to naredbeni gumb. Postoje slučajevi kad ravni gumbi mogu biti prikladni, kao u igrama ili multimedijskim aplikacijama; to je u redu sve dok zadržite dosljednost kroz aplikaciju.

Okviri s tekstom također pružaju neku vrstu prihvatljivosti – korisnik očekuje da će okvir s rubom i bijelom podlogom sadržavati tekst koji se može mijenjati. Iako je moguće prikazati okvir s tekstom bez ruba (BorderStyle = 0), to će mu dati izgled kontrole natpisa i korisniku neće biti očigledno da može mijenjati tekst.

Korištenje praznog prostora

Korištenje *praznog prostora* u vašem korisničkom sučelju može pomoći pri naglašavanju elemenata i poboljšanju upotrebljivosti. Prazan prostor nije doslovno prazan – ovaj izraz označava prostor između i oko kontrola na formi. Previše kontrola na formi može voditi u zbrkano sučelje, otežavajući nalaženje pojedinog polja ili kontrole. Prazan prostor trebate uključiti u vaš dizajn tako da naglašava elemente vašeg oblikovanja.

Dosljedan razmak između kontrola te poravnavanje vodoravnih i okomitih elemenata mogu također učiniti vaš dizajn upotrebljivijim. Kao što je tekst u časopisima uređen u sređenim kolonama s jednakim razmakom među linijama, tako je i sređeno sučelje lakše čitati.

Visual Basic pruža nekoliko alata koje jednostavno podešavaju razmak, poravnanje i veličinu kontrola. U izborniku Format možete pronaći naredbe Align, Make Same Size, Horizontal Spacing, Vertical Spacing i Center In Form.

Neka bude jednostavno

Jednostavnost je možda najvažnije načelo oblikovanja sučelja. Kad je riječ o aplikacijama, ako sučelje izgleda teško, po svoj prilici i jest. Malo razmišljanja unaprijed može vam pomoći u stvaranju sučelja koje izgleda (i je) jednostavno za korištenje. Također, s estetskog stajališta, čist i jednostavan dizajn je uvijek poželjniji.

Uobičajeni početni korak u stvaranju izgleda sučelja je pokušaj oblikovanja prema objektima iz stvarnog svijeta. Zamislite, na primjer, da je od vas zatraženo stvaranje aplikacije za ispunjavanje obrazaca osiguranja. Prirodna reakcija bilo bi oblikovanje sučelja koje na ekranu izgleda identično kao i obrazac na papiru. Takav način stvara nekoliko problema: oblik i veličina papirnatog obrasca razlikuju se od onog na ekranu, kopiranje obrasca prilično vas ograničava na okvire s tekstom i kontrolne kućice, pa nema stvarne prednosti za korisnika.

Daleko je bolje oblikovati vlastito sučelje, s omogućavanjem ispisane kopije (s pregledom ispisa) originalnog papirnatog obrasca. Stvaranjem logičkih grupa polja s originalnog obrasca te korištenjem sučelja s karticama ili s nekoliko povezanih formi, možete predstaviti sve informacije bez potrebe da korisnik odmata ekran. Možete također koristiti i dodatne kontrole, kao okvir s popisom prethodno napunjen izborima, što smanjuje količinu tipkanja korisnika.

Većinu aplikacija možete pojednostaviti postavljanjem rijetko korištenih funkcija na vlastite forme. Pružanje predodređenih vrijednosti može ponekad pojednostaviti aplikaciju; ako devet od deset korisnika odabire podebljano pismo, odredite podebljani tekst kao standard radije nego da to korisnik mora odabrati svaki put (ne zaboravite pružiti opciju za zaobilaženje standarda). Čarobnjaci također mogu pomoći u pojednostavljivanju složenih ili rijetkih zadataka.

Najbolji ispit jednostavnosti je promatranje vaše aplikacije u upotrebi. Ako tipični korisnik ne može odmah obaviti željeni posao bez pomoći, trebalo bi pristupiti ponovnom oblikovanju.

Korištenje boja i slika

Korištenje boja u vašem sučelju može povećati vidljivu privlačnost, ali se lako može pretjerati. Uz puno zaslona sposobnih prikazati milijune boja, primamljivo je upotrijebiti sve boje. Boja, kao i ostala temeljna načela oblikovanja, može biti problematična ako se pažljivo ne uzme u obzir u početnom oblikovanju.

Sklonost bojama je različita; ukus korisnika ne mora biti isti kao i vaš. Boje mogu izazvati jake osjećaje, a ako stvarate za međunarodnu publiku, neke boje mogu imati kulturološko značenje. Obično je najbolje ostati konzervativan, korištenjem mekih, neutralnijih boja.

Naravno, na vaš izbor boja može također utjecati i tip krajnjih korisnika kao i ton i raspoloženje koje želite prenijeti. Sjajne crvene, zelene ili žute boje mogu biti prikladne za aplikacije namijenjene djeci, ali će teško izazvati utisak financijskih obaveza u bankarskoj aplikaciji.

Manje količine sjajnih boja mogu efikasno biti upotrijebljene za naglašavanje ili privlačenje pozornosti na važno područje. Po iskustvenom pravilu, trebali bi ograničiti upotrebu boja u aplikaciji, a vaša shema boja trebala bi ostati dosljedna. Najbolje je stalno koristiti standardnu 16-bojnu paletu ako je moguće; diteriranje može uzrokovati nestajanje nekih boja kod rada s 16-bojnim ekranom.

Pri korištenju boja određenu važnost ima i sljepoća na boje. Mnogo ljudi nije sposobno uočiti razliku između različitih kombinacija primarnih boja kao što su crvena i zelena. Nekome s takvim stanjem, crveni tekst na zelenoj pozadini bit će nevidljiv.

Slike i ikone

Upotreba slika i ikona može također povećati vizualno zanimanje za vašu aplikaciju, ali opet, bitno je pažljivo oblikovanje. Slike mogu izraziti sžete informacije bez potrebe za tekstom, ali različiti ljudi često različito shvaćaju slike.

Alatne trake s ikonama za predstavljanje različitih funkcija su koristan dio sučelja, ali ako korisnik ne može jasno prepoznati funkciju predstavljenu ikonom, one mogu biti kontraproduktivne. Pri oblikovanju ikona alatne trake, pogledajte druge aplikacije kako bi vidjeli koji su standardi već uspostavljeni. Na primjer, puno aplikacija koristi list papira s preklopljenim kutom za predstavljanje ikone New File. Možda postoji i bolja metafora za tu funkciju, ali drugačije predstavljanje moglo bi zbuniti korisnika.

Također je važno uzeti u obzir kulturološko značenje slika. Puno aplikacija koristi sliku poštanskog sandučića seoskog stila za zastavicom (slika 6.21) za predstavljanje funkcija pošte. To je uglavnom američka ikona; korisnici u drugim zemljama ili kulturama vjerojatno je neće prepoznati kao poštanski sandučić.

Slika 6.21 Ikona koja predstavlja poštanski sandučić

Pri oblikovanju vlastitih ikona i slika, pokušajte ih održati jednostavnima. Složene slike s puno boja ne smanjuju se dobro kad se prikazuju kao 16 puta 16 piksela velike ikone na alatnoj traci, ili kad se prikazuju na većim razlučivostima.

Odabir pisama

Pisma (fontovi) su također važan dio vašeg korisničkog sučelja, jer često prenose važne informacije korisniku. Trebate odabrati pisma koja će biti lako čitljiva na različitim razlučivostima i različitim tipovima prikazivanja. Najbolje je zadržati se na jednostavnim pismima tipa sans serif ili serif, gdje je to moguće. Ukrasna pisma tipa script općenito izgledaju bolje pri ispisu nego na ekranu, i mogu biti teško čitljiva pri malim veličinama.

Osim ako ne namjeravate distribuirati pisma zajedno s svojom aplikacijom, trebate se držati standardnih pisama Windowsa kao što su Arial, New Times Roman ili System. Ako računalo korisnika ne sadrži određeno pismo, zamijeniti će ga nekim sličnim pismom, rezultat čega može biti potpuno drugačiji izgled od namjeravanog. Ako oblikujete za međunarodnu publiku, trebate istražiti koja su pisma dostupna u ciljnim jezicima. Također, trebate uzeti u obzir i širenje teksta kod oblikovanja za druge jezike – stringovi teksta mogu se proširiti i do 50% u nekim jezicima.

Ponovno, dosljednost oblikovanja važna je u izboru pisama. U većini slučajeva, u jednoj aplikaciji ne bi trebali koristiti više od dva pisma u dvije ili tri različite veličine. Previše različitih pisama može vašoj aplikaciji dati izgled poruke otmičara za traženje otkupa.

Oblikovanje za upotrebljivost

Upotrebljivost bilo koje aplikacije bezuvjetno određuje sam korisnik. Oblikovanje sučelja je ponavljajući proces; rijetko će prvi pokušaj oblikovanja sučelja za vašu aplikaciju kao rezultat dati savršeno sučelje. Uključivanjem korisnika u ranim koracima procesa oblikovanja, možete s manje napora stvoriti bolje i upotrebljivije sučelje.

Što je dobro sučelje?

Najbolje mjesto za početak kad oblikujete korisničko sučelje je pogled na neke od najprodavanijih aplikacija Microsofta i drugih tvrtki; na kraju krajeva, sve one vjerojatno nisu postale najprodavanije zbog loših sučelja. Pronaći ćete puno zajedničkih osobina, kao što su alatne trake, statusne trake, plutajući savjeti, izbornici osjetljivi na kontekst te dijalozi s karticama. Nije slučajnost da Visual Basic pruža mogućnost dodavanja svega toga u vaše aplikacije.

Možete također posuditi i nešto iz svog iskustva korisnika softvera. Razmislite o nekim aplikacijama koje ste koristili; što je radilo, što nije, i kako bi to mogli popraviti. Zapamtite da, unatoč tome, vaše osobne sklonosti i odbojnosti ne moraju odgovarati korisničkim željama; trebat ćete usporediti svoje ideje s njihovim.

Vjerojatno ste uočili kako većina uspješnih aplikacija pruža mogućnost prilagođavanja različitim željama korisnika. Na primjer, aplikacija Microsoft Windows Explorer dopušta korisnicima kopiranje datoteka uz pomoć izbornika, naredbi tipkovnicom ili povlačenjem i ispuštanjem. Pružanje izbora će proširiti privlačnost vaše aplikacije; kao minimum trebali bi omogućiti pristup svim funkcijama i mišem i tipkovnicom.

Smjernice Windows sučelja

Jedna od najvećih prednosti Windows operativnog sustava je što on predstavlja zajedničko sučelje kroz sve aplikacije. Korisnik koji zna koristiti jednu aplikaciju temeljenu na Windowsima trebao bi biti sposoban lako naučiti bilo koju drugu. Na žalost, aplikacije koje odlutaju predaleko od smjernica ustanovljenog sučelja ne uče se tako lako.

Izbornici su dobar primjer toga – većina aplikacija temeljenih na Windowsima slijedi standard postavljanja izbornika File na lijevoj strani, zatim slijede dodatni izbornici kao što su Edit i Tools, završavajući izbornikom Help na desnoj strani. Moglo bi se raspravljati o tome je li Documents bolje ime od File, i treba li izbornik Help biti prvi. Ništa vas ne sprečava da tako napravite, ali takvim postupkom ćete zbuniti svoje korisnike i smanjiti upotrebljivost svoje aplikacije. Korisnici će morati zastati i razmišljati prebacujući se svaki put između vaše i ostalih aplikacija.

Postavljanje podizbornika je također važno. Korisnici očekuju naredbe Copy, Cut i Paste unutar izbornika Edit; postavljanje u izbornik File bit će potpuno zbunjujuće. Ne odstupajte od ustanovljenih smjernica osim ako za to nemate dobar razlog.

Ispitivanje upotrebljivosti

Najbolji način ispitivanja upotrebljivosti vašeg sučelja je uključivanje korisnika tijekom postupka oblikovanja. Bez obzira oblikujete li veliku sveobuhvatnu aplikaciju ili malu aplikaciju za ograničenu upotrebu, postupak oblikovanja je prilično sličan. Koristeći ustanovljene smjernice oblikovanja, započet ćete oblikovanjem sučelja na papiru.

Idući korak je stvaranje jednog ili više prototipova, oblikujući vaše forme u Visual Basicu. Trebate dodati dovoljno programskog koda kako bi prototip radio: prikazivanje formi, popunjavanje okvira s popisom pokusnim podacima i tako dalje. Nakon toga ste spremni početi ispitivanje upotrebljivosti.

Ispitivanje upotrebljivosti može biti neslužbeni proces, gdje ispitujete dizajn s nekolicinom korisnika, ili službeni proces u postojećem laboratoriju. Na oba načina, svrha je ista – učenje iz prve ruke, od samih korisnika, gdje dizajn djeluje te gdje su potrebna poboljšanja. Umjesto ispitivanja korisnika, djelotvornije je jednostavno pustiti korisnika u nesmetanom radu s aplikacijom i promatrati ga. Neka korisnik izgovara svoj tijek misli dok pokušava napraviti niz zadataka: "Želim otvoriti novi dokument, pa ću pogledati u izborniku File". Zabilježite mjesta gdje dizajn sučelja ne odgovara tijeku misli korisnika. Ispitujte s više korisnika; ako uočite da nekoliko korisnika ima probleme s određenim zadatkom, taj zadatak vjerojatno traži više pažnje.

U idućem koraku, pregledajte svoje bilješke i razmislite kako promijeniti sučelje da bude upotrebljivije. Napravite promjene u vašem sučelju i ponovno ga ispitajte. Jednom kad budete zadovoljni upotrebljivošću voje aplikacije, spremni ste za programiranje. Tijekom procesa razvoja povremeno trebate ispitati jesu li pretpostavke na kojima je stvoren prototip ispravne.

Otkrivanje osobina

Jedan od ključnih pojmova u ispitivanju upotrebljivosti je mogućnost otkrivanja osobina. Ako korisnik ne može otkriti kako upotrijebiti neku osobinu (ili čak da osobina postoji), ta osobina je od male koristi. Na primjer, većina korisnika operativnog sustava Windows 3.1 nije bila svjesna da se kombinacija tipki ALT i TAB može koristiti za prebacivanje između otvorenih aplikacija. Nigdje u sučelju nije postojao nagovještaj koji bi pomogao korisnicima da otkriju tu osobinu.

Za ispitivanje mogućnosti otkrivanja osobine, zatražite od korisnika da izvede zadatak bez vašeg objašnjenja kako to napraviti (na primjer "Stvori novi dokument korištenjem predloška za pismo"). Ako on to ne može napraviti, ili će mu trebati nekoliko pokušaja, trebalo bi poraditi na otkrivanju te osobine.

Kad stvari krenu pogrešno: suradnja s korisnicima

U idealnom svijetu, softver i hardver bi uvijek trebali raditi bezgrešno, a korisnici nikad ne bi griješili. Stvarnost određuje da se pogreške mogu i hoće događati. Dio oblikovanja korisničkog sučelja uključuje odlučivanje kako će aplikacija odgovoriti kad stvari krenu pogrešno.

Uobičajeni odgovor je prikaz dijaloškog okvira koji od korisnika traži odgovor kako će se aplikacija suočiti s problemom. Manje uobičajen (ali bolji) odgovor bio bi jednostavno rješavanje problema bez opterećivanja korisnika. Na kraju krajeva, korisnik je uglavnom usredotočen na obavljanje zadatka, a ne na tehničke detalje. Pri oblikovanju vašeg korisničkog sučelja, razmislite o mogućim pogreškama i odlučite koje od njih traže akciju korisnika, a koje se mogu riješiti programiranjem.

Stvaranje pametnih dijaloških okvira

Ponekad se u vašoj aplikaciji javlja pogreška i tada je potrebno donijeti odluku koja će riješiti situaciju. To se obično događa u grani vašeg programskog koda – izrazu If...Then ili izrazu Case. Ako odluka zahtijeva akciju korisnika, pitanje se obično postavlja korisniku upotrebom dijaloškog okvira. Dijaloški okviri su dio vašeg korisničkog sučelja, i kao kod ostalih dijelova sučelja, njihov dizajn ima ulogu u upotrebljivosti vaše aplikacije.

Ponekad se čini kako su većinu dijaloških okvira oblikovali programeri koji nikad nisu sudjelovali u inteligentnoj raspravi s drugim ljudskim bićem. Poruka kao “Sektor tvrdog diska C: je oštećen ili nedostupan. Odustati, pokušati ponovno, zanemariti?” (pogledajte sliku 6.22) malog je značenja prosječnom korisniku. To je kao da vas konobarica pita “Nemamo više juhe ili je kuhinja zapaljena. Odustati, pokušati ponovno, zanemariti?”. Kako bi odgovorili? Bitno je izraziti pitanja (i izbor odgovora) tako da to korisnik može razumjeti. U prethodnom primjeru, bolja poruka bila bi “Postoji problem s spremanjem datoteke na pogon C. Snimiti datoteku na pogon A, Odustati od snimanja datoteke?”.

Slika 6.22 Koji dijaloški okvir sadrži jasniju poruku?

Kad stvarate dijaloške okvire za svoju aplikaciju, imajte korisnika na umu. Prenosi li poruka korisniku upotrebljivu informaciju? Je li lako razumljiva? Predstavljaju li naredbeni gumbi jasne izbore? Jesu li izbori prikladni danoj situaciji? Imajte na pameti da je dovoljan samo jedan okvir s dosadnom porukom pa da korisnik stekne loš utisak o vašoj aplikaciji.

Kad oblikujete vlastite korisničke dijaloške forme, pokušajte se držati standardnog stila. Ako ćete se previše razlikovati od izgleda standardnih okvira s porukom, korisnik to možda neće prepoznati kao dijaloški okvir.

Za više informacija Da biste naučili više o dijalozima, pogledajte “Dijaloški okviri” ranije u ovom poglavlju.

Rukovanje pogreškama bez dijaloških okvira

Nije uvijek neophodno prekinuti korisnika u radu kad se pojavi pogreška. Ponekad je poželjnije obraditi pogrešku programskim kodom bez obavještavanja korisnika, ili upozoriti korisnika na način koji neće sporiti njegov tijek rada. Dobar primjer ove tehnike je svojstvo automatskog ispravljanja AutoCorrect u aplikaciji Microsoft Word: ako je uobičajena riječ pogrešno napisana, Word će je automatski popraviti; ako je pogrešno napisana manje uobičajena riječ, ona će biti podcrtana crveno tako da je korisnik može ispraviti kasnije.

Postoji niz tehnika koje možete koristiti; na vama je da odlučite koje su tehnike prikladne za vašu vlastitu aplikaciju. Evo nekoliko savjeta:

- Dodajte funkciju Undo u izbornik Edit. Radije nego da prekidate korisnika dijaloškim okvirom s potvrdom za brisanja i slične događaje, vjerujte da je donio pravu odluku i omogućite mu funkciju Undo za slučaj da se kasnije predomisli.
- Prikazujte poruke na statusnoj traci ili ikoni. Ako pogreška ne utječe na posao kojim se korisnik trenutno bavi, ne zaustavljajte aplikaciju. Upotrijebite statusnu traku ili upozoravajuću ikonu blještavih boja kako bi upozorili korisnika – on može obraditi problem kad bude spreman.
- Ispravite problem. Ponekad je rješenje pogreške očigledno. Na primjer, ako je disk pun kad korisnik pokuša spremiti datoteku, provjerom sustava pronađite prostor na drugim pogonima. Ako ima raspoloživog prostora, spremite datoteku; postavite poruku na statusnu traku kako bi korisnik znao što ste napravili.
- Sačuvajte poruku za kasnije. Nisu sve pogreške kritične i ne traže trenutnu pažnju; razmislite o njihovom bilježenju u neku datoteku i prikazivanju korisniku kad izlazi iz aplikacije ili u neko drugo prikladno vrijeme. Ako korisnik napravi moguću pogrešku tijekom upisa (na primjer, Ucila umjesto Ulica), zabilježite to. Dodajte gumb za ponovni pregled upisa i funkciju za prikaz različitosti tako da ih korisnik može ispraviti.
- Nemojte napraviti ništa. Ponekad pogreška nije dovoljno važna da zavrijedi upozorenje. Na primjer, činjenica da je pisač na LPT1 ostao bez papira ne znači puno sve dok niste spremni za ispis. Čekajte sve dok poruka ne bude prikladna trenutnom zadatku.

Za više informacija Kako bi naučili više o tehnikama rukovanja pogreškama, pogledajte 13. poglavlje “Traženje i obrada grešaka”.

Oblikovanje modela za podršku korisnika

Bez obzira na to koliko je krasno vaše sučelje, bit će trenutaka kad će korisnik trebati podršku. Model podrške korisniku za vašu aplikaciju uključuje stvari kao što su stalna pomoć i tiskana dokumentacija; može također sadržavati i sredstva za pomoć korisniku kao što su plutajući savjeti, statusne trake, pomoć “Što je ovo” i čarobnjake.

Model podrške korisniku trebao bi biti oblikovan kao i svaki drugi dio vaše aplikacije: prije nego što počnete s razvojem. Sadržaj vašeg modela će mijenjati ovisno o složenosti aplikacije i tipu krajnjeg korisnika.

Pomoć i dokumentacija

Stalna pomoć je važan dio svake aplikacije – obično je to prvo mjesto gdje će korisnik pogledati ako ima pitanje. Čak i jednostavna aplikacija trebala bi pružiti pomoć; propust pružanja pomoći jednak je pretpostavci da vaši korisnici nikad neće imati pitanja.

Pri oblikovanju vašeg sustava pomoći, imajte na umu da je njezina prvenstvena svrha odgovaranje na pitanja. Kad stvarate imena tema i ulazne indekse pokušajte razmišljati kao korisnik; na primjer, upotreba "Kako ću oblikovati stranicu" bolja je od "izbornik Edit, stavka Page Format" i učinit će vaše teme lakšima za pronalaženje. Ne zaboravite osjetljivost na sadržaj; većina korisnika osjeća se razočaravajuće ako na određenom mjestu pritisnu F1 za pomoć i nađu se na stranici pomoći "Sadržaj".

Pojmovna dokumentacija, tiskana i/ili na kompaktnom disku, korisna je za sve, pa i najmanje, aplikacije. Ona može pružiti informacije koje bi bilo teško pružiti u kraćim temama stalne pomoći. Na kraju, trebali bi pružiti i dokumentaciju u obliku datoteke ReadMe koju korisnik može ispisati ako želi.

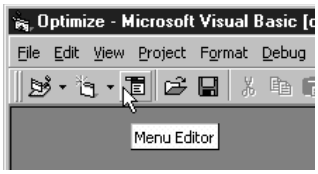
Sredstva korisničke pomoći

Unutar korisničkog sučelja postoji nekoliko tehnika za pružanje pomoći korisniku. Visual Basic pojednostavljuje to dodavanjem plutajućih savjeta, pomoći "Što je to", prikaza statusa i čarobnjaka vašoj aplikaciji. Na vama je da odlučite koja su od tih sredstava prikladna vašoj aplikaciji.

ToolTips

ToolTips, slika 6.23, su odličan način prikazivanja informacije korisniku dok upravlja korisničkim sučeljem. ToolTip je mali natpis koji se prikazuje kad se pokazivač miša određeno vrijeme drži iznad kontrole, i obično sadržava opis funkcija kontrole. Iako se obično koriste u suradnji s alatnim trakama, ToolTipi će također dobro raditi i s gotovo svakim dijelom sučelja.

Slika 6.23 ToolTip za alatnu traku Visual Basica



Većina kontrola Visual Basica sadrži jedno svojstvo za prikazivanje ToolTipa: svojstvo ToolTipText. Sljedeći programski kod će ostvariti ToolTip za naredbeni gumb imena cmdTiskaj:

```
cmdTiskaj.ToolTipText = "Ispisuje trenutni dokument"
```

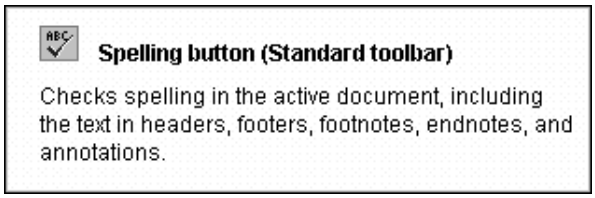
Kao i kod ostalih dijelova sučelja, budite sigurni da tekst jasno izražava željenu poruku korisniku.

Za više informacija Kako bi naučili više o ToolTipu, pogledajte "Svojstvo ToolTipText" u biblioteci *Microsoft Visual Basic 6.0 Langugage Reference*.

Pomoć "Što je to" (What's this)

Pomoć "Što je to" pruža vezu s tekstom pomoći koji se pojavljuje u plutajućem okviru (pogledajte sliku 6.24) kad korisnik odabere pomoć "Što je to" (What's This Help) i klikne pripadajućim kursorom na kontrolu. Pomoć "Što je to" može biti pozvana gumbom na alatnoj traci, stavkom izbornika ili gumbom koji se nalazi na naslovnoj traci dijaloškog okvira.

Slika 6.24 Plutajući prozor pomoći "Što je to"



Kako omogućiti pomoć "Što je to" iz izbornika ili alatne trake

1. Odaberite kontrolu za koju želite pružiti pomoć.
2. U prozoru s svojstvima, odaberite svojstvo **WhatsThisHelpID**.
3. Upišite kontekstni identifikacijski broj za pripadajuću pomoć u plutajućem okviru.
4. Ponovite korake 1 do 3 za svaku dodatnu kontrolu.
5. Odaberite formu.
6. U prozoru s svojstvima, postavite svojstvo forme **WhatsThisHelp** na **True**.
7. U događaju Click izbornika ili gumba alatne trake, upišite sljedeće:

   ```
   imeforme.WhatsThisHelp
   ```

   Kad korisnik klikne na gumb ili izbornik, pokazivač miša će se promijeniti u pokazivač pomoći "Što je to".

Za uključivanje pomoći "Što je to" na naslovnoj traci korisničke dijaloške forme, postavite svojstva forme WhatsThisButton i WhatsThisHelp na True.

Za više informacija Kako bi naučili više o pomoći "Što je to", pogledajte "Svojstvo WhatsThisHelp" i "Svojstvo WhatsThisButton" u biblioteci *Microsoft Visual Basic 6.0 Langugage Reference*.

Prikaz statusa

Prikaz statusa može se također upotrijebiti za pružanje podrške korisniku na otprilike isti način kao i ToolTips. Prikazi statusa su dobar način pružanja uputa ili poruka koje ne bi stale u ToolTip. Kontrola statusne trake uključena u Professional i Enterprise verzije Visual Basica dobro obavlja prikazivanje poruka; kontrola natpisa može također biti upotrijebljena za prikaz statusa.

Tekst ispisan u prikazu statusa može biti ažuriran na jedan od dva načina: događajem GotFocus kontrole ili forme, ili događajem MouseMove. Ako želite upotrijebiti prikaz statusa kao sredstvo učenja, dodajte stavku u izbornik Help koja će uključivati i isključivati njegovo svojstvo Visible.

Kako dodati prikaz statusa

1. Dodajte vašoj formi kontrolu natpisa.
2. Odaberite kontrolu za koju želite prikazati poruku.
3. Dodajte sljedeći programski kod u događaj MouseMove (ili GotFocus) te kontrole:

   ```
   imekontrolenatpisa.Caption = "Upišite datum rođenja u to polje"
   ```

   Kad korisnik pomakne miša iznad kontrole, poruka će biti ispisana u kontroli natpisa.
4. Ponovite korake 2 i 3 za svaku dodatnu kontrolu.

Čarobnjaci

Čarobnjak je sredstvo pomoći korisniku koje korisnika vodi korak po korak kroz postupak, radeći s stvarnim podacima korisnika. Čarobnjaci se obično koriste za pružanje podrške za određeni zadatak. Oni pomažu korisniku u ostvarivanju zadatka koji bi inače zahtijevao poduži (i neželjeni) tečaj učenja; oni pružaju stručne informacije korisniku koji još nije postao stručnjak.

Professional i Enterprise verzije Visual Basica sadržavaju alat za stvaranje čarobnjaka nazvan Wizard Manager.

Za više informacija Kako bi naučili više o čarobnjacima, pogledajte "Korištenje čarobnjaka i dodataka" u 4. poglavlju "Upravljanje projektima".

7. POGLAVLJE

Korištenje standardnih kontrola Visual Basica

Kontrole upotrebljavate za dobivanje podataka od korisnika te za prikazivanje izlaznih podataka. Neke od kontrola koje možete koristiti u vašoj aplikaciji uključuju okvire za tekst, naredbene gumbe i okvire s popisom. Ostale kontrole omogućuju vam pristup podacima drugih aplikacija i procesa kao da je udaljena aplikacija dio vašeg koda. Svaka kontrola ima vlastiti skup svojstava, postupaka i događaja. Ovo poglavlje upoznaje vas s standardnim kontrolama Visual Basica.

Za više informacija Pogledajte "Korištenje ActiveX kontrola" u dijelu *Microsoft Visual Basic 6.0 Component Tools Guide* biblioteke *Microsoft Visual Basic 6.0 Reference Library* za detaljnije informacije o ActiveX kontrolama dostupnim u Professional i Enterprise verzijama Visual Basica.

Sadržaj

- Korištenje kontrola vodoravne i okomite trake za pomicanje
- Korištenje kontrole slike
- Korištenje kontrole natpisa
- Korištenje kontrole linije
- Korištenje kontrole okvira s popisom
- Korištenje kontrole OLE spremnika
- Korištenje kontrole gumba izbora
- Korištenje kontrole okvira za sliku
- Korištenje kontrole udaljenih podataka
- Korištenje kontrole lika
- Korištenje kontrole okvira s tekstom
- Korištenje kontrole mjerača vremena

Primjeri aplikacija: Alarm.vbp, Calc.vbp, Controls.vbp, Flex.vbp, Winseek.vbp

Većina primjera programskog koda u ovom poglavlju uzeta je iz aplikacija Alarm.vbp, Calc.vbp, Controls.vbp, Flex.vbp i Winseek.vbp. Ove aplikacije nalaze se u direktoriju Samples.

Uvod u kontrole Visual Basica

Alatni okvir Visual Basica sadrži alate koje upotrebljavate za stvaranje kontrola na vašoj formi.

Slika 7.1 Alatni okvir Visual Basica

Vrste kontrola

U Visual Basicu postoje tri općenite kategorije kontrola:

- *Ugrađene kontrole*, kao što su naredbeni gumbi i kontrole okvira. Ove kontrole nalaze se unutar datoteke .exe Visual Basica. Ugrađene kontrole uvijek su uključene u alatni okvir, za razliku od ActiveX kontrola i dodatnih objekata, koji mogu biti maknuti iz i dodani u alatni okvir.
- *ActiveX kontrole*, koje postoje kao posebne datoteke s nastavkom .ocx. Ovdje su uključene kontrole koje su dostupne u svim verzijama Visual Basica (kontrole kombiniranog okvira s podacima, okvira s popisom podataka i slične) te kontrole koje su dostupne samo u Professional i Enterprise verzijama (kao kontrole prikaza popisa, alatne trake, animacije i dijaloga s karticama). Dostupno je i puno ActiveX kontrola drugih proizvođača.

  Napomena Kontrole čije datoteke imaju nastavak .vbx koriste stariju tehnologiju i mogu se pronaći u aplikacijama napisanim u starijim verzijama Visual Basica. Kad Visual Basic otvori projekt koji sadrži .vbx kontrolu, standardni postupak je zamjena .vbx kontrole .ocx kontrolom, ali samo ako je dostupna .ocx verzija kontrole. Pogledajte odlomak "Ažuriranje starijih verzija kontrola Visual Basica" kasnije u ovom poglavlju za informacije o ažuriranju kontrola na .ocx format.

- *Dodatni objekti*, kao što je objekt radnog lista Microsoft Excela koji sadrži popis svih zaposlenika vaše firme, ili objekt kalendara Microsoft Projecta koji sadrži informacije planiranja za projekt. Budući da ovi objekti mogu biti dodani u alatni okvir, možemo ih smatrati kontrolama. Neki od ovih objekata također podržavaju automatizaciju (nekad zvanu OLE automatizacija), što vam omogućuje programiranje objekata druge aplikacije iz Visual Basic aplikacije. Pogledajte 9. poglavlje "Programiranje objektima", za više informacija o automatizaciji.

c Pogledajte odlomak "Korištenje ActiveX kontrola" u vodiču *Microsoft Visual Basic 6.0 Component Tools Guide* za više informacija o ActiveX kontrolama dostupnim u Professional i Enterprise verzijama Visual Basica.

Ugrađene kontrole

Sljedeća tablica rezimira ugrađene kontrole koje se mogu pronaći u alatnom okviru Visual Basica.

ikona	ime kontrole	ime klase	opis
	kontrolna kućica	CheckBox	Prikazuje izbor točno/netočno ili da/ne. Istovremeno možete potvrditi bilo koji broj kontrolnih kućica na formi.
	kombinirani okvir	ComboBox	Sjedinjuje okvir s tekstom i okvir s popisom. Dopušta korisniku upis izbora ili odabir stavke s spuštajućeg popisa.

ikona	ime kontrole	ime klase	opis
	naredbeni gumb	CommandButton	Izvršava naredbu ili akciju kad ga korisnik odabere.
	podaci	Data	Omogućuje vam povezivanje s postojećom bazom podataka i prikaz informacija iz nje na vašim formama.
	okvir s popisom direktorija	DirListBox	Prikazuje popis s kojeg korisnik može odabrati direktorije i staze.
	okvir s popisom pogonskih uređaja	DriveListBox	Prikazuje popis s kojeg korisnik može odabrati valjane pogone.
	okvir s popisom datoteka	FileListBox	Prikazuje popis s kojeg korisnik može odabrati datoteke.
	okvir	Frame	Pruža vizualni i funkcionalni spremnik za kontrole.
 	vodoravne i okomite trake za pomicanje	HScrollBar i VScrollBar	Omogućuje korisniku dodavanje traka za pomicanje kontrolama koje ih ne podržavaju automatski (nisu iste kao i trake ugrađene u neke kontrole).
	slika	Image	Prikazuje bitmapirane slike, ikone ili Windows metadatoteke, JPEG ili GIF datoteke; djeluje kao naredbeni gumb kad se klikne na nju.
	natpis	Label	Prikazuje tekst kojeg korisnik ne može mijenjati niti direktno utjecati na njega.
	linija	Line	Dodaje segment ravne linije na formu.
	okvir s popisom	ListBox	Prikazuje popis stavki s kojeg korisnik može birati.
	OLE spremnik	OLE	Umeće podatke u Visual Basic aplikaciju.
	gumb izbora	OptionButton	Kontrola gumba izbora, kao dio grupe s drugim gumbima izbora, prikazuje višestruke izbore, od kojih korisnik može odabrati samo jedan.
	okvir za sliku	PictureBox	Prikazuje bitmapirane slike, ikone ili Windows metadatoteke, JPEG ili GIF datoteke. Također prikazuje i tekst ili djeluje kao vizualni spremnik za druge kontrole.

ikona	ime kontrole	ime klase	opis
	lik	Shape	Dodaje pravokutnik, kvadrat, elipsu ili krug na formu, okvir ili okvir za sliku.
	okvir s tekstom	TextBox	Pruža područje za upis ili prikaz teksta.
	mjerač vremena	Timer	Izvršava događaje mjerača u određenim vremenskim razmacima.

Napomena Alat pokazivača (prvi alat u alatnom okviru) služi za pomicanje formi i kontrola te za promjenu njihovih veličina. On nije kontrola.

Standardne ActiveX kontrole

Learning verzija Visual Basica sadrži niz ActiveX kontrola (označenih kao standardne ActiveX kontrole) koje omogućuju dodavanje naprednih osobina vašim aplikacijama. ActiveX kontrole imaju nastavak .ocx imena datoteke i možete ih upotrijebiti u svom projektu tako da ih ručno dodate u alatni okvir.

Sljedeća tablica rezimira standardne ActiveX kontrole dostupne u Learning verziji Visual Basica.

ikona	ime kontrole	ime klase	opis
	opći dijalog	CommonDialog	Pruža standardan skup dijaloških okvira za operacije kao što su otvaranje i spremanje datoteka, podešavanje ispisa, te odabir boja i pisama.
	kombinirani okvir za podatke	DataCombo	Pruža većinu osobina standardne kontrole kombiniranog okvira, s povećanim sposobnostima pristupa podacima.
	okvir s popisom podataka podacima.	DataList	Pruža većinu osobina standardne kontrole okvira s popisom, s povećanim sposobnostima pristupa
	Microsoft fleksibilna mreža	MSFlexGrid	Slična je kontroli podataka, ali ima dodatne osobine oblikovanja, grupiranja i povezivanja, kao i mogućnost prilagođavanja.

Za više informacija Pogledajte odlomak "Korištenje ActiveX kontrola" u vodiču *Microsoft Visual Basic 6.0 Component Tools Guide* za više informacija o ActiveX kontrolama dostupnim u Professional i Enterprise verzijama Visual Basica.

Dodavanje i uklanjanje ActiveX kontrola

ActiveX kontrole možete dodati i ukloniti iz alatnog okvira korištenjem sljedećeg postupka.

Kako dodati ActiveX kontrolu u alatni okvir

1. U izborniku **Project**, odaberite stavku **Components**.
2. Potvrdite kontrolnu kućicu pored imena .ocx kontrole i odaberite **OK**. Kad je jednom kontrola postavljena u alatni okvir, možete je postaviti na formu kao i bilo koju ugrađenu kontrolu.

Kako ukloniti ActiveX kontrolu

1. Maknite sve primjere te kontrole s formi vašeg projekta. Obrišite sve pozive na kontrolu u programskom kodu projekta. Ako u vašem kodu ostanu pozivi obrisane kontrole, pri prevođenju aplikacije prikazat će se poruka pogreške.
2. U izborniku **Project**, odaberite stavku **Components**.

Odznačite kontrolnu kućicu pored imena .ocx kontrole i odaberite **OK**. Ako je u vašem projektu ostao neki primjer te kontrole, prikazat će se poruka pogreške.

Za više informacija Pogledajte odlomak "Dodavanje kontrola u projekt" u 4. poglavlju "Upravljanje projektima", za više informacija o dodavanju i uklanjanju kontrola i dodatnih objekata u i iz alatnog okvira.

Ažuriranje starijih verzija kontrola Visual Basica

Starije 16-bitne verzije kontrola Visual Basica s nastavkom imena datoteke .vbx nisu sukladne s ovom verzijom Visual Basica. Ako pokušate učitati stari projekt koji sadrži .vbx kontrole, Visual Basic će vas upozoriti da su kontrole nedostupne ili nisu sukladne. Imate mogućnost nastavka učitavanja projekta bez .vbx kontrola ali, naravno, aplikacija neće ispravno raditi.

Ako imate starije projekte Visual Basica koji sadržavaju .vbx kontrole drugih proizvođača, kontaktirajte ih i raspitajte se o .ocx zamjenama.

Pravila imenovanja kontrola

Kad prvi put kreirate *objekt* (formu ili kontrolu), Visual Basic postavlja njegovo svojstvo Name na predodređenu vrijednost. Na primjer, svi naredbeni gumbi početnu vrijednost svojstva Name postavljenu na Command*n*, gdje je *n* 1, 2, 3 i tako dalje. Visual Basic daje prvom naredbenom gumbu kreiranom na formi ime Command1, drugom Command2 i trećem Command3.

Vaš izbor može biti zadržavanje tako određenog imena; međutim, kad imate nekoliko kontrola istog tipa, smisleno je promijeniti njihova svojstva Name u nešto opisnije. Budući da može biti teško razlikovati gumb Command1 na formi MojaForma od gumba Command1 na formi TvojaForma, pomažu pravila imenovanja. To je posebno točno kad se aplikacija sastoji od nekoliko formi, standardnih i klasnih modula.

Možete upotrijebiti prefiks za opis klase, iza kojeg slijedi opisno ime kontrole. Korištenje takvog pravila dodjele imena čini programski kod razumljivijim, a slični objekti su abecedno grupirani u okviru s popisom objekata. Na primjer, kontroli kontrolne kućice mogli bi dati ovakvo ime:

```
chkSamoČitaj
```

Imena koja dajete formama i kontrolama:

- moraju počinjati slovom.
- moraju sadržavati samo slova, brojeve te karakter podvučene linije (_); interpunkcijski karakteri i razmaci nisu dopušteni.
- ne smiju biti duža od 40 karaktera.

Za više informacija Pogledajte Dodatak B “ Pravila programiranja Visual Basica”, za više informacija o pravilima imenovanja.

Korištenje vrijednosti kontrole

Sve kontrole imaju svojstvo koje možete upotrijebiti za spremanje ili dohvaćanje vrijednosti samo pozivom kontrole, bez korištenja svojstva imena. To svojstvo se naziva *vrijednost* kontrole i obično je najvažnije ili najčešće korišteno svojstvo kontrole. Sljedeća tablica sadrži popis svojstava koja se uzimaju u obzir kao vrijednost svake kontrole.

kontrola	vrijednost
kontrolna kućica (Check box)	Value
kombinirani okvir (Combo box)	Text
naredbeni gumb (Command button)	Value
opći dijalog (Common dialog)	Action
podaci (Data)	Caption
kombinirani okvir za podatke (DataCombo)	Text
mreža podataka (DataGrid)	Text
okvir s popisom podataka (DataList)	Text
okvir s popisom direktorija (Directory list box)	Path
okvir s popisom pogonskih uređaja (Drive list box)	Drive

kontrola	vrijednost
okvir s popisom datoteka (File list box)	FileName
fleksibilna mreža (FlexGrid)	Text
okvir (Frame)	Caption
vodoravna traka za pomicanje (Horizontal scroll bar)	Value
slika (Image)	Picture
natpis (Label)	Caption
linija (Line)	Visible
okvir s popisom (List box)	Text
gumb izbora (Option button)	Value
okvir za sliku (Picture box)	Picture
lik (Shape)	Shape
okvir s tekstom (Text box)	Text
mjerač vremena (Timer)	Enabled
okomita klizna traka (Vertical scroll bar)	Value

Kad god se želite pozvati na svojstvo kontrole koje je vrijednost te kontrole, možete to napraviti bez određivanja imena svojstva u vašem programskom kodu. Na primjer, ova linija postavlja vrijednost svojstva Text kontrole okvira s tekstom:

```
Text1 = "Ovaj tekst je dodijeljen svojstvu Text kontrole Text1"
```

U sljedećem primjeru, svojstvu Caption kontrole Label1 dodjeljuje se svojstvo FileName kontrole File1 svaki put kad korisnik klikne datoteku u okviru s popisom datoteka:

```
Private Sub File1_Click()
  Label1 = File1
End Sub
```

Napomena Budući da korištenje vrijednosti kontrole čini vaš programski kod donekle manje čitljivim, nemojte koristiti gornje primjere te se umjesto toga izričito pozivajte na svojstva svih kontrola. Možete pokušati pisati vaš kod na oba načina, te odlučiti koristiti vrijednosti kontrola u vašem kodu ako nemate problema pri njegovom tumačenju i čitanju.

Provjera valjanosti podataka u kontroli ograničavanjem fokusa

Događaj Validate i svojstvo CausesValidation koriste se u tandemu za provjeru unosa u kontrolu prije dopuštanja korisniku da makne fokus s te kontrole. Na primjer, zamislite aplikaciju s nekoliko okvira s tekstom te gumbom za pomoć. Kad svaki okvir s tekstom dobije fokus, želite spriječiti korisnika da makne fokus sve dok nisu ispunjeni određeni uvjeti valjanosti za okvir s tekstom; unatoč tome, također želite omogućiti korisniku da klikne gumb za pomoć u bilo koje vrijeme. Kako bi to napravili, odredite uvjete valjanosti u događaju Validate i postavite svojstvo CausesValidation gumba za pomoć na False. Ako je to svojstvo postavljeno na True (standardno), događaj Validate će se pojaviti na prvoj kontroli. Ako je to svojstvo postavljeno na False, događaj Validate prve kontrole bit će izuzet iz pojavljivanja.

Događaj Validate prikladniji je za provjeru valjanosti unosa podataka od događaja LostFocus jer se događaj LostFocus (u pravilu) pojavljuje nakon gubitka fokusa. Suprotno tome, korištenjem događaja Validate možete spriječiti prebacivanje fokusa na drugu kontrolu sve dok nisu ispunjena sva pravila valjanosti.

Moguće upotrebe

- Aplikacija za unos podataka treba provesti profinjeniju provjeru unosa podataka od one koju pruža kontrola Masked Edit, ili od provjera koje se pojavljuju u poslovnim pravilima.
- Forma treba spriječiti korisnika od micanja s kontrole korištenjem tipke TAB ili tipki za smjer sve dok podaci ne budu uneseni u polje.
- ActiveX dokument koji radi unutar Internet Explorera treba način uz pomoć kojeg će korisnik završiti operaciju na formi prije nego skripta programski pomakne fokus.

Kontrola fokusa u događaju Validate

Događaj Validate uključuje argument *držifokus* (*keepfocus*). Kad je ovaj argument postavljen na True, kontrola će zadržati fokus. Ovaj način djelotvorno sprečava korisnika od klikanja neke druge kontrole.

Rad s matricama kontrola

Matrica kontrola je grupa kontrola koje dijele isto ime i tip. One također dijele i iste potprograme događaja. Matrica kontrola ima barem jedan element i može se povećavati do onoliko elemenata koliko dopuštaju vaši sistemski izvori i memorija; njezina veličina također ovisi i o tome koliko memorije i sistemskih izvora svaka kontrola zahtijeva. Najveći indeks koji možete koristiti u matrici kontrola je 32767. Elementi iste matrice kontrola imaju vlastite vrijednosti svojstava. Uobičajena upotreba matrice kontrola uključuje grupiranje kontrola izbornika ili gumbâ izbora.

Napomena Visual Basic uključuje mogućnost dinamičkog dodavanja nepozivanih kontrola u zbirku Controls tijekom izvođenja aplikacije. Ova tema odnosi se samo na pozivane kontrole dodane tijekom izrade aplikacije odsijecanjem i uljepljivanjem kontrole na formu. Za više informacija o dodavanju kontrola tijekom izvođenja, pogledajte temu “Postupak Add (zbirka Controls)” i “Postupak Add (zbirka Licenses)”, u stalnoj pomoći.

Zašto koristiti matrice kontrola

Dodavanje kontrola korištenjem matrica kontrola koristi manje izvora od jednostavnog dodavanja više kontrola istog tipa tijekom izrade aplikacije. Matrice kontrola su također korisne ako želite da više kontrola dijeli isti programski kod. Na primjer, ako su tri naredbena gumba kreirana kao matrica kontrola, bit će izvršen isti programski kod bez obzira na koji je gumb kliknuo korisnik.

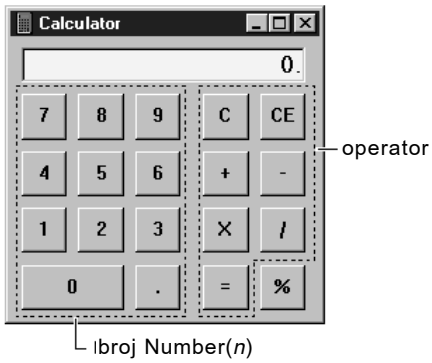
Ako tijekom izvođenja želite stvoriti nov primjer kontrole, ta kontrola mora biti član matrice kontrola. S matricom kontrola, svaki novi element nasljeđuje zajedničke potprograme događaja te matrice.

Korištenjem mehanizma matrice kontrola, svaka nova kontrola nasljeđuje zajedničke potprograme događaja koji su već napisani za matricu. Na primjer, ako vaša forma ima nekoliko okvira s tekstom gdje svaki od njih može primiti vrijednost u obliku datuma, matrica kontrola može biti podešena tako da svi okviri s tekstom dijele isti programski kod za provjeru valjanosti unešenih podataka.

Primjer aplikacije: Calc.vbp

Primjer aplikacije Calculator (koja se nalazi u direktoriju Samples) prikazan na slici 7.2 sadrži dvije matrice kontrola - gumbe s brojevima i gumbe s operatorima.

Slika 7.2 Primjer matrice kontrola



Vrijednosti svojstava Name i Index za matrice kontrola u primjeru Calculator ispisane su u sljedećoj tablici.

Number(*n*)	Operator(*n*)
0 = Number(0)	+ = Operator(1)
1 = Number(1)	- = Operator(2)
2 = Number(2)	X = Operator(3)
3 = Number(3)	/ = Operator(4)

Number(*n*)	Operator(*n*)
4 = Number(4)	= = Operator(5)
5 = Number(5)	
6 = Number(6)	
7 = Number(7)	
8 = Number(8)	
9 = Number(9)	

Uočite da se svaka kontrola poziva sintaksom *objekt(indeks)*. Indeks kontrole određujete kad je stvarate. Zapravo, određivanje indeksa kontrole tijekom izrade čini tu kontrolu dijelom matrice.

Svojstvo Index razlikuje jedan element matrice kontrola od drugog. Kad jedna od kontrola u matrici prepozna događaj, Visual Basic poziva zajednički potprogram događaja i prosljeđuje argument (vrijednost svojstva Index) za identificiranje koja je kontrola matrice zapravo prepoznala događaj.

Na primjer, prva linija potprograma događaja Number_Click je:

```
Private Sub Number_Click(Index As Integer)
```

Ako kontrola Number(0) prepozna događaj, Visual Basic prosljeđuje 0 kao argument *index*, a ako događaj prepozna kontrola Number(1), Visual Basic će proslijediti 1 kao argument *index*. Osim vrijednosti indeksa, ostatak programskog koda potprograma Number_Click koji će se izvršiti jednak je za sve kontrole od Number(0) do Number(9).

Stvaranje matrice kontrola tijekom izrade

Postoje tri načina stvaranja matrice kontrola tijekom izrade aplikacije:

- Dodijelite isto ime više od jednoj kontroli.
- Kopirajte postojeću kontrolu i zatim je ulijepite na formu.
- Postavite vrijednost svojstva Index kontrole na vrijednost koja nije Null (0).

  Napomena Matrice kontrola izbornika morate kreirati u editoru izbornika. Za detalje o tom postupku, pogledajte odlomak “Stvaranje i mijenjanje izbornika tijekom izvođenja” u 6. poglavlju “Stvaranje korisničkog sučelja”.

Kako dodati element matrice kontrola promjenom njegovog imena

1. Kreirajte kontrole koje želite imati u matrici kontrola (kontrole moraju biti istog tipa). Odlučite koja će kontrola postati prvi element matrice.
2. Odaberite jednu od kontrola i promijenite joj svojstvo Name u ime prvog elementa matrice.

3. Kad upišete postojeće ime za kontrolu u matrici, Visual Basic će prikazati dijaloški okvir tražeći od vas potvrdu želite li stvoriti matricu kontrola. Odaberite **Yes** za potvrdu akcije.

Na primjer, ako je ime prvog elementa u matrici kontrola cmdKonMat, odabrat ćete naredbeni gumb koji želite dodati u matricu kontrola i odrediti mu ime cmdKonMat. Prikazat će se poruka "Već imate kontrolu s imenom 'cmdKonMat'. Želite li stvoriti matricu kontrola?". Odaberite Yes za potvrdu operacije.

Kontrole dodane ovakvim postupkom dijele samo svojstvo Name i tip kontrole; sva ostala svojstva ostaju ista kao kad je kontrola prvi put kreirana.

Kako dodati element matrice kontrola kopiranjem postojeće kontrole

1. Kreirajte kontrolu u matrici kontrola.
2. Dok kontrola ima fokus, odaberite **Copy** u izborniku **Edit**.
3. U izborniku **Edit**, odaberite **Paste**. Visual Basic će prikazati dijaloški okvir tražeći od vas potvrdu da želite stvoriti matricu kontrola. Odaberite **Yes** za potvrdu akcije.

Ta kontrola dobiva vrijednost indeksa 1. Kontrola koju ste prvu kreirali ima vrijednost indeksa 0.

Vrijednost indeksa svakog novog elementa matrice odgovara redoslijedu po kojem je element dodan u matricu kontrola. Kad su kontrole dodane na ovaj način, većina vidljivih svojstava, kao visina, širina i boja, kopiraju se od prve kontrole u matrici kontrola novim kontrolama.

Dodavanje u matricu kontrola tijekom izvođenja

Možete dodati i maknuti kontrole iz matrice kontrola tijekom izvođenja aplikacije korištenjem naredbi Load i Unload. Ipak, kontrola koja se dodaje mora biti element postojeće matrice kontrola. Tijekom izrade aplikacije morate stvoriti kontrolu s svojstvom Index postavljenim, u većini slučajeva, na 0. Nakon toga, tijekom izvođenja aplikacije, upotrijebite sljedeću sintaksu:

Load *objekt*(*indeks%*)

Unload *objekt*(*indeks%*)

argument	opis
objekt	Ime kontrole koja se dodaje ili miče iz matrice kontrola.
indeks%	Vrijednost indeksa kontrole u matrici.

Kad učitate novi element u matricu kontrola, većina postavki svojstava kopira se od najnižeg postojećeg elementa u matrici - u ovom primjeru, elementa s vrijednošću indeksa 0. Vrijednosti svojstava Visible, Index i TabIndex ne kopiraju se automatski novim elementima u matrici kontrola, pa kako bi novododanu kontrolu učinili vidljivom, morate joj postaviti svojstvo Visible na True.

Napomena Visual Basic stvara pogrešku ako pokušate upotrijebiti naredbu Load s brojem indeksa koji je već u upotrebi u matrici.

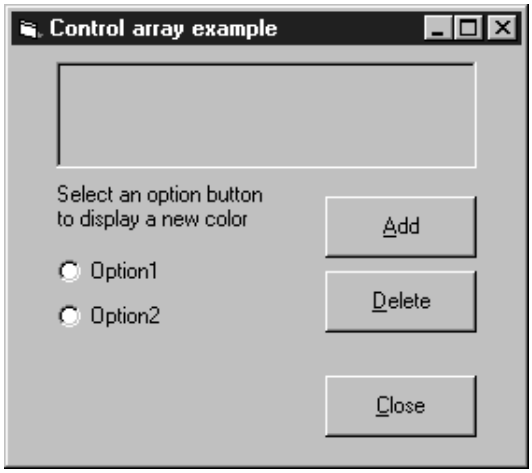
Važno Naredbu Unload možete koristiti za izbacivanje svake kontrole kreirane naredbom Load. Međutim, ne možete upotrijebiti naredbu Unload za izbacivanje kontrola stvorenih tijekom izrade aplikacije, bez obzira jesu li ili nisu dio matrice kontrola.

Aplikacija Controls.vbp: Dodavanje i brisanje kontrola u matrici kontrola

Primjer matrice kontrola pokazuje kako se kontrole - u ovom slučaju, gumbi izbora - dodaju i brišu tijekom izvođenja aplikacije. Primjer omogućuje korisniku dodavanje gumbi izbora koji mijenjaju boju pozadine okvira za sliku.

Počnite s formom, pa zatim kreirajte okvir za sliku, kontrolu natpisa, dva gumba izbora, i tri naredbena gumba, kao što je prikazano na slici 7.3.

Slika 7.3 Dodavanje kontrola tijekom izvođenja



Sljedeća tablica ispisuje postavke svojstava za objekte u aplikaciji.

objekt	svojstvo	postavka
forma	Caption	Control array example
okvir za sliku	Name	picDisplay
natpis	Caption	Select an option button to display, a new color
prvi gumb izbora	Name Index	optButton 0

objekt	svojstvo	postavka
drugi gumb izbora	Name Index	optButton 1
prvi naredbeni gumb	Name Caption	cmdAdd &Add
drugi naredbeni gumb	Name Caption	cmdDelete &Delete
treći naredbeni gumb	Name Caption	cmdClose &Close

Događaji u aplikaciji Controls

Idući korak je dodavanje potprograma događaja gumbima izbora i naredbenim gumbima. Počnite dodavanjem deklaracija za formu:

```
Dim MaxId As Integer
```

Potprogram događaja Click dijele svi gumbi izbora:

```
Private Sub optButton_Click (Index As Integer)
  picDisplay.BackColor = QBColor(Index + 1)
End Sub
```

Novi gumbi izbora dodaju se potprogramom događaja Click naredbenog gumba Add. U ovom primjeru, programski kod prije izvršavanja naredbe Load provjerava je li učitano najviše deset gumbâ izbora. Jednom kad je kontrola učitana, svojstvo Visible mora biti postavljeno na True.

```
Private Sub cmdAdd_Click()
  If MaxId = 0 Then MaxId = 1    ‘ Određivanje ukupnog broja
                                 ‘ gumbâ izbora.
  If MaxId > 8 Then Exit Sub     ‘ Dopušteno je samo
                                 ‘ deset gumbâ izbora.
  MaxId = MaxId + 1              ‘ Povećanje brojača gumbâ.
  Load optButton(MaxId)          ‘ Stvaranje novog gumba.
  optButton(0).SetFocus          ‘ Pokretanje odabira gumbâ.
  ‘ Postavljanje novog gumba ispod prethodnog gumba.
  optButton(MaxId).Top = optButton(MaxId - 1).Top + 400
  optButton(MaxId).Visible = True        ‘ Prikaz novog gumba.
  optButton(MaxId).Caption = “Option” & MaxId + 1
End Sub
```

Gumbi izbora uklanjaju se potprogramom događaja Click naredbenog gumba Delete:

```
Private Sub cmdDelete_Click()
  If MaxId <= 1 Then Exit Sub    ‘ Čuva prva dva gumba.
  Unload optButton(MaxId)        ‘ Brisanje posljednjeg gumba.
  MaxId = MaxId - 1              ‘ Smanjenje brojača gumbâ.
  optButton(0).SetFocus          ‘ Pokretanje odabira gumbâ.
End Sub
```

Događaj Click gumba Close završava aplikaciju:

```
Private Sub cmdClose_Click()
  Unload Me
End Sub
```

Korištenje kontrole ADO podataka

Kontrola ADO podataka (ADO Data) koristi objekte Microsoft ActiveX podataka (ActiveX Data Objects, ADO) za brzo stvaranje veza između kontrola povezanih s podacima i davatelja podataka. Kontrole povezane s podacima su one kontrole koje sadrže svojstvo DataSource. Davatelj podataka može biti bilo koji izvor pisan po specifikaciji OLE baze podataka. Vi također možete stvoriti vlastitog davatelja podataka korištenjem modula klase u Visual Basicu.

Iako u vašoj aplikaciji možete izravno koristiti objekte ActiveX podataka, prednost kontrole ADO podataka je što je to grafička kontrola (s gumbima za naprijed i natrag), a njezino sučelje jednostavno za korištenje dopušta vam stvaranje aplikacija baza podataka s minimumom programskog koda.

Slika 7.4 Kontrola ADO podataka

Neke kontrole iz alatnog okvira Visual Basica mogu biti povezane s podacima, uključujući kontrole kombiniranog okvira, kontrolne kućice, slike, natpisa, okvira s popisom, okvira za sliku i okvira s tekstom. Dodatno, Visual Basic uključuje nekoliko ActiveX kontrola povezanih s podacima kao što su kontrole mreže s podacima, kombiniranog okvira za podatke, grafikona i okvira s popisom podataka. Vi također možete stvoriti vlastite ActiveX kontrole povezane s podacima, ili ih nabaviti od drugih proizvođača.

Prethodne verzije Visual Basica sadržavale su ugrađenu kontrolu podataka i kontrolu udaljenih podataka (RDC) za pristup podacima. Obje kontrole i dalje su uključene u Visual Basic za sukladnost prema nazad. Međutim, zbog fleksibilnosti kontrole ADO, preporučljivo je stvarati nove baze podataka korištenjem kontrole ADO podataka.

Za više informacija Potpun popis kontrola s pristupom podacima može se pronaći u odlomku “Kontrole koje se povezuju prema kontroli ADO podataka” kasnije u ovom poglavlju. Kako koristiti ugrađenu kontrolu podataka i kontrolu udaljenih podataka pogledajte “Korištenje kontrole podataka” i “Korištenje kontrole udaljenih podataka” kasnije u ovom poglavlju. Za detalje o stvaranju davatelja podataka, pogledajte “Stvaranje klasa svjesnih podataka” u 9. poglavlju “Programiranje objektima”.

Moguće upotrebe

- Povezivanje s lokalnim ili udaljenim bazama podataka.
- Otvaranje određene tablice iz baze podataka ili određivanje skupa slogova temeljenog na strukturiranom jeziku upita (structured query langugage, SQL) ili pohranjenom potprogramu ili pregled tablica u toj bazi podataka.
- Proslјeđivanje vrijednosti polja podataka kontrolama povezanim s podacima, gdje možete prikazati ili promijeniti vrijednosti.
- Dodavanje novih slogova ili ažuriranje baze podataka temeljeno na svim promjenama koje ste napravili nad podacima prikazanim u povezanim kontrolama.

Kako bi stvorili klijenta ili konačnu aplikaciju, dodajte kontrolu ADO podataka svojoj formi kao što bi dodali i bilo koju drugu kontrolu Visual Basica. Na formi možete imati onoliko kontrola ADO podataka koliko trebate. Međutim, znajte da je ova kontrola razmjerno "skup" postupak stvaranja veza, jer koristi barem dvije veze za prvu kontrolu, te po jednu vezu više za svaku dodatnu kontrolu.

Stvaranje konačne aplikacije kao baze podataka s minimumom koda

Moguće je stvoriti aplikaciju baze podataka korištenjem minimuma programskog koda te određivanjem nekih svojstava tijekom izrade. Ako kao izvor podataka koristite OLE bazu podataka, na vašem računalu mora biti stvorena datoteka tipa .MDL (Microsoft Data Link). Pogledajte dio "Stvaranje podatkovne veze s Northwind OLE bazom podataka", u stalnoj pomoći, za primjer postupka korak-po-korak.

Kako stvoriti jednostavnu konačnu aplikaciju baze podataka

1. Kreirajte **kontrolu ADO podataka** na formi (ToolTip ikone je "ADODC").

   Ako kontrola nije dostupna u alatnom okviru, pritisnite CTRL + T za prikaz dijaloškog okvira **Components**. U dijalogu **Components**, kliknite **Microsoft ADO Data Control**.

2. U alatnom okviru kliknite na **kontrolu ADO podataka** kako bi je odabrali. Zatim pritisnite F4 za prikaz prozora s svojstvima.

3. U prozoru s svojstvima, kliknite **ConnectionString** za prikaz dijaloškog okvira **ConnectionString**.

4. Ako ste kreirali .MDL datoteku, odaberite **Use OLE DB File** i kliknite **Browse** kako bi pronašli datoteku na računalu. Ako koristite DSN, kliknite **Use ODBC Data Source Name** i odaberite DSN u okviru, ili kliknite **New** za stvaranje nove datoteke. Ako želite stvoriti string za povezivanje, odaberite **Use ConnectionString**, zatim kliknite **Build**, te upotrijebite dijaloški okvir **Data Link Properties** za stvaranje stringa za povezivanje. Nakon stvaranja stringa za povezivanje, kliknite **OK**. Svojstvo **ConnectionString** bit će popunjeno stringom poput ovog:

   ```
   driver={SQL Server};server=bigsmile;uid=s;pwd=pwd;database=pubs
   ```

5. U prozoru s svojstvima, postavite svojstvo **RecordSource** na SQL izraz. Na primjer:

   ```
   SELECT * FROM Titles WHERE AuthorID = 72
   ```

   Kad pristupate tablici uvijek bi trebali uključiti klauzulu WHERE. Ako to ne učinite, zaključat ćete cijelu tablicu, što će biti velika zapreka ostalim korisnicima.

6. Kreirajte kontrolu **okvira s tekstom** na formi za prikaz informacija iz baze podataka.

7. U prozoru s svojstvima, u svojstvo **DataSource** kontrole Text1 upišite ime kontrole ADO podataka (ADODC1). Na taj način povezat ćete okvir s tekstom s kontrolom ADO podataka.

8. U prozoru s svojstvima, kliknite **DataField** i spustit će se popis polja koja su na raspolaganju. Odaberite ime polja koje želite prikazati.

9. Ponovite korake 6, 7 i 8 za svako dodatno polje kojem želite pristupiti.

10. Pritisnite F5 za pokretanje aplikacije. Četiri gumba s strelicama na kontroli ADO podataka možete upotrijebiti za pomak na početak podataka, kraj podataka ili pomicanje s sloga na slog kroz podatke.

Određivanje svojstava ConnectionString, RecordSource, DataSource i DataField programski

Programski kod koji slijedi pokazuje kako se ova četiri svojstva mogu odrediti programski. Uočite da je postavljanje svojstva DataSource obavezna naredba Set.

```
Private Sub Form_Load()
  With ADODC1
    .ConnectionString = “Driver={SQL driver};” & _
    “server=bigsmile;uid=s;pwd=pwd;database=pubs”
    .RecordSource = “Select * From Titles Where AuthorID = 7”
  End With
  Set Text1.DataSource = ADODC1
  Text1.DataField = “Title”
End Sub
```

Događaji kontrole ADO podataka

Kontrola ADO podataka sadrži nekoliko događaja koje možete programirati. Tablica koja slijedi prikazuje događaje te navodi kad se pojavljuju; unatoč tome svrha ove tablice nije cjelovit popis stanja kad se događaji pojavljuju. Za cjelovitu informaciju, pogledajte pripadajući odlomak u pomoći za pojedini događaj.

događaj	pojavljuje se
WillMove	Kod Recordset.Open, Recordset.MoveNext, Recordset.Move, Recordset.MoveLast, Recordset.MoveFirst, Recordset.MovePrevious, Recordset.Bookmark, Recordset.AddNew, Recordset.Delete, Recordset.Requery, Recordset.Resync
MoveComplete	Nakon WillMove
WillChangeField	Prije promjene svojstva Value
FieldChangeComplete	Nakon WillChangeField
WillChangeRecord	Kod Recordset.Update, Recordset.Delete, Recordset.CancelUpdate, Recordset.UpdateBatch, Recordset.CancelBatch
RecordChangeComplete	Nakon WillChangeRecord
WillChangeRecordset	Kod Recordset.Requery, Recordset.Resync, Recordset.Close, Recordset.Open, Recordset.Filter
RecordsetChangeComplete	Nakon WillChangeRecordset
InfoMessage	Kad davatelj podataka vrati rezultat.

Za više informacija Za isprobavanje ostalih korak-po-korak postupaka korištenjem kontrole ADO podataka, pogledajte u stalnoj pomoći odlomke “Stvaranje jednostavne aplikacije baze podataka korištenjem mreže s podacima i kontrole ADO podataka” “Stvaranje jednostavne aplikacije s kombiniranim okvirom za podatke” te “Stvaranje mreže s podacima vezane s kontrolom okvira s popisom podataka”.

Određivanje svojstava kontrole ADO podataka vezanih za bazu podataka

Kod stvaranja veze, možete upotrijebiti jedan od tri izvora: string za povezivanje, datoteku OLE baze podataka (.MDL) ili ODBC DSN datoteku. Kad koristite DSN, vjerojatno nećete trebati mijenjati ni jedno od ostalih svojstava kontrole.

Ipak, ako ste upoznati s tehnologijom baza podataka, možete izmijeniti neka od ostalih svojstava izloženih u kontroli ADO podataka. Sljedeći popis opisuje svojstva te kontrole koja su povezana s bazama podataka. Ovaj popis također savjetuje logički red postavljanja svojstava.

Napomena Tehnologija baza podataka je složena, i savjeti koji slijede ne bi trebali biti shvaćeni kao pravila.

1. **ConnectionString** – Svojstvo ConnectionString je string koji sadrži sve postavke potrebne za uspostavljanje veze. Parametri proslijeđeni ovom stringu ovisni su o upravljačkim programima. Na primjer, ODBC upravljački programi dopuštaju stringu da sadrži goniča, davatelja, predodređenu bazu podataka, poslužitelja, korisničko ime i zaporku.
2. **UserName** – Ime korisnika, neophodno ako je baza podataka zaštićena zaporkom. Kao i svojstvo Provider, ovo svojstvo može biti određeno u svojstvu ConnectionString. Ako pružite oba svojstva, ConnectionString i UserName, vrijednost svojstva ConnectionString će nadjačati vrijednost svojstva UserName.
3. **Password** – Također potrebno za pristupanje zaštićenoj bazi podataka. Kao kod svojstava Provider i UserName, vrijednost ovog svojstva bit će nadjačana vrijednošću određenom u svojstvu ConnectionString.
4. **RecordSource** – Ovo svojstvo općenito sadrži izraz koji određuje što će biti dohvaćeno iz baze podataka.
5. **CommandType** – Svojstvo CommandType upućuje davatelja podataka je li izvor SQL izraz, ime tablice, pohranjeni potprogram ili je nepoznatog tipa.
6. **CursorLocation** – Ovo svojstvo određuje gdje se nalazi pokazivač, na klijentu ili na poslužitelju. Posljedice ove odluke utječu na idućih nekoliko svojstava koje određujete.
7. **CursorType** – Svojstvo CursorType određuje hoće li slog biti statičkog, dinamičkog ili ključnog tipa pokazivača.
8. **LockType** – Svojstvo LockType određuje kako će podaci biti zaključani kad drugi pokušaju promijeniti podatke koje editirate. Kako odrediti ovo svojstvo je složena odluka, ovisna o brojnim čimbenicima.
9. **Mode** – Svojstvo Mode određuje što namjeravate uraditi s slogovima. Na primjer, možete postići neka poboljšanja izvedbe postavljajući ovo svojstvo kao samo za čitanje, ako ste zainteresirani samo za stvaranje izvještaja.
10. **MaxRecords** – Ovo svojstvo određuje maksimalnu veličinu pokazivača. Kako ćete to odrediti ovisi o veličini sloga koje dohvaćate te o izvorima dostupnim na vašem računalu (memorija). Veliki slog (s puno stupaca i velikim stringovima) uzet će više izvora od malog sloga. Zbog toga svojstvo MaxRecords ne bi trebalo biti veće od neophodnog.
11. **ConnectionTimeout** – Postavite svojstvo ConnectionTimeout na broj sekundi potrebnih za čekanje uspostavljanja veze. Pojavit će se pogreška ako to vrijeme istekne.
12. **CacheSize** – Svojstvo CacheSize određuje koliko slogova može biti dohvaćeno iz pokazivača. Ako svojstvo CursorLocation postavite na klijenta, onda ovo svojstvo može biti postavljeno na manji broj (čak na 1) bez štetnih posljedica. Ako je postavljeno na strani poslužitelja, trebali bi prilagoditi tu vrijednost tako da odgovara broju redaka koje želite istovremeno vidjeti. Na primjer, ako koristite kontrolu

mreže s podacima za pregled 30 redaka, postavite svojstvo CacheSize na 60, što vam omogućuje pomicanje kroz mrežu bez stalnog pozivanja novih podataka.

13. **BOFAction**, **EOFAction** – Ova dva svojstva određuju što će se dogoditi kad je kontrola na početku i kraju pokazivača. Izbori uključuju ostajanje na početku ili kraju, pomicanje na prvi ili posljednji slog, ili dodavanje novog sloga (samo na kraju).

Kontrole koje se povezuju s kontrolom ADO podataka

Svaka kontrola koja ima svojstvo DataSource može se povezati s kontrolom ADO podataka. Sljedeće ugrađene kontrole mogu se povezati s kontrolom ADO podataka:

- kontrolna kućica
- kombinirani okvir
- slika
- natpis
- okvir s popisom
- okvir za sliku
- okvir s tekstom

Sljedeće ActiveX kontrole povezane s podacima su također isporučene s svim verzijama Visual Basica:

- okvir s popisom podataka
- kombinirani okvir s podacima
- mreža za podatke
- Microsoft hijerarhijska fleksibilna mreža
- okvir s tekstom tipa RichText
- Microsoft grafikon
- prebirač datuma i vremena
- kombinirana slika
- mjesečni pregled

Na kraju, možete stvoriti svoje vlastite ActiveX kontrole koje će biti povezane s podacima korištenjem objekta DataBinding.

Za više informacija Pogledajte odlomak “Povezivanje kontrole s izvorom podataka” u 9. poglavlju vodiča *Microsoft Visual Basic 6.0 Component Tools Guide* za detalje o stvaranju vlastitih kontrola povezanih s podacima.

Korištenje kontrole kontrolne kućice

Kontrola kontrolne kućice (Check Box) prikazuje kvačicu kad je odabrana. Obično se koristi za predstavljanje odabira tipa da/ne ili točno/netočno korisniku. Kontrole kontrolnih kućica možete koristiti u grupama za prikaz višestrukih izbora od kojih korisnik može odabrati jedan ili više njih.

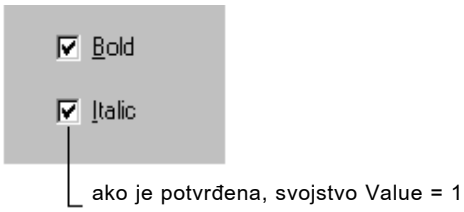
Slika 7.5 Kontrola kontrolne kućice

Kontrola kontrolne kućice slična je kontroli gumba izbora jer se obje kontrole koriste za naznaku odabira koji je napravio korisnik. Razlikuju se po tome što samo jedan gumb izbora u grupi može istovremeno biti odabran. Kod kontrola kontrolne kućice, međutim, može biti odabran bilo koji broj kontrolnih kućica.

Za više informacija Pogledajte "Odabir pojedinih izbora kontrolnim kućicama" u 3. poglavlju "Forme, kontrole i izbornici" za jednostavnu demonstraciju kontrole kontrolne kućice.

Svojstvo Value

Svojstvo Value kontrolne kućice pokazuje je li kontrolna kućica potvrđena, odznačena ili nedostupna (zasjenjena). Kad je potvrđena, vrijednost je postavljena na 1.
Na primjer:



Sljedeća tablica ispisuje vrijednosti i istoznačne konstante Visual Basica koje se koriste za određivanje svojstva Value.

postavka	vrijednost	konstanta
odznačena	0	vbUnchecked
potvrđena	1	vbChecked
nedostupna	2	vbGrayed

Korisnik klikom na kontrolu kontrolne kućice naznačuje potvrđeno ili odznačeno stanje. Nakon toga možete ispitati stanje kontrole i programirati svoju aplikaciju da izvrši neku akciju temeljenu na toj informaciji.

Kontrola kontrolne kućice je standardno postavljena na vbUnchecked. Ako unaprijed želite potvrditi nekoliko kontrolnih kućica u nizu kontrolnih kućica, možete to napraviti postavljanjem svojstva Value na vbChecked u potprogramima Form_Load ili Form_Initialize.

Možete također postaviti svojstvo Value na vbGrayed za onemogućavanje kontrolne kućice. Na primjer, možete isključiti kontrolnu kućicu sve dok se ne ispuni određeni uvjet.

Događaj Click

Kad god korisnik klikne na kontrolu kontrolne kućice, pokreće se događaj Click. Možete zatim programirati svoju aplikaciju da izvede neke akcije ovisno o stanju kontrolne kućice. U sljedećem primjeru, svojstvo Caption kontrole kontrolne kućice mijenja se svaki put kad se klikne na kontrolu, i ukazuje na potvrđeno ili odznačeno stanje.

```
Private Sub Check1_Click()
  If Check1.Value = vbChecked Then
    Check1.Caption = "Potvr|eno"
  ElseIf Check1.Value = vbUnchecked Then
    Check1.Caption = "Odzna~eno"
  End If
End Sub
```

Napomena Ako korisnik pokuša dvaput kliknuti kontrolu kontrolne kućice, svaki klik će biti obrađen odvojeno; to znači da kontrola kontrolne kućice ne podržava događaj dvoklika.

Odgovaranje na miša i tipkovnicu

Događaj Click kontrole kontrolne kućice također se pokreće kad se fokus prebaci na kontrolu korištenjem tipke TAB na tipkovnici te pritiskom na tipku SPACEBAR.

Kontrolnu kućicu možete potvrđivati i odznačivati dodavanjem karaktera & ispred nekog slova u svojstvu Caption kojim ćete stvoriti prečicu tipkovnicom. Na primjer:

U ovom primjeru pritisak na kombinaciju tipki ALT + C naizmjenično mijenja potvrđeno i odznačeno stanje.

Vizualno poboljšanje kontrole kontrolne kućice

Kontrola kontrolne kućice, poput kontrola naredbenog gumba i gumba izbora, može biti vizualno poboljšana promjenom vrijednosti svojstva Style te zatim korištenjem svojstava Picture, DownPicture i DisabledPicture. Na primjer, možda poželite dodati ikonu ili sliku kontrolnoj kućici ili želite prikazati različitu sliku kad je kontrola kliknuta ili onemogućena.

Korištenje kontrole kombiniranog okvira

Kontrola kombiniranog okvira (Combo Box) spaja osobine okvira s tekstom i okvira s popisom. Ova kontrola omogućuje korisniku odabir stavke ili upisom teksta u kombinirani okvir, ili odabirom s popisa.

Slika 7.6 Kontrola kombiniranog okvira

Kombinirani okviri predstavljaju korisniku popis izbora. Ako je broj stavki veći od onog koji se može prikazati u kombiniranom okviru, na kontroli će se automatski prikazati trake za pomicanje. Korisnik se zatim njima može kretati gore-dolje ili lijevo-desno po popisu.

Kad koristiti kombinirani okvir umjesto okvira s popisom

Općenito, kombinirani okvir je prikladniji kad postoji popis *predloženih* izbora, a okvir s popisom je prikladniji kad želite ograničiti unos na ono što je na popisu. Kombinirani okvir sadrži polje za upis podataka, pa izbori koji nisu na popisu mogu biti upisani u to polje.

Kao dodatak, kombinirani okviri štede prostor na formi. Budući da se ne prikazuje puna lista sve dok korisnik ne klikne na strelicu prema dolje (osim stila 1, gdje je popis uvijek otvoren), kombinirani okvir se lako može smjestiti na mali prostor gdje okvir o popisom ne bi stao.

Za više informacija Pogledajte "Korištenje okvira s popisom i kombiniranih okvira" u 3. poglavlju "Forme, kontrole i izbornici", za jednostavnu demonstraciju tih kontrola. Također pogledajte odlomak "Korištenje kontrole okvira s popisom" kasnije u ovom poglavlju za više informacija o kontroli okvira s popisom.

Osobine vezanja s podacima

Visual Basic uključuje i standardni kombinirani okvir i kombinirani okvir za povezivanje s podacima. Iako vam obje verzije omogućuju prikazivanje, editiranje i ažuriranje informacija iz većine standardnih tipova baza podataka, kombinirani okvir povezan s podacima pruža naprednije osobine pristupa podacima. On također podržava skup svojstava i postupaka različit od standardne kontrole kombiniranog okvira.

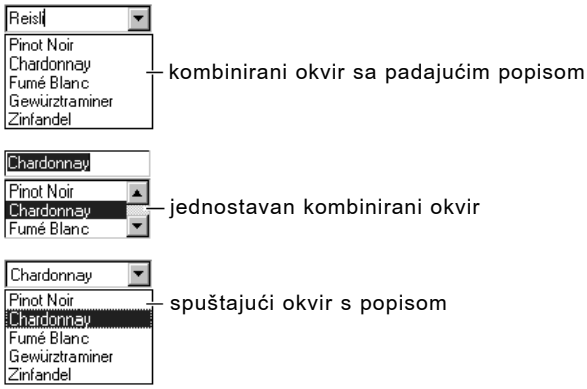
Za više informacija Pogledajte "Korištenje kontrola kombiniranog okvira za podatke i okvira s popisom podataka", kasnije u ovom poglavlju, za više informacija.

Stilovi kombiniranog okvira

Postoje tri stila kombiniranog okvira. Svaki stil može biti određen tijekom izrade aplikacije, i koristi vrijednosti, ili istoznačne konstante Visual Basica, za određivanje stila kombiniranog okvira.

stil	vrijednost	konstanta
Kombinirani okvir s padajućim popisom	0	vbComboDropDown
Jednostavan kombinirani okvir	1	vbComboSimple
Okvir s padajućim popisom	2	vbComboDropDownList

Slika 7.7 Stilovi kombiniranih okvira



Kombinirani okvir s padajućim popisom

Kombinirani okvir s standardnom postavkom stila (Style = 0 – Dropdown Combo) je kombinirani okvir s padajućim popisom. Korisnik može ili direktno upisati tekst (kao u okviru s tekstom) ili kliknuti odijeljenu strelicu na desnoj strani kombiniranog okvira za otvaranje liste izbora. Odabir jednog od izbora ubacuje ga u tekstualni dio na vrhu kombiniranog okvira. Korisnik može otvoriti popis i pritiskom kombinacije ALT + STRELICA DOLJE kad kontrola ima fokus.

Jednostavan kombinirani okvir

Postavljanje svojstva Style kombiniranog okvira na 1 – Simple Combo određuje stil jednostavnog kombiniranog okvira u kojem je popis stalno prikazan. Za prikaz svih stavki popisa, morate kreirati okvir s popisom dovoljno velik da može prikazati sve stavke. Okomita traka za pomicanje se automatski dodaje ako postoji više stavki nego što ih se može prikazati. Korisnik i dalje može direktno upisati tekst ili odabrati stavku s liste. Kao i kombinirani okvir s padajućim popisom, i jednostavni kombinirani okvir također dopušta korisniku upis izbora koji nisu na popisu.

Padajući okvir s popisom

Padajući okvir s popisom (Style = 2 – Dropdown List) sličan je regularnom okviru s popisom – prikazuje listu stavki s kojeg korisnik mora odabrati jednu. Međutim, za razliku od okvira s popisom, lista se ne prikazuje sve dok ne kliknete strelicu na desnoj strani okvira. Ključna razlika između ovog stila i stila kombiniranog okvira s padajućim popisom je da ovdje korisnik ne može upisati tekst u okvir, već može samo odabrati stavku s liste. Koristite ovaj stil okvira s popisom kad vam je prostor najbitniji.

Dodavanje stavki

Za dodavanje stavki u kombinirani okvir, upotrijebite postupak AddItem, koji ima sljedeću sintaksu:

okvir.**AddItem** *stavka* [, *indeks*]

argument	opis
okvir	Ime okvira s popisom ili kombiniranog okvira.
stavka	Izraz tipa stringa koji se dodaje popisu. Ako je *stavka* slovčana konstanta, zatvorite je u navodnike.
indeks	Određuje gdje će u popis biti ubačena nova stavka. *Indeks* jednak 0 predstavlja prvu poziciju. Ako je *indeks* izostavljen, stavka će biti dodana na kraj (ili prema redoslijedu sortiranja).

Iako se stavke liste obično dodaju u potprogramu događaja Form_Load, postupak AddItem možete upotrijebiti u bilo koje vrijeme. To vam daje mogućnost dinamičkog dodavanja stavki na popis (kao odgovor na akcije korisnika).

Sljedeći programski kod dodaje stavke “Chardonnay” “Fumé Blanc” “Gewürztraminer” i “Zinfandel” u kombinirani okvir imena Combo1 s svojstvom Style postavljenim na 0 (vbComboDropDown):

```
Private Sub Form_Load()
  Combo1.AddItem “Chardonnay”
  Combo1.AddItem “Fumé Blanc”
  Combo1.AddItem “Gewürztraminer”
  Combo1.AddItem “Zinfandel”
End Sub
```

Svaki put kad se forma učita tijekom rada aplikacije i korisnik klikne strelicu prema dolje, pojavit će se popis kao što je prikazano na slici 7.8.

Slika 7.8 Kombinirani okvir s “listom vina”

Dodavanje stavki tijekom izrade aplikacije

Stavke također možete unijeti na popis tijekom izrade aplikacije određivanjem svojstva List kontrole kombiniranog okvira u prozoru s svojstvima. Kad odaberete svojstvo List i kliknete pripadajuću strelicu prema dolje, možete upisati stavke i pritisnuti kombinaciju tipki CTRL + ENTER za početak nove linije.

Možete dodati stavke samo na kraj popisa. Dakle, ako želite abecedno sortirati popis, postavite svojstvo Sorted na True. Pogledajte "Sortiranje popisa" u nastavku za više informacija.

Dodavanje stavke na određenu poziciju

Kako bi dodali stavku u popis na određenu poziciju, odredite vrijednost indeksa nakon nove stavke. Na primjer, sljedeća linija programskog koda ubacuje "Pinot Noir" na prvu poziciju, pomičući pozicije ostalih stavki prema dolje:

```
Combo1.AddItem = "Pinot Noir", 0
```

Uočite da 0, a ne 1, određuje prvu stavku u listi (pogledajte sliku 7.9).

Slika 7.9 Dodavanje stavke na popis

Sortiranje popisa

Dodavanje stavki u popis abecednim redoslijedom možete odrediti postavljanjem svojstva Sorted na True te izostavljanjem indeksa. Sortiranje nije osjetljivo na vrstu slova; zbog toga će riječi "chardonnay" i "Chardonnay" biti jednako tretirane.

Kad je svojstvo Sorted postavljeno na True, korištenje postupka AddItem s argumentom *indeks* može dovesti do nepredvidljivih, nesortiranih rezultata.

Brisanje stavki

Možete upotrijebiti postupak RemoveItem za brisanje stavki iz kombiniranog okvira. Postupak RemoveItem ima jedan argument, *indeks*, koji određuje stavku za brisanje:

okvir.**RemoveItem** *indeks*

Argumenti *okvir* i *indeks* isti su kao i za postupak AddItem.

Na primjer, za brisanje prve stavke na listi, dodat ćete sljedeću liniju programskog koda:

```
Combo1.RemoveItem 0
```

Za brisanje svih stavki kombiniranog okvira, upotrijebite postupak Clear:

```
Combo1.Clear
```

Dohvaćanje sadržaja popisa svojstvom Text

Obično je najlakši način dohvaćanja vrijednosti trenutno odabrane stavke upotreba svojstva Text. Svojstvo Text podudara se s svime što je upisano u okvir za tekst kao dio kontrole tijekom izvođenja aplikacije. To može biti ili odabrana stavka ili string kojeg je korisnik upisao u okvir za tekst.

Na primjer, sljedeći programski kod prikazuje informaciju u vinu Chardonnay ako korisnik odabere “Chardonnay” u okviru s popisom:

```
Private Sub Combo1_Click()
  If Combo1.Text = “Chardonnay” Then
    Text1.Text = “Chardonnay je srednje bijelo vino.”
  End If
End Sub
```

Svojstvo Text sadrži trenutno odabranu stavku u kombiniranom okviru Combo1. Programski kod provjerava je li “Chardonnay” odabran te, ako je, prikazuje informaciju u okviru s tekstom Text1.

Pristup stavkama popisa svojstvom List

Svojstvo List pruža pristup svim stavkama u popisu. Ovo svojstvo sadrži matricu u kojoj je svaka stavka popisa jedan element matrice. Svaka stavka predstavljena je u obliku stringa. Za pozivanje stavke iz popisa, upotrijebite ovu sintaksu:

okvir.**List** (*indeks*)

Argument *okvir* je upućivanje na kombinirani okvir, a *indeks* je pozicija stavke u popisu. Prva, najviša, stavka ima indeks 0; sljedeća ima indeks 1 i tako dalje. Na primjer, sljedeći izraz prikazuje treću stavku (*indeks* = 2) popisa u okviru s tekstom:

```
Text1.Text = Combo1.List(2)
```

Određivanje pozicije svojstvom ListIndex

Ako želite znati poziciju odabrane stavke u popisu unutar kombiniranog okvira, upotrijebite svojstvo ListIndex. Ovo svojstvo postavlja ili vraća indeks trenutno odabrane stavke u kontroli i na raspolaganju je samo tijekom rada aplikacije. Određivanje svojstva ListIndex za kombinirani okvir će također prouzročiti događaj Click kontrole.

Vrijednost ovog svojstva je 0 ako je odabrana prva (najviša) stavka, 1 ako je odabrana druga stavka prema dolje i tako dalje. Svojstvo ListIndex ima vrijednost -1 ako ni jedna stavka nije odabrana ili ako korisnik upiše izbor u kombinirani okvir (stila 0 ili 1) umjesto odabira postojeće stavke s popisa.

> **Napomena** Svojstvo NewIndex vam omogućuje praćenje indeksa posljednje stavke dodane u popis. To može biti korisno kad dodajete stavku u sortirani popis.

Vraćanje broja stavki svojstvom ListCount

Kako bi dobili broj stavki u kombiniranom okviru, upotrijebite svojstvo ListCount. Na primjer, sljedeći izraz koristi svojstvo ListCount za utvrđivanje broja stavki u kombiniranom okviru:

```
Text1.Text = "Imate " & Combo1.ListCount & " stavki."
```

Korištenje kontrole naredbenog gumba

Kontrola naredbenog gumba (Command Button) se koristi za započinjanje, prekidanje ili završavanje procesa. Kad je kliknuta, poziva naredbu koja je napisana u njezinom potprogramu događaja Click.

Slika 7.10 Kontrola naredbenog gumba

Većina Visual Basic aplikacija ima naredbene gumbe koji omogućuju korisniku da ih jednostavno klikne za izvođenje akcija. Kad korisnik odabere gumb, on ne samo da pokreće odgovarajuću akciju, nego i izgleda kao da je pritisnut i otpušten pa se zbog toga ponekad i naziva gumb za pritiskanje.

Za više informacija Pogledajte "Pokretanje akcije klikom na gumb" u 3. poglavlju "Forme, kontrole i izbornici", za jednostavnu demonstraciju kontrole naredbenog gumba.

Dodavanje naredbenog gumba na formu

U svojim aplikacijama vjerojatno ćete koristiti jedan ili više naredbenih gumbâ. Kako bi dodali naredbeni gumb na formu, kreirajte ga kao i svaku drugu kontrolu. Naredbenim gumbima može se mijenjati veličina uz pomoć miša ili određivanjem njihovih svojstava Height i Width.

Određivanje natpisa

Upotrijebite svojstvo Caption za promjenu teksta koji se prikazuje na naredbenom gumbu. Tijekom izrade aplikacije, ovo svojstvo možete postaviti ako ga odaberete u prozoru s svojstvima kontrole. Kad svojstvo Caption odredite tijekom izrade aplikacije, tekst na gumbu će se odmah ažurirati.

Svojstvo Caption možete postaviti na najviše 255 karaktera. Ako vaš natpis prekorači širinu naredbenog gumba, bit će prelomljen u iduću liniju. Unatoč tome, bit će odrezan ako kontrola ne može prihvatiti njegovu ukupnu visinu.

Pismo kojim se ispisuje natpis na naredbenom gumbu možete promijeniti određivanjem njegovog svojstva Font.

Stvaranje prečica tipkovnicom

Svojstvo Caption možete upotrijebiti za stvaranje prečica pristupnim tipkama za vaše naredbene gumbe dodavajući znak & prije slova koje želite upotrijebiti kao pristupnu tipku. Na primjer, kako bi stvorili pristupnu tipku za natpis "Pokreni" dodajte znak & prije slova "P": "&Pokreni". Tijekom izvođenja aplikacije, slovo "P" će biti podvučeno i korisnik može odabrati taj naredbeni gumb istovremenim pritiskom ALT + P.

Napomena Za uključivanje znaka & u natpis bez stvaranja pristupne tipke, napišite dva znaka & (&&). U natpisu će biti prikazan samo jedan znak & i nijedan karakter neće biti podvučen.

Određivanje svojstava Default i Cancel

Na svakoj formi možete odrediti jedan naredbeni gumb kao predodređeni naredbeni gumb – to znači da će, uvijek kad korisnik pritisne tipku ENTER, taj naredbeni gumb biti kliknut neovisno o tome koja kontrola na formi ima fokus. Za određivanje naredbenog gumba kao predodređenog postavite njegovo svojstvo Default na True.

Možete također odrediti i predodređeni gumb za odustajanje. Kad je svojstvo Cancel naredbenog gumba postavljeno na True, on će biti kliknut uvijek kad korisnik pritisne tipku ESC, neovisno o tome koja kontrola na formi ima fokus.

Odabir naredbenog gumba

Naredbeni gumb može biti odabran tijekom izvođenja aplikacije korištenjem miša ili tipkovnice na sljedeće načine:

- Upotrijebite miša za klik na gumb.
- Pomaknite fokus na gumb pritiskanjem tipke TAB, te zatim odaberite gumb pritiskom na tipku SPACEBAR ili ENTER.
- Pritisnite pristupnu tipku (ALT + podvučeno slovo) naredbenog gumba.
- Ako je naredbeni gumb *predodređeni naredbeni gumb* za formu, pritisak na ENTER odabire gumb, čak i ako promijenite fokus na drugu kontrolu.
- Ako je naredbeni gumb *predodređeni gumb za odustajanje* za formu, pritisak na ESC odabire gumb, čak i ako promijenite fokus na drugu kontrolu.

Svojstvo Value

Uvijek kad je naredbeni gumb odabran, njegovo svojstvo Value je postavljeno na True i pokreće se događaj Click. Vrijednost False (standardno) pokazuje da gumb nije odabran. Svojstvo Value možete upotrijebiti u programskom kodu za pokretanje događaja Click naredbenog gumba. Na primjer:

```
cmdPokreni.Value = True
```

Događaj Click

Kad je gumb kliknut, pojavljuje se događaj Click naredbenog gumba i poziva se programski kod koji ste napisali u potprogramu događaja Click.

Klik na naredbeni gumb također stvara događaje MouseDown i MouseUp. Ako namjeravate dodati potprograme događaja za te srodne događaje, budite sigurni da se njihove akcije neće sukobljavati. Redoslijed po kojem se ova tri događaja pojavljuju ovisi od kontrole do kontrole. Kod kontrole naredbenog gumba, ovi događaji pojavljuju se po ovom redu: MouseDown, Click, MouseUp.

> **Napomena** Ako korisnik pokuša dvaput kliknuti kontrolu naredbenog gumba, svaki klik će biti obrađen odvojeno; to znači da kontrola naredbenog gumba ne podržava događaj dvoklika.

Za više informacija Pogledajte 11. poglavlje "Odgovor na događaje miša i tipkovnice", za više informacija o događajima MouseDown i MouseUp.

Vizualno poboljšanje naredbenog gumba

Kontrola naredbenog gumba, poput kontrola kontrolne kućice i gumba izbora, može biti vizualno poboljšana promjenom vrijednosti svojstva Style te zatim korištenjem svojstava Picture, DownPicture i DisabledPicture. Na primjer, možda poželite dodati ikonu ili sliku naredbenom gumbu ili želite prikazati različitu sliku kad je gumb kliknut ili onemogućen.

Korištenje kontrole općeg dijaloga

Kontrola općeg dijaloga (Common Dialog) pruža standardan skup dijaloških okvira za operacije kao što su otvaranje i spremanje datoteka, određivanje opcija za ispis, te odabir boja i pisama. Ova kontrola također ima sposobnost prikaza pomoći pokretanjem mehanizma za Windows pomoć.

Slika 7.11 Kontrola općeg dijaloga

Kontrola općeg dijaloga pruža međusklop između Visual Basica i rutina u dinamičkoj biblioteci Microsoft Windowsa Commdlg.dll. Kako bi kreirali dijaloški okvir korištenjem ove kontrole, datoteka Commdlg.dll mora se nalaziti u direktoriju Windows\System.

Kontrolu općeg dijaloga upotrebljavate u vašoj aplikaciji tako da je dodate formi i odredite svojstva kontrole. Dijalog koje se prikazuje pozivom te kontrole određen je postupcima kontrole. Tijekom rada aplikacije, kad se pozove odgovarajući postupak, prikazuje se dijaloški okvir ili se izvršava mehanizam pomoći; tijekom izrade aplikacije, kontrola općeg dijaloga se prikazuje kao ikona na formi. Veličina te ikone se ne može mijenjati.

Kontrola općeg dijaloga omogućuje vam prikaz sljedećih obično korištenih dijaloških okvira:

- Open
- Save As
- Color
- Font
- Print

Kako upotrijebiti kontrolu općeg dijaloga

1. Ako to već niste napravili, dodajte kontrolu općeg dijaloga u alatni okvir odabirom stavke **Components** u izborniku **Project**. Pronađite i odaberite kontrolu u dijalogu s karticama **Controls**, i kliknite gumb **OK**.
2. U alatnom okviru, kliknite kontrolu **CommonDialog** i kreirajte je na formi.

   Kad stvarate kontrolu općeg dijaloga na formi, kontrola će automatski odrediti svoju veličinu. Kao i kontrola mjerača vremena i kontrola općeg dijaloga je nevidljiva tijekom izvođenja aplikacije.
3. Tijekom rada aplikacije, upotrijebite prikladan postupak, kao što je ispisano u sljedećoj tablici, za prikaz željenog dijaloga.

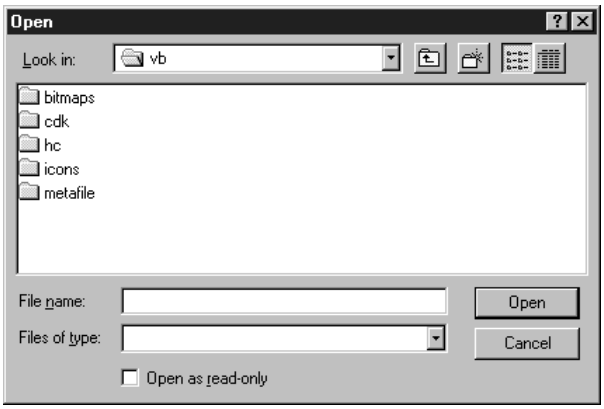
postupak	prikazani dijalog
ShowOpen	Open
ShowSave	Save As
ShowColor	Color
ShowFont	Font
ShowPrinter	Print
ShowHelp	Poziva Windows Help

Prikaz dijaloških okvira Open i Save As

Dijaloški okvir Open omogućuje korisniku određivanje pogona, direktorija, nastavka imena datoteke i imena datoteke.

Dijaloški okvir Save As izgleda isto kao i dijaloški okvir Open, osim naslova dijaloga, i imena datoteka koje su ispisana zasjenjeno. Tijekom rada aplikacije, kad korisnik odabere datoteku i zatvori dijaloški okvir, svojstvo FileName se koristi za čuvanje imena odabrane datoteke.

Slika 7.12 Dijaloški okvir Open



Kako prikazati dijaloški okvir Open

1. Odredite popis filtera koji će biti prikazani u okviru s popisom **Files of type**.

   To možete napraviti određivanjem svojstva Filter koristeći sljedeći oblik:

   opis1 | *filter1* | *opis2* | *filter2* ...

   Opis je string koji se ispisuje u okviru s popisom – na primjer "Tekstualne datoteke (*.txt)". *Filter* je stvarni filter – na primjer "*.txt". Svaki skup *opis* | *filter* mora biti razdvojen simbolom cijevi (|).

2. Upotrijebite postupak ShowOpen za prikaz dijaloškog okvira.

Nakon što korisnik odabere datoteku, upotrijebite svojstvo FileName za dobivanje imena odabrane datoteke.

Kod općih dijaloških okvira, kad je svojstvo CancelError postavljeno na True, stvara se pogreška kad korisnik klikne gumb Cancel dijaloškog okvira. Hvatanjem pogreške kad je prikazan dijaloški okvir, možete odrediti da je pritisnut gumb Cancel.

Sljedeći programski kod prikazuje dijaloški okvir Open i koristi odabrano ime datoteke kao argument za potprogram koji će otvoriti datoteku:

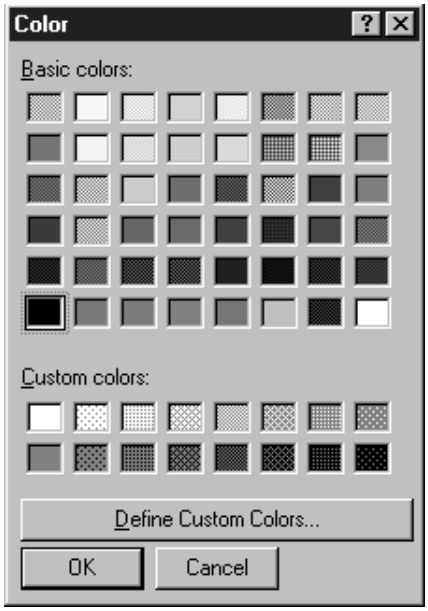
```
Private Sub mnuFileOpen_Click()
   ‘ Svojstvo CancelError je postavljeno na True.
   CommonDialog1.CancelError = True
   On Error GoTo ObradaPogreške
   ‘ Postavljanje filtera.
   CommonDialog1.Filter = “Sve datoteke (*.*)|*.*|Tekstualne _
   datoteke (*.txt)|*.txt|Batch datoteke (*.bat)|*.bat”
   ‘ Određivanje podrazumijevanog filtera.
   CommonDialog1.FilterIndex = 2
```

```
    ‘ Prikaz dijaloškog okvira Open.
    CommonDialog1.ShowOpen
    ‘ Poziv potprograma za otvaranje datoteke.
    OpenFile (CommonDialog1.FileName)
    Exit Sub
ObradaPogreške:
    ‘ Korisnik je pritisnuo gumb Cancel.
    Exit Sub
End Sub
```

Korištenje dijaloškog okvira Color

Dijaloški okvir Color omogućuje korisniku odabir boje iz palete ili stvaranje i odabir korisničke boje. Tijekom rada aplikacije, kad korisnik odabere boju i zatvori dijaloški okvir, trebate upotrijebiti svojstvo Color za dobivanje odabrane boje.

Slika 7.13 Dijaloški okvir Color



Kako prikazati dijaloški okvir Color

1. Postavite svojstvo Flags kontrole dijaloškog okvira na konstantu Visual Basica cdlCCRGBInit.
2. Upotrijebite postupak ShowColor za prikaz dijaloškog okvira.

Upotrijebite svojstvo Color za dobivanje RGB vrijednosti boje koju je korisnik odabrao. Sljedeći programski kod prikazuje dijaloški okvir Color kad korisnik klikne naredbeni gumb Command1:

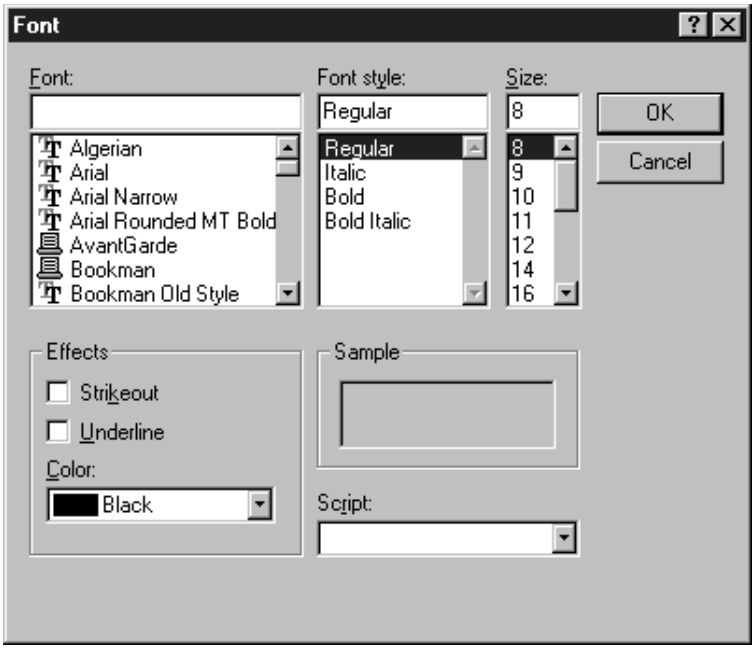
```
Private Sub Command1_Click()
  ‘ Postavljanje svojstva Cancel na True.
  CommonDialog1.CancelError = True
  On Error GoTo ObradaPogreške
  ‘ Odre|ivanje svojstva Flags.
  CommonDialog1.Flags = cdlCCRGBInit
  ‘ Prikaz dijaloškog okvira Color.
  CommonDialog1.ShowColor
  ‘ Postavljanje boje pozadine forme na odabranu boju.
  Form1.BackColor = CommonDialog1.Color
  Exit Sub

ObradaPogreške:
  ‘ Korisnik je pritisnuo gumb Cancel.
  Exit Sub
End Sub
```

Korištenje dijaloškog okvira Font

Dijaloški okvir Font omogućuje korisniku odabir pisma određivanjem njegove veličine, boje i stila. Jednom kad korisnik napravi odabire u dijaloškom okviru Font, sljedeća svojstva sadržavaju informacije o odabiru korisnika.

svojstvo	određuje
Color	Odabrana boja. Za korištenje ovog svojstva, najprije morate postaviti svojstvo Flags na cdlCFEffects.
FontBold	Je li odabrano **podebljavanje**.
FontItalic	Je li odabrano *nakošenje*.
FondStrikethru	Je li odabrano ~~precrtavanje~~.
FontUnderline	Je li odabrano podvlačenje.
FontName	Ime odabranog pisma.
FontSize	Veličina odabranog pisma.

Slika 7.14 Dijaloški okvir Font



Kako prikazati dijaloški okvir Font

1. Postavite svojstvo Flags na jednu od sljedećih vrijednosti konstanti Visual Basica:
   - cdlCFScreenFonts (pisma na ekranu)
   - cdlCFPrinterFonts (pisma na pisaču)
   - cdlCFBoth (za pisma i na ekranu i na pisaču)

     Oprez Svojstvo Flags morate postaviti na jednu od ove tri vrijednosti prije prikaza dijaloškog okvira Font, inače će se pojaviti pogreška No fonts exist (ne postoje pisma).
2. Upotrijebite postupak ShowFont za prikaz dijaloškog okvira.

   Sljedeći programski kod određuje svojstva pisma za okvir s tekstom temeljena na izborima korisnika u dijaloškom okviru Fonts:

```
Private Sub Command1_Click()
   ‘ Postavljanje svojstva Cancel na True.
   CommonDialog1.CancelError = True
   On Error GoTo ObradaPogreške
   ‘ Određivanje svojstva Flags.
   CommonDialog1.Flags = cdlCFBoth Or cdlCFEffects
   ‘ Prikaz dijaloškog okvira Font.
```

```
    CommonDialog1.ShowFont
    ‘ Određivanje svojstava teksta u skladu
    ‘ s odabirima korisnika.
    Text1.Font.Name = CommonDialog1.FontName
    Text1.Font.Size = CommonDialog1.FontSize
    Text1.Font.Bold = CommonDialog1.FontBold
    Text1.Font.Italic = CommonDialog1.FontItalic
    Text1.Font.Underline = CommonDialog1.FontUnderline
    Text1.Font.Strikethru = CommonDialog1.FontStrikethru
    Text1.ForeColor = CommonDialog1.Color
    Exit Sub
ObradaPogreške:
    ‘ Korisnik je pritisnuo gumb Cancel.
    Exit Sub
End Sub
```

Korištenje dijaloškog okvira Print

Dijaloški okvir Print omogućuje korisniku određivanje izgleda izlaza na pisač. Korisnik može odrediti opseg strana koje će biti ispisane, kvalitetu ispisa, broj kopija i tako dalje. Ovaj dijaloški okvir također prikazuje informacije o trenutno instaliranom pisaču te omogućuje korisniku postavljanje ili reinstaliranje novog predodređenog pisača.

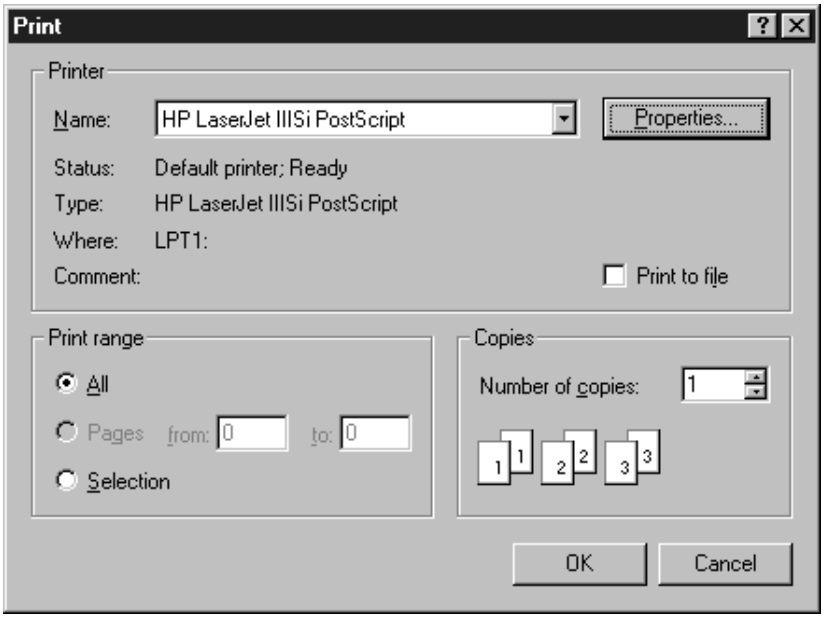
Napomena Ovaj dijaloški okvir ne šalje stvarno podatke pisaču. On omogućuje korisniku da odredi kako će podaci biti otisnuti. Morate napisati programski kod za ispis podataka u obliku koji je određen.

Za više informacija Pogledajte 12. poglavlje “Rad s tekstom i grafikom” za informacije o ispisu podataka.

Tijekom rada aplikacije, kad korisnik napravi odabire u dijaloškom okviru Print, sljedeća svojstva sadržavaju informacije o odabirima korisnika.

svojstvo	određuje
Copies	Broj kopija koje će se ispisati.
FromPage	Početna stranica ispisa.
ToPage	Posljednja stranica ispisa.
hDC	Kontekst sredstva za odabrani pisač.
Orientation	Orijentacija stranice (uspravna ili položena).

Slika 7.15 Dijaloški okvir Print



Kako prikazati dijaloški okvir Print

1. Odredite sve željene standardne postavke za dijalog postavljanjem odgovarajućih svojstava dijaloga Print.

   Na primjer, za ispis broja 2 u okviru Copies kad se prikaže dijalog, postavite svojstvo Copies na 2:

   ```
   CommonDialog1.Copies = 2
   ```

2. Upotrijebite postupak ShowPrinter za prikaz dijaloškog okvira Print.

Sljedeći programski kod prikazuje dijaloški okvir Print kad korisnik klikne naredbeni gumb Command1:

```
Private Sub Command1_Click()
  Dim PrvaStrana, PosljednjaStrana, BrojKopija, Orijentacija, i
  ‘ Postavljanje svojstva Cancel na True.
  CommonDialog1.CancelError = True
  On Error GoTo ObradaPogreške
  ‘ Prikaz dijalo{kog okvira Print.
  CommonDialog1.ShowPrinter
  ‘ Dobivanje vrijednosti iz dijaloškog okvira
  ‘ koje je odredio korisnik.
  PrvaStranica = CommonDialog1.FromPage
  PosljednjaStranica = CommonDialog1.ToPage
  BrojKopija = CommonDialog1.Copies
  Orijentacija = CommonDialog1.Orientation
```

```
    For i = 1 To BrojKopija
      ‘ Ovdje postavite kod za slanje podataka pisaču.
    Next
    Exit Sub
ObradaPogreške:
    ‘ Korisnik je pritisnuo gumb Cancel.
    Exit Sub
End Sub
```

Napomena Ako je svojstvo PrinterDefault postavljeno na True, možete ispisati na objekt Printer Visual Basica. Kao dodatak, kad je svojstvo PrinterDefault postavljeno na True, svaka promjena koju korisnik napravi u dijelu Setup dijaloškog okvira Print koristi se za promjenu odrednica u korisnikovim postavkama pisača.

Korištenje postupka ShowHelp za prikaz pomoći

Postupak ShowHelp kontrole općeg dijaloga omogućuje vam prikaz datoteke pomoći.

Kako prikazati datoteku pomoći korištenjem postupka ShowHelp

1. Odredite svojstva HelpCommand i HelpFile.
2. Upotrijebite postupak ShowHelp za prikaz određene datoteke pomoći.

Sljedeći programski kod prikazuje određenu datoteku pomoći kad korisnik klikne naredbeni gumb Command1:

```
Private Sub Command1_Click()
    ‘ Postavljanje svojstva Cancel na True.
    CommonDialog1.CancelError = True
    On Error GoTo ObradaPogreške
    ‘ Određivanje svojstva HelpCommand.
    CommonDialog1.HelpCommand = cdlHelpForceFile
    ‘ Određivanje datoteke pomoći.
    CommonDialog.HelpFile = “c:\Windows\Cardfile.hlp”
    ‘ Prikaz mehanizma Windows Help
    CommonDialog1.ShowHelp
    Exit Sub

ObradaPogreške:
    ‘ Korisnik je pritisnuo gumb Cancel.
    Exit Sub
End Sub
```

Za više informacija Pogledajte odlomke “Svojstvo HelpCommand” “Svojstvo HelpFile” i “Postupak ShowHelp” u dijelu *Microsoft Visual Basic 6.0 Language Reference* biblioteke *Microsoft Visual Basic 6.0 Reference Library* za više informacija o prikazivanju datoteka pomoći kontrolom općeg dijaloga.

Korištenje kontrole podataka

Ugrađena kontrola podataka (Data) ostvaruje pristup podacima korištenjem mehanizma Microsoft Jet Database – isti mehanizam baze podataka koji pokreće Microsoft Access. Ta tehnologija pruža vam neograničen pristup većini standardnih oblika baza podataka i dopušta vam stvaranje aplikacija svjesnih podataka bez pisanja bilo kakvog programskog koda. Ugrađena kontrola podataka najbolje odgovara malim (lokalnim) bazama podataka, kao što su Access i ISAM baze podataka.

Ugrađenu kontrolu podataka možete upotrijebiti za stvaranje aplikacija koje prikazuju, editiraju i ažuriraju informacije iz većine tipova postojećih baza podataka, uključujući Microsoft Access, Btrieve, dBASE, Microsoft FoxPro i Paradox. Možete je također upotrijebiti za pristup Microsoft Excelu, Lotusu 1-2-3, te standardnim ASCII tekstualnim datotekama kao da su prave baze podataka. Kao dodatak, kontrola podataka dopušta vam pristup i obradu udaljenih ODBC baza podataka (Open Database Connectivity) kao što su Microsoft SQL poslužitelj i Oracle.

> Napomena U Visual Basic su uključene i kontrola podataka i kontrola udaljenih podataka za sukladnost unazad. Međutim, zbog fleksibilnosti kontrole ActiveX objekata podataka (ActiveX Data Objects, ADO), preporučljivo je da novu aplikaciju baze podataka stvarate korištenjem kontrole ADO podataka. Za više detalja, pogledajte “Korištenje kontrole ADO podataka” ranije u ovom poglavlju.

Kontrola podataka, kontrola udaljenih podataka i kontrola ADO podataka su sadržajno slične: sve tri su “kontrole podataka” koje povezuju izvor podataka s kontrolom povezanom s podacima. Sve tri također dijele isti izgled – skup od četiri gumba koji omogućuju korisniku trenutan skok na početak niza slogova, kraj niza slogova, te pomicanje prema naprijed i nazad kroz niz slogova.

Stvaranje jednostavne aplikacije kao baze podataka kontrolom podataka

Kako stvoriti jednostavnu aplikaciju baze podataka kontrolom podataka

1. Stvorite kontrolu **Data** na formi. Kontrola podataka je ugrađena kontrola i uvijek je dostupna.
2. Kliknite kontrolu **Data** za odabir, i pritisnite F4 za prikaz prozora s svojstvima.
3. U prozoru s svojstvima postavite svojstvo **Connect** na tip baze podataka koju želite koristiti.
4. U prozoru s svojstvima, u svojstvo **DatabaseName** upišite ime datoteke ili direktorija baze podataka s kojom se želite povezati.
5. U prozoru s svojstvima, u svojstvo **RecordSource** upišite ime tablice u bazi podataka kojoj želite pristupiti.
6. Kreirajte kontrolu **okvira s tekstom** (**TextBox**) na formi.
7. Kliknite kontrolu **okvira s tekstom** za odabir, i u prozoru s svojstvima postavite svojstvo **DataSource** na kontrolu podataka.

8. U prozoru s svojstvima, u svojstvo **DataField** upišite ime polja u bazi podataka kojeg želite pregledati ili promijeniti.
9. Ponovite korake 6, 7 i 8 za svako dodatno polje kojem želite pristupiti.
10. Pritisnite F5 za pokretanje aplikacije.

Određivanje svojstava povezanih s podacima za kontrolu podataka

Sljedeća svojstva koja su povezana s podacima mogu biti određena tijekom izrade aplikacije. Ova lista savjetuje logički red postavljanja svojstava:

Napomena Tehnologija baze podataka je složena vještina i savjeti koji slijede ne bi trebali biti shvaćeni kao pravila.

1. **RecordsetType** – Svojstvo RecordsetType određuje je li skup slogova tablica, dinamički skup slogova ili trenutni ispis. Izbor utječe na raspoloživa svojstva skupa slogova. Na primjer, tip trenutnog ispisa skupa slogova je ograničeniji od dinamičkog skupa slogova.
2. **DefaultType** – Svojstvo DefaultType određuje hoće li se koristiti radni prostor tipa JET ili ODBCDirect.
3. **DefaultCursorType** – Svojstvo DefaultCursorType određuje mjesto pokazivača. Možete dopustiti ODBC goniču da sam odredi mjesto pokazivača, odredi poslužitelja ili ODBC pokazivač. Svojstvo DefaultCursorType je valjano samo kad se koriste radni prostori tipa ODBCDirect.
4. **Exclusive** – Određuje je li podatak za okolinu s jednim ili više korisnika.
5. **Options** – Ovo svojstvo određuje karakteristike skupa slogova. Na primjer, u okolini s više korisnika, možete postaviti svojstvo Options tako da ne priznaje promjene koje su napravili drugi korisnici.
6. **BOFAction**, **EOFAction** – Ova dva svojstva određuju što će se dogoditi kad je kontrola na početku i kraju pokazivača. Izbori uključuju ostajanje na početku ili kraju, pomicanje na prvi ili posljednji slog, ili dodavanje novog sloga (samo na kraju).

Korištenje kontrola kombiniranog okvira za podatke i okvira s popisom podataka

Kontrole kombiniranog okvira za podatke (DataCombo) i okvira s popisom podataka (DataList) jako sliče standardnim kontrolama okvira s popisom i kombiniranog okvira opisanim u odlomku “Tipovi kontrola za povezivanje”, ali postoje neke važne razlike koje im daju veliku fleksibilnost i iskoristivost u aplikacijama baza podataka. Obje kontrole mogu se automatski popuniti iz polja baze podataka ili kontrole podataka s kojom su povezane. Kao dodatak, one mogu i proslijediti odabrano polje u drugu kontrolu podataka, što ih čini idealnima za aplikacije s “preglednim tablicama”.

Moguće primjene

- U relacijskoj bazi podataka, upotrijebite podatak iz jedne tablice za dostavu vrijednosti koje će biti ubačene u drugu (povezanu) tablicu. Na primjer, u bazi podataka s inventarom, imena dobavljača su spremljena u jednoj tablici gdje svaki dobavljač ima jedinstveni identifikacijski broj. Druga tablica koja prikazuje proizvode koristi taj broj za određivanje firme koja dobavlja proizvod. Upotrijebite kontrolu okvira s popisom podataka za prikaz imena dobavljača dok (nevidljivo) dostavljate identifikacijski broj dobavljača tablici s proizvodima.
- Korisnik može suziti traženje odabirom kriterija iz spuštajuće popisa. Na primjer, aplikacija baze podataka s izvješćima prodaje može koristiti kontrolu okvira s popisom podataka kako bi korisniku dopustila odabir države ili područja prodaje. Kad korisnik napravi odabir, izbor se automatski prosljeđuje drugoj kontroli podataka koja se postavlja na zapise o prodaji iz odabranog područja.

Kao i kod njihovih ugrađenih duplikata, glavna razlika između kontrola kombiniranog okvira za podatke i okvira s popisom podataka je u tome da kontrola kombiniranog okvira za podatke sadrži okvir za tekst čiji sadržaj može biti mijenjan.

Za više informacija Za objašnjenje o mogućnostima povezivanja ovih kontrola s tablicama u bazi podataka pogledajte "Povezivanje dvije tablice korištenjem kontrola DataList i DataCombo", u stalnoj pomoći. Za stvaranje jednostavne aplikacije korištenjem povezanih tablica, pogledajte "Stvaranje jednostavne aplikacije s kontrolom DataCombo", također u stalnoj pomoći.

Značajnija svojstva kontrola

Neka važna svojstva kontrola kombiniranog okvira za podatke i okvira s popisom podataka uključuju:

svojstvo	opis
BoundText	Sadrži vrijednost polja označenog u svojstvu BoundColumn.
SelectedItem	Vraća oznaku reda odabrane stavke.
MatchEntry	Omogućuje mod proširenog traženja za određivanje stavki u popisu stvorenom od kontrole DataCombo.
IntegralHeight	Mijenja veličinu kontrole tako da se može prikazati točan broj cijelih redova.
VisibleCount	Određuje broj vidljivih stavki u popisu.

Napomena Svojstvo DataFormat kontrole kombiniranog okvira za podatke je produženo svojstvo. Zbog toga je uvijek vidljivo u popisu svojstava i može se odrediti iz programskog koda. Unatoč tome kontrola kombiniranog okvira za podatke oblikuje samo najvišu stavku na popisu. To može biti zbuniti krajnjeg korisnika, koji vidi oblikovanu najgornju stavku, ali za odabir dobiva popis neoblikovanih stavki. Oblikovana stavka može zavarati krajnjeg korisnika koji pret-

postavlja da će stavka biti unesena u bazu podataka kao oblikovana. Zbog tih razloga, preporučljivo je ne određivati svojstvo DataFormat kod korištenja kontrole kombiniranog okvira za podatke.

Za više informacija Da biste isprobali vodič koji korak-po-korak pokazuje korištenje svojstva BoundText, pogledajte "Stvaranje mreže podataka povezane s kontrolom okvira s popisom podataka", kasnije u ovom poglavlju. Za potpuni popis svojstava i postupaka za ove kontrole pogledajte "Kontrola DataList" i "Kontrola DataCombo", u stalnoj pomoći.

Povezivanje dvije tablice korištenjem kontrola kombiniranog okvira za podatke i okvira s popisom podataka

Karakteristika koja razlikuje kontrolu kombiniranog okvira za podatke od kontrole okvira s popisom podataka je sposobnost pristupa dvjema različitim tablicama i povezivanje podatka iz prve tablice s poljem u drugoj. To se ostvaruje korištenjem dva izvora podataka (kao što su kontrola ADO podataka ili okolina za podatke).

Relacijske tablice i "neprijateljske" vrijednosti

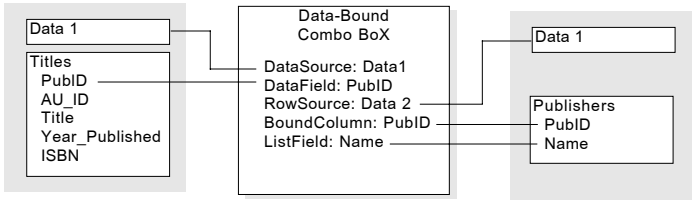
Informacije koje se ponavljano koriste u relacijskoj bazi podataka ne spremaju se u cjelovitom obliku na više mjesta. Umjesto toga, veći dio informacije spremljen je u skupu slogova sastavljenom od puno polja; unutar tih polja nalazi se polje s identifikacijskim brojem koje na jedinstven način označava skup slogova. Na primjer, baza podataka Biblio koja se isporučuje s Visual Basicom sadrži imena nekoliko izdavačkih firmi u tablici imena "Publishers". Tablica sadrži puno polja, kao što su adresa, grad, poštanski broj i broj telefona. Međutim, zbog jednostavnosti, pretpostavite da su polja Name i PubID dva glavna polja u tablici. Polje Name sadrži ime izdavača, dok polje PubID sadrži razmjerno "neprijateljsku" vrijednost kao što je broj ili kod. Ipak, ta neprijateljska vrijednost je važnija jer na jedinstven način označava izdavača, i služi kao veza za cijeli skup slogova. To je vrijednost koja je spremljena u više skupova slogova u drugoj tablici.

Druga tablica ima ime "Titles" i svaki skup slogova sadrži informacije kao što su naslov knjige, godina izdavanja i ISBN oznaku. Unutar tih polja uključeno je i polje imena "PubID". To polje ima isto ime kao i odgovarajuće polje u tablici Publishers zato jer sadrži istu vrijednost koja povezuje naslov knjige s određenim izdavačem.

Shema djelovanja predstavlja mali problem: u aplikaciji baze podataka koja omogućuje korisniku ubacivanje novih naslova, korisnik će ponekad morati upisati brojeve koji označuju izdavača. To je u redu ako je korisnik zapamtio jedinstveni identifikacijski broj svakog izdavača, ali bilo bi lakše za korisnika kad bi umjesto toga vidjeli ime izdavača, a aplikacija bi spremala pripadajuću vrijednost u bazu podataka. Kontrole okvira s popisom podataka i kombiniranog okvira za podatke jednostavno rješavaju taj problem.

Dva izvora podataka, tri polja, bez programiranja

Kontrole okvira s popisom podataka i kombiniranog okvira za podatke koriste dva izvora za svladavanje problema. Dok ove kontrole prikazuju samo ime izdavača (iz tablice Publishers), one zapisuju samo vrijednost polja PubID u tablicu Titles. U prozoru s svojstvima, postavite svojstvo RowSource na izvor podataka koji će dobavljati podatke koji će biti zapisani (tablica Publishers). Nakon toga, u svojstvu DataSource odredite izvor podataka u koji će se zapisivati (tablica Titles). Na kraju, odredite svojstva DataField, ListField i BoundColumn. Slika ispod pokazuje kako se kontroli kombiniranog okvira za podatke dodjeljuju dva izvora podataka (u obliku kontrole podataka) i tri polja:



Ukratko, svojstvo ListField određuje koje se polje zapravo prikazuje u kontroli. U ovom slučaju, to je ime izdavača. Svojstvo BoundColumn, s druge strane, određuje koje će polje u tablici Publishers dati trenutnu vrijednost tablici Titles. Uočite da polje PubID u tablici Publishers ne može (i ne smije) biti mijenjano. Umjesto toga, vrijednost pronađena u polju PubID zapisuje se u polje određeno svojstvom DataField. U ovom slučaju, to je polje PubID tablice Titles.

Sljedeća tablica sžima svojstva i objašnjava kako ih koristiti.

svojstvo	opis
DataSource	Ime kontrole podataka s kojim je povezana kontrola okvira s popisom podataka ili kontrola kombiniranog okvira za podatke.
DataField	Ime polja u skupu slogova određenom u svojstvu DataSource. To polje će biti korišteno za određivanje elementa koji će biti označen u popisu. Ako je napravljen novi odabir, ovo polje će biti ažurirano kad se pomaknete na novi zapis.
RowSource	Ime kontrole podataka koja će biti upotrijebljena za popunjavanje popisa.
BoundColumn	Ime polja u skupu slogova određenom u svojstvu RowSource. To polje mora biti istog tipa kao i podatak u svojstvu DataField koji će biti upotrijebljen za ažuriranje popisa.
ListField	Ime polja u skupu slogova određenom u svojstvu RowSource koji će biti upotrijebljen za popunjavanje popisa.

Napomena Kontrola okvira s popisom podataka i kontrola kombiniranog okvira za podatke mogu se također upotrijebiti i s jednom kontrolom podataka. Da biste to napravili, odredite svojstva DataSource i RowSource na istu kontrolu podataka, a svojstva DataField i BoundColumn postavite na isto polje u skupu slogova kontrole podataka. U takvom slučaju, lista će biti popunjena s vrijednostima svojstva ListField iz istog skupa slogova iz kojeg je i ažurirana. Ako je određeno svojstvo ListField, ali nije svojstvo BoundColumn, svojstvo BoundColumn će automatski biti postavljeno na vrijednost svojstva ListField.

Za više informacija Za isprobavanje postupka korak-po-korak koji izgrađuje jednostavnu aplikaciju baze podataka s kontrolom kombiniranog okvira za podatke, pogledajte sljedeći odlomak "Stvaranje jednostavne aplikacije s kontrolom kombiniranog okvira za podatke".

Stvaranje jednostavne aplikacije s kontrolom kombiniranog okvira za podatke

Sljedeći primjer koristi kontrolu kombiniranog okvira za podatke kako bi stvorio ekran za unos podataka u tablicu Titles iz baze podataka Northwind.mdb. On dopušta korisniku unos novih proizvoda i dodjeljivanje postojećim dobavljačima pružanjem pregledne tablice s imenima svih dobavljača. Kad korisnik dođe do polja Supplier u formi za unos, može odabrati dobavljača iz okvira s popisom. Kad odabere dobavljača, polje SupplierID koje sadrži identifikacijski broj tog dobavljača kopira se u polje SupplierID tablice Products.

Kako stvoriti preglednu tablicu s kontrolom kombiniranog okvira za podatke

1. Stvorite OLEDB izvor podataka za bazu podataka Northwind.

   Ako izvor podataka nije kreiran, slijedite upute u "Kako stvoriti Northwind OLE DB izvor podataka" kasnije u ovom poglavlju.

2. Stvorite novi standardni EXE projekt u Visual Basicu.

   Ako u alatnom okviru ne postoje kontrole **DataGrid**, **DataCombo** ili **ADO Data**, kliknite desnom tipkom miša na **alatni okvir**, i upotrijebite dijaloški okvir **Components** za njihovo dodavanje.

3. Dodajte na svoju formu kontrolu kombiniranog okvira za podatke, kontrolu mreže podataka, te dvije kontrole ADO podataka.

4. U prozoru s svojstvima, odredite svojstva prve kontrole podataka (Adodc1) prema sljedećoj tablici.

svojstvo	postavka
Name	adoDataSource
ConnectionString	Northwind.mdl
RecordSource	Select * From Products;
Caption	Products

5. U prozoru s svojstvima odredite svojstva druge kontrole podataka (Adodc2) prema sljedećoj tablici.

svojstvo	postavka
Name	adoRowSource
ConnectionString	Northwind.mdl
RecordSource	Select CompanyName, SupplierID From Suppliers;
Caption	Suppliers
Visible	False

6. U prozoru s svojstvima odredite svojstva kontrole mreže podataka prema sljedećoj tablici.

svojstvo	postavka
Name	grdProducts
DataSource	adoDataSource
Caption	Products

7. U prozoru s svojstvima odredite svojstva kontrole kombiniranog okvira za podatke prema sljedećoj tablici.

svojstvo	postavka
Name	dcbSuppliers
DataSource	adoDataSource
DataField	SupplierID
RowSource	adoRowSource
ListField	CompanyName
BoundColumn	SupplierID

8. Na kraju, dodajte sljedeći programski kod u modul forme:

```
Private Sub Form_Load()
  ‘ Sakrivanje polja SupplierID u kontroli mreže podataka,
  ‘ tako da se korisnik ne pita koju vrijednost promijeniti.
  grdProducts.Columns(“SupplierID”).Visible = False
End Sub
```

9. Pokrenite projekt.

Kroz skup slogova se možete kretati klikanjem na strelice vidljivih kontrola ADO podataka. Dok to radite, kontrola kombiniranog okvira za podatke će ažurirati i prikazivati ime dobavljača za svaki proizvod. Za mijenjanje polja SupplierID, kliknite strelicu na kontroli kombiniranog okvira za podatke za prikaz spuštajuće popisa, te ponovno kliknite na drugog dobavljača kako bi promijenili vrijednost zapisanu u polju SupplierID.

Korištenje kontrole mreže s podacima

Kontrola mreže s podacima (DataGrid) slična je proračunskoj tablici i prikazuje nizove redova i stupaca predstavljajući zapise i polja nekog objekta skupa slogova. Kontrolu mreže podataka možete upotrijebiti za stvaranje aplikacije koja će korisniku omogućiti čitanje i zapisivanje u većinu baza podataka. Kontrola mreže podataka može se brzo uobličiti tijekom izrade aplikacije uz malo ili bez koda. Kad tijekom izrade aplikacije odredite svojstvo DataSource kontrole mreže s podacima, kontrola će se automatski popuniti, a zaglavlja stupaca će automatski biti određena iz skupa slogova koji je izvor podataka. Nakon toga možete mijenjati stupce mreže; brisati, prerasporediti i dodati zaglavlja stupaca, ili prilagoditi širinu bilo kojeg stupca.

Tijekom rada aplikacije, svojstvo DataSource može se programski prebaciti na pregled druge tablice, ili možete promijeniti upit aktivnoj bazi podataka kako bi dobili drugačiji niz zapisa.

Napomena Kontrola mreže podataka je po programskom kodu uskladiva s kontrolom DBGrid iz Visual Basica 5.0 uz jednu iznimku: kontrola mreže podataka ne podržava pojam “nepovezani mod” (unbound mode). Kontrola DBGrid sadržana je u Visual Basicu u direktoriju Tools.

Moguće primjene

- Pregled i editiranje podataka u udaljenoj ili lokalnoj bazi podataka.
- Upotrijebite kontrolu mreže podataka zajedno s drugim kontrolama povezanim s podacima, kao što je kontrola okvira s popisom podataka, za prikaz zapisa iz jedne tablice, koji su povezani kroz zajedničko polje, u drugoj tablici prikazanoj drugom kontrolom povezanom s podacima.

Korištenje oblikovanja tijekom izrade aplikacije

Uz pomoć kontrole mreže podataka možete stvoriti aplikaciju baze podataka bez pisanja programskog koda koristeći prednosti njezinih osobina oblikovanja. Sljedeće upute izlažu u glavnim crtama općenite korake potrebne za korištenje ove kontrole u tipičnoj upotrebi. Za potpune upute korak-po-korak, pogledajte “Primjer mreže s podacima 1: stvaranje jednostavne aplikacije baze podataka korištenjem mreže podataka”, u stalnoj pomoći.

Kako ostvariti kontrolu mreže podataka tijekom izrade

1. Stvorite .MDL datoteku (Microsoft Data Link) za bazu podataka kojoj želite pristupiti. Pogledajte “Stvaranje Northwind OLE DB veze podataka”, kasnije u ovom poglavlju, za primjer.
2. Postavite kontrolu ADO podataka na formu, i postavite svojstvo **ConnectionString** na OLE DB izvor podataka stvoren u 1. koraku.
3. U polju **RecordSource** kontrole **ADO podataka**, upišite SQL izraz koji vraća skup slogova. Na primjer:

   ```
   Select * From ImeMojeTablice Where CustID = 12
   ```

4. Kreirajte kontrolu **mreže podataka** na formi i postavite svojstvo **DataSource** na kontrolu **ADO podataka**.
5. Kliknite desnom tipkom miša na kontrolu **mreže podataka** i kliknite **Retrieve Fields**.
6. Kliknite desnom tipkom miša na kontrolu **mreže podataka** i kliknite **Edit**.
7. Promijenite veličinu mreže, obrišite ili dodajte stupce.
8. Kliknite desnom tipkom miša na kontrolu **mreže podataka** i kliknite **Properties**.
9. Korištenjem dijaloškog okvira **Property Pages**, odredite prikladna svojstva kontrole za uobličivanje mreže prema vašim željama izgleda i ponašanja.

Promjena prikazanih podataka tijekom izvođenja

Jednom kad ste stvorili mrežu koristeći mogućnost oblikovanja pri izradi aplikacije, možda ćete poželjeti dinamički mijenjati izvor podataka za mrežu tijekom izvođenja. Općeniti postupci za ostvarivanje toga su raspravljeni u nastavku.

Promjena izvora zapisa za izvor podataka

Najčešći postupak promjene prikazanih podataka je promjena upita u svojstvu DataSource. Na primjer, ako kontrola mreže podataka koristi kontrolu ADO podataka kao izvor, ponovno određivanje svojstva RecordSource i obnavljanje kontrole ADO podataka promijenit će prikazane podatke:

```
‘ Kontrola ADO podataka se povezuje s tablicom Products
‘ baze podataka Northwind. Novi upit također traži
‘ sve zapise u kojima je SupplierId = 12.
Dim strQuery As String
strQuery = “SELECT * FROM Suppliers WHERE SupplierID = 12”
Adodc1.RecordSource = strQuery
Adodc1.Refresh
```

Promjena izvora podataka

Tijekom rada aplikacije možete ponovno postaviti svojstvo DataSource na druge izvore podataka. Na primjer, možete imati nekoliko kontrola ADO podataka, svaku povezanu s drugom bazom podataka, ili postavljenu na različita svojstva RecordSource. Jednostavno ponovno postavite svojstvo DataSource s jedne kontrole ADO podataka na drugu:

```
‘ Ponovno postavljanje kontrole ADO podataka koja je povezana
‘ s bazom podataka Pubs i koristi tablicu Authors.
Set DataGrid1.DataSource = adoPubsAuthors
```

Ponovno povezivanje s izvorom podataka

Kad koristite kontrolu mreže podataka s udaljenim bazama podataka kao što je SQL poslužitelj, moguće je da se promijeni struktura tablice. Na primjer, u tablicu može biti dodano polje. U tom slučaju, možete pozvati postupak Rebind za ponovno stvaranje mreže iz nove strukture. Zapamtite da će, ako ste mijenjali izgled stupaca u mreži tijekom izrade aplikacije, kontrola mreže podataka pokušati ponovno stvoriti trenutni izgled, uključujući i prazne stupce. Unatoč tome, možete prisiliti mrežu da obnovi sve stupce ako najprije pozovete postupak ClearFields.

Vraćanje vrijednosti iz mreže podataka

Kad se jednom mreža podataka poveže s bazom podataka, možete poželjeti nadzirati koju je ćeliju korisnik kliknuo. Upotrijebite događaj RowColChange – a ne događaj Click – kao što je prikazano.

```
Private Sub DataGrid1_RowColChange(LastRow As Variant, _
ByVal LastCol As Integer)
  ‘ Ispis teksta, reda i stupca ćelije koju je korisnik kliknuo.
  Debug.Print DataGrid1.Text; DataGrid1.Row; DataGrid1.Col
End Sub
```

Korištenje postupaka CellText i CellValue

Svojstva CellText i CellValue korisna su kad je stupac oblikovan korištenjem svojstva NumberFormat. Svojstvo NumberFormat mijenja oblik svakog stupca koji sadrži broj bez promjene oblika stvarnih podataka. Na primjer, u mreži s stupcem imena ProductID koji sadrži cijele brojeve, programski kod koji slijedi će za posljedicu imati prikaz vrijednosti u mreži podataka oblika “P-0000”. Drugim riječima, iako je stvarna vrijednost u polju ProductID jednaka “3”, vrijednost prikazana od mreže bit će “P-0003”.

```
Private Sub Form_Load()
  DataGrid1.Columns(“ProductID”).NumberFormat = “P-0000”
End Sub
```

Za vraćanje stvarne vrijednosti sadržane u bazi podataka, upotrijebite postupak CellValue, kako je ovdje pokazano:

```
Private Sub DataGrid1.RowColChange(LastRow As Variant, _
ByVal LastCol As Integer)
  Debug.Print _
  DataGrid1.Columns("ProductID").CellValue(DataGrid1.Bookmark)
End Sub
```

Napomena Oba postupka, CellValue opisan gore, i CellText upotrijebljen u nastavku odlomka, zahtijevaju oznaku svojstva kao argument za ispravan rad.

Suprotno tome, ako želite vratiti oblikovanu vrijednost polja, upotrijebite postupak CellText:

```
Private Sub DataGrid1_RowColChange(LastRow As Variant, _
ByVal LastCol As Integer)
  Debug.Print _
  DataGrid1.Columns("ProductID").CellText(DataGrid1.Bookmark)
End Sub
```

Napomena Ovdje opisani postupak CellText jednak je korištenju postupka Text kontrole mreže podataka.

Kamo dalje

Za postupak korak-po-korak stvaranja jednostavne aplikacije ovom kontrolom, pogledajte "Stvaranje jednostavne aplikacije baze podataka kontrolama mreže podataka i ADO podataka", ili "Stvaranje mreže podataka povezane s kontrolom okvira s popisom podataka", kasnije u ovom poglavlju.

Kako bi naučili više o razdijeljenim objektima i kako ih programirati, pogledajte "Rukovanje izgledima mreže podataka", kasnije u ovom poglavlju.

Stvaranje Northwind OLE DB veze podataka

Bitan korak u pristupu podacima je stvaranje OLE DB izbora podataka za svaku bazu podataka kojoj želite pristupiti. Koraci koji slijede stvaraju takav objekt za datoteku Nwind.mdb (Northwind), isporučenu s Visual Basicom. Ovaj izvor podataka koristi se u nekim primjerima postupaka u dokumentaciji Visual Basica. Na računalu trebate samo jednom stvoriti OLE DB izvor podataka.

Kako stvoriti Northwind OLE DB izvor podataka

1. Otvorite Windows Explorer ili Windows NT Explorer.
2. Otvorite direktorij u kojem želite stvoriti OLE DB izvor podataka. U ovom primjeru, otvorite Program Files, Microsoft Visual Studio i VB98.
3. Kliknite desnom tipkom miša da desno okno Explorera i na moćnom izborniku kliknite **New**. Sa popisa tipova datoteka, kliknite **Microsoft Data Link**.
4. Promijenite ime novoj datoteci u **Northwind.MDL**.

5. Kliknite desnom tipkom miša na tu datoteku i u pomoćnom izborniku kliknite na **Properties** za prikaz dijaloškog okvira **Northwind.MDL Properties**.
6. Kliknite karticu **Connection**.
7. Kliknite okvir **Provider** i odaberite **Microsoft Jet 3.51 OLE DB Provider**.
8. U okviru **Data Source** upišite stazu do datoteke nwind.mdb.
9. Kliknite **Test Connection** za provjeru povezivanja.
10. Ako je test uspješan, kliknite **OK**.

Napomena Izvor podataka OLE DB tipa možete također kreirati ako u kontrolnom panelu kliknete na ikonu **Data Links**. Na dijaloškom okviru **Organize Data Link Files**, kliknite **New** za stvaranje novog izvora podataka.

Stvaranje jednostavne aplikacije baze podataka kontrolama mreže podataka i ADO podataka

Korištenjem samo mreže podataka i kontrole ADO podataka možete stvoriti aplikaciju baze podataka koja će omogućavati krajnjem korisniku čitanje i zapisivanje u skup slogova.

Kako stvoriti jednostavnu bazu podataka korištenjem kontrole ADO podataka

1. Stvorite izvor podataka OLE DB tipa za bazu podataka Northwind.

   Ako izvor podataka nije stvoren, slijedite korake u “Stvaranje Northwind OLE DB veze podataka”.
2. Stvorite novi standardni EXE projekt u Visual Basicu.

   Ako u alatnom okviru nije prisutna kontrola **DataGrid**, kliknite desnom tipkom miša na **alatni okvir** i upotrijebite dijaloški okvir **Components** za njezino učitavanje. Također dodajte i kontrolu ADO podataka ako ne postoji.
3. Postavite primjer te dvije kontrole na praznu formu.
4. Postavite svojstvo **ConnectionString** kontrole ADO podataka na izvor podataka Northwind.

   Kliknite kontrolu ADO podataka kako bi je odabrali, i pritisnite F4 za pojavljivanje prozora s svojstvima. Kliknite **ConnectionString** te zatim kliknite **OLE DB File**. Kliknite izvor podataka Northwind.
5. Odredite svojstvo **RecordSource** kontrole ADO podataka.

   U prozoru s svojstvima, kliknite **RecordSource** i upišite SQL izraz za popunjavanje kontrole mreže podataka. U ovom slučaju, upišite **Select * From Products**.
6. Postavite svojstvo **DataSource** kontrole mreže podataka na kontrolu ADO podataka.

   Kliknite na kontrolu mreže podataka kako bi je odabrali. U prozoru s svojstvima, kliknite **DataSource** i bit će prikazana spuštajuća lista s svim kontrolama podataka – u ovom slučaju samo s kontrolom ADO podataka. Odaberite tu kontrolu.
7. Pritisnite F5 za pokretanje projekta.

Stvaranje mreže podataka povezane s kontrolom okvira s popisom podataka

Uobičajena upotreba mreže podataka je prikazivanje "detalja" koje pruža jedna tablica u bazi podataka. Na primjer, baza podataka Northwind (Nwind.mdb) sadrži dvije tablice, jednu s imenom "Suppliers" i drugu imena "Products". U ovom primjeru, upotrijebit ćemo kontrolu okvira s popisom podataka za prikaz imena firmi dobavljača iz tablice "Suppliers". Kad korisnik klikne na ime bilo koje firme, kontrola okvira s popisom podataka će se snabdjeti podatkom SupplierID firme. Korištenjem tog broja, može se sastaviti upit koji će pronaći sve zapise u tablici "Products" koji imaju odgovarajući podatak SupplierID. Drugim riječima, kad korisnik klikne firmu (u kontroli okvira s popisom podataka), svi proizvodi te firme pojavit će se u kontroli mreže podataka.

Kako popuniti kontrolu mreže podataka proizvodima određenog dobavljača

1. Provjerite postoji li izvor podataka tipa OLE DB za bazu podataka Northwind na računalu; ako takav izvor podataka nije stvoren, slijedite korake u "Stvaranje Northwind OLE DB veze podataka".
2. Stvorite novi standardni EXE projekt u Visual Basicu.

   Ako u alatnom okviru ne postoje kontrole **mreže podataka**, **okvira s popisom podataka** i **ADO podataka**, kliknite desnom tipkom miša na **alatni okvir** i kliknite **Components**. U dijaloškom okviru **Components** dvaput kliknite **Microsoft DataGrid Control**, **Microsoft DataList Control** i **Microsoft ADO Control**.
3. Postavite primjere **mreže podataka** i **okvira s popisom podataka** na praznu formu.

   Postavite kontrolu **okvira s popisom podataka** u gornji lijevi kut forme, a kontrolu mreže podataka negdje ispod nje.
4. Postavite dva primjera **kontrole ADO podataka** na formu.

   Odaberite prvu **kontrolu ADO podataka** i pritisnite F4 za prikaz njezinih svojstava. Postavite svojstvo **Name** ove kontrole na **adoSuppliers**. Odaberite drugu **kontrolu ADO podataka** i postavite njezino svojstvo **Name** na **adoProducts**. Postavite prvu kontrolu točno ispod **kontrole okvira s popisom podataka**, a drugu točno ispod **kontrole mreže podataka**.
5. Postavite svojstvo **ConnectionString** obje **kontrole ADO podataka** na Northwind OLE DB izvor podataka.

   Odaberite kontrolu imena **adoSuppliers** i odredite njezino svojstvo **ConnectString** na Northwind OLE DB izvor podataka (Nortwind.mdl). Odaberite kontrolu imena **adoProducts** i ponovite postupak.
6. Odredite svojstvo **RecordSource** obje **kontrole ADO podataka**.

   Odaberite kontrolu **adoSuppliers** i kliknite **RecordSource** na stranici **Properties**. Upišite `Select * From Suppliers`. Ovaj upit upućuje **kontrolu ADO podataka** da vrati sve zapise u tablici Suppliers. Odaberite kontrolu **adoProducts**, kliknite **RecordSource** i upišite `Select * From Products`. Ovaj upit vraća sve zapise iz tablice Products.

7. Postavite svojstvo **RowSource** kontrole **okvira s popisom podataka** na **adoSuplliers**.

   Svojstvo **RowSource** određuje koji će izvor podataka dati podatke za svojstvo **ListField**.

8. Postavite svojstvo **ListField** kontrole okvira s popisom podataka na **CompanyName**.

   Svojstvo **ListField** je postavljeno na ime polja u tablici Suppliers. Tijekom izvođenja aplikacije, kontrola **okvira s popisom podataka** prikazuje vrijednost polja određenog u ovom svojstvu. U ovom primjeru, svojstvo će prikazivati ime tvrtke pronađeno u tablici Suppliers.

9. Postavite svojstvo **BoundColumn** kontrole okvira s popisom podataka na **SupplierID**.

   Svojstvo **BoundColumn** je postavljeno na drugo polje u tablici Suppliers. U ovom slučaju, svojstvo je postavljeno na polje SupplierID. Kad je kliknuta kontrola okvira s popisom podataka, svojstvo BoundColumn vraća vrijednost polja SupplierID povezanog s firmom prikazanom u kontroli okvira s popisom podataka. Ta vrijednost bit će upotrijebljena u upitu tablice Products, koja pruža podatke za **kontrolu mreže podataka**.

10. Postavite svojstvo **DataSource** kontrole **mreže podataka** na **adoProducts**.

    Svojstvo **DataSource** određuje izvor podataka za kontrolu. U ovom slučaju, svojstvo je postavljeno na kontrolu ADO podataka s imenom adoProducts, koja vraća sve zapise u tablici Products.

11. U kodnom modulu forme, dodajte ovaj programski kod:

```
Private Sub DataList1_Click()
   ‘ Određivanje varijable stringa koja će sadržavati novi upit.
   ‘ Novi upit koristi svojstvo BoundText kontrole DataList za
   ‘ dodjelu vrijednosti SupplierID. Novi upit jednostavno pita
   ‘ za sve proizvode s istim SupplierID. Ovaj upit se dodjeljuje
   ‘ svojstvu RecordSource kontrole ADO podataka imena adoProducts.
   ‘ Nakon obnavljanja kontrole, mreža podataka se ažurira novim
   ‘ skupom slogova svih proizvoda koje jeisporučila ista tvrtka.
   Dim strQuery As String
   strQuery = “Select * FROM Products WHERE SupplierID = & _
   DataList1.BoundText
   With adoProducts
      .RecordSource = strQuery
      .Refresh
   End With
   With DataGrid1
      .ClearFields
      .ReBind
   End With
End Sub
```

12. Pokrenite projekt.

    Kliknite ime bilo koje tvrtke u kontroli **okvira s popisom podataka** i kontrola **mreže podataka** će se automatski ažurirati svim proizvodima isporučenim od te tvrtke.

Rad s stupcima

Podatke prikazane u kontroli mreže s podacima možete dinamički mijenjati promjenom svojstva DataSource. Na primjer, možete prikazati drugu tablicu iz iste baze podataka. Ako to napravite, kontrola mreže podataka će prikazati podatke samo s standardnim svojstvima.

Dodavanje, brisanje i skrivanje stupaca

Možete programski dodati, brisati ili skrivati stupce korištenjem postupaka zbirke Columns i objekta Column.

Dodavanje i brisanje stupca

Za dodavanje stupca tijekom rada aplikacije, upotrijebite postupak Add. Ako najprije odredite varijablu i dodijelite novi objekt varijabli, možete odrediti razna svojstva kratkim kodom.

```
Private Sub DodajStupac()
  ‘ Dodavanje stupca na krajnji desni položaj. Slijedi određivanje
  ‘ svojstava Visible, Width, Caption i Alignment. Svojstvo
  ‘ DataField određuje s kojim će poljem biti povezan stupac.
  Dim c As Column
  Set c = DataGrid1.Columns.Add(DataGrid1.Columns.Count)
  With c
    .Visible = True
    .Width = 1000
    .Caption = “Moj novi stupac”
    .DataField = Adodc1.Recordset.Fields(“ProductName”).Name
    .Alignment = dbgRight
  End With
End Sub
```

Bilo koji stupac možete obrisati korištenjem postupka Remove. Obavezno odredite koji stupac ćete brisati korištenjem argumenta. Sljedeći programski kod će obrisati stupac koji je kliknut.

```
Private Sub DataGrid1_HeadClick(ByVal StuIndeks As Integer)
  DataGrid1.Columns.Remove StuIndeks
End Sub
```

Skrivanje stupca

Možete skriti bilo koji stupac postavljanjem njegovog svojstva Visible na False. To je posebno korisno kad želite ograničiti broj stupaca tako da ih korisnik može vidjeti ili editirati. Sljedeći primjer prolazi kroz zbirku Columns, skrivajući sve osim nekoliko stupaca.

```
Private Sub SakrijStupac()
   ‘ Korištenje svojstva DataField za određivanje koji je stupac
   ‘ ispitan. Prikazivanje samo tri stupca: ImeProizvoda,
   ‘ CijenaProizvoda i ProizvodaNaSkladištu.
   Dim c As Column
   For Each c In DataGrid1.Columns
      Select Case c.DataField
      Case “ImeProizvoda”
         c.Visible = True
      Case “CijenaProizvoda”
         c.Visible = True
      Case “ProizvodaNaSkladi{tu”
         c.Visible = True
         c.Caption = “Na skladištu” ‘ Promjena zaglavlja stupca.
      Case Else                    ‘ Skrivanje svih ostalih stupaca.
         c.Visible = False
      End Select
   Next c
End Sub
```

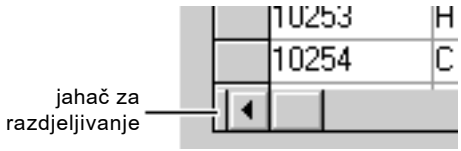
Rukovanje izgledima mreže podataka

Mreža koja je “razdijeljena” omogućuje krajnjem korisniku da ima više od jednog izgleda istih podataka. Na primjer, zamislite da imate veliku tablicu koja se sastoji od deset polja. U takvom slučaju, izgled skupa zapisa u kontroli mreže podataka bit će širok deset stupaca, te, osim ako vaša forma nije jako široka, korisnik neće moći vidjeti više od nekoliko stupaca. Nadalje, zamislite da korisnika zanima samo prvi i posljednji stupac (na primjer, ime u prvom stupcu i broj telefona u posljednjem). Kako bi se mogla vidjeti oba stupca jedan uz drugog (bez promjene rasporeda stupaca), mreža može biti razdijeljena.

Stvaranje razdijeljenog objekta

Tijekom izrade aplikacije, razdjeljivanje možete stvoriti desnim klikom na mrežu, klikom na stavku Edit, ponovnim desnim klikom i klikom stavke Split. Nakon toga razdjeljivanje možete prilagoditi desnim klikom na kontrolu za prikaz dijaloškog okvira Property Pages. Na kartici Splits prilagodite razdjeljivanje. Za uklanjanje razdjeljivanja, kliknite desnom tipkom miša na razdjeljivanje i odaberite Remove.

Tijekom izvođenja aplikacije, krajnji korisnik također može ručno razdijeliti mrežu (osim ako takva akcija nije onemogućena) klikom na jahač (tab) koji se nalazi desno od donjeg lijevog kuta kontrole, kao što je prikazano na ovoj slici:



U pravilu, kontrola mreže podataka sadrži jedan objekt Split. Ovo je programski kod koji krajnjem korisniku onemogućuje dodavanje razdjeljivanja:

```
DataGrid1.Splits(0).AllowSizing = False
```

Programsko dodjeljivanje i brisanje razdjeljivanja

Kontrola mreže podataka sadrži zbirku objekata Split. Kako bi programski dodali razdjeljivanje, upotrijebite postupak Add, na sljedeći način:

```
DataGrid1.Splits.Add 1
```

> **Napomena** Postupak Add zahtjeva indeks novog razdjeljivanja kao argument. Za dodavanje razdjeljivanja, odredite argument indeksa jednak svojstvu Count zbirke Splits.

Korištenjem postupka Add zbirke Splits, možete programski dodati razdjeljivanja kad ih trebate. Budući da dodavanje više od dva razdjeljivanja može učiniti mrežu teškom za korištenje, možete ograničiti broj razdjeljivanja korištenjem svojstva Count zbirke.

```
If DataGrid1.Splits.Count < 3 Then       ‘ Dodavanje razdjeljivanja.
  DataGrid1.Splits.Add DataGrid1.Splits.Count
End If
```

Usklađivanje razdjeljivanja

Kad imate više od jednog razdjeljivanja, trebat ćete nadzirati kako se razdjeljivanja pomiču. Na primjer, u mreži s tri razdjeljivanja, možete odlučiti uskladiti samo prvo i treće razdjeljivanje, dopuštajući srednjem razdjeljivanju da se neovisno pomiče. Za usklađivanje dva (ili više) razdjeljivanja, postavite svojstvo ScrollGroup svakog objekta Split na istu vrijednost.

```
‘ Usklađivanje prvog i tre}eg Split objekta.
With DataGrid1
  .Splits(0).ScrollGroup = 1
  .Splits(1).ScrollGroup = 2
  .Splits(2).ScrollGroup = 1
End With
```

Izgled razdjeljivanja možete osim toga i prilagoditi postavljanjem svojstva ScrollBars tako da prikazuje samo jednu traku za pomicanje za usklađenu grupu.

Kontroliranje ponašanja tipke TAB i kursorskih tipki

Korištenjem svojstava WrapCellPointer, TabAcrossSplits i TabAction možete odrediti kako će se mreža ponašati kad krajnji korisnik pritisne tipku tab ili kursorske tipke.

Od ova tri svojstva, svojstvo TabAction je na najvišoj razini jer odlučuje hoće li svojstva WrapCellPointer i TabAcrossSplits uopće imati efekta. Svojstvo TabAction ima tri postavke: Control Navigation, Column Navigation i Grid Navigation. Kad je ovo svojstvo postavljeno na Control Navigation, pritisak na tipku TAB mijenja fokus na iduću kontrolu po vrijednosti svojstva TabIndex. Ova postavka ima veću prednost od svojstava WrapCellPointer i TabAcrossSplits.

Svojstvo WrapCellPointer određuje kako će se ponašati tipka TAB i kursorske tipke unutar jednog razdjeljivanja. Ako je ovo svojstvo postavljeno na True, a aktivna ćelija je u zadnjem stupcu, kad korisnik pritisne tipku TAB, sljedeći red prvog stupca će postati aktivna ćelija. Međutim, ako je aktivna ćelija u zadnjem redu posljednjeg stupca, neće biti mjesta kamo se treba “prebaciti”.

Svojstvo TabAcrossSplits određuje kako djeluju tipka TAB i kursorske tipke kad na mreži postoje dva ili više razdjeljivanja. Ako je ovo svojstvo postavljeno na True, a aktivna ćelija je u posljednjem stupcu bilo kojeg razdjeljivanja osim posljednjeg, pritisak na tipku TAB ili kursorsku tipku će za posljedicu imati “skok” aktivne ćelije na prvi stupac idućeg razdjeljivanja. Aktivna ćelija će ostati u istom redu.

> **Napomena** Ako su oba svojstva WrapCellPointer i TabAcrossSplits postavljena na True, aktivna ćelija neće se prebaciti osim ako nije u posljednjem stupcu posljednjeg razdjeljivanja. Ona će se tad prebaciti na idući red u prvom stupcu prvog razdjeljivanja.

Prilagođavanje zbirke stupaca

Svaki objekt Split ima svojstvo Column koje vam omogućuje rukovanje zbirkom objekata stupaca. Čineći to, možete promijeniti izgled svakog objekta Split. Na primjer, jedno razdjeljivanje može sadržavati dva stupca koji prikazuju polja s imenima i prezimenima, dok drugo razdjeljivanje može sadržavati stupce koji prikazuju telefonski broj i adresu. Kako bi to postigli, sakrijte ostale stupce postavljanjem njihovih svojstava Visible na False, kao što je ovdje prikazano.

```
‘ Prolaz kroz zbirku Columns za ispitivanje svojstva DataField
‘ svakog stupca. Ako test nije uspješan, skrivanje stupca.
Dim i As Integer
‘ Skrivanje svih stupaca osim stupca Proizvod.
For i = 0 To DataGrid1.Splits(0).Columns.Count - 1
  If DataGrid1.Splits(0).Columns(i).DataField <> “Proizvod” Then
    DataGrid1.Splits(0).Columns(i).Visible = False
  End If
Next i
```

```
‘ Skrivanje svih stupaca osim stupca Cijena.
For i = 0 To DataGrid1.Splits(0).Columns.Count - 1
  If DataGrid1.Splits(1).Columns(i).DataField <> “Cijena” Then
    DataGrid1.Splits(1).Columns(i).Visible = False
  End If
Next i
```

Praćenje zapisa oznakama i zbirkom oznaka

Oznake (bookmarks) i zbirka oznaka SelBookmarks pružaju način za praćenje zapisa. To je nužno kad programirate posebne posljedatke unutar aplikacije, kao što je omogućavanje krajnjem korisniku da ručno odabere nekoliko nepovezanih zapisa, i izvođenje opsežnih ažuriranja odabranih zapisa. U takvom slučaju trebate pratiti zapise koji su odabrani, i zbog toga ćete koristiti zbirku oznaka SelBookmarks i njezina svojstva.

Dvije funkcije, postupci CellText i CellValue, zahtijevaju oznake kako bi ispravno radile.

Praćenje korisničkih odabira

Zbirka SelBookmarks sadrži oznake svih odabranih zapisa. Kad korisnik ručno odabere zapise (držanjem pritisnute tipke CTRL dok klikanjem odabire zapise) oznaka svakog odabranog zapisa se dodaje u zbirku. Korištenjem uobičajenih postupaka ponavljanja, možete odrediti što je bilo odabrano, spremiti oznake (u slučaju da trebate obnoviti vrijednosti), i izvesti operaciju:

```
Dim i As Integer   ‘ Brojač
Dim intCount As Integer
intCount = DataGrid1.SelBookmarks.Count - 1
ReDim arrSelBK(intCount)  ‘ Određivanje matrice za čuvanje oznaka.
For i = 0 To intCount
  arrSelBK(i) = DataGrid1.SelBookmarks(i)
  ‘ Ovdje izvedite operacije. Ako operacije trebaju biti
  ‘ poništene, izaći iz petlje iz upotrijebite matricu
  ‘ za povratak vrijednosti prije izmjene.
Next i
```

Programski odabir zapisa dodavanjem u zbirku oznaka SelBookmarks

Možete također programski odabrati zapise dodajući ih u zbirku. Na primjer, možete imati mrežu koja pokazuje sve narudžbe određenog kupca. Za označavanje svih zapisa u kojima je kupac potrošio više od 100 kuna, filtrirajte zapise, i dodajte rezultate kao oznake u zbirku SelBookmarks.

```
Dim rs As Recordset
Set rs = Adodc1.Recordset

While Not rs.EOF
  If rs!SupplierID = 12 Then
    DataGrid1.SelBookmarks.Add rs.Bookmark
  End If
  rs.MoveNext
Wend
```

Prikaz izračunatih polja

Pretpostavimo da imate polje u tablici imena “Cijena” i želite izračunati porez na svaku stavku u tablici korištenjem lokalnog iznosa poreza. To je izračunato polje, koje možete stvoriti prilagođivanjem upita u svojstvu DataSource za izračunavanje vrijednosti, i vraćanjem te vrijednosti kontroli mreže podataka.

Kako stvoriti izračunato polje u kontroli mreže podataka

1. Provjerite postoji li izvor podataka tipa OLE DB za bazu podataka Northwind na računalu; ako takav izvor podataka nije stvoren, slijedite korake u “Stvaranje Northwind OLE DB veze podataka”, ranije u ovom poglavlju.
2. Kreirajte kontrolu ADO podataka i kontrolu mreže podataka na formi.
3. Postavite svojstvo ConnectionString kontrole ADO podataka na izvor podataka Northwind.
4. Odredite svojstvo RecordSource kontrole ADO podataka.

   U prozoru s svojstvima, kliknite svojstvo RecordSource i upišite **Select ProductName, UnitPrice, (UnitPrice * .082) As Tax From Products**.
5. Postavite svojstvo DataSource kontrole mreže podataka na kontrolu ADO podataka.
6. Pokrenite projekt.

Korištenje kontrole mreže podataka s modulom klase

Ako podaci kojima želite pristupiti postoje u korisničkom obliku, ili u obliku koji nije izravno podržan ODBC goničem, za spremanje podataka možete stvoriti klasu. Nakon toga ćete programirati klasu s prilagođenim funkcijama za pronalaženje podataka. Klasa tad postaje izvor podataka kojeg može koristiti bilo koji potrošač podataka, kao što je kontrola mreže podataka.

U događaju Initialize modula klase, trebate najprije stvoriti objekt ADODB skupa slogova određivanjem varijable kao New ADODB.Recordset. Nakon stvaranja objekta skupa slogova, dodajte mu polja, po jedno za svako polje u vašem izvoru podataka. Zatim popunite skup slogova prikladnim podacima.

Napomena Izvor podataka također možete stvoriti korištenjem jednostavnog davatelja tipa OLEDB. Pogledajte "Stvaranje klasa svjesnih podataka" u 9. poglavlju za više informacija o jednostavnim OLEDB davateljima.

Modul klase predstavlja događaj GetDataMember koji se pojavljuje svaki put kad potrošač podataka (kao kontrola mreže podataka) zatraži podatke. U događaju, argument Data postavlja se na objekt skupa slogova stvoren u događaju Initialize.

Za korištenje modula klase, stvorite formu s kontrolom mreže podataka. U događaju forme Load, postavite programski kod koji određuje svojstvo DataSource kontrole mreže podataka na klasu.

Napomena Modul klase nije dostupan tijekom izrade aplikacije. Na primjer, kod kontrole mreže podataka, svi dostupni izvori podataka pojavljuju se u spuštajućoj popisu kad korisnik klikne svojstvo DataSource u prozoru s svojstvima. Modul klase neće se pojaviti među njima, i može biti određen samo kroz programski kod.

Stvaranje izvora podataka korištenjem modula klase

Primjer koji slijedi koristi modul klase za stvaranje jednostavnog izvora podataka. Kontrola mreže podataka se zatim povezuje s modulom kroz svojstvo DataSource.

Kako stvoriti klasu za korištenje s mrežom podataka

1. Stvorite novi **standardni EXE** projekt.
2. Dodajte kontrolu **mreže podataka** na formu.

   Ako kontrola mreže podataka nije dostupna u **alatnom okviru**, kliknite stavku **Components** u izborniku **Project**. Kliknite **Microsoft DataGrid Control**, pa odaberite **OK**.
3. U izborniku **Project** kliknite stavku **References**. U dijaloškom okviru References, kliknite **Microsoft ActiveX Data Objects 2.0 Library**.
4. U izborniku **Project** kliknite **Add Class Module** za dodavanje modula klase projektu.
5. U projektnom prozoru kliknite ikonu klase kako bi je odabrali, i pritisnite F4 za prikaz prozora s svojstvima.
6. U prozoru s svojstvima, promijenite ime klase u **NamesData**.
7. U prozoru s svojstvima, kliknite **DataSourceBehavior** i promijenite svojstvo u **vbDataSource**.
8. U odjeljku Declaration modula klase, stvorite varijablu ADODB tipa skupa slogova, na ovaj način:

   ```
   Option Explicit
   Private WithEvents rsNames As ADODB.Recordset
   ```

   Određivanje varijable korištenjem ključne riječi WithEvents dopušta vam programiranje događaja objekta skupa slogova.

9. U događaj klase Initialize dodajte sljedeći programski kod:

```
Private Sub Class_Initialize()
   ‘ Dodavanje imena novih članova podataka u zbirku DataMember.
   ‘ To omogućuje ostalim objektima da vide raspoložive članove.
   DataMembers.Add “Names”
   Set rsNames = New ADODB.Recordset    ‘ Odre|ivanje varijable objekta
   ‘ Stvaranje skupa slogova s dva polja i otvaranje skupa
   ‘ slogova. Prvi zapis je podatak tipa cijelog broja, a drugi je
   ‘ string, s najviše 256 karaktera. CursorType se postavlja na
   ‘ OpenStatic - pregled niza zapisa s ažuriranjem. LockType je
   ‘ postavljen LockOptimistic kako bi ažuriranja skupa slogova
   ‘ bila dopuštena.
   With rsNames
      .Fields.Append “ID”, adInteger
      .Fields.Append “Name”, adBSTR, 255
      .CursorType = adOpenStatic
      .LockType = adLockOptimistic
      .Open
   End With

   Dim i As Integer
   For i = 1 to 10              ‘ Dodavanje 10 zapisa.
      rsNames.AddNew
      rsNames!ID = i
      rsNames!Name = “Name “ & i
      rsNames.Update
   Next i
   rsNames.MoveFirst     ‘ Pomicanje na početak skupa slogova.
End Sub
```

Programski kod najprije stvara objekt skupa slogova, zatim dodaje dva polja u taj skup. Kod zatim dodaje deset zapisa u skup slogova.

10. U događaju klase GetDataMember, upišite sljedeći programski kod:

```
Private Sub Class_GetDataMembet(ByVal DataMember As String, _
Data As Object)
   Set Data = rsNames
End Sub
```

Programski kod vraća objekt skupa slogova svaki put kad se pojavi događaj – kad god je objekt klase povezan s potrošačem podataka, kao što je kontrola mreže podataka.

11. U kodnom modulu objekta forme, odredite varijablu objekta za klasu:

```
Option Explicit
Private datNames As NamesData  ‘ Varijabla klase.
```

12. U događaju Load forme, dodajte programski kod koji će postaviti svojstvo DataSource kontrole mreže podataka na objekt klase.

```
Private Sub Form_Load()
   ‘ Stvaranje novog objekta NamesData.
   Set datNames = New NamesData
   ‘ Povezivanje mreže podataka s novim izvorom podataka.
   Set Datagrid1.DataSource = datNames
End Sub
```

13. Pritisnite F5 za pokretanje projekta.

Programiranje događaja skupa slogova

Možete također programirati događaje skupa slogova, objekta Recordset. U modulu klase, kliknite okvir Object (u gornjem lijevom kutu), i odaberite **rsNames**. U okviru Procedures/Events (gornji desni kut), spuštajuća lista će prikazati sve događaje objekta skupa slogova.

Dodavanje svojstva klasi

Modul klase također može biti promijenjen tako da odgovara na događaje ili pozive funkcija. Sljedeći programski kod pokazuje kako najprijemožete dodati svojstvo klasi. Kad je pozvano od drugog objekta, svojstvo vraća broj zapisa u klasi.

```
Public Property Get RecordCount() As Long
  RecordCount = rsNames.RecordCount
End Sub
```

Korištenje svojstva DataMember

Događaj GetDataMember također uključuje argument DataMember. Korištenjem tog argumenta, možete uključiti više od jednog skupa slogova u modul klase, i vratiti odgovarajući skup slogova korištenjem naredbe Select Case s argumentom DataMember:

```
Private Sub Class_GetDataMember(ByVal DataMember As String, _
Data As Object)
  Select Case DataMember
  Case “Names”
    Set Data = rsNames
  Case “Dates”
    Set Data = rsDates
  Case Else
    ‘ Određivanje podrazumijevanog podatka.
    Set Data = rsYears
  End Select
End Sub
```

Kako bi odredili kojeg člana podataka želite, postavite svojstvo DataMember potrošača podataka na odgovarajući string, pa zatim odredite svojstvo DataSource na uobičajen način. Za kontrolu mreže podataka, to bi izgledalo ovako:

```
Private Sub Form_Load()
  ‘ Stvaranje novog objekta NamesData.
  Set datNames = New NamesData

  ‘ Određivanje željenog DataMember, pa određivanje DataSource.
  DataGrid1.DataMember = “Names”
  Set DataGrid1.DataSource = datNames
End Sub
```

Korištenje kontrola datotečnog sustava

Većina aplikacija mora predstaviti informacije o pogonskim uređajima, direktorijima i datotekama. Da biste omogućili korisnicima vaših aplikacija da istraže datotečni sustav, Visual Basic pruža dvije mogućnosti. Možete upotrijebiti standardne dijaloške okvire koje pruža kontrola općeg dijaloga, ili možete izgraditi korisničke dijaloge koristeći svoje vlastite kombinacije tri specijalizirane kontrole: okvira s popisom pogonskih uređaja, okvira s popisom direktorija i okvira s popisom datoteka.

Kontrole datotečnog sustava možete koristiti kako bi u vašim aplikacijama omogućili korisnicima istraživanje i odabir neke od raspoloživih datoteka. Razmislite o korištenju kontrole općeg dijaloga ako samo trebate standardni dijaloški okvir File Open ili Save.

Za više informacija Pogledajte “Korištenje kontrole općeg dijaloga” ranije u ovom poglavlju za više informacija.

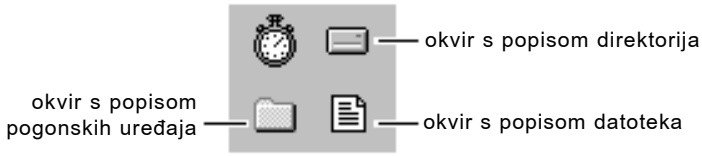
Primjer aplikacije: Winseek.vbp

Većina primjera programskog koda uzeta je iz primjera aplikacije Winseek (Winseek.vbp) koja se nalazi u direktoriju Samples.

Istraživanje datotečnog sustava

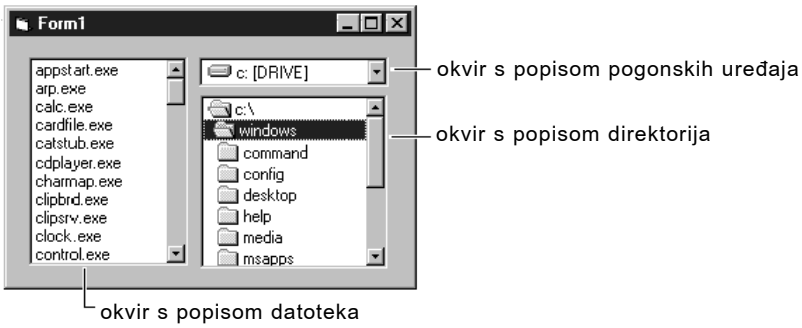
Svaka od kontrola datotečnog sustava pažljivo je oblikovana tako da spoji fleksibilne i usavršene mogućnosti nadzora datotečnog sustava s laganim programiranjem. Svaka kontrola automatski izvodi svoje posljedatke pronalaženja datotečnih podataka, ali možete napisati kod za prilagođavanje njihovog izgleda i određivanje informacija koje će prikazati.

Slika 7.16 Kontrole datotečnog sustava



Kontrole datotečnog sustava možete koristi pojedinačno ili u kombinaciji. Sa kombinacijama, možete napisati programski kod u potprogramima događaja svake kontrole kako bi odredili njihovo međusobno djelovanje. Možete ih i pustiti da rade neovisno jedna o drugoj. Slika 7.17 prikazuje zajedničko korištenje sve tri kontrole.

Slika 7.17 Zajedničko korištenje kontrola datotečnog sustava



Kontrole datotečnog sustava automatski dobivaju sve informacije od operativnog sustava; možete pristupiti tim informacijama ili odrediti što će biti prikazano u svakoj kontroli kroz njezina svojstva. Na primjer, sadržaj trenutnog radnog direktorija se prikazuje standardno (to je direktorij iz kojeg je pokrenuta aplikacija, ili koji je postao trenutan direktorij kao rezultat naredbe ChDir).

Vaša aplikacija može također prikazati popis datoteka čija imena odgovaraju predlošku, kao što je *.frm. Jednostavno stvorite okvir s popisom datoteka na formi i odredite njegovo svojstvo Pattern na *.frm. Svojstvo Pattern možete odrediti tijekom rada aplikacije sljedećim programskim kodom:

```
File1.Pattern = "*.FRM"
```

Kontrole datotečnog sustava daju vam fleksibilnost koja nije dostupna s kontrolom općeg dijaloga. Ove kontrole možete miješati i prilagođavati ih na puno različitih načina, gdje vi nadzirete njihov izgled i način kako međusobno djeluju.

Ako je vaš cilj samo omogućiti korisniku da otvara i sprema datoteke, kontrola općeg dijaloga pruža niz gotovih dijaloških okvira za te i slične operacije. Oni su isti kao i dijaloški okviri koje koristi puno drugih aplikacija temeljenih na Microsoft Windowsima, tako da pružaju standardizirani izgled i osjećaj. Oni također prepoznaju mrežne pogone kad nisu na raspolaganju.

Za više informacija Pogledajte "Korištenje kontrole općeg dijaloga" ranije u ovom poglavlju za više informacija.

Okvir s popisom pogonskih uređaja

Okvir s popisom pogonskih uređaja (Drive List Box) je okvir s spuštajućom listom. U pravilu, prikazan je aktivni pogon na sistemu korisnika. Kad ova kontrola ima fokus, korisnik može upisati bilo koju valjanu oznaku pogona ili kliknuti strelicu na desnoj strani okvira. Kad korisnik klikne strelicu, spušta se okvir s listom svih valjanih pogona. Ako korisnik odabere novi pogon s popisa, taj pogon će se pojaviti na vrhu okvira s popisom.

Možete upotrijebiti programski kod za utvrđivanje svojstva Drive okvira s popisom pogonskih uređaja kako bi odredili koji je pogon trenutno odabran. Vaša aplikacija može također odrediti koji će se pogon pojaviti na vrhu okvira s popisom sljedećim jednostavnim izrazom:

```
Drive1.Drive = “c:\”
```

Okvir s popisom pogonskih uređaja prikazuje valjane dostupne pogone. Odabir pogona s popisa ne mijenja automatski trenutni radni pogonski uređaj; međutim, možete upotrijebiti svojstvo Drive za promjenu pogona na razini operativnog sustava ako ga odredite kao argument za naredbu ChDrive:

```
ChDrive Drive1.Drive
```

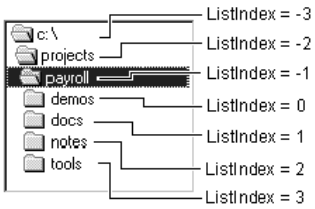
Okvir s popisom direktorija

Okvir s popisom direktorija (Directory List Box) prikazuje ustroj direktorija na aktivnom pogonu korisnikovog sustava, počinjući od direktorija najviše razine. U početku, ime aktivnog direktorija pojavljuje se istaknuto i uvučeno od direktorija iznad u hijerarhiji, prema korijenskom direktoriju. Poddirektoriji su uvučeni ispod aktivnog direktorija u okviru s popisom direktorija. Kako se korisnik miče gore ili dolje po popisu, svaka odabrana stavka bit će označena.

Označavanje pojedinih direktorija

Svaki direktorij u okviru ima pridruženu cjelobrojnu oznaku koja vam omogućuje prepoznavanje pojedinih direktorija. Ova sposobnost nije sadržana u kontroli općeg dijaloga. Direktorij određen svojstvom Path (Dir1.Path) uvijek ima vrijednost svojstva ListIndex od -1. Direktorij odmah iznad njega ima vrijednost svojstva ListIndex od -2, još jedan iznad -3 i tako dalje sve do korijenskog direktorija. Prvi poddirektorij odabranog direktorija Dir1.Path ima vrijednost svojstva ListIndex od 0. Ako postoji više direktorija na prvoj razini poddirektorija, idući ima vrijednost svojstva ListIndex od 1, zatim 2 i tako dalje, kako je prikazano na slici 7.18.

Slika 7.18 Ustroj direktorija prikazan u okviru s popisom direktorija



Određivanje aktivnog direktorija

Upotrijebite svojstvo Path okvira s popisom direktorija za određivanje ili vraćanje aktivnog direktorija u okviru (ListIndex = -1). Na primjer, ako svojstvu `Drive1.Path` dodijelite "c:\payroll" na slici 7.18, direktorij \Payroll postaje odabran kao trenutan radni direktorij.

Slično tome, svojstvo Drive okvira s popisom pogona može biti dodijeljeno svojstvu Path okvira s popisom direktorija:

```
Dir1.Path = Drive1.Drive
```

Kad se izvrši ovo dodjeljivanje, okvir s popisom direktorija prikazuje sve raspoložive direktorije i poddirektorije na tom pogonu. U pravilu, okvir s popisom direktorija također prikazuje sve direktorije iznad, i sve poddirektorije odmah ispod, aktivnog direktorija na pogonu dodijeljenog svojstvu Dir1.Path. Okvir s popisom direktorija ne *postavlja* aktivni direktorij na razini operativnog sustava; on samo označava direktorij i daju mu vrijednost svojstva ListIndex od -1.

Kako bi postavili aktivni radni direktorij, upotrijebite naredbu ChDir. Na primjer, sljedeći izraz mijenja aktivni direktorij na onaj prikazan u okviru s popisom direktorija:

```
ChDir Dir1.Path
```

U aplikaciji koja koristi kontrole datotečnog sustava, možete postaviti aktivni direktorij na direktorij u kojem se nalazi izvršna (.exe) datoteka aplikacije objektom Application:

```
ChDrive App.Path   ' Postavljanje pogona.
ChDir App.Path     ' Postavljanje direktorija.
```

Napomena Svojstvo Path dostupno je samo tijekom izvođenja aplikacije, a ne tijekom izrade aplikacije.

Za više informacija Pogledajte "Objekt App" u biblioteci *Microsoft Visual Basic 6.0 Language Reference* za više informacija o objektu Application.

Klik na stavku direktorija

Kad korisnik klikne na stavku u okviru s popisom direktorija, ta stavka će biti osvijetljena. Kad se dvaput klikne na stavku, ona se dodjeljuje svojstvu Path, njezino svojstvo

ListIndex dobiva vrijednost -1, a okvir s popisom direktorija se ponovno iscrtava kako bi prikazao neposredne poddirektorije.

Pronalaženje relativnog položaja direktorija

Svojstvo ListCount vraća broj direktorija ispod trenutno proširenog direktorija, a ne ukupan broj stavki u okviru s popisom direktorija. Budući da je vrijednost svojstva ListIndex trenutno proširenog direktorija uvijek -1, možete napisati programski kod za utvrđivanje koliko je daleko ispod korijena trenutno prošireni direktorij u hijerarhiji. Na primjer:

```
‘ Početak trenutno proširenog direktorija.

IdiGore = 0
‘ Dir1.List(x) vra}a prazni string ako
‘ direktorij ne postoji.
Do Until Dir1.List(IdiGore) = “”
  IdiGore = IdiGore - 1
Loop
‘ Pretvaranje u pozitivni broj, ako je potrebno.
RazinaGore = Abs(IdiGore)
```

Okvir s popisom datoteka

Okvir s popisom datoteka (File List Box) prikazuje datoteke sadržane u direktoriju određenom svojstvom Path tijekom rada aplikacije. Možete prikazati sve datoteke u aktivnom direktoriju aktivnog pogona korištenjem sljedećeg izraza:

```
File1.Path = Dir1.Path
```

Nakon toga možete prikazati podskup tih datoteka određivanjem svojstva Pattern – na primjer, *.frm prikazuje samo datoteke s tim sufiksom. Svojstvo Pattern može također prihvatiti popis razdvojenu točkom-zarezom (;). Na primjer, sljedeća linija programskog koda prikazuje sve datoteke s sufiksima .frm i .bas:

```
File1.Pattern = “*.frm; *.bas”
```

Visual Basic podržava zamjenski znak upitnika (?). Na primjer, ???.txt prikazuje datoteke kojima se ime sastoji od tri karaktera i ima sufiks .txt.

Rad s atributima datoteka

Atributi trenutno odabrane datoteke (Archive, Normal, System, Hidden i ReadOnly) su također dostupni kroz svojstva okvira s popisom datoteka. Ova svojstva možete upotrijebiti za određivanje vrste datoteka koje će biti prikazane u okviru s popisom datoteka. Standardna vrijednost za atribute System i Hidden je False. Standardna vrijednost za atribute Normal, Archive i ReadOnly je True.

Za prikaz datoteka koje se mogu samo čitati u okviru s popisom datoteka, na primjer, jednostavno postavite svojstvo ReadOnly na True, a ostale atribute na False.

```
File1.ReadOnly = True
File1.Archive = False
File1.Normal = False
File1.System = False
File1.Hidden = False
```

Kad je argument Normal postavljen na True, bit će prikazane datoteke bez atributa System ili Hidden. Kad je argument Normal postavljen na False, i dalje možete prikazati datoteke s atributima ReadOnly i/ili Archive postavljanjem tih atributa na True.

Napomena Ne možete upotrijebiti svojstva atributa za promjenu atributa datoteke. Za određivanje atributa datoteke, upotrijebite naredbu SetAttr.

U pravilu, možete odabrati samo jednu datoteku u okviru s popisom datoteka. Za odabir više datoteka istovremeno, upotrijebite svojstvo MultiSelect.

Za više informacija Za više informacija o naredbi SetAttr, pogledajte odlomak "Naredba SetAttr", u stalnoj pomoći. Pogledajte i odlomak "Svojstvo MultiSelect" u biblioteci *Microsoft Visual Basic 6.0 Language Reference*.

Zajednička upotreba kontrola datotečnog sustava

Ako koristite kombinaciju kontrola datotečnog sustava, možete usklađivati informacije koje te kontrole prikazuju. Na primjer, ako imate okvir s popisom pogonskih uređaja, okvir s popisom direktorija i okvir s popisom datoteka s predodređenim imenima Drive1, Dir1 i File1, niz događaja mogao bi raditi na sljedeći način:

1. Korisnik odabire pogon u okviru s popisom pogonskih uređaja Drive1.
2. Prouzročen je događaj Drive1_Change, i ažurira se prikaz u okviru Drive1 kako bi pokazao novi pogonski uređaj.
3. Programski kod u potprogramu događaja Drive1_Change dodjeljuje novi odabir (svojstvo Drive1.Drive) svojstvu Path okviru s popisom direktorija Dir1 sljedećim izrazima:

   ```
   Private Sub Drive1_Change()
     Dir1.Path = Drive1.Drive
   End Sub
   ```

4. Dodjela vrijednosti svojstvu Path uzrokuje događaj Dir1_Change i ažurira prikaz u okviru Dir1 kako bi se prikazao trenutan direktorij novog pogonskog uređaja.
5. Programski kod u potprogramu događaja Dir1_Change dodjeljuje novu stazu (svojstvo Dir1.Path) svojstvu File1.Path okvira s popisom datoteka File1:

   ```
   Private Sub Dir1_Change()
     File1.Path = Dir1.Path
   End Sub
   ```

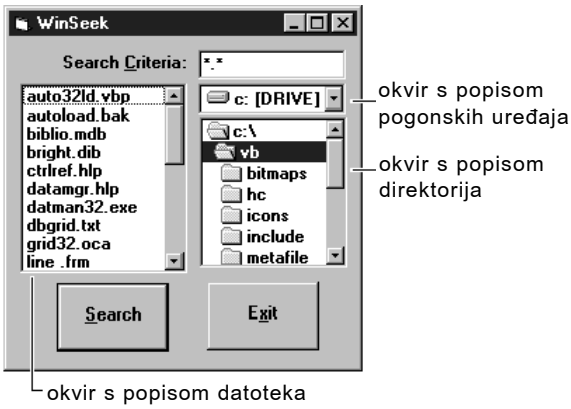
6. Dodjela svojstvu File1.Path uzrokuje promjenu u okviru File1 čiji prikaz odražava odrednice staze okvira Dir1.

Potprogrami događaja koje koristite i svojstva koja mijenjate ovise o tome na koji način vaša aplikacija koristi kombinaciju kontrola datotečnog sustava. Programski kod u odlomku "Primjer kontrola datotečnog sustava: aplikacija za traganje datoteka" pokazuje usklađivanje ovdje opisanih kontrola.

Primjer kontrola datotečnog sustava: Aplikacija za traženje datoteka

Budući da korisnici često žele brzo pronaći datoteku ili grupu datoteka dostupnu aplikaciji, puno aplikacija sadrži mogućnost istraživanja datotečnog sustava. Primjer aplikacije Winseek.vbp pomaže korisniku u pretraživanju pogonskih uređaja i direktorija, i prikazuje bilo koju kategoriju datoteka.

Slika 7.19 Kontrole datotečnog sustava u aplikaciji Winseek



Sljedeća tablica rezimira kontrole na formi Seek.frm aplikacije Winseek.

kontrola	svojstvo	postavka
Okvir s popisom pogona	Name	drvList
Okvir s popisom direktorija	Name	dirList
Okvir s popisom datoteka	Name Pattern	filList *.*
Prvi naredbeni gumb	Name Caption Default	cmdSearch &Search True
Drugi naredbeni gumb	Name Caption	cmdExit E&xit
Okvir s popisom	Name	lstFoundFiles

Napomena Kontrole datotečnog sustava nemaju svojstva natpisa, iako ih možete označiti i dati im pristupne tipke. Za više informacija o korištenju natpisa na ovaj način, pogledajte "Korištenje kontrole natpisa" kasnije u ovom poglavlju.

Pisanje koda za aplikaciju Winseek

U okviru s popisom pogonskih uređaja, događaj Change se pokreće jednim klikom miša na stavku. Događaj Change se također pojavljuje kad korisnik odabere stavku i zatim promijeni fokus na formu. U okviru s popisom direktorija, događaj DblClick je neophodan za stvaranje događaja Change.

Kad korisnici žele promijeniti direktorij bez korištenja miša, obično koriste kursorske tipke za odabir željenog direktorija i zatim pritisnu tipku ENTER.

Budući da je ENTER obično povezan s kontrolom predodređenog naredbenog gumba, aplikacija Winseek mora razlikovati želi li korisnik samo promijeniti direktorij ili početi potragu za datotekama.

Aplikacija Winseek razlučuje tu dvosmislenost utvrđivanjem razlikuje li se staza u okviru dirList od trenutno odabranog direktorija. Takva situacija može se dogoditi kad korisnik jednostrukim klikom odabere stavku u okviru s popisom direktorija ili se kreće u tom okviru korištenjem kursorskih tipki. Sljedeći programski kod određuje je li svojstvo dirList.Path različito od staze odabranog direktorija. Ako su staze različite, ažurira se svojstvo dirList.Path. Ako su staze iste, pokreće se traženje.

```
Private Sub cmdSearch_Click()
  .
  .
  .
‘ Ako je dirList.Path različit od trenutno odabranog
‘ direktorija, ažurira ga; inače izvodi traganje.
If dirList.Path <> dirList.List(dirList.ListIndex) Then
  dirList.Path = dirList.List(dirList.ListIndex)
  Exit Sub
End If
‘ Nastavak s traženjem.
  .
  .
  .
End Sub
```

Aplikacija Winseek koristi sljedeće potprograme za obradu značajnih događaja:

- Potprogram drvList_Change
- Potprogram dirList_Change
- Potprogram cmdSearch_Click

Događaj Change okvira s popisom pogonskih uređaja

Kad korisnik klikne na stavku u okviru s popisom pogonskih uređaja, stvara se događaj Change. Poziva se potprogram događaja drvList_Change, i izvodi sljedeći kôd:

```
Private Sub drvList_Change()
  On Error GoTo DriveHandler
  ‘ Ako je odabran novi pogon, okvir Dir1
  ‘ ažurira svoj prikaz.
  dirList.Path = drvList.Drive
  Exit Sub
‘ Ako postoji pogreška, obnavlja drvList.Drive s
‘ pogonom iz dirList.Path
DriveHandler:
  drvList.Drive = dirList.Path
  Exit Sub
End Sub
```

Uočite da se događaj Change u okviru s popisom pogonskih uređaja pojavljuje kad je odabran novi pogonski uređaj, ili jednostrukim klikom miša ili kad korisnik pomakne odabir (na primjer, kursorskom tipkom). Rukovatelj pogrešom pokreće se akcijama kao što su pokušaj pristupa disketnom pogonu u kojem nema diskete ili odabir mrežnog pogonskog uređaja koji je nenamjerno isključen. Budući da pogreška sprječava originalno dodjeljivanje, svojstvo dirList.Path i dalje sadržava valjani pogonski uređaj. Dodjeljivanje svojstva dirList.Path svojstvu drvList.Drive ispravlja pogrešku.

Za više informacija Pogledajte 13. poglavlje “Traženje i obrada pogrešaka”, za više informacija.

Događaj Change okvira s popisom direktorija

Ako korisnik dvaput klikne na stavku u okviru s popisom direktorija, ili ako je svojstvo Path okvira dirList promijenjeno programskim kodom (kao u potprogramu drvList_Change), pokreće se događaj dirList_Change. Sljedeći programski kod odgovara na taj događaj:

```
Private Sub dirList_Change()
  ‘ Ažuriranje okvira s popisom datoteka za usklađivanje
  ‘ s okvirom s popisom direktorija.
  filList.Path = dirList.Path
End Sub
```

Ovaj potprogram događaja dodjeljuje svojstvo Path okvira dirList svojstvu Path okvira filList. To uzrokuje događaj PathChange u okviru filList, koji se ponovno iscrtava; ne trebate dodavati programski kod u potprogram filList_PathChange, jer se u ovoj aplikaciji lanac događaja završava u okviru filList.

Događaj Click naredbenog gumba

Ovaj potprogram događaja određuje je li odabrana stavka u okviru dirList ista kao i vrijednost svojstva dirList.Path. Ako su stavke različite, ažurira se svojstvo dirList.Path. Ako su stavke iste, izvodi se traženje.

```
Private Sub cmdSearch_Click()
   .
   .
   .
   ‘ Ako je dirList.Path različit od trenutno odabranog
   ‘ direktorija, ažurira ga; inače izvodi traganje.
If dirList.Path <> dirList.List(dirList.ListIndex) Then
   dirList.Path = dirList.List(dirList.ListIndex)
   Exit Sub
End If
‘ Nastavak s traženjem.
   .
   .
   .
End Sub
```

Napomena Aplikaciju Winseek možete poboljšati dodatnim osobinama. Na primjer, možete upotrijebiti svojstva atributa za datoteke. Možete upotrijebiti kontrolne kućice kako bi omogućili korisniku postavljanje različitih kombinacija atributa datoteka tako da okvir s popisom datoteka prikazuje datoteke koje su skrivene, sistemske i tako dalje. To će smanjiti traženje na datoteke koje odgovaraju tim uvjetima.

Korištenje kontrole okvira

Kontrole okvira (Frame) koriste se za omogućavanje prepoznatljivog grupiranja drugih kontrola. Na primjer, kontrole okvira možete upotrijebiti za funkcionalno dijeljenje forme – za odvajanje grupa kontrola gumbâ izbora.

Slika 7.20 Kontrola okvira

U većini slučajeva, kontrolu okvira koristit ćete pasivno – za grupiranje drugih kontrola – i neće biti potrebe za tim da kontrola odgovara na njezine događaje. Međutim, vrlo vjerojatno ćete mijenjati njezina svojstva Name, Caption i Font.

Za više informacija Pogledajte "Grupiranje izbora gumbima izbora" u 3. poglavlju "Forme, kontrole i izbornici", za jednostavnu demonstraciju korištenja kontrole okvira za grupiranje gumbâ izbora.

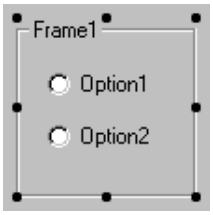
Dodavanje kontrole okvira na formu

Kad koristite kontrolu okvira za grupiranje drugih kontrola, najprije kreirajte kontrolu okvira, pa tek onda kreirajte kontrole unutar nje. Taj način omogućuje vam zajedničko pomicanje okvira i kontrola koje on sadrži.

Stvaranje kontrola unutar okvira

Kako bi okviru dodali druge kontrole, stvorite ih unutar okvira. Ako stvorite kontrolu izvan okvira, ili upotrijebite postupak dvostrukog klika na ikonu za dodavanje kontrole na formu, pa je zatim pokušate pomaknuti unutar kontrole okvira, kontrola će se nalaziti iznad okvira i morat ćete posebno micati okvir i kontrole.

Slika 7.21 Kontrole unutar okvira

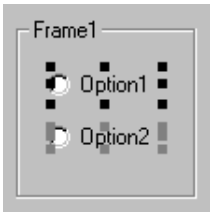


Napomena Ako imate postojeće kontrole koje želite grupirati unutar okvira, možete odabrati sve kontrole, izrezati ih u odlagalište, odabrati kontrolu okvira i ulijepiti izrezane kontrole u kontrolu okvira.

Odabir više kontrola u okviru

Kako bi unutar okvira odabrali više kontrola istovremeno, držite pritisnutu tipku CTRL i upotrijebite miša za uokviravanje kontrola. Kad otpustite tipku miša, kontrole unutar okvira bit će odabrane, kao na slici 7.22.

Slika 7.22 Odabir kontrola unutar okvira



Korištenje kontrole hijerarhijske fleksibilne mreže

Kontrola hijerarhijske fleksibilne mreže (Microsoft Hierarchical FlexGrid, MSHFlexGrid) i kontrola fleksibilne mreže (Microsoft FlexGrid, MSFlexGrid) predstavljaju podatke skupa slogova, iz jedne ili više tablica, u obliku mreže.

Kontrola hijerarhijske fleksibilne mreže pruža vam napredne osobine prikazivanja podataka u mreži. Slična je kontroli mreže povezanih podataka (Microsoft Data Bound grid, DataGrid), ali s izrazitom razlikom da kontrola hijerarhijske fleksibilne mreže ne dopušta korisniku promjenu podataka s kojima je povezana ili koje sadrži. Zbog toga vam ova kontrola omogućuje prikazivanje podataka korisniku uz istovremeno osiguravanje da će originalni podaci ostati sigurni i nepromijenjeni. Međutim, također je moguće dodati osobine editiranja ćelija vašoj kontroli hijerarhijske fleksibilne mreže kombiniranjem s okvirom s tekstom.

Iako je kontrola hijerarhijske fleksibilne mreže temeljena na kontroli fleksibilne mreže korištenoj u Visual Basicu 5.0, kontrola hijerarhijske fleksibilne mreže je fleksibilnija, uspoređujući ih međusobno. Kontrola hijerarhijske fleksibilne mreže također pruža više mogućnosti prikazivanja s kojima možete odrediti korisnički oblik koji će najbolje odgovarati vašim potrebama.

Ove teme se uglavnom koncentriraju na korištenje hijerarhijske fleksibilne mreže. Za informacije o bivšoj kontroli fleksibilne mreže, pogledajte dokumentaciju Visual Basica 5.0.

Kontrole fleksibilne mreže Visual Basica

ikona	kratica	ime kontrole
	kontrola MSHFlexGrid	kontrola Microsoft Hierarchical FlexGrid
	kontrola MSFlexGrid	kontrola Microsoft FlexGrid

Kontrola hijerarhijske fleksibilne mreže podržava sljedeće osobine:

- Povezivanje s podacima samo za čitanje.
- Dinamičko preraspoređivanje stupaca i redova.
- Automatsko regrupiranje podataka tijekom podešavanja stupaca.
- Prilagođivanje postojećem kodu Visual Basica za mrežu podataka (DataGrid).
- Svaka ćelija može sadržavati tekst, sliku ili oboje.
- Promjena trenutnog teksta ćelije programskim kodom ili tijekom izvođenja aplikacije.
- Automatsko čitanje podataka kad je hijerarhijska fleksibilna mreža dodijeljena kontroli podataka.
- Prijelom riječi u tekstu unutar ćelija.

- Povezivanje s ActiveX podacima kad su svojstva kontrole **DataSource** i **DataMember** povezana s određenim davateljem podataka. *
- Povezivanje uz pomoć upravitelja Data Binding Manager u Visual Basicu. *
- Direktno povezivanje s grupiranim i povezanim skupovima slogova tipa ADO iz hijerarhije Command. *
- Dodatne mogućnosti prikaza kad je hijerarhijska fleksibilna mreža povezana s hijerarhijom skupova slogova. Te dodatne mogućnosti dopuštaju različit prikaz za grupirane i povezane skupove slogova, uključujući trake. *

  Napomena Za iskorištavanje osobina označenih asteriskom (*) morate koristiti kontrolu hijerarhijske fleksibilne mreže. Te osobine nisu dostupne u kontroli fleksibilne mreže.

Zbog ograničenja prethodne kontrole fleksibilne mreže, neke osobine, kao što su trake, nisu dostupne pri korištenju te kontrole. Za pristup svim osobinama, preporučujemo korištenje kontrole hijerarhijske fleksibilne mreže kad stvarate novu kontrolu mreže podataka. Kontrola fleksibilne mreže ne ažurira se automatski na kontrolu hijerarhijske fleksibilne mreže.

Pristup kontroli hijerarhijske fleksibilne mreže

Upotrijebite sljedeći postupak za ugradnju i pristup kontroli hijerarhijske fleksibilne mreže u Visual Basicu.

Kako ugraditi i pristupiti kontroli hijerarhijske fleksibilne mreže

1. U izborniku **Project** odaberite **Components**. Pojavit će se dijaloški okvir **Components**.
2. Na kartici **Controls**, odaberite **Microsoft Hierarchical FlexGrid Control 6.0**, i zatim kliknite **OK**. Kontrola **MSHFlexGrid** će biti dodana u alatni okvir Visual Basica.
3. U alatnom okviru Visual Basica, kliknite kontrolu **MSHFlexGrid**, i kreirajte je na formi Visual Basica.

   - ili -

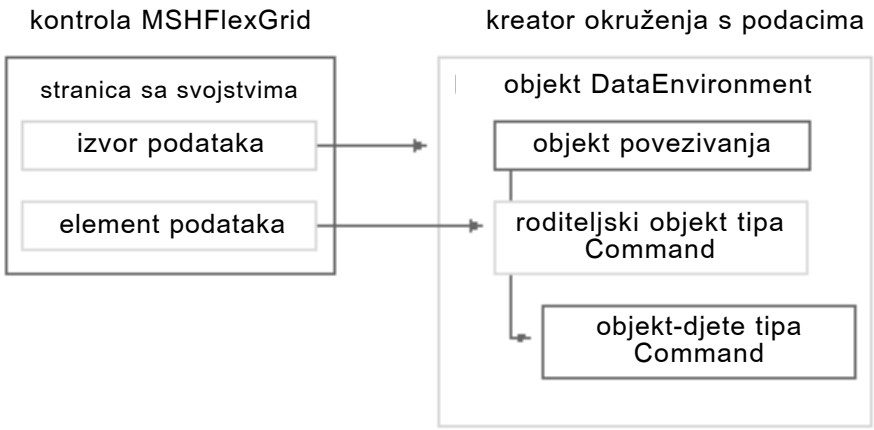
   U alatnom okviru Visual Basica, dvokliknite kontrolu **MSHFlexGrid** za dodavanje na formu.

Povezivanje podataka s hijerarhijskom fleksibilnom mrežom

Prije nego što iskoristite njene osobine, morate povezati podatke s hijerarhijskom fleksibilnom mrežom. Podatke možete povezati s kontrolom ili korištenjem upravitelja Data Binding Manager ili programski.

Jednom kad je vaša hijerarhijska fleksibilna mreža povezana s izvorom podataka, prikaz tijekom izrade unutar hijerarhijske fleksibilne mreže sastoji se od jednog praznog stupca i jednog praznog reda. Informacije polja i trake se ne uspostavljaju automatski (za dobivanje tih informacija pogledajte "Uspostavljanje strukture", kasnije u ovom poglavlju). Ako se hijerarhijska fleksibilna mreža pokrene bez informacija o polju i traci, podaci se prikazuju korištenjem podrazumijevanih postavki svojstava. To znači da se, ako je hijerarhijska fleksibilna mreža povezana s hijerarhijom Command, trake podataka prikazuju vodoravno, a svaka traka sadrži stupac za svako polje u skupu slogova.

Hijerarhijska fleksibilna mreža povezana s izvorom podataka



Korištenje upravitelja za povezivanje podataka s mrežom

Ovaj dio opisuje kako upotrijebiti Data Binding Manager upravitelja iz Visual Basica za povezivanje podataka s vašom hijerarhijskom fleksibilnom mrežom. Taj upravitelj pruža korisničko sučelje jednostavno za korištenje kod povezivanja podataka.

Kako odrediti izvor podataka korištenjem upravitelja

1. Stvorite izvor podataka za svoju hijerarhijsku fleksibilnu mrežu.

   Izvor podataka može biti objekt DataEnvironment, kontrola ADO podataka (ActiveX Data Object) ili nova osobina Visual Basica. Za ovaj postupak stvorite izvor podataka kao objekt DataEnvironment.

2. U alatnom okviru Visual Basica, kliknite kontrolu **MSHFlexGrid**, i kreirajte je na formi Visual Basica.

   - ili -

   U alatnom okviru Visual Basica dvokliknite kontrolu **MSHFlexGrid** za dodavanje na formu.

3. U prozoru s svojstvima Visual Basica postavite svojstvo **DataSource** na objekt DataEnvironment koji sadrži objekt Command koji želite povezati s hijerarhijskom fleksibilnom mrežom.

   Oprez Ako se svojstvo **DataSource** ponovno odredi, svi korisnički promijenjeni podaci u ćelijama hijerarhijske fleksibilne mreže bit će izgubljeni.

4. U prozoru s svojstvima Visual Basica, postavite svojstvo **DataMember** na objekt Command sadržan u objektu DataEnvironment. Ako u vašoj hijerarhijskoj fleksibilnoj mreži želite vidjeti hijerarhijske podatke, morate odrediti najviši, roditeljski objekt Command u Command hijerarhiji kao **DataMember**.

5. Kako bi vidjeli podatke u hijerarhijskoj fleksibilnoj mreži, u izborniku **Run**, kliknite **Start**.

   - ili -

   Pritisnite F5.

Programsko povezivanje podataka s hijerarhijskom fleksibilnom mrežom

Ovaj dio opisuje kako programski povezati podatke s hijerarhijskom fleksibilnom mrežom.

Kako programski odrediti izvor podataka

1. U alatnom okviru Visual Basica, dvokliknite kontrolu **MSHFlexGrid** kako bi je postavili na formu Visual Basica.

2. Kliknite desnom tipkom miša na svoju hijerarhijsku fleksibilnu mrežu i iz pomoćnog izbornika odaberite stavku **View Code**. Otvorit će se kodni prozor.

3. U događaju Form_Load, dodajte programski kod za stvaranje skupa slogova tipa ADO i dodijelite ga hijerarhijskoj fleksibilnoj mreži. Taj kod ispisan je u idućim točkama.

   Napomena Kako bi programski odredili izvor podataka, projekt mora sadržavati pokazivač na Microsoft ActiveX objekte podataka: u izborniku **Project**, odaberite **Select References**, i zatim odaberite **Microsoft ActiveX Data Objects 2.0 Library**.

   - Stvorite vezu i skup slogova tipa ADO ubacivanjem sljedećeg koda, te zamijenite tumačenja prikladnim izrazima (na primjer, zamijenite <mojIzvorPodataka> stvarnim imenom vašeg izvora podataka):

```
Dim Cn As New Connection, Rs As New Recordset
‘ Trebate zamijeniti <mojIzvorPodataka> valjanim
‘ imenom izvora podataka na vašem sustavu.
Cn.ConnectionString = “DSN=<mojIzvorPodataka>”
‘ Upotrijebite sljedeći kod za SHAPE naredbe.
Cn.Provider = “MSDataShape”
Cn.CursorLocation = adUseNone
```

```
‘ Drugi način, za SQL naredbe, je sljedeći kod.
Cn.CursorLocation = adUseNone

Cn.Open
‘ Za sljedeće povezivanje trebate odrediti
‘ valjani izvor podataka za vaš skup slogova.
Rs.Source = <valjana SQL naredba SELECT>

‘ Slijedi spajanje objekta Command i objekta Connection
‘ te njihovo izvršavanje.
Set Rs.ActiveConnection = Cn
Rs.Open
```

- Dodijelite otvoreni skup slogova u varijabli Rs hijerarhijskoj fleksibilnoj mreži ubacivanjem sljedećeg programskog koda:

  ```
  Set MSHFlexGrid1.DataSource = Rs
  ```

4. Kako bi vidjeli podatke u hijerarhijskoj fleksibilnoj mreži, u izborniku **Run** kliknite **Start**.

   - ili -

   Pritisnite F5.

Korištenje hijerarhijske fleksibilne mreže s hijerarhijskim skupom slogova

Hijerarhijsku fleksibilnu mrežu možete upotrijebiti zajedno s hijerarhijskim skupom slogova za pregled srodnih informacija. Te informacije možete prikazivati korisniku uz istovremenu sigurnost da su originalni podaci ostali sigurni i nepromijenjeni, ili možete dodati osobine editiranja ćelija vašoj hijerarhijskoj fleksibilnoj mreži dodavanjem okvira s tekstom na formu. Kad je hijerarhijska fleksibilna mreža povezana s hijerarhijom skupa slogova, možete prikazati grupirane i srodne skupove slogova korištenjem traka.

Napomena Prije izvođenja sljedećeg postupka, podaci moraju biti povezani s vašom hijerarhijskom fleksibilnom mrežom. Kako bi to učinili, pogledajte “Povezivanje podataka s hijerarhijskom fleksibilnom mrežom”.

Kako koristiti trake kod prikazivanja hijerarhijskih skupova slogova u hijerarhijskoj fleksibilnoj mreži

1. Kliknite desnom tipkom miša na hijerarhijsku fleksibilnu mrežu, i odaberite **Properties** u pomoćnom izborniku. Pojavit će se dijaloški okvir **Property Pages** hijerarhijske fleksibilne mreže.
2. Na kartici **General**, odredite **BandDisplay**. Pogledajte odlomak “Oblikovanje trake” kasnije u poglavlju za opis prikaza svake trake.

3. Na kartici **Bands**, odaberite traku u okviru **Band**. Taj popis raspoloživih traka je temeljen na skupovima slogova u hijerarhiji Command. Kod svih traka, ime objekta Command koji uzrokuje skup slogova ispisano je u zagradama.
4. Prilagodite svojstva svake trake prema potrebama. Pogledajte “Oblikovanje trake” za više informacija.
5. Kliknite **OK** za prihvaćanje svojstava trake u vašoj hijerarhijskoj fleksibilnoj mreži i zatvaranje dijaloškog okvira **Property Pages**.

Oblikovanje trake

U vašoj hijerarhijskoj fleksibilnoj mreži možete upotrijebiti oblikovanje trake kako bi odredili mogućnosti prikaza u određenom skupu slogova. Traka se stvara za svaki skup slogova u hijerarhijskom skupu slogova tipa ADO. Na primjer, kad povezujete hijerarhijsku fleksibilnu mrežu s hijerarhijskim skupom slogova tipa ADO koji sadrži kupce i narudžbe, hijerarhijska fleksibilna mreža u početku sadrži dvije trake.

Izgled hijerarhijske fleksibilne mreže možete prilagoditi oblikovanjem trake. To vam omogućuje označavanje važnih informacija unutar više traka. Elementi trake koje možete oblikovati uključuju zaglavlja stupaca, linije mreže, boje i uvlačenja.

Kod povezivanja s nehijerarhijskim skupom slogova, postoji samo jedna traka, i ta traka se označava kao traka 0 (band 0). Druge trake nisu dostupne, jer su trake temeljene na hijerarhiji skupova slogova ili objekata Command.

Promjena izgleda trake

Možete promijeniti način na koji se polja prikazuju u traci promjenom izgleda trake. Standardno, polja se prikazuju vodoravno unutar traka, kao u standardnoj mreži.

Vodoravan prikaz trake

Okomit prikaz za posljedicu može imati širenje visine trake kako bi traka prihvatila sva polja. Sve ostale trake koje se prikazuju u hijerarhijskoj fleksibilnoj mreži također se proširuju, osiguravajući jednaku visinu svih traka.

Okomit prikaz trake

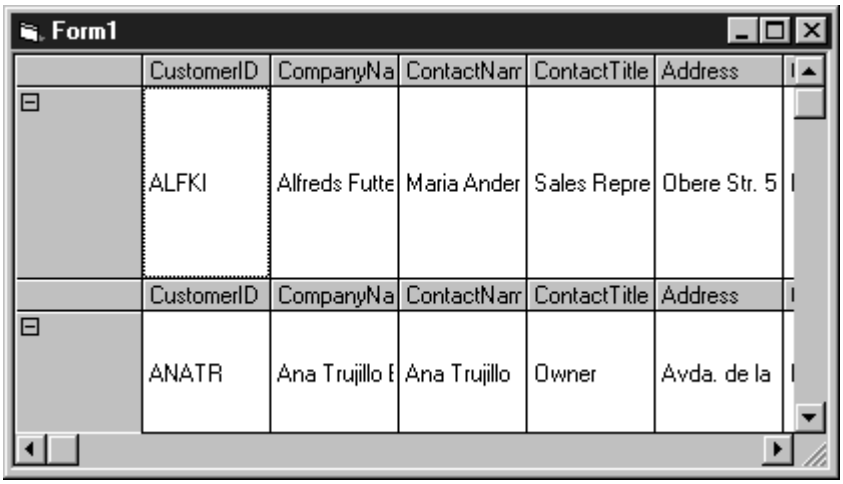
Kako odrediti prikaz trake

1. Kliknite desnom tipkom miša na hijerarhijsku fleksibilnu mrežu, i odaberite **Properties** u pomoćnom izborniku. Pojavit će se dijaloški okvir **Property Pages** hijerarhijske fleksibilne mreže.
2. Na kartici **General** odaberite **BandDisplay**.
3. Kliknite **OK** za prihvaćanje svojstava prikaza traka u vašoj hijerarhijskoj fleksibilnoj mreži i zatvorite dijaloški okvir **Property Pages**.

Prikaz zaglavlja stupaca

Kad je prikaz trake vodoravan, možete prikazati zaglavlja u vašoj hijerarhijskoj fleksibilnoj mreži. Zaglavlja se prikazuju neposredno iznad trake, ponavljajući se za svaku traku u hijerarhijskoj fleksibilnoj mreži. Kako bi prikazali samo jedan skup zaglavlja za svaku traku, na gornjoj ili lijevoj strani hijerarhijske fleksibilne mreže, upotrijebite nepomične ćelije umjesto zaglavlja. Pogledajte “Prilagođavanje izgleda nepomičnog područja” (u odlomku “Prilagođavanje područja hijerarhijske fleksibilne mreže”), kasnije u ovom poglavlju za informacije o nepomičnim ćelijama.

Vodoravna zaglavlja stupaca



1. Kliknite desnom tipkom miša na hijerarhijsku fleksibilnu mrežu, i odaberite **Properties** u pomoćnom izborniku. Pojavit će se dijaloški okvir **Property Pages** hijerarhijske fleksibilne mreže.
2. Na kartici **Bands** odaberite **ColumnHeaders**, i odaberite stil zaglavlja stupca u okviru s popisom **TextStyleHeader**.
3. Kliknite **OK** za prihvaćanje svojstava zaglavlja stupaca vaše hijerarhijske fleksibilne mreže i zatvaranje dijaloškog okvira **Property Pages**.

Promjena reda stupaca

Možete promijeniti redoslijed stupaca unutar trake vaše hijerarhijske fleksibilne mreže.

Kako promijeniti redoslijed stupaca unutar trake

1. Odaberite stupac koji želite pomaknuti.
2. Koristeći strelice prema gore i dolje pomaknite stupac na novi položaj unutar trake.

Promjena boja i mrežnih linija

Informacija o boji i mrežnim linijama trake može se odrediti općenito ili pojedinačno. Možete promijeniti boje i mrežne linije kako bi istaknuli važne informacije unutar vaše hijerarhijske fleksibilne mreže te povećali čitljivost. Kao dodatak, možete odrediti želite li prikazati mrežne linije između ćelija unutar trake. Upotrijebite sljedeći postupak za promjenu boja i mrežnih linija za sve trake u vašoj hijerarhijskoj fleksibilnoj mreži.

Napomena Za promjenu informacije o boji pojedine trake, morate programski izvesti promjenu koristeći svojstvo BackColorBand.

Kako općenito promijeniti boju i mrežne linije

1. Kliknite desnom tipkom miša na hijerarhijsku fleksibilnu mrežu, i odaberite **Properties** u pomoćnom izborniku. Pojavit će se dijaloški okvir **Property Pages** hijerarhijske fleksibilne mreže.
2. Na kartici **Bands** odaberite stil u okviru **Gridlines** i kliknite **Apply**. Ovaj stil određuje tip linije koja se iscrtava između standardnih područja popunjenih tekstom vaše hijerarhijske fleksibilne mreže za određenu traku.
3. Na kartici **Style**, odaberite stil u okviru **GridLinesFixed**. Zatim, odaberite stil u okviru **GridLinesUnpopulated**, i kliknite **Apply**. Ovi stilovi određuju tip linija koje se iscrtavaju između nepomičnih i praznih područja vaše hijerarhijske fleksibilne mreže.
4. Na kartici **Color**, dodijelite boju svakom svojstvu mrežne linije. Kako bi to napravili, najprije odredite **Color Set**. Zatim, odredite svojstvo koje želite promijeniti (na primjer, **GridColor**), odaberite boju u **Color Palette**, i kliknite **Apply**. Ponovite to za svaku mrežnu liniju hijerarhijske fleksibilne mreže koju želite promijeniti.

   Napomena Kad koristite **standardne Windows** boje (**Windows Default**), hijerarhijska fleksibilna mreža pojavljuje se u bojama određenim u vašem kontrolnom panelu prikaza. Kao dodatak promjeni boja hijerarhijske fleksibilne mreže u standardne ili unaprijed određene boje, možete stvoriti svoje vlastite definicije boja klikom na **Edit Custom Color** te korištenjem rezultirajućeg dijaloškog okvira **Color**.
5. Kliknite **OK** za prihvaćanje svojstava mrežnih linija i boja za vašu hijerarhijsku fleksibilnu mrežu i zatvaranje dijaloškog okvira **Property Pages**.

Uvlačenje trake

Kod okomitog prikaza traka, možete uvući traku određen broj stupaca. To vam omogućuje jasno predstavljanje traka s informacijama korisniku. Uvučeni stupac prije svake trake sadržava prazne ćelije koje ne rade ništa. Zbog toga, korisnik ne može postaviti fokus na ta područja. Osobine oblika tih ćelija određene su svojstvima oblikovanja uvlačenja, kao što je svojstvo **GridLinesIndent**.

Pogledajte sliku okomitog prikaza traka u “Promjeni izgleda trake”, ranije.

Kako uvući traku

1. Kliknite desnom tipkom miša na hijerarhijsku fleksibilnu mrežu, i odaberite **Properties** u pomoćnom izborniku. Pojavit će se dijaloški okvir **Property Pages** hijerarhijske fleksibilne mreže.
2. Na kartici **Bands** odaberite **BandIndent** i odredite za koliko će stupaca traka biti uvučena.
3. Kliknite **OK** za prihvaćanje svojstava trake za vašu hijerarhijsku fleksibilnu mrežu i zatvaranje dijaloškog okvira **Property Pages**.

Korištenje funkcionalnosti trake s nehijerarhijskim skupom slogova

Traku nehijerarhijskog skupa slogova možete oblikovati korištenjem bilo kojeg svojstva u dijaloškom okviru Property Pages hijerarhijske fleksibilne mreže.

Nehijerarhijski skup slogova sadrži samo jednu traku, traku 0. Druge trake nisu dostupne jer su trake temeljene na skupovima slogova u hijerarhiji Command.

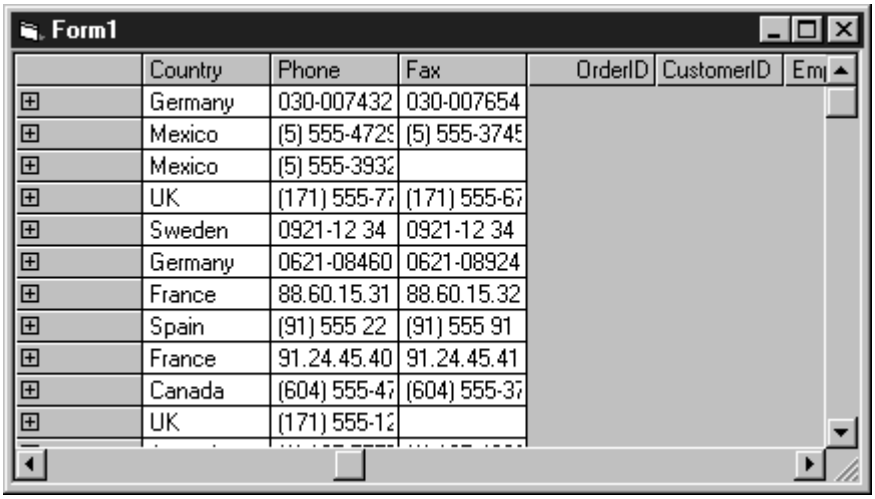
Korištenje mogućnosti širenja i skupljanja trake

Uz mogućnosti širenja i skupljanja trake lakše je vidjeti organizaciju skupa slogova i kretati se kroz hijerarhijsku fleksibilnu mrežu. Te mogućnosti omogućuju korisnicima pregled velike količine podataka ili zgušnjavanje informacija. Kad se traka može proširiti, u gornjem lijevom kutu pojavljuje se sličica za oznaku širenja (+) ili skupljanja (-). Mogućnost širenja i skupljanja dostupna je za okomite i vodoravne trake.

Kad su trake proširene pojavljuje se sličica skupljanja (-). U proširenom stanju, trake prikazuju maksimalnu količinu podataka. Kad su skupljene, pojavljuje se sličica proširivanja (+), a trake prikazuju minimalnu količinu podataka.

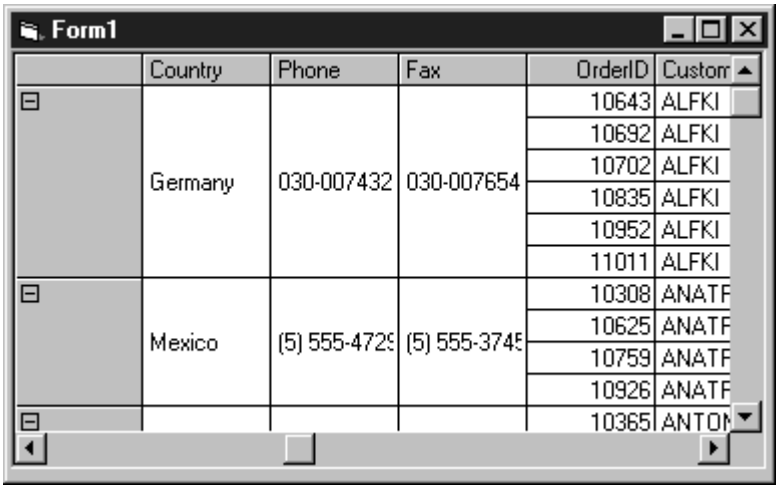
Kad su skupljene, trake mogu pokazivati nepopunjena područja. Ta područja oblikovana su u skladu s specifikacijama oblikovanja nepopunjenih područja.

Skupljena traka



Kad je traka skupljena tako da se ne prikazuje nijedan od njenih zapisa, stupci te trake neće biti prikazani. Kako korisnik proširuje traku, prikazuju se stupci. Kad su trake raširene, održavaju jednoliku visinu. Ako je potrebno, manje trake će se povećati kako bi posljedržale istu visinu kao i veće trake.

Proširena traka



Proširivanje i skupljanje traka

Ovaj dio opisuje kako dodati mogućnost širenja i skupljanja traci u vašoj hijerarhijskoj fleksibilnoj mreži. Također je opisano i kako upotrijebiti mogućnost širenja i skupljanja.

Kako traci dodati mogućnost širenja i skupljanja

1. Kliknite desnom tipkom miša na hijerarhijsku fleksibilnu mrežu, i odaberite **Properties** u pomoćnom izborniku. Pojavit će se dijaloški okvir **Property Pages** hijerarhijske fleksibilne mreže.
2. Na kartici **Bands**, odaberite **BandExpandable**. Zatim odaberite traku koju želite proširiti u popisu traka **Band**.

   Napomena Da bi traka bila proširiva, morate imati bar jednu podtraku. Zbog toga je kontrolna kućica **BandExpandable** onemogućena ako postoji samo traka 0.
3. Kliknite **OK** za prihvaćanje svojstava trake vaše hijerarhijske fleksibilne mreže i zatvaranje dijaloškog okvira **Property Pages**.

Kako upotrijebiti mogućnost širenja i skupljanja

1. Kad je jednom vašoj hijerarhijskoj fleksibilnoj mreži dodana ta mogućnost preko dijaloškog okvira **Property Pages**, u izborniku **Run**, odaberite **Start**.

   - ili -

Pritisnite F5.

2. Kliknite sličicu širenja (+) u gornjem lijevom kutu trake kako bi vidjeli maksimalne (proširene) informacije skupa slogova.
3. Kliknite sličicu skupljanja (–) u gornjem lijevom kutu trake kako bi vidjeli minimalne (skupljene) informacije skupa slogova.

   Napomena Ako proširite traku koja sadrži skupljenu podtraku, podtraka će ostati skupljena sve dok je ne proširite.

Uspostavljanje strukture

Informacije o strukturi vaše hijerarhijske fleksibilne mreže uključuju detaljne informacije o podešavanju vaše trake i stupca. U pravilu, poredak stupaca unutar svake trake odgovara poretku u pripadajućem skupu slogova tipa ADO. Kad je struktura uspostavljena, možete je upotrijebiti za kontroliranje prikaza podataka u vašoj hijerarhijskoj fleksibilnoj mreži.

Kako uspostaviti informacije trake i polja

- Kliknite desnom tipkom miša na vašu hijerarhijsku fleksibilnu mrežu, i u pomoćnom izborniku odaberite stavku **Retrieve Structure**.

  - ili -

  Otvorite dijaloški okvir **Property Page** i odaberite karticu **Bands**. Ta kartica sadrži informacije o strukturi vaše hijerarhijske fleksibilne mreže.

Nakon uspostavljanja strukture podataka, informacije se spremaju s hijerarhijskom fleksibilnom mrežom. Kad je struktura informacija uspostavljena, hijerarhijska fleksibilna mreža tijekom izrade aplikacije prikazuje ažurirane informacije za svaku traku i polje.

Napomena Ako se promijeni struktura vrijednosti svojstva **DataSource**, morate ponovno učitati strukturu prije nego što se promjene odraze u vašoj hijerarhijskoj fleksibilnoj mreži.

Brisanje informacija trake i stupaca

Kad obrišete informacije trake i stupca, one se obnavljaju na predodređene postavke.

Kako obrisati informacije trake i stupca

- Kliknite desnom tipkom miša na svoju hijerarhijsku fleksibilnu mrežu, i u pomoćnom izborniku odaberite **Clear Structure**.

  Napomena Ako ste promijenili svojstva trake ili stupca u vašoj hijerarhijskoj fleksibilnoj mreži, pojavit će se poruka upozorenja. U tom trenutku, možete kliknuti **OK** za ponovno postavljanje hijerarhijske fleksibilne mreže na standardno stanje i odbacivanje svih korisničkih postavki.

Prilagođavanje područja hijerarhijske fleksibilne mreže

Hijerarhijska fleksibilna mreža sadrži različita područja koja možete prilagođavati. Ta područja možete prilagoditi korištenjem dijaloškog okvira **Property Pages** hijerarhijske fleksibilne mreže ili programski, korištenjem **kodnog prozora**. Prilagođavanje područja može poboljšati čitljivost i povećati iskoristivost vaše hijerarhijske fleksibilne mreže.

Područja hijerarhijske fleksibilne mreže uključuju:

- **Standardno**

  Standardno područje vaše hijerarhijske fleksibilne mreže su ćelije koje sadrže informacije povezane s podacima.

- **Traka**

  Područje trake sadrži informacije prikaza za svaki skup slogova unutar vaše hijerarhijske fleksibilne mreže. Pogledajte “Oblikovanje trake” ranije u ovom poglavlju za više informacija.

- **Nepomično**

  Nepomično područje sastoji se od redova i stupaca koji su nepomični, ili statični.

- **Zaglavlje**

  Područje zaglavlja stupaca označuje informacije o skupu slogova koji je povezan s vašom hijerarhijskom fleksibilnom mrežom. Kad se koriste, zaglavlja stupaca ponavljaju se za svaku traku u vašoj hijerarhijskoj fleksibilnoj mreži. Pogledajte “Oblikovanje trake” ranije u ovom poglavlju za više informacija.

- **Uvlačenje**

  Kad se trake prikazuju okomito, područje uvlačenja vaše hijerarhijske fleksibilne mreže je područje koje uvlači traku podataka za određen broj stupaca. Uvučen stupac prije svake trake sadrži prazne ćelije koje ne rade ništa. Pogledajte “Oblikovanje trake” ranije u ovom poglavlju za više informacija.

- **Nepopunjeno**

  Nepopunjeno područje vaše hijerarhijske fleksibilne mreže sadrži ćelije desno i ispod radnih područja vaše hijerarhijske fleksibilne mreže. To područje je prazno i sadrži ćelije koje ne rade ništa.

Prilagođavanje izgleda standardnog područja

Možete prilagoditi izgled standardnog područja vaše hijerarhijske fleksibilne mreže, čineći informacije jasnijim i pristupačnijim. Standardno područje sadrži ćelije koje su povezane s podacima.

Kako prilagoditi izgled standardnog područja hijerarhijske fleksibilne mreže

1. Kliknite desnom tipkom miša na hijerarhijsku fleksibilnu mrežu, i odaberite **Properties** u pomoćnom izborniku. Pojavit će se dijaloški okvir **Property Pages** hijerarhijske fleksibilne mreže.
2. Na kartici **General** odredite sljedeća standardna svojstva:

svojstvo	opis
Rows	Broj redova u hijerarhijskoj fleksibilnoj mreži.
Columns	Broj stupaca u hijerarhijskoj fleksibilnoj mreži.
AllowBigSelection	Omogućuje korisniku odabir cijelog stupca ili reda unutar hijerarhijske fleksibilne mreže klikom na zaglavlje stupca ili reda.
MousePointer	Određuje tip pokazivača miša. Napomena Za upotrebu korisničkog pokazivača, morate dodijeliti korisničku ikonu svojstvu **MouseIcon** na kartici **Picture**.
FillStyle	Određuje hoće li oblikovanje teksta i ćelije vrijediti za pojedinu ćeliju ili za sve odabrane ćelije.
SelectionMode	Određuje hoće li vaša hijerarhijska fleksibilna mreža dopustiti odabir korisnika.
AllowUserResizing	Određuje smije li korisnik mijenjati veličinu redova i stupaca korištenjem miša.
ScrollBars	Određuje ima li hijerarhijska fleksibilna mreža traku za pomicanje, i ako ima, je li vodoravna, okomita ili obje.
HighLight	Određuje kad ćelije trebaju biti istaknute.
FocusRect	Određuje tip pravokutnika fokusa koji vaša hijerarhijska fleksibilna mreža iscrtava oko trenutne ćelije.

3. Na kartici **Style** odredite sljedeća standardna svojstva:

svojstvo	opis
MergeCells	Određuje hoće li ćelije s istim sadržajem biti grupirane u jednu ćeliju koja se proteže preko više ćelija ili redova.
RowHeightMin	Minimalna visina reda za cijelu kontrolu, u twipsima.
PictureType	Određuje prikazuje li se slika u boji ili kao jednobojna.
WordWrap	Određuje dopušta li vaša hijerarhijska fleksibilna mreža tekstu u ćeliji prijelom teksta ili višestruke linije.

4. Na kartici **Font** odredite sljedeća standardna svojstva:

svojstvo	opis
Font	Ime stila pisma.
Size	Veličina pisma, u točkama.
Effects	Efekti pisma; podvučeno ili prekriženo, ili stil teksta; podebljan ili nakošen. Možete potvrditi do četiri kontrolne kućice **Effects**.
Sample Text	Pregled odabranog pisma.

5. Kliknite **OK** za prihvaćanje standardnih svojstava vaše hijerarhijske fleksibilne mreže i zatvaranje dijaloškog okvira **Property Pages**.

Prilagođavanje izgleda nepomičnog područja

Možete stvoriti nepomične redove ili stupce za prikazivanje u vašoj hijerarhijskoj fleksibilnoj mreži. Za povećanje čitljivosti, trebali bi upotrijebiti nepomično područje za prikaz jednog niza zaglavlja umjesto korištenja ponavljajućih zaglavlja stupaca za svaku traku. Ovo područje ostaje statično, označavajući redove, ispod, ili stupce, prema desno.

Kako prilagoditi izgled nepomičnog područja hijerarhijske fleksibilne mreže

1. Kliknite desnom tipkom miša na vašu hijerarhijsku fleksibilnu mrežu, i odaberite **Properties** u pomoćnom izborniku. Pojavit će se dijaloški okvir **Property Pages** hijerarhijske fleksibilne mreže.
2. Na kartici **General**, odredite broj nepomičnih redova i stupaca u okvirima s tekstom **Fixed Rows** i **Fixed Cols**, te zatim kliknite **Apply**.
3. Na kartici **Style** odaberite stil nepomičnog teksta u okviru **TextStyleFixed**, te zatim kliknite **Apply**.
4. Na kartici **Font** odredite sljedeća standardna svojstva:

svojstvo	opis
Font	Ime stila pisma.
Size	Veličina pisma, u točkama.
Effects	Efekti pisma; podvučeno ili prekriženo, ili stil teksta; podebljan ili nakošen. Možete potvrditi do četiri kontrolne kućice **Effects**.
Sample Text	Pregled odabranog pisma.

5. Na kartici **Color**, dodijelite boju svakom nepomičnom svojstvu. Kako bi to napravili, najprije odaberite **Color Set**. Zatim, kliknite svojstvo koje želite promijeniti (na primjer, **BackColorFixed**), kliknite boju u **Color Palette**, te zatim kliknite **Apply**. Ponovite to za svako nepomično područje koje želite promijeniti.

   Napomena Kad koristite **standardne Windows** boje (**Windows default**), hijerarhijska fleksibilna mreža pojavljuje se u bojama određenim u vašem kontrolnom panelu prikaza. Kao dodatak promjeni boja hijerarhijske fleksibilne mreže u standardne ili unaprijed određene boje, možete stvoriti vaše vlastite definicije boja klikom na **Edit Custom Color** te korištenjem rezultirajućeg dijaloškog okvira **Color**. Za informacije o svakom svojstvu, pogledajte temu svojstava hijerarhijske fleksibilne mreže.

6. Kliknite **OK** za prihvaćanje nepomičnih svojstava vaše hijerarhijske fleksibilne mreže i zatvaranje dijaloškog okvira **Property Pages**.

Prilagođavanje izgleda zaglavlja

Možete promijeniti svojstva oblika i prikaza za zaglavlja trake u vašoj hijerarhijskoj fleksibilnoj mreži.

Kako prilagoditi izgled zaglavlja u hijerarhijskoj fleksibilnoj mreži

1. Kliknite desnom tipkom miša na vašu hijerarhijsku fleksibilnu mrežu, i odaberite **Properties** u pomoćnom izborniku. Pojavit će se dijaloški okvir **Property Pages** hijerarhijske fleksibilne mreže.

2. Na kartici **Bands** odaberite stil zaglavlja u okviru **TextStyleHeader**, te zatim odaberite **ColumnHeaders**.

   Napomena Kako bi spriječili prikazivanje dupliciranih zaglavlja u vašoj hijerarhijskoj fleksibilnoj mreži, morate u okvir s tekstom **Fixed Row**, na kartici **General**, upisati 0. Kao dodatak, ako želite prikazati samo jedan niz zaglavlja u traci na vrhu hijerarhijske fleksibilne mreže, moraju se koristiti nepomični redovi umjesto zaglavlja stupaca.

3. U okvirima s popisom **Column Caption** i **Column Name**, odredite stupce koji će biti prikazani. U pravilu, ova lista uključuje sva polja u skupu slogova s njihovim imenima. Kako bi maknuli objekt Field iz prikaza ili promijenili njegovo ime, upotrijebite popis **Column Caption**. Za odznačavanje polja, kliknite pripadajuću kvačicu. Kako bi polju promijenili ime, kliknite ime za odabir, i ponovno ga kliknite za pristup modu izmjene gdje mu možete dati novo ime.

4. Kliknite **OK** za prihvaćanje svojstava trake vaše hijerarhijske fleksibilne mreže i zatvaranje dijaloškog okvira **Property Pages**.

Prilagođavanje nepopunjenih područja

Možete promijeniti svojstva oblikovanja i prikaza nepopunjenih područja vaše hijerarhijske fleksibilne mreže. Ta nepopunjena područja su prazna i ne sadrže podatke.

Kako prilagoditi izgled nepopunjenih područja u hijerarhijskoj fleksibilnoj mreži

1. Kliknite desnom tipkom miša na vašu hijerarhijsku fleksibilnu mrežu i odaberite **Properties** u pomoćnom izborniku. Pojavit će se dijaloški okvir **Property Pages** hijerarhijske fleksibilne mreže.
2. Na kartici **Style** promijenite svojstvo **GridLinesUnpopulated**.
3. Na kartici **Color**, odredite boju svakom nepomičnom svojstvu. Da biste to napravili, najprije odaberite **Color Set**. Zatim, kliknite svojstvo koje želite promijeniti (na primjer, **BackColorUnpopulated**), kliknite boju u **Color Palette**, i kliknite **Apply**. Ponovite ovaj postupak za svako nepopunjeno područje koje želite promijeniti.

   Napomena Kad koristite **standardne Windows** boje (**Windows default**), hijerarhijska fleksibilna mreža pojavljuje se u bojama određenim u vašem kontrolnom panelu prikaza. Kao dodatak promjeni boja hijerarhijske fleksibilne mreže u standardne ili unaprijed određene boje, možete stvoriti vaše vlastite definicije boja klikom na **Edit Custom Color** te korištenjem rezultirajućeg dijaloškog okvira **Color**.
4. Kliknite **OK** za prihvaćanje nepomičnih svojstava vaše hijerarhijske fleksibilne mreže i zatvaranje dijaloškog okvira **Property Pages**.

Povezivanje vaše hijerarhijske fleksibilne mreže s okolinom podataka

Prije korištenja traka u vašoj hijerarhijskoj fleksibilnoj mreži, trebate povezati fleksibilnu mrežu s okolinom podataka, koja sadrži glavni, roditeljski objekt Command.

Kako povezati vašu hijerarhijsku fleksibilnu mrežu s okolinom podataka

1. Stvorite **MSHFlexGrid** kontrolu na formi Visual Basica.
2. U prozoru s svojstvima Visual Basica, postavite svojstvo **DataSource** na objekt okoline podataka s objektom Command kojeg želite povezati s kontrolom hijerarhijske fleksibilne mreže. Na primjer, DataEnvironment1.
3. U prozoru s svojstvima Visual Basica, postavite svojstvo **DataMember** na glavni, roditeljski objekt Command u vašoj okolini podataka. Vaši podaci su sad povezani s kontrolom hijerarhijske fleksibilne mreže.
4. Kako bi vidjeli podatke u hijerarhijskoj fleksibilnoj mreži, odaberite **Start** iz izbornika **Run** ili pritisnite F5. Struktura se uspostavlja i pojavljuje u hijerarhijskoj fleksibilnoj mreži.

Sortiranje i spajanje podataka u hijerarhijskoj fleksibilnoj mreži

U ovom primjeru, možete sortirati i spajati podatke u hijerarhijskoj fleksibilnoj mreži. U većini slučajeva, podacima pristupate učitavanjem iz baze podataka u hijerarhijsku fleksibilnu mrežu. Međutim, ovdje ćete unijeti primjere podataka korištenjem kodnog prozora kako bi popunili stupce i redove vaše hijerarhijske fleksibilne mreže.

Kako stvoriti ovakav prikaz podataka

1. Odredite svojstva hijerarhijske fleksibilne mreže.
2. Stvorite podatke.
3. Odredite rutine za izračunavanje indeksa i sortiranje.
4. Odredite rutine za upis podataka (iz koraka 2) u hijerarhijsku fleksibilnu mrežu.
5. Omogućite kontroli mijenjanje pregleda organizacije podataka.

Kako bi dovršili ovaj primjer, slijedite postupke u ovom dijelu po prikazanom redu.

Određivanje svojstava kontrole

Upotrijebite postavke u sljedećoj tablici za određivanje broja stupaca i redova, informacije pisma, te za stvaranje zaglavlja stupaca vaše hijerarhijske fleksibilne mreže.

Kontrola MSHFlexGrid

svojstvo	postavka
Name	Fg1
Cols	4
Rows	20
MergeCells	2 – Restrict Rows
FormatString	<Region|<Product|<Employees|>Sales
FontName	Arial

Sortiranje i spajanje podataka

Upotrijebite sljedeće procedure za završavanje primjera sortiranja i spajanja podataka u vašoj hijerarhijskoj fleksibilnoj mreži.

Kako sortirati i spajati podatke

1. Stvorite matricu za spremanje primjera podataka. Kako bi to napravili, ubacite sljedeću rutinu u događaj Form_Load vašeg kodnog prozora:

```
Sub Form_Load()
  Dim I As Integer
  ‘ Stvaranje matrice.
  For I = Fg1.FixedRows To Fg1.Rows - 1
     ‘ Područje.
     Fg1.TextArray(fgi(I, 0)) = RandomString(0)
     ‘ Proizvod.
     Fg1.TextArray(fgi(I, 1)) = RandomString(1)
     ‘ Djelatnik.
     Fg1.TextArray(fgi(I, 2)) = RandomString(2)
     Fg1.TextArray(fgi(I, 3)) = Format(Rnd * 10000 “#.00”)
  Next

  ‘ Određivanje spajanja.
  Fg1.MergeCol(0) = True
  Fg1.MergeCol(1) = True
  Fg1.MergeCol(2) = True

  ‘ Sortiranje kako bi vidjeli efekte.
  DoSort

  ‘ Oblikovanje mreže.
  Fg1.ColWidth(0) = 1000
  Fg1.ColWidth(1) = 1000
  Fg1.ColWidth(2) = 1000
  Fg1.ColWidth(3) = 1000
End Sub
```

2. Izračunajte indeks i završite sortiranje. Kako bi to napravili, odredite rutinu za izračunavanje indeksa i obavljanje sortiranja. Indeks se koristi s svojstvom **TextArray** za sortiranje podataka. Ubacite sljedeću rutinu za izračunavanje indeksa:

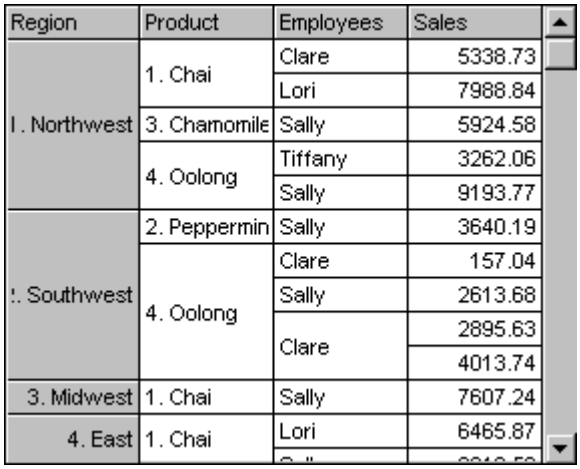
```
Function Fgi (r As Integer, c As Integer) As Integer
  Fgi = c + Fg1.Cols * r
End Function
```

```
Sub DoSort()
  Fg1.Col = 0
  Fg1.ColSel = Fg1.Cols - 1
  Fg1.Sort = 1          ‘ Opće rastuće sortiranje.
End Sub
```

3. Unesite podatke u vašu hijerarhijsku fleksibilnu mrežu. Kako bi to napravili, odredite rutinu koja će popuniti hijerarhijsku fleksibilnu mrežu primjerima podataka:

```
Function RandomString (kind As Integer)
  Dim s As String
  Select Case kind

    Case 0      ‘ Područje.
      Select Case (Rnd * 1000) Mod 5
        Case 0: s = “1. Northwest”
        Case 1: s = “2. Southwest”
        Case 2: s = “3. Midwest”
        Case 3: s = “4. East”
        Case Else: s = “5. Overseas”
      End Select

    Case 1      ‘ Proizvod.
      Select Case (Rnd * 1000) Mod 5
        Case 0: s = “1. Chai”
        Case 1: s = “2. Peppermint”
        Case 2: s = “3. Chamomile”
        Case Else: s = “4. Oolong”
      End Select

    Case 2      ‘ Djelatnik.
      Select Case (Rnd * 1000) Mod 4
        Case 0: s = “Clare”
        Case 1: s = “Tiffany”
        Case 2: s = “Sally”
        Case Else: s = “Lori”
      End Select
  End Select
  RandomString = s
End Function
```

Ako pokrenete projekt u ovom trenutku, trebao bi izgledati otprilike ovako:



Region	Product	Employees	Sales
1. Northwest	1. Chai	Clare	5338.73
		Lori	7988.84
	3. Chamomile	Sally	5924.58
	4. Oolong	Tiffany	3262.06
		Sally	9193.77
2. Southwest	2. Peppermin	Sally	3640.19
	4. Oolong	Clare	157.04
		Sally	2613.68
		Clare	2895.63
			4013.74
3. Midwest	1. Chai	Sally	7607.24
4. East	1. Chai	Lori	6465.87

Zatim, trebate omogućiti korisniku da reorganizira podatke. To znači da morate dopustiti hijerarhijskoj fleksibilnoj mreži prebacivanje pregleda organizacije podataka.

4. Reorganizirajte podatke dodavanjem sljedeće rutine, koja će odvući stupce na nove položaje. Ova rutina koristi svojstvo **Tag** s spremanje broja stupca kad korisnik pritisne tipku miša, pokrećući događaj MouseDown.

```
Sub Fg1_MouseDown (Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  Fg1.Tag = “”
  If Fg1.MouseRow <> 0 Then Exit Sub
  Fg1.Tag = Str(Fg1.MouseCol)
  MousePointer = vbSizeWE
End Sub
```

5. Dodajte sljedeću rutinu za ponovno namještanje stupaca i sortiranje podataka kad korisnik otpusti tipku miša, pokrećući događaj MouseUp.

```
Sub Fg1_MouseUp (Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  MousePointer = vbDefault
  If Fg1.Tag = “” Then Exit Sub
  Fg1.Redraw = False
  Fg1.ColPosition(Val(Fg1.Tag)) = Fg1.MouseCol
  DoSort
  Fg1.Redraw = True
End Sub
```

Jednom kad su procedure ovog primjera završene, podaci se automatski ponovno organiziraju uvijek kad povučete stupac na novi položaj tijekom izvođenja aplikacije. Na primjer, ako povučete stupac Employee prema lijevo, izgledat će ovako:

Employees	Region	Product	Sales
Clare	1. Northwest	1. Chai	5338.73
	2. Southwes	4. Oolong	157.04
			2895.63
			4013.74
	5. Overseas	2. Peppermin	3820.11
Lori	4. East	1. Chai	6465.87
	5. Overseas	2. Peppermin	2268.66
	1. Northwest	1. Chai	7988.84
	5. Overseas	4. Oolong	6478.21
Sally	4. East	1. Chai	9619.53
		4. Oolong	5833.59
	2. Southwes	4. Oolong	2613.68

Editiranje ćelija u proračunskoj tablici hijerarhijske fleksibilne mreže

U ovom primjeru, možete editirati ćelije u proračunskoj tablici hijerarhijske fleksibilne mreže. Ovaj primjer pokazuje neke od sposobnosti događaja i spremnika kontrole hijerarhijske fleksibilne mreže, i pokazuje kako ta kontrola može biti upotrijebljena za stvaranje proračunske tablice s mogućnošću editiranja ćelija korištenjem standardnih kontrola Visual Basica.

Napomena Primjer aplikacije (Flex.vbp) prikazuje funkcionalnost potrebnu za kretanje naokolo te za odabir opsega ćelija.

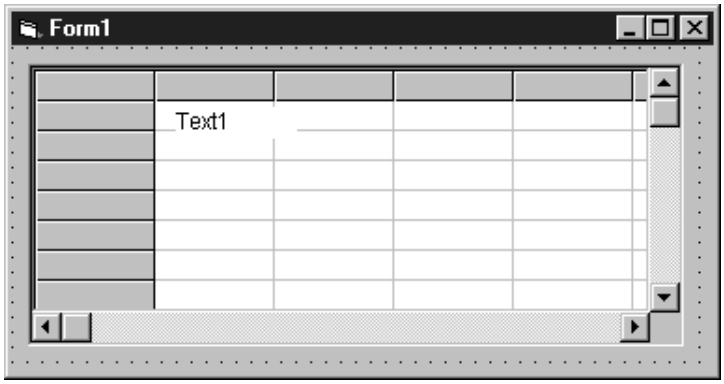
Kako stvoriti ovaj prikaz podataka

1. Stvorite kontrole hijerarhijske fleksibilne mreže i okvira s tekstom.
2. Odredite svojstva tih kontrola.
3. Dodajte zaglavlja za redove i stupce hijerarhijskoj fleksibilnoj mreži.
4. Dodajte mogućnost editiranja unutar ćelija hijerarhijskoj fleksibilnoj mreži.
5. Dodajte kontroli okvira s tekstom (okviru za “editiranje”) funkcionalnost ažuriranja podataka.
6. Kopirajte podatke iz okvira s tekstom u hijerarhijsku fleksibilnu mrežu.

Kako bi završili ovaj primjer, slijedite postupke u ovom dijelu po prikazanom redoslijedu.

Stvaranje kontrola

Dodajte vašem projektu hijerarhijsku fleksibilnu mrežu, zatim dodajte kontrolu okvira s tekstom kako bi stvorili odnos roditelj-dijete, kako je prikazano:



Određivanje svojstava kontrola

Odredite svojstva kontrola hijerarhijske fleksibilne mreže i okvira s tekstom kako slijedi:

Kontrola MSHFlexGrid

svojstvo	postavka
Name	Fg2
Cols	6
Rows	20
FillStyle	1 – Repeat
FocusRect	2 – Heavy
FontName	Arial
FontSize	9

Kontrola TextBox

svojstvo	postavka
Name	TxtEdit
FontName	Arial
FontSize	9
BorderSize	0 – None
Visible	False

Editiranje ćelija u proračunskoj tablici

Upotrijebite sljedeće procedure za editiranje ćelija u vašoj hijerarhijskoj fleksibilnoj mreži.

Kako editirati ćelije u proračunskoj tablici

1. Preinačite vašu hijerarhijsku fleksibilnu mrežu, tako da sliči proračunskoj tablici, dodavanjem sljedećeg koda potprogramu Form_Load u kodnom prozoru:

```
Sub Form_Load()
  Dim i As Integer

  ‘ Sužavanje prvog stupca.
  Fg2.ColWidth(0) = Fg2.ColWidth(0) / 2
  Fg2.ColAlignment(0) = 1      ‘ Centriranje središta.

  ‘ Označavanje redova i stupaca.
  For i = Fg2.FixedRows To Fg2.Rows - 1
     Fg2.TextArray(fgi(i, 0)) = i
  Next
  For i = Fg2.FixedCols To Fg2.Cols - 1
     Fg2.TextArray(fgi(0, i)) = i
  Next

  ‘ Pokretanje okvira za editiranje (pa se učitava sad).
  txtEdit = “”
End Sub
```

2. Stvorite funkciju za izračunavanje indeksa za svojstvo TextArray na ovaj način:

```
Function Fgi (r As Integer, c As Integer) As Integer
  Fgi = c + Fg2.Cols * r
End Function
```

3. Dodajte sljedeći kod u događaje **KeyPress** i **DblClick** hijerarhijske fleksibilne mreže:

```
Sub Fg2_KeyPress (KeyAscii As Integer)
  MSHFlexGridEdit Fg2, txtEdit, KeyAscii
End Sub

Sub Fg2_DblClick()
  MSHFlexGridEdit Fg2, txtEdit, 32     ‘ Simuliranje razmaka.
End Sub
```

4. Dodajte sljedeću rutinu za pokretanje okvira s tekstom i prosljeđivanje fokusa s hijerarhijske fleksibilne mreže na kontrolu okvira s tekstom:

```
Sub MSHFlexGridEdit (MSHFlexGrid As Control, _
Edt As Control, KeyAscii As Integer)

  ‘ Korištenje karaktera koji je upisan.
  Select Case keyascii

  ‘ Razmak znači editiranje trenutnog teksta.
  Case 0 To 32
     Edt = MSHFlexGrid
     Edt.SelStart = 1000

  ‘ Sve drugo znači zamjenu trenutnog teksta.
  Case Else
     Edt = Chr(keyascii)
     Edt.SelStart = 1
  End Select

  ‘ Prikaz Edt na pravom mjestu.
  Edt.Move MSHFlexGrid.Left + MSHFlexGrid.CellLeft, _
     MSHFlexGrid.Top + MSHFlexGrid.CellTop, _
     MSHFlexGrid.CellWidth - 8, _
     MSHFlexGrid.CellHeight - 8
  Edt.Visible = True

  ‘ Početak rada.
  Edt.SetFocus
End Sub
```

5. Dodajte mogućnost ažuriranja podataka okviru s tekstom dodavanjem sljedeće rutine u njegove događaje KeyPress i DblClick:

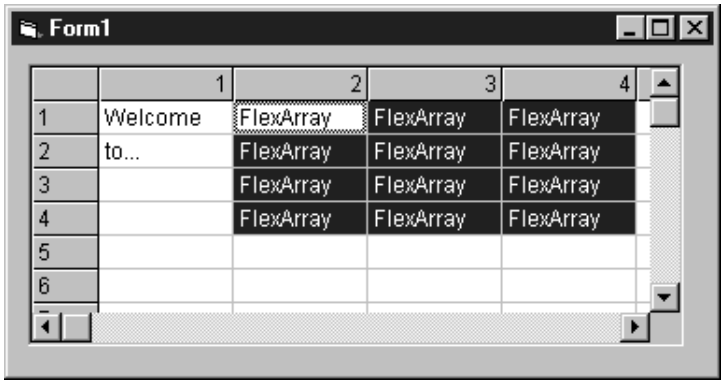
```
Sub txtEdit_KeyPress (KeyAscii As Integer)
  ‘ Brisanje tipke Return za prekid pištanja.
  If KeyAscii = Asc(vbCr) Then KeyAscii = 0
End Sub

Sub txtEdit_KeyDown (KeyCode As Integer, Shift As Integer)
  EditKeyCode Fg2, txtEdit, KeyCode, Shift
End Sub

Sub EditKeyCode (MSHFlexGrid As Control, Edt As _
Control, KeyCode As Integer, Shift As Integer)

  ‘ Standardna obrada kontrole editiranja.

  Select Case KeyCode
  Case 27               ‘ ESC; sakrij, vrati fokus na MSHFlexGrid.
     Edt.Visible = False
     MSHFlexGrid.SetFocus

  Case 13               ‘ ENTER vraća fokus na MSHFlexGrid.
     MSHFlexGrid.SetFocus

  Case 38               ‘ Gore.
     MSHFlexGrid.SetFocus
     DoEvents
     If MSHFlexGrid.Row > MSHFlexGrid.FixedRows Then
         MSHFlexGrid.Row = MSHFlexGrid.Row - 1
     End If

  Case 40               ‘ Dolje.
     MSHFlexGrid.SetFocus
     DoEvents
     If MSHFlexGrid.Row < MSHFlexGrid.Rows - 1 Then
         MSHFlexGrid.Row = MSHFlexGrid.Row + 1
     End If
  End Select
End Sub
```

Zatim, trebate reći hijerarhijskoj fleksibilnoj mreži što napraviti s podacima kad su uneseni u okvir s tekstom. Fokus se vraća kontroli nakon što korisnik unese podatke i pritisne ENTER ili klikne drugu ćeliju u hijerarhijskoj fleksibilnoj mreži. Zbog toga morate kopirati podatke iz okvira s tekstom u aktivnu ćeliju; da bi to napravili, nastavite s sljedećim koracima.

6. Kopirajte podatke iz okvira s tekstom u hijerarhijsku fleksibilnu mrežu dodavanjem sljedećeg koda u potprograme događaja GotFocus i LeaveCell:

```
Sub Fg2_GotFocus()
   If txtEdit.Visible = False Then Exit Sub
   Fg2 = txtEdit
   txtEdit.Visible = False
End Sub

Sub Fg2_LeaveCell()
   If txtEdit.Visible = False Then Exit Sub
   Fg2 = txtEdit
   txtEdit.Visible = False
End Sub
```

Kad su procedure ovog primjera završene, podaci mogu biti uneseni i pojedine ćelije tijekom izvođenja aplikacije, kako je ovdje prikazano.



Prikaz obrisa s zaglavljima korištenjem hijerarhijske fleksibilne mreže

U ovom primjeru možete upotrijebiti hijerarhijsku fleksibilnu mrežu za stvaranje prikaza s stilom obrisa uz točke zaglavlja koje možete skupiti ili proširiti.

Kako stvoriti ovaj prikaz podataka

1. Odredite svojstva hijerarhijske fleksibilne mreže.
2. Stvorite podatke.
3. Dodajte mogućnost skupljanja i širenja.

Kako bi završili ovaj primjer, slijedite postupke u ovom dijelu po prikazanom redu.

Određivanje svojstava kontrole

Postavite četiri zaglavlja stupaca korištenjem svojstva **FormatString**. Odredite prvi stupac kao sužen i prazan (slično proračunskoj tablici), a ostala tri za sadržavanje podataka (uključujući razmak između svakog zaglavlja). Kao dodatak, odredite sljedeće:

Kontrola MSHFlexGrid

svojstvo	postavka
Name	Fg3
Cols	4
Rows	2
SelectionMode	1 – By Row
FillStyle	1 – Repeat
FocusRect	0 – None
GridLines	0 – None
GridLinesFixed	2 – Inset
FormatString	^|Description|>Date|>Amount
FontName	Arial

Stvaranje prikaza s obrisom s zaglavljima

Upotrijebite sljedeće procedure kako bi završili primjer stvaranja prikaza s obrisom s zaglavljima koristeći hijerarhijsku fleksibilnu mrežu.

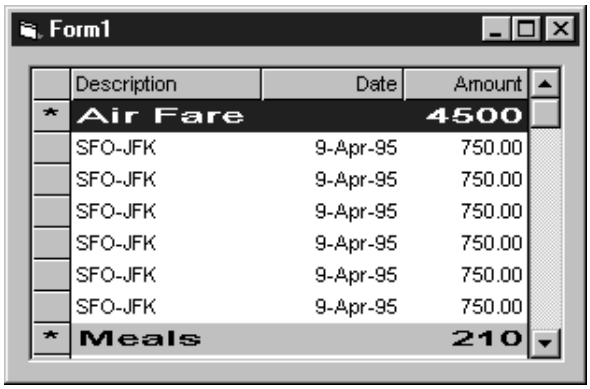
Kako stvoriti prikaz s obrisom s zaglavljima

1. Stvorite primjere podataka, i odredite svojstva zaglavlja, stupaca i redova. Kako bi to napravili, ubacite sljedeći kod u događaj Form_Load uz pomoć kodnog prozora. Ovaj kod će stvoriti primjere podataka, postaviti i izračunati zaglavlja na vrhu kontrole, te odrediti svojstva **Col** i **Row** kako bi tijekom rada aplikacije bila odabrana prva ćelija.

```
Sub Form_Load()
   Dim i As Integer, tot As Integer
   Dim t As String, s As String

   ‘ Stvaranje primjera podataka.
   t = Chr(9)
   Fg3.Rows = 1
```

```
   Fg3.AddItem "*" + t + "Airfare"
   s = "" + t + "SFO-JFK" + t + "9-Apr-95" + t + "750.00"
   For i = 0 To 5
      Fg3.AddItem s
   Next

   Fg3.AddItem "*" + t + "Meals"
   s = "" + t + "Flint's BBQ" + t + "25-Apr-95" + t + "35.00"
   For i = 0 to 5
      Fg3.AddItem s
   Next

   Fg3.AddItem "*" + t + "Hotel"
   s = "" + t + "Center Plaza" + t + 25-Apr-95" + t + "817.00"
   For i = 0 to 5
      Fg3.AddItem s
   Next

   ' Dodavanje zbrojeva i oblikovanje unosa zaglavlja.
   For i = Fg3.Rows - 1 To 0 Step -1
      If Fg3.TextArray(i * Fg3.Cols) = "" Then
          tot = tot + Val(fg3.TextArray(i * Fg3.Cols + 3))
      Else
          Fg3.Row = i
          Fg3.Col = 0
          Fg3.ColSet = Fg3.Cols - 1
          Fg3.CellBackColor = &HC0C0C0
          Fg3.CellFontBold = True
          Fg3.CellFontWidth = 8
          Fg3.TextArray(i * Fg3.Cols + 3) = Format(tot "0")
          tot = 0
      End If
   Next
   Fg3.ColSel = Fg3.Cols - 1

   ' Oblikovanje mreže.
   Fg3.ColWidth(0) = 300
   Fg3.ColWidth(1) = 1500
   Fg3.ColWidth(2) = 1000
   Fg3.ColWidth(3) = 1000
End Sub
```

Tijekom izvođenja aplikacije, redovi su sortirani u tri odjela, grupirana prema njihovim pojedinim zaglavljima: Air Fare, Meals i Hotels, kako je ovdje prikazano:

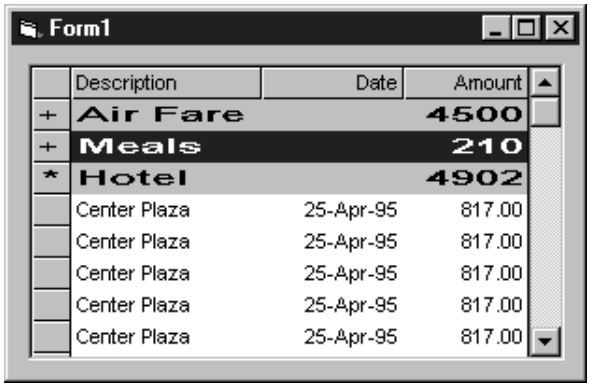


2. Dodajte mogućnost skupljanja i širenja zaglavljima redova ubacivanjem sljedećeg programskog koda u potprogram događaja DblClick hijerarhijske fleksibilne mreže:

```
Sub Fg3_DblClick()
  Dim i As Integer, r As Integer

  ‘ Zanemarivanje gornjeg reda.
  r = Fg3.MouseRow
  If r < 1 Then Exit Sub

  ‘ Pronalaženje polja za sužavanje ili proširivanje.
  While r > 0 And Fg3.TextArray(r * Fg3.Cols) = “”
     r = r - 1
  Wend

  ‘ Prikaz simbola za sužavanje/širenje u prvom stupcu.
  If Fg3.TextArray(r * Fg3.Cols) = “*” Then
     Fg3.TextArray(r * Fg3.Cols) = “+”
  Else
     Fg3.TextArray(r * Fg3.Cols) = “*”
  End If

  ‘ Širenje stavki ispod trenutnog zaglavlja.
  r = r + 1
  If Fg3.RowHeight(r) = 0 Then
     Do While Fg3.TextArray(r * Fg3.Cols) = “”
        Fg3.RowHeight(r) = -1         ‘ Predodređena visina reda.
        r = r + 1
        If r >= Fg3.Rows Then Exit Do
     Loop
```

```
' Sužavanje stavki ispod trenutnog zaglavlja.
  Else
     Do While Fg3.TextArray(r * Fg3.Cols) = ""
         Fg3.RowHeight(r) = 0          ' Skrivanje reda.
         r = r + 1
         If r >= Fg3.Rows Then Exit Do
     Loop
  End If
End Sub
```

Kad završite sve korake ovog primjera, tijekom izvođenja aplikacije možete širiti ili sužavati zaglavlja redova dvostrukim klikom na simbole "+" ili "*" u prvom stupcu, kao što je ovdje prikazano:



Napomena Ovaj primjer možete promijeniti tako da prikazuje slike umjesto karaktera "+" i "*", ili možete dodati dodatne razine obrisu.

Korištenje kontrola vodoravne i okomite klizne trake

Klizne trake (Scroll Bars) omogućuju lako kretanje kroz dugu popis stavki ili velik iznos informacija vodoravnim ili okomitim pomicanjem unutar aplikacije ili kontrole. Klizne trake su uobičajen element sučelja Windowsa 95/98 i Windowsa NT.

Slika 7.23 Kontrole vodoravne i okomite klizne trake

Kontrole vodoravne i okomite klizne trake nisu iste kao i ugrađene klizne trake koje se mogu pronaći u Windowsima ili trake koje su dodijeljene okvirima s tekstom, okvirima s popisom, kombiniranim okvirima ili MDI formama unutar Visual Basica. Te klizne trake pojavljuju se automatski uvijek kad dana aplikacija ili kontrola sadrži više informacija nego što može biti prikazano u trenutnoj veličini prozora (ili, u slučaju okvira s tekstom i MDI formi, kad je svojstvo ScrollBars također postavljeno na True).

U prethodnim verzijama Visual Basica, klizne trake često su korištene kao sredstva za unos. Smjernice Windows sučelja sad savjetuju, međutim, da se kao sredstva za unos koriste kontrole klizača umjesto kontrola kliznih traka. Kontrola klizača uključena je u Professional i Enterprise verzije Visual Basica.

Kontrole kliznih traka i dalje imaju vrijednost u Visual Basicu jer omogućuju pomicanje kroz aplikacije ili kontrole koje to ne osiguravaju automatski. Pogledajte "Primjer kontrola kliznih traka: stvaranje pomičnog grafičkog prozora", kasnije u ovom poglavlju, za informacije o korištenju kliznih traka na takav način.

Kako rade klizne trake

Kontrole kliznih traka koriste događaje Scroll i Change za nadzor pomicanja gumba za pomicanje (ponekad označenog kao kotačić) uzduž klizne trake.

događaj	opis
Change	Pojavljuje se nakon pomaka gumba za pomicanje.
Scroll	Pojavljuje se tijekom pomaka gumba za pomicanje. Ne pojavljuje se ako su kliknute strelice za pomicanje ili klizna traka.

Korištenje događaja Scroll pruža pristup vrijednosti klizne trake tijekom pomicanja. Događaj Change pojavljuje se nakon otpuštanja gumba za pomicanje te kad je kliknuta klizna traka ili strelice za pomicanje.

Svojstvo Value

Svojstvo Value (koje ima predodređenu vrijednost 0) je cjelobrojna vrijednost koja odgovara položaju gumba za pomicanje na kliznoj traci. Kad je položaj tog gumba na najmanjoj vrijednosti, on se nalazi na krajnjem lijevom položaju (na vodoravnoj kliznoj traci) ili najgornjem položaju (na okomitoj kliznoj traci). Kad gumb za pomicanje ima najveću vrijednost, nalazi se na krajnjem desnom ili donjem položaju. Slično tome, vrijednost na pola puta između dna i vrha opsega postavlja gumb za pomicanje na sredinu klizne trake.

Kao dodatak korištenju klikanja mišem za promjenu vrijednosti klizne trake, korisnik također može povući gumb za pomicanje na bilo koji položaj na traci. Rezultirajuća vrijednost ovisi o položaju gumba, ali uvijek je unutar opsega svojstava Min i Max koje je odredio korisnik.

> **Napomena** Vrijednost svojstva Min može biti veća od vrijednosti svojstva Max ako želite da vaša klizna traka prikazuje informacije koje se mijenjaju od veće k manjoj vrijednosti.

Svojstva LargeChange i SmallChange

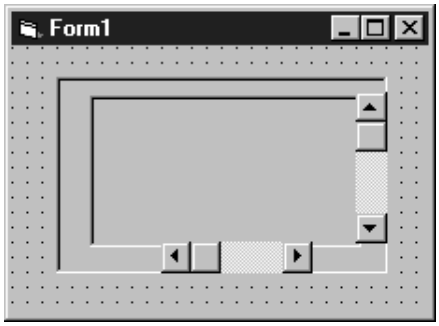
Za određivanje iznosa promjene koja je pokazana u kliznoj traci, upotrijebite svojstvo LargeChange za klikanje na traku za pomicanje, i svojstvo SmallChange za klikanje strelica na krajevima klizne trake. Svojstvo Value klizne trake povećava se ili smanjuje za vrijednosti određene u svojstvima LargeChange i SmallChange. Položaj gumba za pomicanje možete odrediti postavljanjem svojstva Value na vrijednost između 0 i 32.767, uključivo.

Primjer kontrola kliznih traka: Stvaranje pomičnog grafičkog prozora

Kontrole vodoravne i okomite klizne trake mogu biti upotrijebljene, kao dodatak kontroli okvira za sliku, za stvaranje aplikacije pomičnog grafičkog prozora. Sama kontrola okvira za sliku ne omogućuje vam pomicanje slike ako je slika veća od okvira – kontrola okvira za sliku ne dodaje automatski klizne trake.

Ova aplikacija koristi dva okvira za sliku. Prvi okvir se označuje kao nepomična *roditeljska* kontrola okvira za sliku. Drugi okvir, koji je sadržan unutar roditeljskog, označuje se kao *dječja* kontrola okvira za sliku. Dijete okvir za sliku sadrži grafičku sliku i pomiče se unutar roditeljske kontrole okvira za sliku korištenjem klizna traka.

Slika 7.24 Dodavanje klizna traka tijekom izrade aplikacije



Počnite stvaranjem novog projekta u kojem ćete na formi stvoriti dva okvira za sliku, te po jednu vodoravnu i okomitu kliznu traku, kao što je prikazano na slici 7.24.

Događaj forme Form_Load koristi se za određivanje moda skale, veličine dječjeg okvira za sliku unutar roditeljskog okvira za sliku, položaja i veličine vodoravne i okomite klizne trake, te za učitavanje bitmapirane slike. Dodajte sljedeći kod u potprogram događaja forme Form_Load:

```
Private Sub Form_Load()

   ‘ Postavljanje svojstva ScaleMode na piksele.
   Form1.ScaleMode = vbPixels
   Picture1.ScaleMode = vbPixels
```

```
    ‘ Svojstvo AutoSize se postavlja na True kako bi se
    ‘ okvir kontrole Picture2 proširio na veličinu slike.
    Picture2.AutoSize = True

    ‘ Postavljanje svojstva BorderStyle oba okvira na None.
    Picture1.BorderStyle = 0
    Picture2.BorderStyle = 0

    ‘ Učitavanje slike.
    Picture2.Picture = LoadPicture(“c:\Windows\Winlogo.bmp”)

    ‘ Pokretanje pozicioniranja oba okvira.
    Picture1.Move 0, 0, ScaleWidth - VScroll1.Width, _
    ScaleHeight - HScroll1.Height
    Picture2.Move 0, 0

    ‘ Postavljanje vodoravne klizne trake.
    HScroll1.Top = Picture1.Height
    HScroll1.Left = 0
    HScroll1.Width = Picture1.Width

    ‘ Postavljanje okomite klizne trake.
    VScroll1.Top = 0
    VScroll1.Left = Picture1.Width
    VScroll1.Height = Picture1.Height

    ‘ Određivanje svojstva Max za klizne trake.
    HScroll1.Max = Picture2.Width - Picture1.Width
    VScroll1.Max = Picture2.Height - Picture1.Height
    ‘ Određivanje hoće li slika-dijete popuniti ekran. Ako hoće,
    ‘ nema potrebe za korištenjem klizna traka.
    VScroll1.Visible = (Picture1.Height < Picture2.Height)
    HScroll1.Visible = (Picture1.Width < Picture2.Width)

End Sub
```

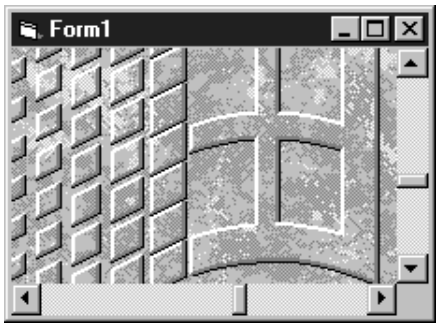
Događaj Change vodoravne i okomite klizne trake koristi se za pomicanje dječjeg okvira za sliku gore i dolje ili lijevo i desno unutar roditeljskog okvira za sliku. Dodajte sljedeći programski kod u događaj Change obje klizne trake:

```
Private Sub HScroll1_Change()
  Picture2.Left = -HScroll1.Value
End Sub

Private Sub VScroll1_Change()
  Picture2.Top = -VScroll1.Value
End Sub
```

Svojstva Left i Top dječjeg okvira za sliku postavljaju se na negativne vrijednosti vodoravne i okomite klizne trake tako da se, kad se pomičete gore, dolje, lijevo ili desno, odgovarajuće pomiče i prikaz.

Tijekom izvođenja aplikacije, slika će biti prikazana kao na slici 7.25.

Slika 7.25 Pomicanje slike tijekom izvođenja aplikacije



Promjena veličine forme tijekom izvođenja

U malo prije opisanom primjeru, veličina vidljivog dijela slike ograničena je originalnom veličinom forme. Kako bi promijenili veličinu grafičkog prozora u aplikaciji kad korisnik promijeni veličinu forme tijekom rada aplikacije, dodajte sljedeći kod u potprogram događaja Form_Resize forme.

```
Private Sub Form_Resize()
  ‘ Kad se promijeni veličina forme,
  ‘ mijenjaju se dimenzije kontrole Picture1.
  Picture1.Height = Form1.Height
  Picture1.Width = Form1.Width
```

```
    ‘ Ponovno pokretanje položaja okvira
    ‘ za sliku i kliznu traka.
    Picture1.Move 0, 0, ScaleWidth - VScroll1.Width, _
    ScaleHeight - HScroll1.Height
    Picture2.Move 0, 0
    HScroll1.Top = Picture1.Height
    HScroll1.Left = 0
    HScroll1.Width = Picture1.Width
    VScroll1.Top = 0
    VScroll1.Left = Picture1.Width
    VScroll1.Height = Picture1.Height
    HScroll1.Max = Picture2.Width - Picture1.Width
    VScroll1.Max = Picture2.Height - Picture1.Height

    ‘ Provjera trebaju li klizne trake.
    VScroll1.Visible = (Picture1.Height < Picture2.Height)
    HScroll1.Visible = (Picture1.Width < Picture2.Width)

End Sub
```

Korištenje kontrole slike

Kontrola slike (Image) koristi se za prikaz grafike. Kontrole slike mogu prikazivati slike sljedećih oblika: bitmapirane slike, ikone, metadatoteke, poboljšane metadatoteke, te JPEG ili GIF datoteke.

Slika 7.26 Kontrola slike

Kao dodatak, kontrole slike odgovaraju na događaj Click i mogu se koristiti kao zamjena za naredbene gumbe, kao dijelovi alatne trake, ili za stvaranje jednostavnih animacija.

Za više informacija Pogledajte “Grafičke kontrole lake kategorije” u 3. poglavlju “Forme, kontrole i izbornici”, za jednostavnu demonstraciju korištenja kontrole slike kao naredbenog gumba. Pogledajte “Stvaranje alatne trake” u 6. poglavlju “Stvaranje korisničkog sučelja” za informacije o korištenju kontrole slike pri stvaranju alatne trake. Pogledajte “Stvaranje jednostavne animacije” u 12. poglavlju “Rad s tekstom i grafikom” za više informacija o korištenju kontrole slike za stvaranje jednostavnih animacija.

Kad koristiti kontrolu slike umjesto kontrole okvira za sliku

Kontrola slike koristi manje sistemskih izvora i brže iscrtava sliku od kontrole okvira za sliku, ali podržava samo manji dio svojstava, događaja i postupaka kontrole okvira za sliku. Obje kontrole podržavaju iste formate slika. Međutim, slike u kontroli slike možete razvući na veličinu kontrole. To ne možete napraviti s kontrolom okvira za sliku.

Za više informacija Pogledajte "Redukcija u grafici" u 15. poglavlju "Oblikovanje u korist izvođenja i sukladnosti", za informacije o korištenju kontrola slike za očuvanje sistemskih izvora.

Podržani grafički formati

Kontrola slike može prikazati slike u bilo kojem od sljedećih standardnih formata:

format slike	opis
Bitmapirana slika	*Bitmapirana slika* određuje sliku kao uzorak točaka (piksela). Ima sufiks imena datoteke .bmp ili .dib. Bitmapirane slike se također zovu i grafika " obojenog tipa". Možete koristiti bitmapirane slike raznih dubina boja, uključujući 2, 4, 8, 16, 24 i 32-bitne, ali bitmapirana slika se ispravno prikazuje samo ako sredstvo prikazivanja podržava dubinu boje koju koristi slika. Na primjer, slika s 8 bitova po pikselu (256 boja) prikazuje samo 16 boja kad se pokazuje na ekranu koji podržava samo 4 bita po pikselu (16 boja).
Ikona	*Ikona* je poseban tip bitmapirane slike. Najveća veličina ikona je 32 x 32 piksela, ali kod Microsoft Windowsa 95/98, postoje ikone veličine 16 x 16 piksela. Ikona ima sufiks imena datoteke .ico.
Kursor	Kursori su, kao i ikone, zapravo bitmapirane slike. Međutim, kursori sadrže i vruću točku, piksel koji prati položaj kursora svojim x i y koordinatama. Kursori imaju sufiks imena datoteke .cur.
Metadatoteka	*Metadatoteka* određuje sliku kao kodirane linije i oblike. Uobičajene metadatoteke imaju sufiks imena datoteke .wmf. Poboljšane metadatoteke imaju sufiks imena datoteke .emf. Samo datoteke koje su sukladne s Microsoft Windowsima mogu biti učitane. Metadatoteke se također nazivaju i grafikom "crtanog tipa".
JPEG	JPEG (Joint Photographic Experts Group) je format sžete bitmapirane slike koji podržava 8- i 24-bitnu boju. To je popularan format datoteke slike na Internetu.
GIF	GIF (Graphic Interchange Format) je format sžete bitmapirane slike originalno razvijen od CompuServea. Podržava do 256 boja i popularan je format datoteka na Internetu.

Učitavanje grafike u kontrolu slike

Slike mogu biti učitane u kontrolu slike tijekom izrade aplikacije određivanjem svojstva Picture u prozoru s svojstvima kontrole, ili tijekom izvođenja aplikacije korištenjem svojstva Picture i funkcije LoadPicture.

```
Set Image1.Picture = LoadPicture("c:\Windows\Winlogo.cur", _
vbLPLarge, vbLPColor)
```

Kad se slika učita u kontrolu slike, kontrola automatski mijenja svoju veličinu tako da odgovara slici – neovisno o tome koliko je mala ili velika kontrola stvorena na formi.

Možete poželjeti upotrijebiti datoteke ikona (.ico) ili kursora (.cur) koje sadrže odvojene slike raznih veličina i dubina boja za podršku nizu sredstava prikaza. Postavke funkcije LoadPicture omogućuju vam odabir slika određene dubine boja i veličine iz .ico ili .cur datoteke. U slučajevima kad točan par traženim postavkama nije dostupan, funkcija LoadPicture učitava sliku koja najviše odgovara traženoj.

Za brisanje grafike iz kontrole slike, upotrijebite funkciju LoadPicture bez određivanja imena datoteke. Na primjer:

```
Set Image1.Picture = LoadPicture
```

To će očistiti kontrolu slike čak i ako je grafika učitana u svojstvo Picture tijekom izrade aplikacije.

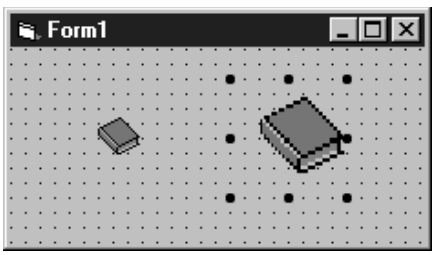
Korištenje odlagališta

Grafiku također možete dodati u kontrolu slike tijekom izrade aplikacije tako da je ulijepite iz druge aplikacije. Na primjer, možda želite dodati bitmapiranu sliku koja je stvorena u aplikaciji Windows Paint. Jednostavno kopirajte sliku u odlagalište, odaberite kontrolu slike, te upotrijebite prečicu tipkama CTRL + V ili naredbu Paste iz izbornika Edit.

Svojstvo Stretch

Svojstvo Stretch određuje hoće li slika biti razvučena kad kontrola slike promijeni veličinu tijekom izrade aplikacije. Ako je postavljeno na True, slika učitana u kontrolu slike svojstvom Picture bit će razvučena. Razvlačenje slike (posebno formata bitmapirane slike) može kao posljedicu imati gubitak kvalitete slike, kao što je prikazano na slici 7.27. Metadatoteke, koje su grafika "crtanog tipa", bolje su prilagođene razvlačenju.

Slika 7.27 Razvlačenje bitmapirane slike



Korištenje kontrole natpisa

Kontrole natpisa (Label) koriste se za prikaz teksta i korisnik ih ne može mijenjati. Koriste se za označavanje objekata na formi – pružajući opis što će pojedina kontrola napraviti kad bude kliknuta, na primjer – ili tijekom rada aplikacije, mogu prikazivati informacije kao odgovor na događaje ili procese u vašoj aplikaciji.

Slika 7.28 Kontrola natpisa

Natpisi se koriste u puno slučajeva, za puno različitih namjena. Najčešće se koriste za označavanje kontrola koje nemaju vlastito svojstvo Caption. Na primjer, možete upotrijebiti kontrolu natpisa za dodavanje opisnog natpisa okvirima s tekstom, okvirima s popisom, kombiniranim okvirima i tako dalje. Kontrole natpisa mogu također biti upotrijebljene za dodavanje opisnog teksta formi, na primjer, za pružanje pomoćnih informacija korisniku.

Možete također napisati kod koji mijenja tekst prikazan u kontroli natpisa kao odgovor na događaje tijekom rada aplikacije. Na primjer, ako vaša aplikacija treba nekoliko minuta za obradu neke promjene, u kontroli natpisa možete prikazivati poruku o stanju obrade.

Budući da kontrole natpisa ne mogu imati fokus, mogu biti upotrijebljene i za stvaranje pristupnih tipki drugim kontrolama.

Određivanje teksta natpisa

Kako bi promijenili tekst prikazan u kontroli natpisa, upotrijebite svojstvo Caption. Tijekom izrade, ovo svojstvo možete odrediti ako ga odaberete u prozoru s svojstvima kontrole.

Duljina svojstva Caption može biti postavljena na najviše 1024 bajta.

Poravnavanje teksta

Svojstvo Alignment omogućuje vam određivanje poravnavanja teksta unutar kontrole natpisa na lijevo poravnavanje (Left Justify, 0, standardno), centriranje (Center, 1) ili desno poravnavanje (Right Justify, 2).

Svojstva AutoSize i WordWrap

U pravilu, kad tekst upisan u svojstvo Caption prekorači širinu kontrole, tekst se prelama u iduću liniju i odrezuje ako prekoračuje visinu kontrole.

Da biste dopustili kontroli da automatski prilagodi svoju veličinu sadržaju, postavite svojstvo AutoSize na True. Kontrola će se proširiti vodoravno tako da odgovara cijelom sadržaju svojstva Caption. Kako bi omogućili prijelom sadržaja prema dolje i okomito širenje kontrole, postavite svojstvo WordWrap na True.

Za više informacija Pogledajte “Korištenje natpisa za prikaz teksta” u 3. poglavlju “Forme, kontrole i izbornici”, za jednostavnu demonstraciju svojstava AutoSize i WordWrap.

Korištenje natpisa za stvaranje pristupnih tipki

Postavite svojstvo UseMnemonic na True ako želite odrediti karakter u svojstvu Caption natpisa kao pristupnu tipku. Kad u kontroli natpisa odredite pristupnu tipku, korisnik može pritisnuti i držati pritisnutim ALT + karakter kojeg ste odredili za pomicanje fokusa na iduću kontrolu u tabulatornom redu.

Možete također stvoriti pristupne tipke za bilo koju kontrolu koja ima svojstvo Caption dodavanjem znaka & prije slova koje želite upotrijebiti kao pristupnu tipku. Za dodjelu pristupnih tipki kontrolama koje nemaju naslove, upotrijebite kontrolu natpisa s tom kontrolom. Budući da natpisi ne mogu primiti fokus, fokus se automatski pomiče na sljedeću kontrolu u tabulatornom redu. Upotrijebite ovu tehniku za dodjelu pristupnih tipki okvirima s tekstom, okvirima za sliku, kombiniranim okvirima, okvirima s popisom, okvirima s popisom pogonskih uređaja i direktorija, kontrolama mreže i slike.

Kako dodijeliti pristupnu tipku kontroli s natpisom

1. Najprije stvorite kontrolu natpisa, a zatim stvorite kontrolu.

   - ili -

   Stvorite kontrole po bilo kakvom redoslijedu i postavite svojstvo TabIndex kontrole natpisa za jedan manje od kontrole.

2. Upotrijebite znak & u svojstvu Caption kontrole natpisa s dodjelu pristupne tipke kontroli natpisa.

   Napomena Možda ćete trebati prikazati znak & u kontroli natpisa, umjesto da ga koristite za stvaranje pristupne tipke. To se može dogoditi ako povežete kontrolu natpisa s poljem u skupu slogova gdje podaci uključuju znak &. Kako bi prikazali znak & u kontroli natpisa, postavite njezino svojstvo UseMnemonic na False.

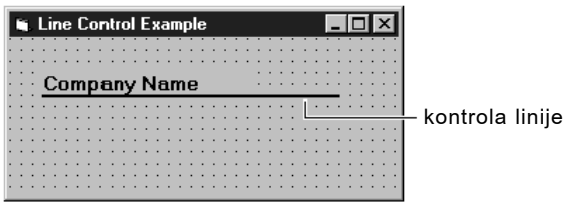
Korištenje kontrole linije

Kontrola linije (Line) koristi se za stvaranje jednostavnih dijelova linije na formi, okviru ili u okviru za sliku.

Slika 7.29 Kontrola linije

Možete kontrolirati položaj, duljinu, boju i stil kontrola linije kako bi prilagodili izgled aplikacije. Slika 7.30 prikazuje kontrolu linije upotrijebljenu za grafičko odvajanje natpisa s tekstom “Company Name” od ostatka forme.

Slika 7.30 Kontrola linije na formi



Kontrola linije ima ograničenu upotrebljivost i namijenjena je za jednostavnu upotrebu – prikaz i ispis. Dijelovi linije mogu biti spojeni tako da oblikuju druge likove, na primjer. Za naprednije upotrebe trebat ćete upotrijebiti postupak Line.

Za više informacija Pogledajte "Korištenje grafičkih postupaka" u 12. poglavlju "Rad s tekstom i grafikom", za više informacija o crtanju linija, pravokutnika i popunjenih okvira tijekom rada aplikacije koristeći postupak Line, ili za više informacija o crtanju krugova, elipsi i lukova tijekom rada aplikacije koristeći postupak Circle.

Određivanje stila ruba i boje

Boju i stil dijela linije određujete korištenjem svojstava BorderStyle i BorderColor.

Svojstvo BorderStyle omogućuje vam sedam stilova linije:

- Proziran (transparent)
- Pun (solid)
- S crtama (dash)
- S točkama (dot)
- Crta-točka (dash-dot)
- Crta-točka-točka (dash-dot-dot)
- Popunjen iznutra (inside solid)

Stil linije možete odrediti tijekom izrade aplikacije odabirom svojstva BorderStyle u prozoru s svojstvima kontrole linije ili, tijekom rada aplikacije, određivanjem stila koristeći odgovarajuću konstantu Visual Basica u programskom kodu.

Svojstvo BorderColor koristi se za određivanje boje linije.

Tijekom izrade aplikacije, boju linije možete odrediti odabirom svojstva BorderColor u prozoru s svojstvima kontrole linije te zatim odabirom iz raspoložive palete ili iz sistemskih boja.

Kako bi odredili boje tijekom rada aplikacije, upotrijebite konstante boja Visual Basica (vbGreen, na primjer) ili konstante sistemskih boja (vbWindowBackground, na primjer) ili funkciju RGB za određivanje boje ruba.

> **Napomena** Kad je svojstvo BorderStyle postavljeno na 0 (Transparent), svojstvo BorderColor se zanemaruje.

Za više informacija Pogledajte "Funkcija RGB" za informacije o određivanju RGB boja. Također, pogledajte 12. poglavlje "Rad s tekstom i grafikom" za detaljne informacije o stvaranju grafike u Visual Basicu.

Pomicanje i određivanje veličine linije

Kontrolu linije možete pomicati ili joj mijenjati veličinu tijekom rada aplikacije promjenom njezinih svojstava X1, X2, Y1 i Y2. Svojstva X1 i Y1 određuju vodoravne i okomite pozicije lijevog kraja dijela linije. Svojstva X2 i Y2 određuju vodoravne i okomite pozicije desnog kraja dijela linije. Kontrolu linije ne možete micati korištenjem postupka Move.

Crtanje linija na formi

Kontrolu linije možete upotrijebiti za crtanje jednostavnih linija na formi.

Kako nacrtati liniju na formi

1. U alatnom okviru, odaberite kontrolu linije.

   Kad se pokazivač pomakne na formu, mijenja se u križni pokazivač.

2. Kliknite formu na mjestu gdje želite početak linije i držite pritisnutu tipku miša.
3. Povucite križni pokazivač do mjesta gdje želite kraj linije i otpustite tipku miša.
4. U prozoru s svojstvima, odredite svojstvo **BorderStyle** ako želite promijeniti izgled linije.
5. U okviru **Settings**, odredite željeni stil.

Korištenje kontrole okvira s popisom

Kontrola okvira s popisom (List Box) prikazuje popis stavki od kojih korisnik može odabrati jednu ili više njih.

Slika 7.31 Kontrola okvira s popisom

Okviri s popisom predstavljaju korisniku popis izbora. U pravilu, izbori se prikazuju okomito u jednom stupcu, iako možete također postaviti i više stupaca. Ako je broj stavki veći od onog koji se može prikazati u okviru s popisom, na kontroli će se automatski pojaviti klizne trake. Korisnik se zatim može pomicati gore i dolje, ili lijevo i desno kroz popis. Slika 7.32 prikazuje okvir s popisom s jednim stupcem.

Slika 7.32 Kontrola s popisom s jednim stupcem

Osobine povezivanja s podacima

Visual Basic uključuje obje verzije kontrole okvira s podacima, standardnu i verziju za povezivanje s podacima. Iako vam obje verzije kontrole okvira s popisom omogućuju prikazivanje, editiranje i ažuriranje informacija iz većine standardnih tipova baza podataka, kontrola okvira s popisom podataka (DataList) također podržava i različit skup svojstava i postupaka od standardne kontrole okvira s podacima.

Za više informacija Pogledajte "Korištenje kontrola kombiniranog okvira za podatke i okvira s popisom podataka" za više informacija o verziji kontrole okvira s popisom za povezivanje s podacima.

Događaji Click i DoubleClick

Preporučena praksa za događaje okvira s popisom, posebno kad se okvir s popisom pojavljuje kao dio dijaloškog okvira, je dodavanje naredbenog gumba za upotrebu s okvirom s popisom. Potprogram događaja Click tog gumba trebao bi pokretati upotrebu odabira iz okvira s popisom, provodeći bilo koju akciju koja je prikladna vašoj aplikaciji.

Dvoklik stavke na popisu trebao bi imati isti učinak kao i odabir stavke te klik naredbenog gumba. Kako bi to napravili, neka potprogram DblClick okvira s popisom poziva potprogram Click naredbenog gumba:

```
Private Sub List1_DblClick()
  Command1_Click
End Sub
```

Ili, postavite vrijednost svojstva Value naredbenog gumba na True, što će automatski pozvati potprogram događaja:

```
Private Sub List1_DblClick()
  Command1.Value = True
End Sub
```

Taj način pruža prečicu korisnicima miša, a ipak ne sprečava korisnike tipkovnice u izvođenju iste akcije. Uočite da ne postoji akcija tipkovnicom jednaka događaju DblClick.

Dodavanje stavki na popis

Kako bi dodali stavke u okvir s popisom, upotrijebite postupak AddItem, koji ima sljedeću sintaksu:

okvir.**AddItem** *stavka* [, *indeks*]

argument	opis
okvir	Ime okvira s popisom.
stavka	Izraz tipa stringa koji se dodaje popisu. Ako je *stavka* tekstualni izraz, zatvorite ga u navodnike.

argument	opis
indeks	Određuje gdje će u popis biti ubačena nova stavka. *Indeks* jednak 0 predstavlja prvu poziciju. Ako je *indeks* izostavljen, stavka će biti dodana na kraj (ili prema redoslijedu sortiranja).

Iako se stavke popisa obično dodaju u potprogramu događaja Form_Load, postupak AddItem možete koristiti u bilo koje vrijeme. To vam daje mogućnost dinamičkog dodavanja stavki (kao odgovor na akcije korisnika).

Sljedeći programski kod postavlja stavke “Germany” “India” “France” i “USA” u okvir s popisom imena List1.

```
Private Sub Form_Load()
  List1.AddItem “Germany”
  List1.AddItem “India”
  List1.AddItem “France”
  List1.AddItem “USA”
End Sub
```

Uvijek kad se forma učita tijekom rada aplikacije, popis će izgledati kao na slici 7.33

Slika 7.33 Okvir s popisom “Države”

Dodavanje stavke na određen položaj

Kako bi dodali stavku u popis na određen položaj, odredite vrijednost indeksa za novu stavku. Na primjer, iduća linija koda ubacuje stavku “Japan” na prvu poziciju, pomičući pozicije ostalih stavki prema dolje:

```
List1.AddItem “Japan”, 0
```

Uočite da vrijednost 0, a ne 1, određuje prvu stavku u popisu (pogledajte sliku 7.34).

Slika 7.34 Dodavanje stavke popisu

Dodavanje stavki tijekom izrade aplikacije

Stavke također možete unijeti u popis tijekom izrade aplikacije određivanjem svojstva List u prozoru s svojstvima kontrole okvira s popisom. Kad odaberete svojstvo List i zatim kliknete pripadajuću strelicu prema dolje, možete upisati stavke popisa i pritisnuti kombinaciju tipki CTRL + ENTER za početak nove linije.

Možete dodati stavke samo na kraj popisa. Dakle, ako želite abecedno sortirati popis, postavite svojstvo Sorted na True. Pogledajte "Sortiranje popisa" u nastavku za više informacija.

Sortiranje popisa

Dodavanje stavki u popis abecednim redoslijedom možete odrediti postavljanjem svojstva Sorted na True te izostavljanjem indeksa. Sortiranje nije osjetljivo na vrstu slova; zbog toga će riječi "japan" i "Japan" biti jednako tretirane.

Kad je svojstvo Sorted postavljeno na True, korištenje postupka AddItem s argumentom *indeks* može dovesti do nepredvidljivih, nesortiranih rezultata.

Brisanje stavki iz popisa

Možete upotrijebiti postupak RemoveItem za brisanje stavki iz okvira s popisom. Postupak RemoveItem ima jedan argument, *indeks*, koji određuje stavku za brisanje:

okvir.**RemoveItem** *indeks*

Argumenti *okvir* i *indeks* isti su kao i za postupak AddItem.

Na primjer, za brisanje prve stavke popisa, dodat ćete sljedeću liniju programskog koda:

```
List1.RemoveItem 0
```

Za brisanje svih stavki standardnih i podatkovnih verzija okvira s popisom i kombiniranih okvira, upotrijebite postupak Clear:

```
List1.Clear
```

Dohvaćanje sadržaja popisa svojstvom Text

Obično je najlakši način dohvaćanja vrijednosti trenutno odabrane stavke upotreba svojstva Text. Svojstvo Text uvijek se podudara s stavkom popisa koju korisnik odabere tijekom izvođenja aplikacije.

Na primjer, sljedeći programski kod prikazuje informaciju o stanovništvu Kanade ako korisnik odabere "Canada" u okviru s popisom:

```
Private Sub List1_Click()
  If List1.Text = "Canada" Then
    Text1.Text = "Kanada ima 24 milijuna stanovnika."
  End If
End Sub
```

Svojstvo Text sadrži trenutno odabranu stavku u okviru s popisom List1. Programski kod provjerava je li odabrana stavka "Canada" te, ako je, prikazuje informaciju u okviru s tekstom Text1.

Pristup stavkama popisa svojstvom List

Svojstvo List pruža pristup svim stavkama u popisu. Ovo svojstvo sadrži matricu u kojoj je svaka stavka popisa jedan element matrice. Svaka stavka predstavljena je u obliku stringa. Za pozivanje stavke iz popisa, upotrijebite ovu sintaksu:

okvir.**List** (*indeks*)

Argument *okvir* je upućivanje na okvir s popisom, a *indeks* je pozicija stavke u popisu. Prva, najviša, stavka ima indeks 0; sljedeća ima indeks 1 i tako dalje. Na primjer, sljedeći izraz prikazuje treću stavku (*indeks* = 2) popisa u okviru s tekstom:

```
Text1.Text = List1.List(2)
```

Određivanje pozicije svojstvom ListIndex

Ako želite znati poziciju odabrane stavke u popisu, upotrijebite svojstvo ListIndex. Ovo svojstvo postavlja ili vraća indeks trenutno odabrane stavke u kontroli i na raspolaganju je samo tijekom rada aplikacije. Određivanje svojstva ListIndex za okvir s popisom će također prouzročiti događaj Click kontrole.

Vrijednost ovog svojstva je 0 ako je odabrana prva (najviša) stavka, 1 ako je odabrana druga stavka prema dolje i tako dalje. Svojstvo ListIndex ima vrijednost -1 ako ni jedna stavka nije odabrana.

> Napomena Svojstvo NewIndex omogućuje vam praćenje indeksa posljednje stavke dodane popisu. To može biti korisno kad dodajete stavku u sortirani popis.

Vraćanje broja stavki svojstvom ListCount

Kako bi dobili broj stavki u okviru s popisom, upotrijebite svojstvo ListCount. Na primjer, sljedeći izraz koristi svojstvo ListCount za utvrđivanje broja stavki u okviru s popisom:

```
Text1.Text = “Imate “ & List1.ListCount & “ stavki.”
```

Stvaranje okvira s popisom s više stupaca i više odabira

Svojstvo Columns omogućuje vam određivanje broja stupaca u okviru s popisom. Ovo svojstvo može imati sljedeće vrijednosti:

vrijednost	opis
0	Jednostupčan okvir s popisom s okomitim pomicanjem.
1	Jednostupčan okvir s popisom s vodoravnim pomicanjem.
>1	Višestupčan okvir s popisom s vodoravnim pomicanjem.

Visual Basic vodi brigu o prijelomu stavki popisa u iduću liniju te o dodavanju vodoravne klizne trake popisu ako je potrebno; ako popis popunjava jedan stupac, ne dodaje se klizna traka. Prijelom u idući stupac također se automatski pojavljuje prema potrebi. Zapamtite da će dio teksta stavke biti odrezan ako je tekst stavke širi od širine stupca.

Možete omogućiti korisnicima da iz popisa odaberu više stavki. Višestruki odabir u standardnim okvirima s popisom obrađuje se određivanjem svojstva MultiSelect, koje može imati sljedeće vrijednosti.

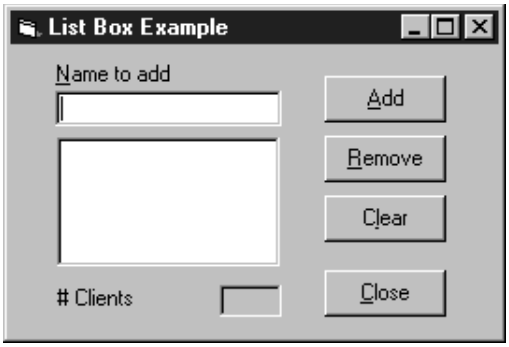
vrijednost	tip odabira	opis
0	nikakav	Standardan okvir s popisom.
1	jednostavan višestruki odabir	Klik ili SPACEBAR odabiru ili odznačuju dodatne stavke popisu.
2	proširen višestruki odabir	SHIFT + klik ili SHIFT + kursorska tipka proširuju odabir tako da uključuje sve stavke između trenutnog i prethodnih odabira. CTRL + klik označuju ili odznačuju stavke popisu.

Za više informacija Pogledajte "Primjer kontrole okvira s popisom 2: Stvaranje okvira s popisom s više stupaca", kasnije u ovom poglavlju, za više informacija o svojstvima Columns i MultiSelect.

Primjer kontrole okvira s popisom 1: Dodavanje i brisanje stavki

Ovaj primjer pokazuje kako možete upotrijebiti postupke AddItem, RemoveItem i Clear s svojstvima ListIndex i ListCount za dodavanje i micanje stavki popisa tijekom rada aplikacije. Primjer na slici 7.35 dopušta korisniku upis imena kupca u okvir s tekstom, koje će biti dodano u okvir s popisom ako je kliknut gumb Add. Korisnik može obrisati trenutnu stavku popisa tako da je odabere i klikne gumb Remove, ili odabirom gumba Clear može obrisati sve stavke popisa.

Slika 7.35 Okvir s popisom koristi postupke AddItem, RemoveItem i Clear



Broj kupaca u okviru s popisom prikazuje se u kontroli natpisa koja izgleda kao okvir s tekstom (svojstvo BorderStyle je postavljeno na 1 – Fixed Single). Ovaj natpis ažurira se svaki put kad se ime kupca doda ili obriše. Budući da je svojstvo Sorted okvira s popisom postavljeno na True, stavke se dodaju u okvir s popisom po abecednom redu.

Stvorite formu s okvirom s tekstom, okvirom s popisom, tri kontrole natpisa, i četiri naredbena gumba. Sljedeća tablica ispisuje postavke svojstava za objekte u aplikaciji.

objekt	svojstvo	postavka
gornji okvir s tekstom	Name Text	txtName (prazno)
gornji natpis	Name Caption	lblName &Name to add
okvir s popisom	Name Sorted	lstCLient True
donji natpis	Name Caption	lblClients # Clients
natpis s brojem kupaca (izgleda kao okvir s tekstom)	Name Caption BorderStyle	lblDisplay (prazno) 1 – Fixed Single
prvi naredbeni gumb	Name Caption	cmdAdd &Add
drugi naredbeni gumb	Name Caption	cmdRemove &Remove
treći naredbeni gumb	Name Caption	cmdClear C&lear
četvrti naredbeni gumb	Name Caption	cmdClose &Close

Događaji u aplikaciji s okvirom s popisom

Dodajte sljedeći programski kod u potprogram događaja cmdAdd_Click:

```
Private Sub cmdAdd_Click()
  lstClient.AddItem txtName.Text         ‘ Dodavanje u popis.
  txtName.Text = “”                      ‘ Brisanje okvira s tekstom.
  txtName.SetFocus
  ‘ Prikaz broja kupaca.
  lblDisplay.Caption = lstClient.ListCount
End Sub
```

Dodajte sljedeći programski kod u potprogram događaja cmdRemove_Click:

```
Private Sub cmdRemove_Click()
  Dim Ind As Integer

  Ind = lstCLient.ListIndex              ‘ Dohvaćanje indeksa.
  ‘ Je li odabrana stavka popisa.
  If Ind >= 0 Then
    ‘ Brisanje s popisa.
    lstClient.RemoveItem Ind
    ‘ Prikaz broja kupaca.
    lblDisplay.Caption = lstClient.ListCount
  Else
    Beep
  End If
  ‘ Onemogućavanje gumba ako u popisu nema stavki.
  cmdRemove.Enabled = (lstClient.ListIndex <> -1)
End Sub
```

Dodajte sljedeći programski kod u potprogram događaja cmdClear_Click:

```
Private Sub cmdClear_Click()
  ‘ Brisanje okvira s popisom.
  lstClient.Clear
  ‘ Onemogućavanje gumba Remove.
  cmdRemove.Enabled = False
  ‘ Prikaz broja kupaca.
  lblDisplay.Caption = lstClient.ListCount
End Sub
```

Dodajte sljedeći programski kod u potprogram događaja cmdClose_Click:

```
Private Sub cmdClose_Click()
  Unload Me
End Sub
```

Dodajte sljedeći programski kod u potprogram događaja lstClient_Click:

```
Private Sub lstClient_Click()
  cmdRemove.Enabled = (lstClient.ListIndex <> -1)
End Sub
```

Dodajte sljedeći programski kod u potprogram događaja txtName_Change:

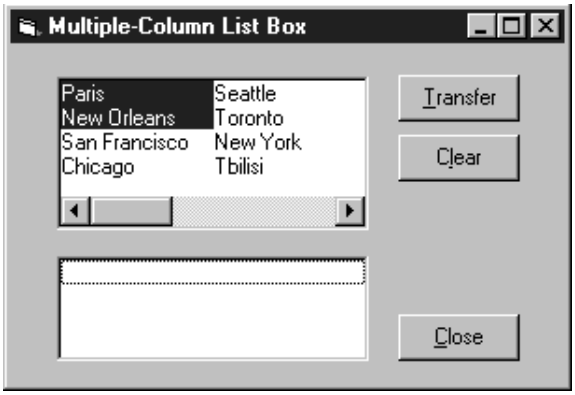
```
Private Sub txtName_Change()
  ‘ Omogućavanje gumba Add ako u imenu
  ‘ postoji bar jedan karakter.
  cmdAdd.Enabled = (Len(txtName.Text) > 0)
End Sub
```

Primjer kontrole okvira s popisom 2: Stvaranje okvira s popisom s više stupaca

Kako bi stvorili okvir s popisom s više stupaca i mogućnošću višestrukog odabira, trebate odrediti njegova svojstva Columns i MultiSelect. U sljedećem primjeru, ta svojstva se koriste za stvaranje takvog okvira s popisom.

Uočit ćete da kad pokrenete aplikaciju, okvir s popisom sadrži dva stupca, kao što je prikazano na slici 7.36.

Slika 7.36 Okvir s popisom s više stupaca



Ako stvorite okvir s popisom dovoljno velik da prikazuje sve stavke u jednom stupcu, drugi stupac bit će prazan; ostale stavke će biti prelomljene, a vodoravna klizna traka će se automatski pojaviti samo ako okvir s popisom nije dovoljno dug. Pokušajte promijeniti veličinu gornjem okviru s popisom i dodati mu dodatne stavke u popis da vidite kako Visual Basic automatski rukuje višestrukim stupcima.

Ovaj primjer koristi svojstvo Selected – matricu vrijednosti tipa Boolean koja sadrži stanja odabira okvira s popisom – za određivanje koje su stavke odabrane. Svaki zapis u matrici odgovara stavci popisa i postavljen je na True ako je stavka odabrana, ili na False ako nije odabrana. Nakon što korisnik odabere stavke iz popisa, provjerava se svaki zapis matrice kako bi se vidjelo je li odabran (True). Ako jest, zapis se dodaje drugom popisu, normalnom okviru s popisom s jednim stupcem, korištenjem postupka AddItem.

Postavite svojstva objekata u ovom primjeru kao što je naznačeno u sljedećoj tablici.

objekt	svojstvo	postavka
forma	Caption	Multiple-Column List Box
gornji okvir s popisom	Name	lstTop
	Columns	2
	MultiSelect	2 – Extended

objekt	svojstvo	postavka
donji okvir s popisom	Name	lstBottom
prvi naredbeni gumb	Name Caption	cmdTransfer &Transfer
drugi naredbeni gumb	Name Caption	cmdClear C&lear
treći naredbeni gumb	Name Caption	cmdClose &Close

Svojstvo MultiSelect omogućuje vam odabir niza vrijednosti u okviru s popisom. Ako kliknete na prvu stavku popisa, te zatim pritisnete SHIFT i kliknete posljednju stavku u nizu (ili upotrijebite tipke SHIFT + STRELICA DOLJE), bit će odabrane sve stavke u nizu.

Događaji u aplikaciji s višestupčanim okvirom s popisom

Dodajte sljedeći kod u potprogram Form_Load za pokretanje gornjeg popisa, lstTop:

```
Private Sub Form_Load()
  lstTop.AddItem “Paris”
  lstTop.AddItem “New Orleans”
  lstTop.AddItem “San Francisco”
  lstTop.AddItem “Chicago”
  lstTop.AddItem “Seattle”
  lstTop.AddItem “Toronto”
  lstTop.AddItem “New York”
  lstTop.AddItem “Tbilisi”
  lstTop.AddItem “Moscow”
  lstTop.AddItem “Portland”
  ‘ Odabir nekoliko stavki.
  lstTop.Selected(0) = True
  lstTop.Selected(1) = True
End Sub
```

Napomena Možete dodati stavke okvirima s popisom bez ponavljajućeg korištenja postupka AddItem tako da upišete stavke u svojstvo List u prozoru s svojstvima. Nakon upisa svake stavke, pritisnite CTRL + ENTER za prijelaz u iduću liniju. Ovakav postupak omogućuje vam upis višestrukih unosa u višestupčani okvir s popisom.

Dodajte sljedeći programski kod u potprogram događaja lstTop_DblClick:

```
Private Sub lstTop_DblClick()
  cmdTransfer.Value = True       ‘ Pritisak gumba Transfer.
End Sub
```

Dodajte sljedeći kod u potprogram događaja Click naredbenog gumba Transfer:

```
Private Sub cmdTransfer_Click()
  For n = 0 To (lstTop.ListCount - 1)
‘ Ako je stavka odabrana, dodavanje donjem popisu.
    If lstTop.Selected(n) = True Then
       lstBottom.AddItem = lstTop.List(n)
    End If
  Next
  cmdClear.Enabled = True
End Sub
```

Uočite da vrijednosti matrice indeksa počinju od 0 i idu do ListCount – 1.

Dodajte sljedeći kod u potprogram događaja Click naredbenog gumba Clear:

```
Private Sub cmdClear_Click()
  lstBottom.Clear
End Sub
```

Dodajte sljedeći kod u potprogram događaja Click naredbenog gumba Close:

```
Private Sub cmdClose_Click()
  Unload Me
End Sub
```

Korištenje kontrole OLE spremnika

U kontrolu OLE spremnika (OLE Container) možete povezati ili umetnuti bilo koji objekt koji podržava automatizaciju (prije zvanu OLE automatizacija). Koristeći ovu kontrolu, vaša Visual Basic aplikacija može prikazati i upravljati podacima iz drugih Windows temeljenih aplikacija, kao što su Microsoft Excel i Microsoft Word za Windowse.

Slika 7.37 Kontrola OLE spremnika

Kontrola OLE spremnika koristi se za stvaranje aplikacije usmjerene na dokumente. U takvoj aplikaciji, korisnik kombinira podatke iz različitih aplikacija kako bi stvorio jedan dokument. Takav tip aplikacije može biti obradnik teksta koji korisniku omogućuje upis teksta te zatim umetanje proračunske tablice ili grafikona.

Kontrola OLE spremnika dopušta vam dodavanje objekata iz drugih aplikacija u vaše Visual Basic aplikacije. S ovom kontrolom možete:

- Stvoriti u svojoj aplikaciji oznaku mjesta za objekt. Možete stvoriti objekt koji se pojavljuje u kontroli OLE spremnika tijekom rada aplikacije, ili možete promijeniti objekt kojeg ste postavili u kontrolu OLE spremnika tijekom izrade aplikacije.
- Stvoriti povezani objekt u svojoj aplikaciji.

- Povezati kontrolu OLE spremnika s bazom podataka.
- Izvesti akciju ako korisnik, pomakne, promijeni veličinu, ili ažurira objekt u kontroli OLE spremnika.
- Stvoriti objekte iz podataka koji su kopirani u odlagalište.
- Prikazati objekte kao ikone.
- Pružiti sukladnost prema natrag s aplikacijom koja uključuje puno kontrola OLE spremnika (nazvanih OLE klijent kontrola u prethodnim verzijama Visual Basica).

Za više informacija Pogledajte "Korištenje vidljivog sučelja sastavnih dijelova" u 10. poglavlju "Programiranje sastavnim dijelovima", za detaljnu raspravu i primjere kako se koristi kontrola OLE spremnika.

Korištenje kontrole gumba izbora

Kontrole gumba izbora (Option Button) koriste se za prikaz izbora, obično u grupama gumbâ izbora, od kojih korisnik može izabrati jedan.

Slika 7.38 Kontrola gumba izbora

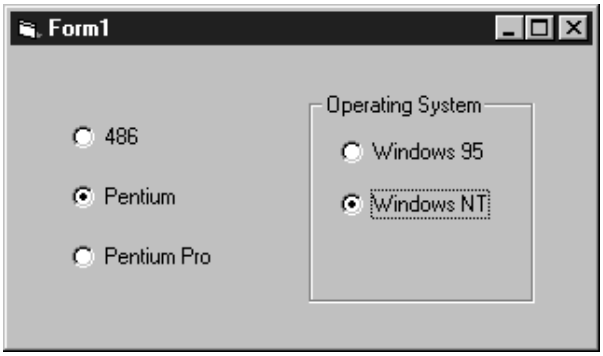
Iako kontrola gumba izbora i kontrola kontrolne kućice izgledaju kao da djeluju slično, postoji važna razlika; kad korisnik odabere gumb izbora, ostale kontrole gumba izbora u istoj grupi bit će automatski odznačene. Suprotno tome, istovremeno može biti odabran bilo koji broj kontrola kontrolne kućice.

Za više informacija Pogledajte "Grupiranje izbora gumbima izbora" u 3. poglavlju "Forme, kontrole i izbornici" za prikaz upotrebe gumbâ izbora.

Stvaranje grupa gumbâ izbora

Kontrole gumbâ izbora možete grupirati tako da ih kreirate unutar spremnika kao što je kontrola okvira, kontrola okvira za sliku ili forma. Tijekom izvođenja aplikacije, korisnik može odabrati jedan gumb izbora iz svake zasebne grupe gumbâ izbora. Na primjer, ako dodate gumbe izbora na formu te gumbe izbora u kontrolu okvira na formi, stvorili ste dvije zasebne grupe gumbâ izbora.

Slika 7.39 Stvaranje grupa gumbâ izbora



Svi gumbi izbora koji su dodani direktno na formu postaju jedna grupa. Kako bi dodali dodatne grupe, morate ih postaviti unutar kontrola okvira ili okvira za sliku.

Kako bi grupirali kontrole gumbâ izbora u okviru ili okviru za sliku, najprije stvorite okvir ili okvir za sliku, pa onda stvorite kontrole gumba izbora unutar njih. Tijekom izrade aplikacije, gumbi izbora sadržani unutar okvira ili okvira za sliku mogu biti odabrani i pomaknuti kao samostalna jedinica.

Kako bi odabrali više kontrola sadržanih unutar kontrole okvira, kontrole okvira za sliku ili forme, držite pritisnutu tipku CTRL dok mišem crtate okvir oko kontrola.

Za više informacija Pogledajte "Korištenje kontrole okvira" ranije u ovom poglavlju za više informacija o stvaranju kontrola unutar okvira.

Odabir gumba izbora tijekom rada aplikacije

Gumb izbora može biti odabran tijekom rada aplikacije na više načina: klikom mišem, korištenjem tipke TAB za prebacivanje fokusa na kontrolu, korištenjem tipke TAB za odabir grupe kontrola gumbâ izbora te korištenjem kursorskih tipki za odabir jednog gumba unutar grupe, stvaranjem pristupne tipke u natpisu gumba izbora, ili postavljanjem svojstva Value gumba izbora na True u programskom kodu.

Događaj Click

Kad je odabran gumb izbora, pokreće se događaj Click. Ovisno o namjeni u vašoj aplikaciji, možete odgovoriti ili ne odgovoriti na taj događaj. Odgovor na taj događaj koristan je kad želite ažurirati tekst kontrole natpisa za pružanje informacija korisniku o izboru koji je odabran, na primjer.

Svojstvo Value

Svojstvo Value kontrole gumba izbora pokazuje je li gumb izbora odabran. Kad je odabran, vrijednost ovog svojstva mijenja se na True. Gumb izbora možete odabrati iz programskog koda postavljanjem njegovog svojstva Value. Na primjer:

```
optPentium.Value = True
```

Kako bi gumb izbora bio predodređen unutar grupe gumbâ izbora, odredite njegovo svojstvo Value tijekom izrade aplikacije korištenjem prozora s svojstvima, ili tijekom rada aplikacije iz programskog koda, kako je malo prije pokazano.

Kad korisniku pokažete dijaloški okvir koji sadrži gumbe izbora, tražite od njega da odabere izbore koji će odrediti daljnji rad aplikacije. Možete upotrijebiti svojstvo Value svake kontrole gumba izbora kako bi odredili koji je izbor ili izbori odabran te odgovoriti sukladno tome.

Stvaranje prečica tipkovnicom

Svojstvo Caption možete iskoristiti za stvaranje prečica s pristupnim tipkama za vaše gumbe izbora tako da dodate znak & prije slova koje želite upotrijebiti kao pristupnu tipku. Na primjer, kako bi stvorili pristupnu tipku gumbu izbora s natpisom "Pentium" dodajte znak & prije slova "P": "&Pentium". Tijekom rada aplikacije, slovo "P" bit će podvučeno i korisnik može odabrati taj gumb izbora istovremenim pritiskom tipki ALT + P.

Napomena Za uključivanje znaka & u natpis bez stvaranja pristupne tipke, napišite dva znaka & (&&). U natpisu će biti prikazan samo jedan znak & i nijedan karakter neće biti podvučen.

Onemogućavanje gumba izbora

Kako bi onemogućili gumb izbora, postavite njegovo svojstvo Enabled na False. Tijekom rada aplikacije, taj gumb izbora će izgledati zasjenjen, što znači da je nedostupan.

Vizualno poboljšanje kontrole gumba izbora

Izgled kontrole gumba izbora može biti poboljšan promjenom vrijednosti svojstva Style te zatim korištenjem svojstava Picture, DownPicture i DisabledPicture.

Korištenje kontrole okvira za sliku

Kontrola okvira za sliku (Picture Box) koristi se za prikazivanje grafike, djelovanje kao spremnik drugih kontrola, te za prikazivanje izlaznih rezultata grafičkih postupaka ili teksta korištenjem postupka Print.

Slika 7.40 Kontrola okvira za sliku

Kontrola okvira za sliku slična je kontroli slike po tome što obje mogu biti upotrijebljene za prikaz grafike u vašoj aplikaciji – obje podržavaju iste grafičke formate. Međutim, kontrola okvira za sliku sadržava funkcionalnost koju nema kontrola slike, na primjer: sposobnost da djeluje kao spremnik za druge kontrole i podrška grafičkim postupcima.

Za više informacija Pogledajte "Rad s kontrolom okvira za sliku" u 3. poglavlju "Forme, kontrole i izbornici" za prikaz upotrebe okvira za sliku.

Podržani grafički formati

Kontrola okvira za sliku može prikazivati datoteke slika u bilo kojem od sljedećih formata: bitmapirane slike, kursori, ikone, metadatoteke, poboljšane metadatoteke, te JPEG ili GIF datoteke.

Za više informacija Pogledajte "Korištenje kontrole slike", ranije u ovom poglavlju, za detaljne opise ovih grafičkih formata.

Učitavanje grafike u kontrolu okvira za sliku

Slike mogu biti učitane u kontrolu okvira za sliku tijekom izrade aplikacije odabirom svojstva Picture u prozoru s svojstvima kontrole, ili tijekom rada aplikacije korištenjem svojstva Picture i funkcije LoadPicture.

```
Set Picture1.Picture = _
LoadPicture("c:\Windows\Winlogo.cur", vbLPLarge, vbLPColor)
```

Možete poželjeti upotrijebiti datoteke ikona (.ico) ili kursora (.cur) koje sadrže odvojene slike raznih veličina i dubina boja za podršku nizu sredstava prikaza. Postavke funkcije LoadPicture omogućuju vam odabir slika određene dubine boja i veličine iz .ico ili .cur datoteke. U slučajevima kad točan par traženim postavkama nije dostupan, funkcija LoadPicture učitava sliku koja najviše odgovara traženoj.

Za brisanje grafike iz kontrole okvira za sliku, upotrijebite funkciju LoadPicture bez određivanja imena datoteke. Na primjer:

```
Set Picture1.Picture = LoadPicture
```

To će očistiti kontrolu okvira za sliku čak i ako je grafika učitana u svojstvo Picture tijekom izrade aplikacije.

Korištenje odlagališta

Grafiku također možete dodati u kontrolu okvira za sliku tijekom izrade aplikacije tako da je ulijepite iz druge aplikacije. Na primjer, možda želite dodati bitmapiranu sliku koja je stvorena u aplikaciji Windows Paint. Jednostavno kopirajte sliku u odlagalište, odaberite kontrolu okvira za sliku, te upotrijebite prečicu tipkama CTRL + V ili naredbu Paste iz izbornika Edit.

Promjena veličine slike

U pravilu, slike se učitavaju u okvir za sliku u originalnoj veličini, što znači da će dio slike biti odrezan ako je slika veća od kontrole – kontrola okvira za sliku ne pruža klizne trake. Kako bi kontrola okvira za sliku automatski promijenila svoju veličinu tako da može prikazati cijelu sliku, postavite njezino svojstvo AutoSize na True. Kontrola će kod učitavanja slike prilagoditi svoju veličinu slici – povećat će se ili smanjiti.

Za razliku od kontrole slike, kontrola okvira za sliku ne može razvući sliku tako da odgovara veličini kontrole.

Za više informacija Pogledajte "Primjer kontrola kliznih traka: stvaranje pomičnog grafičkog prozora", ranije u ovom poglavlju, za informacije o korištenju okvira za sliku u stvaranju pomičnog grafičkog prozora.

Korištenje kontrole okvira za sliku kao spremnika

Kontrolu okvira za sliku možete upotrijebiti kao spremnik za druge kontrole. Na primjer, budući da se okvir za sliku može postaviti u unutarnje područje MDI forme, često se koristi za ručno stvaranje alatne trake ili statusne trake.

Za više informacija Pogledajte "Stvaranje alatne trake" u 6. poglavlju "Stvaranje korisničkog sučelja", za više informacija o korištenju kontrole okvira za sliku kao spremnika za druge kontrole.

Grafički postupci

Okviri s sliku, kao i forme, mogu biti upotrebljeni za prihvaćanje izlaznih rezultata grafičkih postupaka kao što su Circle, Line i Point. Na primjer, možete upotrijebiti postupak Circle za crtanje kruga u okviru za sliku postavljanjem svojstva AutoRedraw kontrole na True.

```
Picture1.AutoRedraw = True
Picture1.Circle (1200, 1000), 750
```

Postavljanje svojstva AutoRedraw na True omogućuje crtanje izlaznih rezultata tih postupaka na kontroli te automatsko ponovno iscrtavanje kad kontrola okvira za sliku promijeni veličinu ili se ponovno prikaže nakon što je bila skrivena drugim objektom.

Za više informacija Pogledajte "Korištenje grafičkih postupaka" u 12. poglavlju "Rad s tekstom i grafikom", za više informacija o korištenju kontrole okvira za sliku s grafičkim postupcima.

Korištenje postupka Print

Kontrolu okvira s slikom možete upotrijebiti za ispis teksta korištenjem postupka Print te postavljanjem svojstva AutoRedraw na True. Na primjer:

```
Picture1.Print "Nekakav tekst"
```

Kad koristite postupak Print možete također promijeniti stil i veličinu pisma ili upotrijebiti svojstva CurrentX, CurrentY, Height i Width za poravnavanje teksta unutar okvira za sliku.

Za više informacija Pogledajte 12. poglavlje "Rad s tekstom i grafikom".

Korištenje kontrole udaljenih podataka

Kontrola udaljenih podataka (Remote Data) izvršava pristup podacima korištenjem Microsoftovih objekata udaljenih podataka (Microsoft Remote Data Objects). Ta tehnologija daje vam neograničen pristup većini standardnih oblika baza podataka i dopušta vam stvaranje aplikacija svjesnih podataka bez pisanja ikakvog programskog koda. Kontrola udaljenih podataka prilagođena je većim bazama podataka tipa klijent-poslužitelj, uključujući ODBC baze podataka (Open Database Connectivity) kao što su Microsoft SQL poslužitelj i Oracle.

Treba zapamtiti da kontrola udaljenih podataka ne radi s podacima tipa Image.

Napomena U Visual Basic su uključene i kontrola udaljenih podataka i kontrola podataka zbog sukladnosti prema naposljed. Unatoč tome, zbog fleksibilnosti ActiveX objekata podataka (ActiveX Data Objects, ADO), preporučljivo je da novu aplikaciju baze podataka stvarate korištenjem kontrole ADO podataka. Za više detalja, pogledajte "Korištenje kontrole ADO podataka" ranije u ovom poglavlju.

Kontrola podataka, kontrola udaljenih podataka i kontrola ADO podataka su sadržajno slične: sve tri su "kontrole podataka" koje povezuju izvor podataka s kontrolom povezanom s podacima. Sve tri također dijele isti izgled – skup od četiri gumba koji omogućuju korisniku trenutan skok na početak niza slogova, kraj niza slogova, te pomicanje prema naprijed i naposljed kroz niz slogova.

Stvaranje jednostavne aplikacije kao baze podataka kontrolom udaljenih podataka

Sljedeći postupci stvorit će jednostavnu aplikaciju baze podataka korištenjem kontrole udaljenih podataka.

Kako stvoriti aplikaciju baze podataka korištenjem kontrole udaljenih podataka

1. Stvorite **kontrolu udaljenih podataka** na formi (ToolTip ikone je "MSRDC").

   Ako kontrola udaljenih podataka nije u alatnom okviru, pritisnite CTRL + T za prikaz dijaloškog okvira Components. U dijaloškom okviru **Components**, kliknite **Microsoft Remote Data Control**. Kliknite **OK** za dodavanje te kontrole u alatni okvir.

2. Kliknite **kontrolu udaljenih podataka** kako bi je odabrali, i pritisnite F4 za prikaz prozora s svojstvima.

3. U prozoru s svojstvima, postavite svojstvo **DataSourceName** na ime baze podataka s kojom se želite povezati.

4. U prozoru s svojstvima, kliknite svojstvo **SQL** i upišite SQL izraz, na primjer:

   ```
   SELECT * FROM Products WHERE AuthorID = 72
   ```

   Kad pristupate tablici uvijek bi trebali uključiti klauzulu WHERE. Ako to ne učinite, zaključat ćete cijelu tablicu, što će biti velika zapreka ostalim korisnicima.

5. Kreirajte **kontrolu okvira s tekstom** na formi.

6. Kliknite **kontrolu okvira s tekstom** kako bi je odabrali, i u prozoru s svojstvima postavite njezino svojstvo **DataSource** na **kontrolu udaljenih podataka**.

7. U prozoru s svojstvima, u svojstvo **DataField** upišite ime polja u bazi podataka kojeg želite vidjeti ili mijenjati.
8. Ponovite korake 5, 6 i 7 za svako dodatno polje kojem želite pristupiti.
9. Pritisnite F5 za pokretanje aplikacije.

Određivanje svojstava kontrole udaljenih podataka vezanih s podacima

Sljedeća svojstva koja se odnose na podatke mogu biti određena tijekom izrade aplikacije. Ova lista savjetuje logičan red određivanja svojstava:

1. **Connect** – Svojstvo Connect je string koji sadrži sve postavke potrebne za uspostavljanje veze. Parametri proslijeđeni ovom stringu ovisni su o upravljačkim programima. Na primjer, ODBC upravljački programi dopuštaju stringu da sadrži goniča, bazu podataka, korisničko ime i zaporku.
2. **UserName** – Identificira korisnika zaštićenoj bazi podataka. Korisnik također mora pružiti i valjanu zaporku koju prepoznaje sustav upravljanja bazom podataka. Ime korisnika može se nalaziti i u svojstvu Connect, i u tom slučaju je ovdje nepotrebno.
3. **Password** – Zajedno s imenom korisnika, zaporka dopušta korisniku pristup zaštićenim podacima. Zaporka se može nalaziti i u svojstvu Connect, i u tom slučaju je ovdje nepotrebna.
4. **SQL** – Svojstvo SQL sadrži SQL izraz koji se koristi za dohvaćanje niza rezultata. Veličina skupa rezultata može odrediti hoćete li koristiti pokazivač na strani klijenta ili poslužitelja. Na primjer, mali skup rezultata može biti upravljan klijentskim pokazivačem.
5. **RowSetSize** – Određuje broj redova koji se vraćaju u skupu rezultata, ako je pokazivač ključnog tipa. Ovaj broj možete fino prilagoditi izvorima računala (memorija) za dobivanje na učinku.
6. **ReadOnly** – Određuje hoće li se podaci zapisivati. Ako zapisivanje podataka nije potrebno, postavljanje ovog svojstva na True može povećati učinak.
7. **CursorDriver** – određuje položaj i tip upravljačkog goniča. Određivanje ovog svojstva utječe na određivanje ostalih svojstava. Na primjer, odabir pokazivača tipa ODBC na strani klijenta može vam dati povećanje učinka sve dok je skup rezultata mali.
8. **LockType** – Svojstvo LockType određuje kako će podaci biti zaključani kad drugi pokušaju promijeniti podatke. Ako ne očekujete da će drugi mijenjati podatke (dok radite s tim podacima) postavite svojstvo LockType na “optimistic” - ostavljajući podatke slobodne kako bi ih drugi mogli pregledavati i mijenjati. Ako ovo svojstvo postavite na “pesimistic”, drugi ne mogu pristupiti podacima dok im vi pristupate.
9. **BOFAction**, **EOFAction** – Ova dva svojstva određuju što će se dogoditi kad je kontrola na početku i kraju pokazivača. Izbori uključuju ostajanje na početku ili kraju, pomicanje na prvi ili posljednji slog, ili dodavanje novog sloga (samo na kraju).

10. **ResultSetType** – Određuje je li pokazivač statičnog ili ključnog tipa.
11. **KeySetSize** – Ako je pokazivač ključnog tipa, možete poboljšati veličinu vraćenog skupa rezultata svojstvom KeySetSize.
12. **LoginTimeout** – Određuje broj sekundi čekanja prije vraćanja pogreške.
13. **MaxRows** – Određuje koliko će velik biti pokazivač. Kako ćete to odrediti ovisi o veličini zapisa koje dohvaćate te o izvorima dostupnim na vašem računalu (memorija). Veliki zapis (s puno stupaca i velikim stringovima) uzet će više izvora od malog sloga. Svojstvo MaxRows treba shodno tome biti smanjeno.
14. **Options** – Određuje hoće li kontrola neusklađeno izvoditi upite ili neće. Upotrijebite neusklađene operacije kad očekujete da će upitu trebati više od nekoliko minuta da se izvede.
15. **Prompt** – Kad kontrola udaljenih podataka otvori vezu temeljenu na parametrima kontrole, od svojstva Connect se očekuje da sadrži informacije dostatne za uspostavljanje veze. Ako nisu pružene informacije poput imena izvora podataka, imena korisnika ili zaporke, upravitelj ODBC goniča prikazuje jedan ili više dijaloških okvira za dobivanje informacija od korisnika. Ako ne želite prikazivanje tih dijaloških okvira, postavite svojstvo Prompt tako da onemogući tu osobinu.
16. **QueryTimeout** – Određuje koliko će se sekundi čekati završetak upita prije vraćanja pogreške.
17. **BatchSize** – Ovo svojstvo određuje koliko će izraza biti skupno poslano – ako gonič dopušta skupne izraze.

Korištenje kontrole lika

Kontrola lika (Shape) koristi se za stvaranje sljedećih predodređenih likova na formama, okvirima ili okvirima za sliku: pravokutnik, kvadrat, elipsa, krug, pravokutnik s zaobljenim kutovima ili kvadrat s zaobljenim kutovima.

Slika 7.41 Kontrola lika

Svakom liku kojeg stvorite na formi možete odrediti tip lika, boju, stil popune, boju ruba i stil ruba.

Za jednostavne primjene, kontrola lika omogućuje vam stvaranje niza raznih likova bez pisanja koda. Za naprednije mogućnosti trebate upotrijebiti postupke Line i Circle.

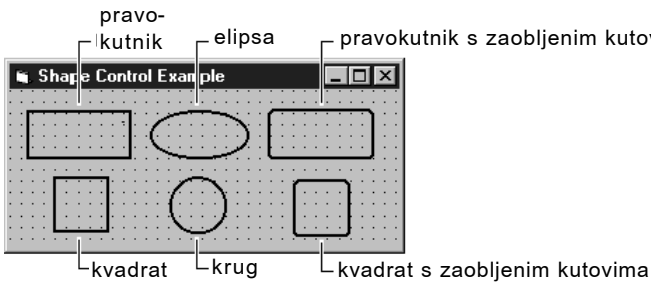
Za više informacija Pogledajte "Korištenje grafičkih postupaka" u 12. poglavlju "Rad s tekstom i grafikom", za više informacija o crtanju linija, pravokutnika i popunjenih okvira tijekom rada aplikacije korištenjem postupka Line ili ta više informacija o crtanju krugova, elipsi i lukova tijekom rada aplikacije korištenjem postupka Circle.

Predodređeni likovi

Svojstvo Shape kontrole lika omogućuje vam šest predodređenih likova. Sljedeća tablica ispisuje sve predodređene likove, njihove vrijednosti i odgovarajuće konstante Visual Basica:

lik	stil	konstanta
pravokutnik	0	vbShapeRectangle
kvadrat	1	vbShapeSquare
elipsa	2	vbShapeOval
krug	3	vbShapeCircle
pravokutnik s zaobljenim kutovima	4	vbShapeRoundedRectangle
kvadrat s zaobljenim kutovima	5	vbShapeRoundedSquare

Slika 7.42 Predodređeni likovi



Stilovi popunjavanja i linije

Svojstva FillStyle i BorderStyle možete upotrijebiti za određivanje stila popunjavanja i ruba bilo koji od likova kojeg nacrtate na formi.

Svojstvo FillStyle, kao i svojstvo Style, pruža vam niz prije određenih uzoraka za stil popunjavanja. Oni uključuju puno (solid), prozirno (transparent), s vodoravnim linijama (horizontal line), s okomitim linijama (vertical line), koso prema gore (upward diagonal), koso prema dolje (downward diagonal), prekriženo (cross) i dijagonalno prekriženo (diagonal cross) popunjavanje.

Svojstvo BorderStyle pruža vam niz prije određenih stilova ruba. Oni uključuju: proziran (transparent), pun (solid), s nizom crta (dash), s nizom točaka (dot), s nizom crta-točka (dash-dot), s nizom crta-točka-točka (dash-dot-dot) te popunjen iznutra (inside solid) rub.

Za više informacija Svojstva FillStyle i BorderStyle pružaju vam konstante koje predstavljaju prije spomenute stilove. Pogledajte "Svojstvo FillStyle" i "Svojstvo BorderStyle" u dijelu *Microsoft Visual Basic 6.0 Controls Reference* biblioteke *Microsoft Visual Basic 6.0 Reference Library* za više informacija.

Određivanje atributa boje

Svojstva BackColor i FillColor omogućuju vam dodavanje boje liku i njegovom rubu.

Tijekom izrade aplikacije, možete odrediti boje popune i ruba biranjem odgovarajućeg svojstva u prozoru s svojstvima te odabirom boje iz raspoložive boje ili sistemskih boja.

Kako bi odredili boje tijekom izvođenja aplikacije, upotrijebite konstante boja Visual Basica (vbGreen, na primjer), konstante sistemskih boja (vbWindowsBackground, na primjer) ili funkciju RGB za određivanje boje popunjavanja.

Napomena Kad su svojstva FillStyle ili BackStyle postavljena na 1 (prozirno, transparent), svojstva FillColor i BackColor se zanemaruju.

Za više informacija Pogledajte "Funkcija RGB" u biblioteci *Microsoft Visual Basic 6.0 Language Reference* za informacije o određivanju RGB boja. Također, pogledajte 12. poglavlje "Rad s tekstom i grafikom" za detaljne informacije o stvaranju grafike u Visual Basicu.

Crtanje likova na formi

Kontrolu lika možete upotrijebiti za crtanje pravokutnika (s pravilnim ili zaobljenim kutovima), kvadrata (s pravilnim ili zaobljenim kutovima), elipsi i krugova na formi.

Kako nacrtati lik na formi

1. U alatnom okviru odaberite **kontrolu lika**.

   Kad se pokazivač pomakne na formu, mijenja se u križni pokazivač.

2. Kliknite i povucite križni pokazivač kako bi napravili željenu veličinu lika.
3. U prozoru s svojstvima odredite svojstvo **Shape**.
4. U okviru **Settings** odaberite željeni stil.

Likovi mogu mijenjati veličinu kao i sve ostale kontrole, odabirom i povlačenjem kontrole na željenu veličinu, ili određivanjem svojstava Height i Width.

Korištenje kontrole okvira s tekstom

Kontrola okvira s tekstom (Text Box) koristi se za prikaz informacija koje je unio korisnik tijekom rada aplikacije, ili informacija dodijeljenih svojstvu Text kontrole tijekom izrade ili izvođenja aplikacije.

Slika 7.43 Kontrola okvira s tekstom

Općenito, kontrola okvira s tekstom trebala bi biti upotrijebljena za tekst koji se može obrađivati, iako ga možete učiniti i tekstom samo za čitanje postavljanjem svojstva Locked na True. Okviri za tekst vam također omogućuju prikaz više linija, prijelom teksta prema veličini kontrole te dodavanje temeljnih oblikovanja teksta.

Svojstvo Text

Tekst upisan u kontrolu okvira s tekstom sadržan je u svojstvu Text. U pravilu, u okvir s tekstom možete unijeti do 2048 karaktera. Ako svojstvo MultiLine ove kontrole postavite na True, možete unijeti do 32 kilobajta teksta.

Oblikovanje teksta

Kad tekst prekorači granice kontrole, možete dopustiti kontroli da automatski prelomi tekst postavljanjem svojstva MultiLine na True te dodati klizne trake postavljanjem svojstva ScrollBars za dodavanje vodoravne ili okomite klizne trake, ili obje. Međutim, automatski prijelom teksta bit će nedostupan ako dodate vodoravnu traku za pomicanje jer se vodoravno područje obrađivanja povećava prisutnošću klizne trake.

Kad je svojstvo MultiLine postavljeno na True, možete također prilagoditi poravnanje teksta na lijevo poravnavanje, centriranje ili desno poravnavanje. Tekst je u pravilu lijevo poravnan. Ako je svojstvo MultiLine postavljeno na False, postavke svojstva Alignment nemaju učinka.

Za više informacija Pogledajte "Rad s okvirima s tekstom" u stalnoj pomoći, za prikaz svojstava MultiLine, ScrollBar i Alignment.

Odabir teksta

Možete kontrolirati mjesto ubacivanja i ponašanje odabranog teksta u okviru s tekstom svojstvima SelStart, SelLength i SelText.

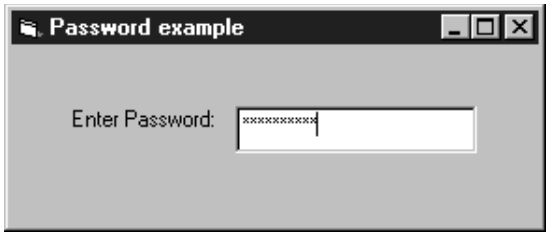
Za više informacija Pogledajte "Rad s okvirima s tekstom" u 3. poglavlju "Forme, kontrole i izbornici", za prikaz svojstava SelStart, SelText i SelLength.

Stvaranje okvira s tekstom za zaporku

Okvir za zaporku je okvir s tekstom koji omogućuje korisniku upis njegove zaporke dok prikazuje zamjenske karaktere, kao što su zvjezdice (asterisci, *). Visual Basic pruža dva svojstva okvira s tekstom, PasswordChar i MaxLength, koja pojednostavljuju stvaranje okvira s tekstom za unos zaporke.

Svojstvo PasswordChar određuje karakter koji će biti prikazan u okviru s tekstom. Na primjer, ako želite prikazati zvjezdice u okviru za zaporku, postavite * za vrijednost svojstva PasswordChar u prozoru s svojstvima. Neovisno o karakteru koji korisnik upiše u okvir s tekstom, bit će prikazana zvjezdica, kao što je prikazano na slici 7.44.

Slika 7.44 Primjer zaporke



Svojstvom MaxLength određujete koliko karaktera može biti upisano u okvir s tekstom. Nakon prekoračenja vrijednosti ovog svojstva, računalo ispušta pisak i okvir s tekstom ne prihvaća daljnje karaktere.

Poništavanje upisa u okvir s tekstom

Možete upotrijebiti događaj KeyPress za ograničenje ili pretvaranje karaktera koji su upisani. Događaj KeyPress koristi jedan argument, *keyascii*. Taj argument je cjelobrojna vrijednost koja predstavlja jednaku brojčanu (ASCII) vrijednost karaktera upisanog u okvir s tekstom.

Sljedeći primjer pokazuje kako poništiti karaktere dok se upisuju. Ako upisani karakter nije unutar određenog opsega, procedura ga poništava postavljajući KeyAscii na 0. Okvir s tekstom za ovaj primjer ima ime txtUpisBroja, a procedura sprječava okvir s tekstom od prihvaćanja bilo kojeg karaktera osim brojaka. Usporedite KeyAscii direktno s brojčanim (Asc) vrijednostima raznih karaktera.

```
Private Sub txtUpisBroja_KeyPress(KeyAscii As Integer)
  If KeyAscii < Asc("0") Or KeyAscii > Asc("9") Then
    KeyAscii = 0          ' poništavanje karaktera.
    Beep                  ' Signalni zvuk pogreške.
  End If
End Sub
```

Za više informacija Pogledajte "Odgovor na događaje tipkovnice" u 11. poglavlju "Odgovor na događaje miša i tipkovnice", za više informacija o događaju KeyPress.

Stvaranje okvira s tekstom samo za čitanje

Možete upotrijebiti svojstvo Locked da spriječite korisnike u editiranju sadržaja okvira s tekstom. Postavite svojstvo Locked na True kako bi korisnicima dopustili pomicanje i označavanje teksta u okviru s tekstom bez dopuštanja mijenjanja. Sa svojstvom Locked postavljenim na True, naredba Copy radit će unutar okvira s tekstom, ali naredbe Cut i Paste neće. Svojstvo Locked utječe samo na *akcije korisnika* tijekom rada aplikacije. Sadržaj okvira s tekstom i dalje možete mijenjati *programski* tijekom izvođenja promjenom svojstva Text okvira s tekstom.

Ispis znaka navodnika u stringu

Znakovi navodnika (" ") se ponekad pojavljuju u tekstualnom nizu.

```
Ona je rekla "Zaslužio si čašćenje!"
```

Budući da su stringovi koji se dodjeljuju varijabli ili svojstvu zatvoreni znakovima navodnika (" "), morate ubaciti dodatan niz znakova navodnika s svaki navodnik koji se prikazuje u stringu. Visual Basic tumači dva znaka navodnika zaredom kao umetnuti znak navodnika.

Na primjer, kako bi ispisali prošli string, upotrijebite sljedeći programski kod:

```
Text1.Text = "Ona je rekla ""Zaslužio si čašćenje!"" "
```

Kako bi postigli isti učinak, možete upotrijebiti ASCII karakter (34) za znak navodnika:

```
Text1.Text = "Ona je rekla " & Chr(34) _
+ "Zaslužio si čašćenje!" & Chr(34)
```

Korištenje kontrole mjerača vremena

Kontrole mjerača vremena (Timer) odgovaraju na tijek vremena. Neovisne su o korisniku, i možete ih programirati tako da poduzmu akcije u pravilnim vremenskim razmacima. Uobičajen odgovor je provjera sistemskog sata za utvrđivanje je li vrijeme za izvođenje nekog posljedatka. Mjerači vremena također su korisni za druge vrste pozadinskih obrada.

Slika 7.45 Kontrola mjerača vremena

Svaka kontrola mjerača vremena ima svojstvo Interval koje određuje broj milisekundi koje će proći između jednog događaja Timer do drugog. Osim ako nije isključen, mjerač vremena nastavlja s prihvaćanjem događaja (odgovarajuće nazvanog događaj Timer) u otprilike jednakim vremenskim razmacima.

Svojstvo Interval ima nekoliko ograničenja koje morate uzeti u obzir pri programiranju kontrole mjerača vremena:

- Ako vaša aplikacija ili druga aplikacija ima velike zahtjeve u sustavu – kao duge petlje, temeljita proračunavanja, ili pristup pogonskom uređaju, mreži ili priključku – možda neće dobivati događaje mjerača tako često kako određuje svojstvo Interval.
- Vremenski razmak može biti između 0 i 64.767, uključivo, što znači da čak i najdulji vremenski razmak ne može biti puno dulji od jedne minute (oko 64,8 sekundi).
- Ne postoji jamstvo za istek vremenskog razmaka točno na vrijeme. Kako bi osigurali točnost, mjerač vremena bi trebao provjeravati sistemski sat kad je to potrebno, umjesto unutarnjeg praćenja nakupljenog vremena.
- Sustav proizvodi 18 otkucaja sata u sekundi – pa iako se svojstvo Interval mjeri u milisekundama, stvarna preciznost vremenskog razmaka nije veća od jedne osamnaestine sekunde.

Svaka kontrola mjerača vremena mora biti povezana s formom. Zbog toga, da bi stvorili aplikaciju s mjeračem vremena, morate stvoriti bar jednu formu (iako ta forma ne mora biti vidljiva ako je ne trebate za neku drugu namjenu).

Napomena Riječ "timer" koristi se na više načina u Visual Basicu, a svaki je blisko povezan s radom kontrole mjerača vremena. Osim imena kontrole i tipa kontrole "timer" se koristi i za događaj Timer te za funkciju Timer.

Primjer aplikacije: Alarm.vbp

Tehnike rada s kontrolom mjerača vremena uključene su u primjeru aplikacije Alarm (Alarm.vbp) koja se nalazi u direktoriju Samples.

Postavljanje kontrole mjerača vremena na formu

Postavljanje kontrole mjerača vremena na formu jednako je postavljanju bilo koje kontrole. Kliknite gumb mjerača vremena u alatnom okviru i stvorite kontrolu na formi.

Mjerač vremena pojavljuje se na formi samo tijekom izrade aplikacije tako da ga možete odabrati, vidjeti njegova svojstva, i napisati potprograme događaja za njega. Tijekom izvođenja aplikacije, mjerač vremena je nevidljiv i njegov položaj i veličina su nebitni.

Pokretanje kontrole mjerača vremena

Kontrola mjerača vremena ima sva ključna svojstva.

svojstvo	postavka
Enabled	Ako želite početak rada mjerača vremena čim se forma učita, postavite ovo svojstvo na True. Inače, ostavite ovo svojstvo postavljeno na False. Možete odabrati da vanjski događaj (kao klik naredbenog gumba) pokrene rad mjerača vremena.
Interval	Broj milisekundi između događaja Timer.

Uočite da je svojstvo Enabled mjerača vremena različito od svojstva Enabled ostalih objekata. Kod većine objekata, svojstvo Enabled određuje hoće li objekt moći odgovoriti na događaj kojeg je uzrokovao korisnik. Kod kontrole mjerača vremena, postavljanje svojstva Enabled na False obustavlja djelovanje mjerača vremena.

Zapamtite da je događaj Timer *periodičan*. Svojstvo Interval ne određuje "kako dugo" koliko određuje "kako često". Duljina vremenskog razmaka treba ovisiti o preciznosti koju želite. Budući da postoje neke ugrađene mogućnosti za izazivanje pogreške, neka vaš vremenski razmak bude jednak polovici željenog iznosa preciznosti.

Napomena Što se češće stvara događaj Timer, to će više procesorskog vremena biti iskorišteno za odgovor na događaj. To može usporiti sveukupno izvođenje. Ne postavljajte osobito male vremenske razmake osim ako ih ne trebate.

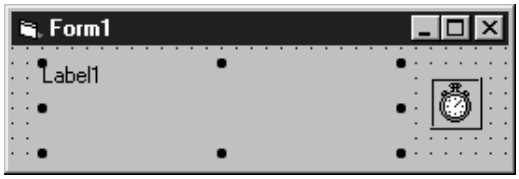
Primjer kontrole mjerača vremena: Odgovor na događaj Timer

Kad istekne vremenski razmak kontrole mjerača vremena, Visual Basic stvara događaj Timer. Tipično, vaš odgovor na taj događaj je provjera nekih općih uvjeta, kao što je sistemski sat.

Digitalni sat je vrlo jednostavna, ali i vrlo korisna aplikacija koja uključuje kontrolu mjerača vremena. Jednom kad shvatite kako aplikacija radi, možete je proširiti da radi kao sat s alarmom, štoperica ili neki drugi uređaj za mjerenje vremena.

Aplikacija Digitalni sat uključuje mjerač vremena i kontrolu natpisa s rubom. Tijekom izrade, aplikacija izgleda kao na slici 7.46.

Slika 7.46 Aplikacija Digitalni sat



Tijekom izvođenja aplikacije, mjerač vremena je nevidljiv. Sljedeća tablica ispisuje vrijednosti svojstava za aplikaciju Digitalni sat.

kontrola	svojstvo	postavka
Label1	BorderStyle	Fixed Single
Timer1	Interval	500 (pola sekunde)
Timer1	Enabled	True

Jedina procedura u ovoj aplikaciji je potprogram događaja za mjerač vremena:

```
Private Sub Timer1_Timer()
  If lblTime.Caption <> CStr(Time) Then
    lblTime.Caption = Time
  End If
End Sub
```

Procedura prikazuje sistemsko vrijeme pozivom ugrađene funkcije Time. Ova funkcija vraća vrijednost tipa Variant koja sadrži trenutno vrijeme kao vrijednost datum/vrijeme (VarType 7). Kad je dodijelite varijabli stringa ili svojstvu, kao što je svojstvo Caption u ovom slučaju, Visual Basic je pretvara u string korištenjem oblika vremena određenog u kontrolnom panelu. Ako ju želite prikazati koristeći drugačiji oblik, možete upotrijebiti funkcije Format.

Za više informacija Pogledajte "Funkcija Format" u biblioteci *Microsoft Visual Basic 6.0 Language Reference*.

Svojstvo Interval mjerača vremena postavljeno je na 500, slijedeći pravilo o određivanju vremenskog razmaka na polovicu najkraćeg perioda kojeg želite razlikovati (jedna sekunda u ovom slučaju). To može uzrokovati ažuriranje kontrole natpisa od programskog koda mjerača vremena dva puta u sekundi. To može biti rasipno i može uzrokovati vidljiva titranja, tako da kod ispituje je li trenutno vrijeme različito od prikazanog u kontroli natpisa prije nego što promijeni natpis.

Izgled Digitalnog sata možete prilagoditi bez potrebe za pisanjem bilo koje dodatne naredbe. Na primjer, možete odabrati drugačije pismo za kontrolu natpisa ili možete promijeniti svojstvo BorderStyle forme.

8. POGLAVLJE

Više o programiranju

Ovo poglavlje ide dalje od temelja programiranja u Visual Basicu i upoznaje vas s nizom osobina koje olakšavaju stvaranje moćnih i fleksibilnih aplikacija.

Na primjer, možete učitati više projekata tijekom jednog rada u okolini programiranja, raditi s postavkama registara Windowsa, ili selektivno prevoditi određene dijelove svoje aplikacije.

Iznad temelja pisanja programskog koda, Visual Basic pruža niz elemenata programskog jezika koji poboljšavaju vaš kod. Posljednje četiri teme ovog poglavlja raspravljaju o tim elementima jezika: korisnički određenim tipovima, nabrojenim konstantama, matricama i zbirkama.

Sadržaj

- Rad s više projekata
- Upravljanje postavkama aplikacije
- Korištenje uvjetnog prevođenja
- Rad s datotekama izvora
- Rad s predlošcima
- Rad s prekidačima naredbene linije
- Prevođenje vašeg projekta u strojni kod
- Stvaranje vlastitih tipova podataka
- Korištenje nabrajanja za rad s nizovima konstanti
- Korištenje zbirki umjesto matrica

Rad s više projekata

Možete stvoriti puno aplikacija radom s jednim projektom. Međutim, kako vaša aplikacija postaje sve složenija, možete poželjeti raditi istovremeno s više projekata u okolini programiranja. Na primjer, možete upotrijebiti jedan projekt za stvaranje izvršne datoteke aplikacije, a drugi projekt će poslužiti kao “privremeno odlagalište” za ispitivanje programskog koda prije nego što ga dodate aplikaciji.

Možete dodati novi ili postojeći projekt tijekom vašeg trenutnog rada tako da ga dodate u *projektnu grupu*. Nakon toga možete spremiti projektnu grupu i nastaviti raditi s njom sljedeći put. Možete otvoriti projektnu grupu ili zaseban projekt iz projektne grupe, te dodati projektnu grupu ili pojedine projekte drugoj projektnoj grupi.

U projektnoj grupi, jedan izvršni projekt služi kao početni projekt. Kad se otvori projektna grupa i odaberete Start iz izbornika Run, kliknete gumb Start na alatnoj traci, ili pritisnete F5, Visual Basic će pokrenuti početni projekt.

U verzijama Professional ili Enterprise, projektne grupe možete upotrijebiti za stvaranje i ispravljanje pogrešaka u aplikacijama s više dijelova. Na primjer, možete stvoriti i ispravljati projektne grupe koje sadrže standardne izvršne projekte, izvršne projekte tipa ActiveX, projekte s dinamičkim ActiveX bibliotekama, ili projekte ActiveX kontrola. Za više informacija, pogledajte "Stvaranje ActiveX sastavnih dijelova" u dijelu *Microsoft Visual Basic 6.0 Component Tools Guide* biblioteke *Microsoft Visual Basic 6.0 Reference Library*.

Dodavanje ili micanje projekta

Kad stvorite novi projekt ili na početku rada ili odabirom naredbe New Project u izborniku File, Visual Basic automatski stvara projektnu grupu za njega. U tu grupu zatim možete dodati dodatne nove ili postojeće projekte.

Kako dodati novi projekt u projektnu grupu

- Kliknite gumb **Add Project** na alatnoj traci ili odaberite **Add Project** u izborniku **File**.

  U verziji Learning Visual Basica, Visual Basic automatski dodaje novi izvršni projekt u projektnu grupu. U verzijama Professional i Enterprise, tip projekta kojeg želite dodati možete odabrati u pomoćnom izborniku gumba **Add Project** ili među ikonama na kartici **New** dijaloškog okvira **Add Project**.

Kako dodati postojeći projekt u projektnu grupu

1. U izborniku **File** odaberite **Add Project**.

   Visual Basic će prikazati dijaloški okvir **Add Project**.

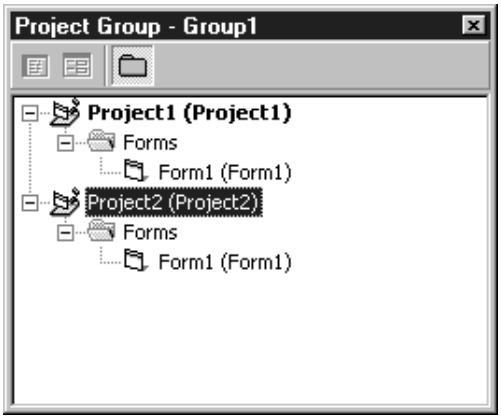
2. Kliknite karticu **Existing**.

3. Odaberite datoteku projekta, i odaberite **Open**.

   Visual Basic će dodati odabrani projekt u projektnu grupu.

Visual Basic prikazuje višestruke projekte u projektnom prozoru hijerarhijskim pregledom. Svaki projekt pojavljuje se na najvišoj razini, s projektnim formama, modulima, kontrolama, stranicama svojstava ili objektima dokumenata grupiranim ispod njega na hijerarhijski način.

Slika 8.1 Projektni prozor s više projekata



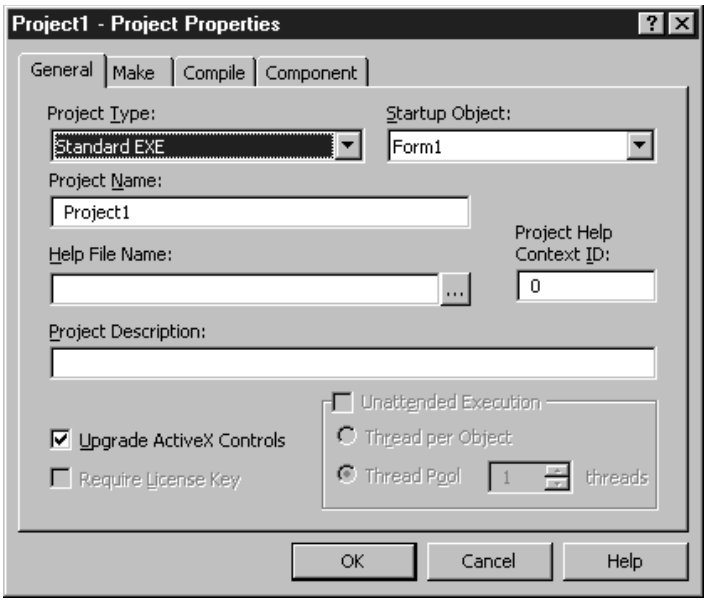
Kako maknuti projekt iz projektne grupe

1. Odaberite ime projekta u projektnom prozoru.
2. U izborniku **File** odaberite **Remove Project**.

Visual Basic će maknuti odabrani projekt iz projektne grupe.

U Professional i Enterprise verzijama Visual Basica, možete promijeniti tip projekta odabirom naredbe Project Properties u izborniku Project, te promjenom izbora Project Type na kartici General dijaloškog okvira Project Properties.

Slika 8.2 Kartica General u dijaloškom okviru Project Properties



Određivanje početnog projekta

Budući da projektna grupa sadrži više projekata, Visual Basic treba znati koji će projekt pokrenuti kad odaberete Start iz izbornika Run, kliknete gumb Start na alatnoj traci ili pritisnete F5. U pravilu, Visual Basic pokreće prvi izvršni (.exe) projekt koji je dodan projektnoj grupi. Međutim, možete odrediti drugi početni sastavni dio grupe.

Kako odrediti početni sastavni dio

1. U projektnom prozoru, odaberite projekt.
2. Kliknite projekt desnom tipkom miša i u pomoćnom izborniku odaberite **Set as Start Up**.

Visual Basic prikazuje ime početnog projekta podebljanim pismom u projektnom prozoru.

Za više informacija Ispravljanje višestrukih projekata raspravljeno je u 7. poglavlju pomoći "Ispravljanje, ispitivanje i razvijanje sastavnih dijelova", u "Stvaranje ActiveX sastavnih dijelova" vodiča *Microsoft Visual Basic 6.0 Component Tools Guide* uključenog s Professional i Enterprise verzijama Visual Basica.

Upravljanje postavkama aplikacije

U Microsoft Windowsima 3.1 i ranijim verzijama Windowsa, postavke aplikacija kao što su položaj prozora, korištene datoteke, te ostale stavke bile su obično spremljene u .ini datotekama. U Windowsima NT, Windowsima 95 i kasnijim verzijama Windowsa te postavke aplikacije spremljene su u sistemskim registrima.

Visual Basic pruža standardno mjesto u registru za spremanje programskih informacija o aplikacijama stvorenim u Visual Basicu:

HKEY_CURRENT_USER\Software\VB and VBA Program Settings*imeaplikacije**odjeljak**ključ*

Visual Basic također pruža četiri naredbe i funkcije za rukovanje postavkama spremljenim na mjestu registra vaše aplikacije.

funkcija ili naredba	opis
funkcija GetSetting	Dohvaćanje postavki registra.
naredba SaveSetting	Spremanje i stvaranje postavki registra.
funkcija GetAllSettings	Vraćanje matrice koja sadrži višestruke postavke registra.
naredba DeleteSetting	Brisanje postavki registra.

Napomena Kako bi vidjeli unose registara, upotrijebite aplikaciju Regedit, uključenu s Windowsima 95/98 i Windowsima NT.

Stvaranje ili spremanje postavki aplikacije

Možete upotrijebiti naredbu SaveSeting za snimanje nove vrijednosti ključa registra spremljenog na mjestu registra vaše aplikacije. Na primjer, možete dodati programski kod u događaj Form_Unload glavne forme aplikacije koji će sčuvati postavke pri zatvaranju aplikacije, ili kodom u događaju Form_Unload dijaloškog okvira Options možete ažurirati korisničke postavke.

Upotrijebite sljedeću sintaksu za naredbu SaveSetting:

SaveSetting *imeaplikacije*, *odjeljak*, *ključ*, *vrijednost*

Sljedeći programski kod snima nove vrijednosti ključeva Rezkopija i PosljednjiUnos u odjeljku Početak registra aplikacije s imenom "RegKor". Ovaj kod pretpostavlja da varijable strDatum i intPosljednjiUnos sadrže nove vrijednosti:

```
Private Sub Form_Unload(Cancel As Integer)
  SaveSetting "RegKor" "Početak" "Rezkopija", strDatum
  SaveSetting "RegKor" "Početak" "PosljednjiUnos", intPosljednjiUnos
End Sub
```

Ako ne postoji unos za aplikaciju "RegKor" ili u odjeljku Software\Microsoft registra ne postoji neki od ovih odjeljaka ili ključeva, ovaj programski kod će ga stvoriti.

Za više informacija Pogledajte "Naredba SaveSetting" u dijelu *Microsoft Visual Basic 6.0 Language Reference* biblioteke *Microsoft Visual Basic 6.0 Reference Library*.

Dohvaćanje postavki aplikacije

Funkcije GetSetting i GetAllSettings možete upotrijebiti za dohvaćanje vrijednosti registra spremljene na mjestu registra vaše aplikacije. Na primjer, vaša aplikacija može dohvatiti postavke registra za obnavljanje stanja u kojem je bila kad je bila zatvorena.

Jedna po jedna postavka

Kako bi dohvatili jednu postavku registra, upotrijebite sljedeću sintaksu funkcije GetSetting:

GetSetting (*imeaplikacije*, *odjeljak*, *ključ* [, *predviđeno*])

Sljedeći programski kod dohvaća vrijednost ključa PosljednjiUnos u odjeljku Početak aplikacije "RegKor", i prikazuje vrijednost u prozoru za neposredan upis naredbi.

```
Private Sub Form_Load()
  Dim intPosljednjiUnos As Integer
  intPosljednjiUnos = GetSetting("RegKor" "Po~etak", _
  "PosljednjiUnos" "0")
  Debug.Print intPosljednjiUnos
End Sub
```

Uočite da možete upotrijebiti neobavezan parametar, *predviđeno*, za određivanje vrijednosti koju vraća Visual Basic ako ne postoji vrijednost ispisana u registru za određeni ključ.

Višestruke postavke

Kako bi dohvatili listu ključeva registra te njihove vrijednosti, upotrijebite sljedeću sintaksu za funkciju GetAllSettings:

GetAllSettings (*imeaplikacije*, *odjeljak*)

Sljedeći programski kod dohvaća dvostupčanu listu ključeva registra i njihove vrijednosti u odjeljku Početak aplikacije "RegKor", te prikazuje rezultate u prozoru za neposredan upis naredbi.

```
Private Sub Form_Load()
  Dim avntPostavke As Variant
  Dim intX As Integer
  avntPostavke = GetAllSettings("RegKor" "Početak")
  For intX = 0 To Ubound(avntPostavke, 1)
    Debug.Print avntPostavke(intX, 0), avntPostavke(intX, 1)
  Next intX
End Sub
```

Za više informacija Pogledajte "Funkcija GetSetting" i "Funkcija GetAllSettings" u biblioteci *Microsoft Visual Basic 6.0 Language Reference*.

Brisanje postavki aplikacije

Naredbu DeleteSetting možete upotrijebiti za brisanje ključa registra, odjeljka, ili mjesta registra aplikacije. Na primjer, možete poželjeti obrisati sve informacije u registru za aplikaciju kad se aplikacija deinstalira.

Upotrijebite sljedeću sintaksu za naredbu DeleteSetting:

DeleteSetting (*imeaplikacije*, *odjeljak*, *ključ*)

Sljedeći programski kod briše ključ PosljednjiUnos u odjeljku Početak aplikacije "RegKor".

```
Private Sub cmdObrišiKljuč_Click()
  DeleteSetting "RegKor" "Početak" "PosljednjiUnos"
End Sub
```

Sljedeći programski kod briše cijeli odjeljak Početak u registru aplikacije "RegKor".

```
Private Sub cmdObrišiOdjeljak_Click()
  DeleteSetting "RegKor" "Početak"
End Sub
```

Sljedeći programski kod briše cijelo mjesto registra za aplikaciju “RegKor”.

```
Private Sub cmdDeinstaliraj_Click()
  DeleteSetting “RegKor”
End Sub
```

Za više informacija Pogledajte “Naredba DeleteSetting” u biblioteci *Microsoft Visual Basic 6.0 Language Reference*.

Korištenje uvjetnog prevođenja

Uvjetno prevođenje dopušta vam selektivno prevođenje određenih dijelova aplikacije. Možete uključiti određene osobine svoje aplikacije u razne verzije, kao što je oblikovanje aplikacije za rad na različitim sustavima, ili promjena filtara za prikaz datuma i valute za aplikacije distribuirane na nekoliko različitih jezika.

Oblikovanje koda za uvjetno prevođenje

Kako bi uvjetno preveli dio svog programskog koda, zatvorite ga između naredbi #If...Then i #End If, koristeći konstantu tipa Boolean za ispitivanje grananja. Za uključivanje dijela koda u prevedeni kod, postavite vrijednost te konstante na -1 (True).

Na primjer, kako bi stvorili verzije iste aplikacije za francuski i njemački jezik iz istog strojnog koda, umetnite dijelove koda specifične za određenu verziju u naredbe #If...Then koristeći unaprijed određene konstante conFrancuskaVerzija i conNjemačkaVerzija.

```
#If conFrancuskaVerzija Then
   ‘ <kod specifičan za verziju na francuskom jeziku.>
#ElseIf conNjemačkaVerzija Then
   ‘ <kod specifičan za verziju na njemačkom jeziku.>
#Else
   ‘ <kod specifičan za ostale verzije.>
#End If
```

Ako je vrijednost konstante conFrancuskaVerzija postavljena na True tijekom prevođenja, bit će preveden uvjetni programski kod za verziju na francuskom jeziku. Ako je vrijednost konstante conNjemačkaVerzija postavljen na True, prevoditelj koristi verziju na njemačkom jeziku.

Određivanje konstanti uvjetnog prevođenja

Postoje tri načina za postavljanje konstanti uvjetnog prevođenja: u polju Conditional Compilation Arguments kartice Make dijaloškog okvira Project Properties, naredbenom linijom, ili u programskom kodu.

Konstante uvjetnog prevođenja imaju posebno područje i ne može im se pristupiti iz standardnog koda. Kako ćete odrediti konstantu uvjetnog prevođenja može ovisiti o području koje želite za konstantu.

kako odrediti	područje
dijaloški okvir Project Properties	Javna za sve module u projektu.
naredbena linija	Javna za sve module u projektu.
naredba #Const u kodu	Privatna za sve module u kojima je određena.

Određivanje konstanti u dijaloškom okviru Project Properties

Prije stvaranja izvršne datoteke, u izborniku Project, odaberite Project Properties, kliknite karticu Make u dijaloškom okviru Project Properties, i unesite argument, kao što je conFrancuskaVerzija = -1, u polje Conditional Compilation Arguments (ako prevodite svoju aplikaciju u verziju na francuskom jeziku). Kad prevodite aplikaciju, ovaj argument će zadovoljiti uvjet #If...Then, i kod između naredbi #If...Then i #End If će biti uključen u prevedenu aplikaciju.

Ako imate složen izraz #If...Then, koji sadrži jednu ili više naredbi #ElseIf, trebat ćete odrediti dodatne konstante. Možete odrediti više konstanti razdvajanjem dvotočkama, kao u sljedećem primjeru:

```
conFrancuskaVerzija=-1:conANSI=0
```

Određivanje konstanti u naredbenoj liniji

Ako želite započeti prevođenje iz naredbene linije, upotrijebite prekidač /d za upis konstanti uvjetnog prevođenja, kao što je ovdje prikazano:

vb6.exe /make MojProj.vbp /d conFrancuskaVerzija=-1:conANSI=0

Nije potreban razmak između prekidača /d i prve konstante. Odredbe naredbene linije nadjačavaju odredbe unesene u dijaloški okvir Project Properties, ali ih ne brišu; argumenti postavljeni u dijaloškom okviru Project Properties ostaju aktivni za iduća prevođenja.

Za više informacija Pogledajte “Smjernica #If...Then...#Else” i “Naredba #Const” u biblioteci *Microsoft Visual Basic 6.0 Language Reference*.

Rad s datotekama izvora

Datoteka izvora omogućuje vam skupljanje svih tekstova i slika aplikacije specifičnih za određenu verziju na jednom mjestu. Tu mogu biti uključene ikone, ekranski tekst, te ostali materijal koji se može razlikovati između lokaliziranih verzija, revizija ili specifičnih konfiguracija.

Dodavanje izvora projektu

Datoteku izvora možete stvoriti korištenjem dodatka Resource Editor. Prevedena datoteka izvora imat će sufiks imena datoteke .res. Svaki projekt može imati samo jednu datoteku izvora.

Stvarna datoteka sastoji se od niza zasebnih stringova, slika ili drugih dijelova, a svaki od njih ima jedinstvenu identifikacijsku oznaku. Ta oznaka je tipa Long ili String, ovisno o tipu podatka koji je predstavljen izvorom. Stringovi, na primjer, imaju oznaku tipa Long, dok slike imaju oznaku tipa Long ili String. Kako bi dohvatili izvore iz vašeg koda, doznajte identifikacijsku oznaku svakog izvora. Parametri funkcije za poziv izvora mogu koristiti podatke tipa Variant.

Kako dodati novu datoteku izvora vašem projektu

1. Odaberite **Resource Editor** u izborniku **Tools**. U prozoru editora izvora otvorit će se prazna datoteka izvora.

   Napomena Dodatak Resource Editor mora biti instaliran. Za informacije o instaliranju dodataka, pogledajte "Korištenje čarobnjaka i dodataka" u 4. poglavlju "Upravljanje projektima".

2. Odaberite gumb **Save** u alatnoj traci editora izvora za snimanje datoteke izvora. Datoteka će biti dodana u projektni prozor u odjeljak Related Documents.

Kako dodati postojeću datoteku izvora vašem projektu

- Odaberite **Add New Resource File** u izborniku **Project**. Svaka postojeća datoteka izvora u vašem projektu bit će zamijenjena.

  Oprez Ako napravite bilo kakve promjene u postojećoj datoteci izvora, one mogu utjecati na ostale projekte koji koriste tu datoteku izvora. Budite sigurni da ste snimili datoteku pod novim imenom.

  Napomena Dodatak Resource Editor mora biti instaliran. Za informacije o instaliranju dodataka, pogledajte "Korištenje čarobnjaka i dodataka" u 4. poglavlju "Upravljanje projektima".

Za više informacija Za više informacija o datotekama izvora, pogledajte "Korištenje datoteka izvora za lokalizaciju" u 16. poglavlju "Međunarodna izdanja".

Napomena Datoteke izvora u Windowsima različite su za 16-bitne i 32-bitne aplikacije. Visual Basic će stvoriti poruku pogreške ako projektu pokušate dodati 16-bitnu datoteku izvora.

Korištenje izvora u kodu

Visual Basic pruža tri funkcije za dohvaćanje podataka iz datoteke izvora za korištenje u programskom kodu.

funkcija	opis
LoadResString	Vraćanje tekstualnog niza.
LoadResPicture	Vraćanje objekta Picture, kao što je bitmapirana slika, ikona ili kursor.
LoadResData	Vraćanje matrice tipa Byte. Koristi se za datoteke tipa .wav, na primjer.

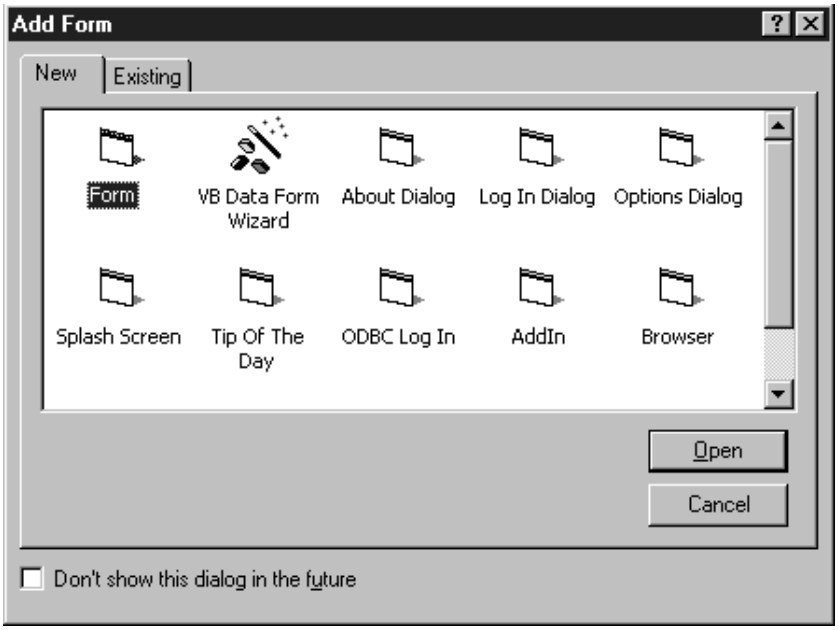
Za više informacija Pogledajte teme o pripadajućim funkcijama u biblioteci *Microsoft Visual Basic 6.0 Language Reference*.

Rad s predlošcima

Visual Basic pruža raznolike predloške za stvaranje uobičajenih sastavnih dijelova aplikacija. Umjesto stvaranja svih dijelova vaše aplikacije ni iz čega, možete prilagoditi postojeći predložak. Možete također ponovno upotrijebiti korisničke sastavne dijelove iz više aplikacija stvaranjem vlastitih predložaka.

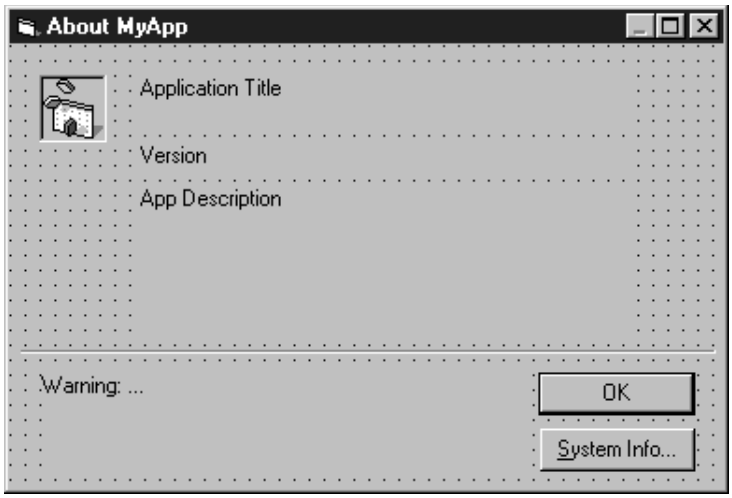
Postojeći predložak možete otvoriti odabirom njegove ikone u dijaloškom okviru Add Object kad stvarate novu formu, modul, kontrolu, stranicu svojstava, ili dokument. Na primjer, Visual Basic pruža ugrađene predloške forme za stvaranje dijaloškog okvira About, dijaloškog okvira Options ili uvodnog ekrana.

Slika 8.3 Dijaloški okvir Add Form



Kad otvorite predložak, Visual Basic prikazuje objekt s oznakama mjesta koja možete prilagoditi. Na primjer, kako bi stvorili dijaloški okvir About, otvorite predložak About Dialog i zamijenite oznake Application Title, Version i App Description informacijama svojstvenim vašoj aplikaciji.

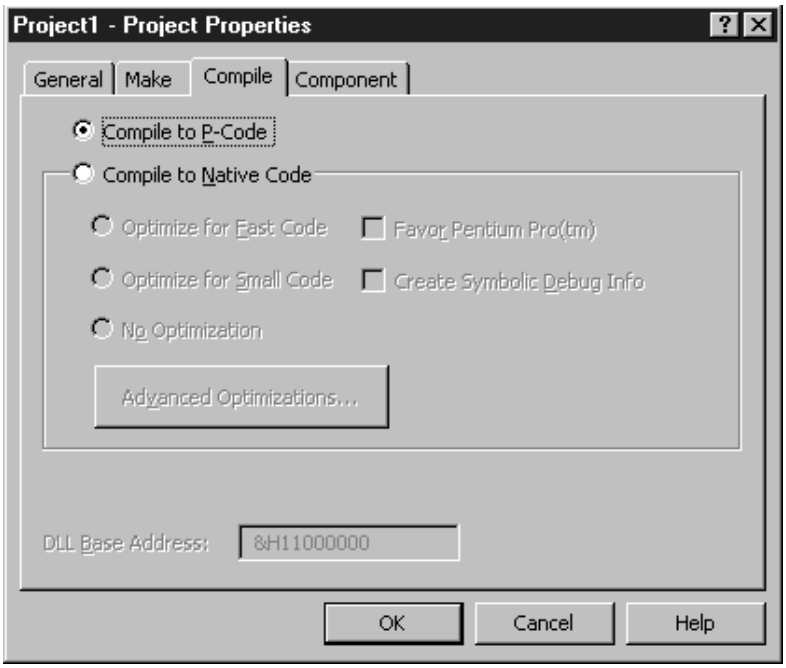
Slika 8.4 Predložak forme About Dialog



Kako bi stvorili vlastiti predložak, snimite objekt koji želite upotrijebiti kao predložak, kopirajte ga u odgovarajući poddirektorij direktorija Visual Basic Template. Na primjer, za stvaranje korisničkog predloška forme MojaForma, snimite formu s imenom MojaForma, te kopirajte datoteku MojaForma.frm u direktorij \VB\Template\Forms. Kad u izborniku Project odaberete naredbu Add Form, Visual Basic će prikazati predložak MojaForma u dijaloškom okviru Add Form, kao što je prikazano na slici 8.3.

Prikaz predložaka u dijaloškom okviru Add možete onemogućiti odabirom naredbe Options u izborniku Tools te brisanjem opcija Show Templates na kartici Environment dijaloškog okvira Options. Na primjer, kako bi onemogućili prikaz predložaka forme, obrišite opciju Forms u dijaloškom okviru.

Slika 8.5 Kartica Environment dijaloškog okvira Options



Rad s prekidačima naredbene linije

Prekidači naredbene linije pružaju način kontroliranja izvođenja Visual Basica. Korištenjem prekidača naredbene linije, možete pokrenuti Visual Basic i izvesti određen projekt, stvoriti izvršnu datoteku ili biblioteku dinamičkih veza, ili odrediti string koji će biti proslijeđen funkciji Command$.

Na primjer, kako bi pokrenuli projekt MojProj.vbp i zatim automatski izašli, pokrenite Visual Basic sljedećom naredbenom linijom:

c:\Program Files\Microsoft Visual Studio\VB\vb6.exe /runexit Mojproj.vbp

Sljedeća tablica sžima prekidače naredbene linije Visual Basica.

prekidač	opis
/cmd *cmdstring*	Određuje naredbeni string koji će biti proslijeđen funkciji Command$. Kad se koristi, mora biti posljednji prekidač naredbene linije.
/d *konstprevođenja*	Određuje jednu ili više konstanti uvjetnog prevođenja za upotrebu s prekidačima /make ili /makedll.
/make *imeprojekta*	Prevodi određeni projekt u izvršnu datoteku.

prekidač	opis
/makedll *imeprojekta*	Prevodi određeni projekt u biblioteku dinamičkih veza.
/mdi	Pokreće Visual Basic koristeći programsko okruženje s sučeljem s više dokumenata (MDI).
/out *imedatoteke*	Ispis pogrešaka u datoteku kad se koristi s prekidačima /make ili /makedll.
/run *imeprojekta*	Pokretanje određenog projekta.
/runexit *imeprojekta*	Pokretanje određenog projekta i zatim automatski izlazak.
/sdi	Pokreće Visual Basic koristeći programsko okruženje s sučeljem s jednim dokumentom (SDI).
/?	Prikaz liste valjanih prekidača naredbene linije.

Prevođenje vašeg projekta u strojni kod

Ako imate Professional ili Enterprise verzije Visual Basica, možete prevesti vaš programski kod u standardni oblik p-koda Visual Basica ili u oblik strojnog koda. Prevođenje u strojni kod pruža nekoliko opcija za optimiziranje i ispravljanje pogrešaka koje nisu dostupne s p-kodom.

P-kod, ili *pseudo kod*, je posrednički korak između instrukcija visoke razine u vašoj Basic aplikaciji i *strojnog koda* niske razine kojeg izvodi procesor vašeg računala. Tijekom izvođenja aplikacije, Visual Basic prevodi svaki izraz p-koda u strojni kod. Prevođenjem direktno u oblik strojnog koda, uklanjate posrednički korak p-koda.

Prevedeni strojni kod možete ispravljati korištenjem standardnih alata za ispravljanje strojnog koda, kao što je okolina za ispravljanje koju pruža Visual C++. Možete također koristiti mogućnosti raspoložive u jezicima kao što je Visual C++ za optimiziranje i ispravljanje strojnog koda. Na primjer, možete optimizirati kod za brzinu ili za veličinu.

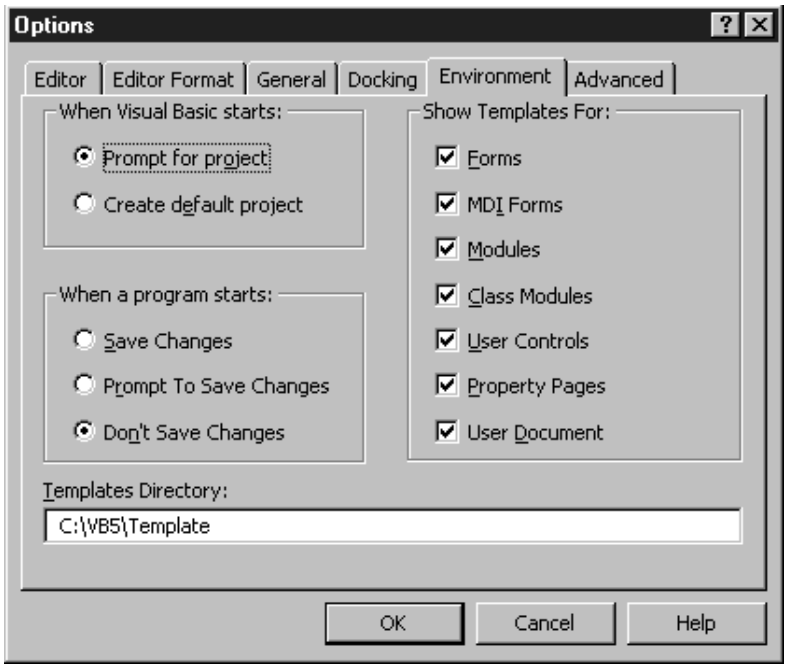
Napomena Svi projekti stvoreni Visual Basicom koriste usluge dinamičke biblioteke gotovih rutina (MSVBVM60.DLL). Među uslugama koje pruža ova biblioteka su kod za pokretanje i završavanje vaše aplikacije, funkcionalnost formi i ugrađenih kontrola, te funkcije izvođenja poput Format i CLng.

Prevođenje projekta s opcijom strojnog koda znači da će programski kod koji ste napisali biti potpuno preveden u strojne instrukcije procesorskog čipa, umjesto da bude preveden u p-kod. To će znatno ubrzati petlje i matematičke proračune, a ponekad može ubrzati pozive usluga koje pruža MSVBVM60.DLL. Unatoč tome, takvo prevođenje ne uklanja potrebu za tom dinamičkom bibliotekom.

Kako prevesti projekt u strojni kod

1. U projektnom prozoru odaberite projekt koji želite prevesti.
2. U izborniku **Project** odaberite **Project Properties**.
3. U dijaloškom okviru **Project Properties** kliknite karticu **Compile**.

Slika 8.6 Kartica Compile u dijaloškom okviru Project Properties



4. Odaberite **Compile to Native Code**.

   Visual Basic omogućuje nekoliko opcija za prilagodbu i optimiziranje izvršne datoteke. Na primjer, za stvaranje prevedenog koda koji će biti optimiziran za veličinu, odaberite opciju **Optimize for Small Code**.

   Za dodatne napredne opcije optimiziranja, kliknite gumb **Advanced Optimizations**.
5. Odaberite opcije koje želite, i kliknite **OK**.
6. U izborniku **File**, odaberite **Make Exe** ili **Make Project Group**.

Sljedeća tablica opisuje opcije strojnog koda za optimizaciju.

opcija	opis
Assume No Aliasing (napredna optimizacija)	Kaže prevoditelju da vaša aplikacija ne koristi pseudonime. Potvrda ove opcije dopušta prevoditelju da primjeni optimizaciju kao što je spremanje varijabli u registre i optimiziranje petlji.
Create Symbolic Debug Info	Stvara .pdb datoteku te .exe ili .dll datoteku s informacijama koje će omogućiti ispravljanje korištenjem Microsoft Visual C++ 5.0 ili druge sukladne aplikacije za pronalaženje pogrešaka.
Favor Pentium Pro	Optimizira kod za podršku Pentium Pro procesoru.
No Optimization	Onemogućuje sve optimizacije.
Optimize for Fast Code	Povećava brzinu .exe ili .dll datoteka kazujući prevoditelju da podrži brzinu prije veličine.
Optimize for Small Code	Smanjuje veličinu .exe ili .dll datoteka kazujući prevoditelju da podrži veličinu prije brzine.
Remove Array Bounds Checks (napredna optimizacija)	Onemogućuje provjeru granica matrica u Visual Basicu.
Remove Floating Point Error (napredna optimizacija)	Onemogućuje provjeru pogreške s pomičnim zarezom u Visual Basicu.
Remove Integer Overflow Checks (napredna optimizacija)	Onemogućuje provjeru prekoračenja cijelih brojeva u Visual Basicu.
Remove Safe Pentium FDIV (napredna optimizacija)	Onemogućuje provjeru sigurnog dijeljenja Pentium procesora s pomičnim zarezom.

Za više informacija Za detalje o opcijama strojnog koda pogledajte Dodatak C "Opcije prevoditelja u strojni kod".

Stvaranje vlastitih tipova podataka

Možete spojiti varijable nekoliko raznih tipova kako bi stvorili korisničke tipove (poznate kao *strukte* - *structs* - u programskom jeziku C). Korisnički određeni tipovi su korisni kad želite stvoriti jednu varijablu koja pamti nekoliko povezanih dijelova informacije.

Korisnički određen tip stvarate naredbom Type, koja mora biti postavljena u odjeljak Declarations modula. Korisnički određeni tipovi mogu biti određeni kao privatne ili javne varijable prikladnom ključnom riječi. Na primjer:

```
Private Type MojTipPodatka
```

- ili -

```
Public Type MojTipPodatka
```

Na primjer, možete stvoriti korisnički određen tip koji pamti informacije o računalnom sustavu:

```
‘ Odjeljak Declarations (standardnog modula).
Private Type SustavInfo
  CPU As Variant
  Memorija As Long
  VideoBoje As Integer
  Cijena As Currency
  DatumKupnje As Variant
End Type
```

Određivanje varijabli korisnički određenog tipa

Možete odrediti lokalne varijable, privatne varijable na razini modula, ili javne varijable na razini modula istog korisnički određenog tipa:

```
Dim MojSustav As SustavInfo, TvojSustav As SustavInfo
```

Sljedeća tablica opisuje gdje, i s kojim područjem, možete odrediti korisnički određene tipove i njihove varijable.

potprogram/modul	Možete *stvoriti* korisnički određen tip kao...	*Varijable* korisnički određenog tipa mogu biti određene kao...
Potprogrami	nije primjenjivo	samo lokalne
Standardni moduli	privatan ili javan	privatne ili javne
Moduli forme	samo privatan	samo privatne
Moduli klase	privatan ili javan	privatne ili javne

Napomena Ako je određen korištenjem naredbe Dim, korisnički određen tip će u standardnim ili klasnim modulima u pravilu biti javan. Ako namjeravate koristiti korisnički određen tip kao privatan, budite sigurni da ste ga odredili korištenjem naredbe Private.

Dodjela i dohvaćanje vrijednosti

Dodjeljivanje i dohvaćanje vrijednosti iz elemenata ovakve varijable slično je postavljanju i dohvaćanju svojstava:

```
MojSustav.CPU = “486”
If MojSustav.DatumKupnje > #1/1/92# Then
```

Možete također dodijeliti jednu varijablu drugoj ako su obje istog korisnički određenog tipa. To će dodijeliti sve elemente jedne varijable istim elementima druge varijable.

```
TvojSustav = MojSustav
```

Korisnički određeni tipovi koji sadrže matrice

Korisnički određen tip može sadržavati uobičajenu (stalne veličine) matricu.
Na primjer:

```
Type SustavInfo
  CPU As Variant
  Memorija As Long
  DiskPogoni(25) As String       ‘ Matrica stalne veličine.
  VideoBoje As Integer
  Cijena As Currency
  DatumKupnje As Variant
End Type
```

Može također sadržavati i dinamičku matricu.

```
Type SustavInfo
  CPU As Variant
  Memorija As Long
  DiskPogoni() As String         ‘ Dinamička matrica.
  VideoBoje As Integer
  Cijena As Currency
  DatumKupnje As Variant
End Type
```

Vrijednostima u matrici unutar korisnički određenog tipa možete pristupiti na isti način kao što pristupate svojstvu objekta.

```
Dim MojSustav As SustavInfo
ReDim MojSustav.DiskPogoni(3)
MojSustav.DiskPogoni(0) = “1.44 MB”
```

Možete također odrediti matricu korisnički određenih tipova:

```
Dim SviSustavi(100) As SustavInfo
```

Slijedite ista pravila za pristup sastavnim dijelovima ove strukture podataka.

```
SviSustavi(5).CPU = “386SX”
SviSustavi(5).DiskPogoni(2) = “100M SCSI”
```

Prosljeđivanje korisnički određenih tipova potprograma

Argumente potprograma možete proslijediti korištenjem korisnički određenog tipa.

```
Sub PopuniSustav(NekiSustav As SustavInfo)
  NekiSustav.CPU = lstCPU.Text
  NekiSustav.Memorija = txtMemorija.Text
  NekiSustav.Cijena = txtCijena.Text
  NekiSustav.DatumKupnje = Now
End Sub
```

Napomena Ako želite proslijediti korisnički određen tip u modulu forme, potprogram mora biti privatan.

Možete vratiti korisnički određene tipove iz funkcija, i možete proslijediti varijable korisnički određenog tipa u potprogram kao jedan od argumenata. Korisnički određeni tipovi se uvijek prosljeđuju upućivanjem, tako da potprogram može promijeniti argument i vratiti ga pozivnom potprogramu, kao što je prikazano u prošlom primjeru.

Napomena Budući da se korisnički određeni tipovi uvijek prosljeđuju upućivanjem, svi podaci sadržani u korisnički određenom tipu bit će proslijeđeni i vraćeni iz potprograma. Za korisnički određene tipove koji sadrže velike matrice, to bi kao rezultat moglo imati slabu izvedbu, posebno u aplikacijama klijent/poslužitelj gdje potprogram može raditi na udaljenom računalu. U takvoj situaciji, bolje je izvući i proslijediti samo nužne podatke iz korisnički određenog tipa.

Za više informacija Kako bi pročitali više o prosljeđivanju upućivanjem, pogledajte "Prosljeđivanje argumenata potprogramima" u 5. poglavlju "Osnove programiranja".

Korisnički određeni tipovi koji sadržavaju objekte

Korisnički određeni tipovi mogu također sadržavati i objekte.

```
Private Type PaketRačuna
  frmUnos As Form
  dbRačunPlaća As DataBase
End Type
```

Savjet Budući da podaci tipa Variant mogu spremiti puno različitih tipova podataka, matrica tipa Variant može biti upotrijebljena u većini situacija gdje očekujete upotrebu korisnički određenog tipa. Matrica tipa Variant je zapravo fleksibilnija od korisnički određenog tipa, jer uvijek možete mijenjati tip podatka kojeg spremate u svaki element matrice, a možete i matricu napraviti dinamičnom tako da možete mijenjati njezinu veličinu prema potrebi. Međutim, matrica tipa Variant uvijek koristi više memorije nego odgovarajući korisnički određen tip.

Gniježđenje struktura podataka

Gniježđenje struktura podataka može biti složeno koliko želite. U stvari, korisnički određeni tipovi mogu sadržavati druge korisnički određene tipove, kao što je prikazano u sljedećem primjeru. Kako bi vaš programski kod učinili čitljivijim i lakšim za ispravljanje, pokušajte održati kod koji određuje korisnički određene tipove u jednom modulu.

```
Type PogonInfo
  Type As String
  Size As Long
End Type
```

```
Type SustavInfo
  CPU As Variant
  Memorija As Long
  DiskPogoni(26) As PogonInfo
  Cijena As Currency
  DatumKupnje As Variant
End Type

Dim SviSustavi(100) As SustavInfo
SviSustavi(1).DiskPogoni(0).Type = “Floppy”
```

Korištenje nabrajanja za rad s nizovima konstanti

Nabrajanje pruža prikladan način za rad s nizovima povezanih konstanti te povezivanje vrijednosti konstanti s imenima. Na primjer, možete odrediti nabrajanje za niz cjelobrojnih konstanti povezanih s danima tjedna, te zatim upotrijebiti imena dana u programskom kodu radije nego njihovih cjelobrojnih vrijednosti.

Nabrajanje stvarate određivanjem tipa nabrajanja naredbom Enum u odjeljku Declarations standardnog modula ili javnog modula klase. Tipovi nabrajanja mogu biti određeni kao privatni ili javni odgovarajućom naredbom. Na primjer:

```
Private Enum MojeNabrajanje
```

- ili -

```
Public Enum MojeNabrajanje
```

U pravilu, prva konstanta u nabrajanju pokreće se s vrijednošću 0, a sljedeće konstante pokreću se s vrijednošću za jedan većom od prethodne konstante. Na primjer, sljedeće nabrajanje, Days, sadrži konstantu imena Sunday s vrijednošću 0, konstantu imena Monday s vrijednošću 1, konstantu imena Tuesday s vrijednošću 2 i tako dalje.

```
Public Enum Days
  Sunday
  Monday
  Tuesday
  Wednesday
  Thursday
  Friday
  Saturday
End Enum
```

> **Savjet** Visual Basic pruža ugrađeno nabrajanje, vbDayOfWeek, koje sadrži konstante dana u tjednu. Kako bi vidjeli predodređene konstante nabrajanja, upišite vbDayOfWeek u kodni prozor, praćeno s točkom. Visual Basic automatski prikazuje listu konstanti nabrajanja.

Možete izričito dodijeliti vrijednosti konstantama u nabrajanju korištenjem izraza dodjeljivanja. Možete dodijeliti bilo koju vrijednost dugog cijelog broja, uključujući negativne brojeve. Na primjer, možete poželjeti da konstante s vrijednostima manjim od 0 predstavljaju stanja pogreške.

U sljedećem nabrajanju, konstanti Invalid izričito je dodijeljena vrijednost -1, a konstanti Sunday dodijeljena je vrijednost 0. Budući da je prva varijabla u nabrajanju, varijabla Saturday također se pokreće s vrijednošću 0. Vrijednost varijable Monday je 1 (jedan više od vrijednosti varijable Sunday), vrijednost varijable Tuesday je 2 i tako dalje.

```
Public Enum WorkDays
  Saturday
  Sunday = 0
  Monday
  Tuesday
  Wednesday
  Thursday
  Friday
  Invalid = -1
End Enum
```

> **Napomena** Visual Basic obrađuje vrijednosti konstanti u nabrajanju kao duge cijele brojeve. Ako konstanti u nabrajanju dodijelite vrijednost s pomičnim zarezom, Visual Basic zaokružuje vrijednost na najbliži dugi cijeli broj.

Organiziranjem skupova povezanih konstanti u nabrajanjima, možete upotrijebiti ista imena konstante u različitim kontekstima. Na primjer, možete upotrijebiti isto ime za konstante dana tjedna u nabrajanjima Days i WorkDays.

Kako bi zaobišli dvosmislene odnose kad upućujete na pojedinu konstantu, označite ime konstante s njezinim nabrajanjem. Sljedeći programski kod upućuje na konstante Saturday u nabrajanjima Days i WorkDays, prikazujući njihove različite vrijednosti u prozoru za trenutan upis naredbi.

```
Debug.Print “Days.Saturday = “ & Days.Saturday
Debug.Print “WorkDays.Saturday = “ & WorkDays.Saturday
```

Možete također upotrijebiti vrijednost konstante u jednom nabrajanju kad dodjeljujete vrijednost konstanti u drugom nabrajanju. Na primjer, sljedeće određivanje nabrajanja WorkDays jednako je prethodnom određivanju.

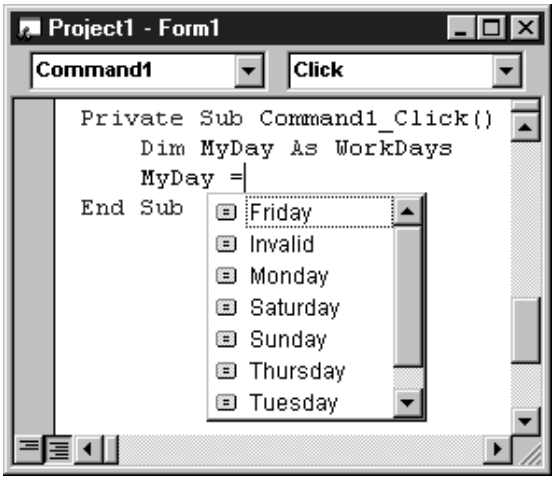
```
Public Enum WorkDays
  Sunday = 0
  Monday
  Tuesday
  Wednesday
  Thursday
  Friday
  Saturday = Days.Saturday - 6
  Invalid = -1
End Enum
```

Nakon određivanja tipa nabrajanja, možete odrediti varijablu tog tipa, te upotrijebiti tu varijablu za spremanje vrijednosti neke od konstanti nabrajanja. Sljedeći programski kod koristi varijablu tipa nabrajanja WorkDays za spremanje cjelobrojne vrijednosti povezane s konstantama nabrajanja WorkDays.

```
  Dim MyDay As WorkDays
  MyDay = Saturday         ‘ Saturday je postavljen na 0.
  If MyDay < Monday Then   ‘ Monday je postavljen na 1, tako da
                           ‘ Visual Basic prikazuje okvir s porukom.
    MsgBox “Vikend je. Pogrešan radni dan!”
End If
```

Uočite da kad u kodnom prozoru upisujete drugu liniju programskog koda ovog primjera, Visual Basic automatski prikazuje konstante nabrajanja WorkDays u listi automatskog završavanja izraza.

Slika 8.7 Visual Basic automatski prikazuje konstante nabrajanja



Budući da je konstanta Sunday također postavljena na 0, Visual Basic će također prikazati okvir s porukom ako zamijenite "Saturday" s "Sunday" u drugoj liniji ovog primjera:

```
MyDay = Sunday            ‘ Sunday je također postavljen na 0.
```

Napomena Iako obično dodjeljujete samo vrijednosti konstanti nabrajanja varijablama određenim s tipom nabrajanja, tim varijablama možete dodijeliti bilo koju vrijednost dugog cijelog broja. Visual Basic neće stvoriti pogrešku ako dodijelite takvu vrijednost varijabli koja nije pridružena konstanti nabrajanja.

Za više informacija Pogledajte "Naredba Enum" u biblioteci *Microsoft Visual Basic 6.0 Language Reference*. Također pogledajte "Pribavljanje imenovanih konstanti za vaše sastavne dijelove" u 6. poglavlju "Opća načela oblikovanja sastavnih dijelova", drugog dijela "Stvaranje ActiveX sastavnih dijelova" vodiča *Microsoft Visual Basic 6.0 Component Tools Guide*, dostupnog u verzijama Professional i Enterprise.

Napredne osobine matrica

Iako se matrice najčešće upotrebljavaju za spremanje grupa varijabli, postoji nekoliko drugih postupaka u kojima su matrice korisne. Možete dodijeliti sadržaj jedne matrice drugoj, stvoriti funkcije koje vraćaju matrice, te stvoriti svojstva koja vraćaju matrice. Takve tehnike mogu u puno slučajeva poboljšati izvođenje vaše aplikacije.

Dodjela matrica

Baš kako dodjeljujete sadržaj jedne varijable drugoj, na primjer strA = strB, možete također dodijeliti sadržaj jedne matrice drugoj. Zamislite, na primjer, da želite kopirati matricu bajtova s jednog mjesta na drugo. Možete to učiniti kopiranjem jednog po jednog bajta, slično ovom:

```
Sub KopiranjeBajta(staraKopija() As Byte, novaKopija() As Byte)
  Dim i As Integer
  ReDim novaKopija(LBound(staraKopija) To Ubound(staraKopija))
  For i = Lbound(staraKopija) To Ubound(staraKopija)
    novaKopija(i) = staraKopija(i)
  Next
End Sub
```

Puno djelotvorniji način da to napravite je dodjeljivanje jedne matrice drugoj:

```
Sub KopiranjeBajta(staraKopija() As Byte, novaKopija() As Byte)
  novaKopija = staraKopija
End Sub
```

Kod dodjeljivanja varijabli postoji nekoliko pravila kojih se trebate držati. Na primjer, iako možete dodijeliti varijablu određenu kao Integer varijabli određenoj kao Long bez problema, dodjeljivanje varijable tipa Long varijabli tipa Integer može lako dovesti do pogreške prekoračenja. Kao dodatak pravilima o tipovima podataka, dodjele matrica imaju dodatna pravila koja uključuju broj dimenzija, veličinu tih dimenzija, te je li matrica stalna ili dinamična.

Pokušaj dodjeljivanja matrica s različitim dimenzijama i/ili tipovima podataka može ili ne mora uspjeti, što ovisi o nekoliko čimbenika:

- Tip matrice koja je upotrijebljena na lijevoj strani dodjele: stalna matrica (Dim x(1 To 10) As Integer) ili dinamička matrica (Dim x() As Integer).
- Odgovara li ili ne broj dimenzija na lijevoj strani broju dimenzija matrice na desnoj strani dodjele.
- Podudara li se ili ne broj elemenata svake dimenzije na svakoj strani dodjele. Dimenzije se mogu podudarati čak i ako su određivanja različita, kao kad je jedna matrica temeljena na početku s nulom (Dim x(0 To 4)), a druga s jedinicom (Dim x(1 To 5)), sve dok imaju isti broj elemenata.
- Tipovi podataka svih elemenata svake strane dodjele moraju biti sukladni. Pravila su ista kao i za dodjeljivanje varijabli.

Sljedeća tablica prikazuje posljedice tih čimbenika:

Lijeva strana	Podudara li se broj dimenzija?	Podudara li se broj elemenata?	Rezultat dodjele
dinamička	ne	da ili ne	Uspješan. Lijeva strana se redimenzionira kako bi odgovarala desnoj strani ako je potrebno.
dinamička	da	ne	Uspješan. Lijeva strana se redimenzionira kako bi odgovarala desnoj strani ako je potrebno.
dinamička	da	da	Uspješan.
stalna	da ili ne	da ili ne	Neuspješan s pogreškom prevođenja.

Pogreške se mogu pojaviti i kod prevođenja i tijekom izvođenja aplikacije (na primjer, ako tipovi podataka ne mogu biti usklađeni ili ako dodjela pokuša redimenzionirati matricu stalne veličine). Na vama je, kao programeru, da dodate obradu pogrešaka kako bi osigurali sukladnost matrica prije pokušaja dodjeljivanja.

Vraćanje matrice iz funkcije

Moguće je za funkciju da vrati matricu podataka. Na primjer, možete poželjeti vratiti matricu bajtova iz funkcije bez potrebe za izvođenjem pretvaranja u string i iz stringa.

Vraćanje matrice iz funkcije neznatno se razlikuje od vraćanja jedne vrijednosti. Zahtjeva izričit tip u određivanju funkcije, te zahtijeva prosljeđivanje povratka izrazom Exit Function.

Ovdje je jednostavan primjer funkcije koja vraća matricu cijelih brojeva:

```
Private Sub Form_Load()
  Dim i As Integer
  Dim MatricaZaVraćanje() As Integer

  i = 1
  MatricaZaVraćanje() = FunkcijaMatrice(i)
End Sub

Public Function FunkcijaMatrice(i As Integer) As Integer()
  Dim x(2) As Integer

  x(0) = i
  x(1) = i + 1

‘ Vrijednost se prosljeđuje izrazom Exit Function.
  Exit Function x
  End Function
```

Nakon izvođenja gornjeg primjera, MatricaZaVraćanje() će biti matrica s dva elementa i sadržavat će 1 i 2. Uočite da izraz Exit Function prosljeđuje matricu kao argument; ta matrica mora biti istog tipa podatka kao i funkcija (u ovom slučaju, Integer). Zato što je to poziv funkcije, možete proslijediti matricu bez zagrada.

Napomena Iako je moguće vratiti matricu dodjeljivanjem drugoj matrici (`FunkcijaMatrice = x()`), to se ne preporučuje zbog izvođenja.

Morate odrediti tip za funkciju koja vraća matricu; taj tip može biti Variant. Zbog toga će `Function X() As Variant()` raditi dok `Function X() As ()` neće uspjeti.

Kad pozivate funkciju koja vraća matricu, varijabla koja će sadržavati vrijednosti rezultata mora biti matrica i mora biti istog tipa podatka kao i funkcija, inače će se pojaviti pogreša “Type Mismatch” (pogrešan tip podatka).

Za više informacija Kako bi naučili više o korištenju matrica, pogledajte “Matrice” u 5. poglavlju “Osnove programiranja”. Za informacije o vraćanju matrica iz svojstava, pogledajte “Postavljanje potprograma svojstava da rade za vas” u 9. poglavlju “Programiranje objektima”.

Korištenje zbirki umjesto matrica

Iako se zbirke često koriste za rad s objektima, možete upotrijebiti zbirku za rad s bilo kojim tipom podataka. U nekim okolnostima, može biti djelotvornije spremiti pojedine stvari u zbirku umjesto u matricu.

Možete poželjeti upotrijebiti zbirku ako radite s malim, promjenjivim skupom stvari. Sljedeći dio programskog koda pokazuje kako možete upotrijebiti zbirku za spremanje i prikazivanje liste URL adresa.

```
‘ Zbirka na razini modula.
Public colURLProšlost As New Collection

‘ Programski kod za dodavanje određenih URL adresa zbirci.
Private Sub SpremiURLProšlost(URLAdresa As String)
  colURLProšlost.Add URLAdresa
End Sub

‘ Programski kod za prikazivanje liste URL adresa
‘ u prozoru za neposredan upis naredbi.
Private Sub IspišiURLProšlost()
  Dim URLAdresa As Variant
  For Each URLAdresa in colURLProšlost
    Debug.Print URLAdresa
  Next URLAdresa
End Sub
```

Za više informacija Za više informacija o korištenju zbirki, pogledajte “Programiranje vašim vlastitim objektima” u 9. poglavlju “Programiranje objektima”. Kako bi naučili više o korištenju matrica, pogledajte “Matrice” u 5. poglavlju “Osnove programiranja”.

9. POGLAVLJE

Programiranje objektima

Objekti su u središtu programiranja Visual Basicom. Forme i kontrole su objekti. Baze podataka su objekti. Objekti su svuda gdje pogledate.

Ako ste neko vrijeme koristili Visual Basic, ili ako ste izradili primjere u prvih pet poglavlja ove knjige, tada ste već programirali objektima, ali tu je još više toga o objektima od onog što ste dosad vidjeli.

U ovom poglavlju, korisnički određeni tipovi će poprimiti vlastite osobnosti i postati klase. Vidjet ćete kako je lako stvoriti vlastite objekte iz klasa koje odredite, te koristiti objekte za pojednostavljivanje vašeg kodiranja i povećavanje ponovne upotrebe koda.

Sadržaj

- Što trebate znati o objektima u Visual Basicu
- Odgonetavanje objekata
- Stvaranje vlastitih klasa
- Dodavanje svojstava i postupaka klasi
- Dodavanje događaja klasi
- Stvaranje klasa svjesnih podataka
- Imenovanje svojstava, postupaka i događaja
- Polimorfizam
- Programiranje vašim vlastitim objektima
- Modeli objekata
- Stvaranje vlastitih klasa zbirki
- ActiveX kreatori

I ne zaustavlja se ovdje. 10. poglavlje “Programiranje sastavnim dijelovima”, uzima idući korak, pokazujući kako možete upotrijebiti Visual Basic za upravljanje objektima koje pružaju druge aplikacije.

Primjeri aplikacija: ProgWOb.vbp, Dataware.vbp

Neki od primjera programskog koda u ovom poglavlju uzeti su iz aplikacija Programming with Objects (ProgWOb.vbp) i Data-aware Classes (Dataware.vbp). Ovi primjeri aplikacija nalaze se u direktoriju Samples.

Što trebate znati o objektima u Visual Basicu

Visual Basic čini korištenje objekata lakim, ali važnije je da čini mogućim stupnjevit prijelaz između proceduralnog kodiranja i programiranja objektima.

Naravno, pomaže što ste koristili objekte cijelo vrijeme kad ste koristili Visual Basic.

Kratka terminologija

Ovo što slijedi je brzopotezno putovanje kroz izraze koje ćete susresti u raspravama o objektima Visual Basica i njihovim sposobnostima. Ako ste u Visual Basic došli iz drugog programskog jezika, ili ste radili s ActiveX (nekoć OLE) terminologijom, ova tema će vam pomoći u prijelazu.

Ako su vam objekti novi pojam, možda će vam sve ovo biti pomalo zbunjujuće. To je u redu – brzim prolazom kroz izraze koje ćete susretati, počet ćete oblikovati sliku o tome kako se oni međusobno uklapaju. U ostatku poglavlja ćete otkrivati više o objektima, pa se možete vratiti na ovu temu kako bi postavili svaki dio informacije na svoje mjesto.

Ovdje počinje

Objekti su *sžeti* – znači da sadržavaju i svoj programski kod i svoje podatke, što ih čini lakšim za održavanje nego tradicionalne načine pisanja koda.

Objekti Visual Basica imaju *svojstva*, *postupke* i *događaje*. Svojstva su podaci koji opisuju objekt. Objektu možete reći da napravi poslove koji se nazivaju postupci. Događaji su djela koja objekt radi; možete napisati kod koji će biti izveden kad se pojave događaji.

Objekti u Visual Basicu stvaraju se iz *klasa*; zbog toga se za objekt kaže da je *primjer klase*. Klasa određuje sučelje objekta, je li objekt javan, te pod kojim okolnostima će biti stvoren. Opisi klasa spremljeni su u *tipskim bibliotekama*, i mogu se pregledavati *pretraživačima objekata*.

Kako bi upotrijebili objekt, morate imati *upućivanje* na njega u *varijabli objekta*. Tip *povezivanja* određuje brzinu kojom će se pristupati postupcima objekta korištenjem varijable objekta. Varijabla objekta može biti *kasno povezana* (*late bound*, najsporije) ili *rano povezana* (*early bound*). Rano povezane varijable mogu biti *povezane načinom DispID* ili *povezane načinom vtable* (najbrže).

Skup svojstava i postupaka naziva se *sučelje*. Podrazumijevano sučelje objekta u Visual Basicu je *dvostruko sučelje* koje podržava sva tri oblika povezivanja. Ako je varijabla objekta *jasno označena* (znači, Dim... As *imeklase*), upotrijebit će najbrži oblik povezivanja.

Kao dodatak njihovom podrazumijevanom sučelju, objekti Visual Basica mogu ostvariti dodatna sučelja kako bi omogućili *polimorfizam*. Polimorfizam vam dopušta rukovanje različitim vrstama objekata bez potrebe za brigom kojeg je tipa svaki od njih. *Višestruka sučelja* su osobina Modela objekta kao sastavnog dijela (Component Object Model, COM); omogućuju vam razvijanje vaših aplikacija tijekom vremena, dodajući novu upotrebljivost bez prekidanja starog programskog koda.

Klase Visual Basica mogu također biti *svjesne podataka*. Klasa može djelovati kao *potrošač* podataka direktnim povezivanjem s vanjskim izvorom podataka, ili kao *izvor* podataka za druge objekte pružajući podatke iz vanjskog izvora.

Idemo na koncert

Uh! Ako vam je sve ovo izgledalo poznato kao vlastiti džep, prošetat ćete s lakoćom kroz ostatak poglavlja. Ako nije, ne brinite – objašnjenja svih ovih pojmova strateški su raspršena kroz ostatak teksta (i predstavljena na puno normalniji način).

Otkrivanje klase kojoj pripada objekt

Opće varijable objekta (znači, varijable koje ste odredili kao As Object) mogu sadržavati objekte puno raznih klasa. Slično tome, varijable određene ugrađenim tipovima Visual Basica Form i Control mogu sadržavati forme i kontrole različitih klasa.

Kad koristite varijable tih tipova, možda ćete trebati poduzeti različite akcije temeljene na klasi objekta – na primjer, neki objekti možda ne podržavaju određeno svojstvo ili postupak. Visual Basic pruža dva načina za to: naredbu TypeOf i funkciju TypeName.

Naredba TypeOf može se koristiti samo u izrazima If...Then...Else. Ime klase morate uključiti izravno u vaš programski kod. Na primjer, If TypeOf MojaKontrola Is CheckBox Then.

Funkcija TypeName je fleksibilnija. Možete ju upotrijebiti bilo gdje u vašem programskom kodu, a jer vraća ime klase kao string, možete je usporediti s vrijednošću u varijabli stringa.

Pozivanje svojstva ili postupka korištenjem imena stringa

Tijekom izrade aplikacije veći dio vremena možete otkrivati svojstva i postupke objekta te pisati kod koji će njima baratati. Međutim, u nekoliko slučajeva, možda nećete unaprijed poznavati svojstva i postupke objekta, ili ćete jednostavno poželjeti fleksibilnost koja omogućuje krajnjem korisniku određivanje svojstava i izvođenje postupaka tijekom rada.

Uzmite u obzir, na primjer, klijentsku aplikaciju koja proračunava izraze koje je unio korisnik prosljeđivanjem operatora poslužiteljskoj aplikaciji. Sad pretpostavite da stalno dodajete nove funkcije poslužitelju koje zahtijevaju nove operatore. Na žalost, trebat ćete ponovno prevesti i distribuirati klijentsku aplikaciju prije nego što bude sposobna koristiti nove operatore. Kako bi to zaobišli, možete upotrijebiti funkciju CallByName za prosljeđivanje novih operatora kao stringova, bez mijenjanja aplikacije.

Funkcija CallByName dopušta vam korištenje stringa za određivanje svojstva ili postupka tijekom rada aplikacije. Sintaksa funkcije CallByName izgleda ovako:

Rezultat = **CallByName** (*Objekt*, *Imepotprograma*, *Tippoziva*, *Argumenti()*)

Prvi argument ove funkcije, *Objekt*, prima ime objekta na koji želite djelovati. Drugi argument, *Imepotprograma*, prima string koji sadrži ime pozvanog potprograma postupka ili svojstva. Argument *Tippoziva* prima konstantu koja predstavlja tip pozvanog potprograma: postupak (vbMethod), puštanje svojstva (vbLet), primanje svojstva (vbGet) ili određivanje svojstva (vbSet). Posljednji argument je neobavezan, i prima matricu tipa Variant koja sadrži argumente za potprogram.

Pretpostavimo da imate poslužiteljsku aplikaciju, MathServer, s novom funkcijom KvadratniKorijen. Vaša aplikacija ima dvije kontrole okvira s tekstom: okvir Text1 sadrži izraz koji će biti proračunat; okvir Text2 se koristi za upis imena funkcije. Možete upotrijebiti sljedeći kod u događaju Click naredbenog gumba za pozivanje funkcije KvadratniKorijen na izraz u okviru Text1:

```
Private Sub Command1_Click()
  Text1.Text = CallByName(MathServer, Text2.Text, _
  vbMethod, Text1.Text)
End Sub
```

Ako korisnik upiše “64 / 4” u okvir Text1 i “KvadratniKorijen” u Text2, gornji programski kod će pozvati funkciju KvadratniKorijen (koju prima string koji sadrži izraz za proračunavanje kao obavezni argument) i vratiti “4” u okviru Text1 (kvadratni korijen iz 16, ili 64 podijeljeno s 4). Naravno, ako korisnik upiše nevaljan string u okvir Text2, ako string sadrži ime svojstva umjesto imena postupka, ili ako postupak ima dodatni obavezni argument, pojavit će se pogreška tijekom izvođenja. Kao što možete pretpostaviti, trebat ćete dodati jak kod za obradu pogrešaka kad koristite funkciju CallByName kako bi preduhitrili ove ili sve druge pogreške.

Iako funkcija CallByName može biti korisna u nekim okolnostima, trebate odvagnuti njezinu korisnost protiv utjecaja na izvođenje – pozivanje funkcije CallByName nešto je sporije od poziva kasnog povezivanja. Ako koristite funkciju koja će biti ponavljano pozivana, kao kad je unutar petlje, funkcija CallByName može imati neke posljedice na izvođenje.

Izvođenje višestrukih akcija na objektu

Često trebate izvesti nekoliko različitih akcija na istom objektu. Na primjer, možete trebati odrediti nekoliko svojstava za isti objekt. Jedan način da to uradite je korištenje nekoliko izraza.

```
Private Sub Form_Load()
  Command1.Caption = "OK"
  Command1.Visible = True
  Command1.Top = 200
  Command1.Left = 5000
  Command1.Enabled = True
End Sub
```

Uočite da svi ovi izrazi koriste istu varijablu objekta, Command1. Ovaj programski kod možete učiniti lakšim za pisanje, lakšim za čitanje, i djelotvornijim za izvođenje korištenjem izraza With...End With.

```
Private Sub Form_Load()
  With.Command1
     .Caption = "OK"
     .Visible = True
     .Top = 200
     .Left = 5000
     .Enabled = True
  End With
End Sub
```

Naredbe With možete također i ugnijezditi postavljanjem jednog izraza With...End With unutar drugog izraza With...End With.

Korištenje podrazumijevanih svojstava

Mnogo objekata ima *podrazumijevana svojstva*. Podrazumijevana svojstva možete upotrijebiti za pojednostavljivanje vašeg koda, jer ne trebate izravno upućivati na svojstvo kad mu određujete vrijednost. Za objekt kojem je podrazumijevano svojstvo Value, sljedeća dva izraza su jednaka:

```
objekt = 20
```

- i -

```
objekt.Value = 20
```

Da bi vidjeli kako ovo radi, stvorite na formi naredbeni gumb i okvir s tekstom. Dodajte sljedeći izraz u događaj Click naredbenog gumba:

```
Text1 = "pozdrav"
```

Pokrenite aplikaciju i kliknite naredbeni gumb. Budući da je svojstvo Text podrazumijevano svojstvo okvira s tekstom, u okviru s tekstom će se ispisati "pozdrav".

Korištenje podrazumijevanih svojstava s varijablama objekta

Kad je pokazivač na objekt spremljen u varijabli objekta, još uvijek možete upotrijebiti podrazumijevano svojstvo. Sljedeći dio programskog koda to pokazuje.

```
Private Sub Command1_Click()
  Dim obj As Object
  ‘ Postavljanje pokazivača na Text1 u objekt.
  set obj = Text1
  ‘ Odre|ivanje vrijednosti podrazumijevanog svojstva (Text).
  obj = “pozdrav”
End Sub
```

U gornjem kodu, izraz obj = “pozdrav” je potpuno jednak izrazu obj.Text = “pozdrav”.

Korištenje podrazumijevanih svojstava s varijablama tipa Variant

Pristup podrazumijevanim svojstvima drugačiji je kad je pokazivač na objekt spremljen u varijabli tipa Variant, umjesto u varijabli objekta. To je zato što varijabla tipa Variant može sadržavati podatke raznih tipova.

Na primjer, možete pročitati podrazumijevano svojstvo okvira Text1 korištenjem pokazivača u varijabli tipa Variant, ali pokušaj dodjeljivanja stringa “zbogom” podrazumijevanom svojstvu neće raditi. Umjesto toga, zamijenit će pokazivač na objekt stringom, i promijeniti tip podatka u varijabli Variant.

Da bi vidjeli kako to radi, upišite sljedeći programski kod u događaj Click naredbenog gumba iz prošlog primjera:

```
Private Sub Command1_Click()
  Dim vnt As Variant
  ‘ Određivanje podrazumijevanog svojstva (Text) na “pozdrav”.
  Text1 = “pozdrav”
  ‘ Postavljanje pokazivača na Text1 u Variant.
  Set vnt = Text1
  ‘ Prikaz podrazumijevanog svojstva, a za Text1, i pokazivanje
  ‘ što sadrži Variant kao pokazivač na objekt.
  MsgBox vnt “Je li objekt? “ & IsObject(vnt)
  ‘ Pokušaj postavljanja podrazumijevanog svojstva za Text1.
  vnt = “zbogom”
  MsgBox vnt “Je li objekt? “ & IsObject(vnt)
End Sub
```

Kad pokrenete aplikaciju i kliknete naredbeni gumb, najprije ćete dobiti okvir s porukom koji prikazuje trenutnu vrijednost podrazumijevanog svojstva okvira Text1 “pozdrav”, koju možete provjeriti ako pogledate okvir Text1. Naslov okvira s porukom potvrđuje da varijabla vnt tipa Variant sadrži pokazivač na objekt – znači, pokazivač na okvir Text1.

Kad kliknete gumb OK na okviru s porukom "zbogom" se dodjeljuje varijabli vnt, uništavajući pokazivač na okvir Text1. Zatim se prikazuje drugi okvir s porukom, pokazujući sadržaj varijable vnt – koji, kao što možete vidjeti, ne odgovara trenutnoj vrijednosti svojstva Text1.Text.

Naslov okvira s porukom potvrđuje da varijabla vnt više ne sadrži pokazivač na objekt – sada sadrži string "zbogom".

Za više informacija Za detalje o varijablama tipa Variant i ostalim tipovima podataka, pogledajte "Uvod u varijable, konstante i tipove podataka" u 5. poglavlju "Osnove programiranja".

Ostala stanovišta o korištenju objekata s varijablama tipa Variant raspravljena su u "Objekt tipa Collection Visual Basica", kasnije u ovom poglavlju.

Stvaranje matrica objekata

Možete odrediti i koristiti matricu tipa objekta isto kao što određujete i koristite matrice bilo kojeg tipa podatka. Te matrice mogu biti stalne ili promjenjive veličine.

Matrice varijabli forme

Možete odrediti matricu formi ključnom riječi Private, Dim, ReDim, Static ili Public na isti način kako određujete matricu bilo kojeg tipa. Ako odredite matricu s ključnom riječi New, Visual Basic automatski stvara nov primjer forme za svaki element u matrici dok koristite elemente u matrici.

```
Private Sub Command1_Click()
  Dim intX As Integer
  Dim frmNova(1 To 5) As New Form1
  For intX = 1 To 5
    frmNova(intX).Show
    frmNova(intX).WindowState = vbMinimized
    ' Kako bi stvorili nove forme a da se
    ' najprije kratko ne prikažu u normalnoj veličini
    ' zamijenite redoslijed gornje dvije linije.
  Next
End Sub
```

Klik naredbenog gumba za izvođenje gornjeg koda stvorit će pet smanjenih primjera forme Form1.

Napomena Ako pogledate na traku s zadaćama, vidjet ćete formu Form1 *šest* puta. Dodatni primjer forme Form1 nije smanjen – to je forma s kojom ste počeli.

Matrice varijabli kontrole

Možete odrediti matricu kontrola ključnom riječi Private, Dim, ReDim, Static ili Public na isti način kako određujete matricu bilo kojeg tipa. Međutim, za razliku od matrica formi, matrice kontrola ne mogu biti određene ključnom riječi New. Na primjer, možete odrediti matricu tako da bude određenog tipa kontrole:

```
ReDim AktivneSlike(10) As Image
```

Kad odredite matricu određenog tipa kontrole, toj matrici možete dodijeliti samo kontrole tog tipa. U slučaju prethodnog primjera određivanja, na primjer, matrici možete dodijeliti samo kontrole slike, ali te kontrole slike mogu doći s različitih formi.

Suprotnost ovome je ugrađena zbirka Controls, koja može sadržavati puno raznih tipova kontrola -, ali sve te kontrole moraju biti na istoj formi.

Alternativno tome, možete odrediti matricu općih varijabli kontrole. Na primjer, možete poželjeti pratiti svaku kontrolu koja je ispuštena na određenu kontrolu, i ne dozvoliti bilo kojoj kontroli da bude ispuštena više od jednom. Možete to napraviti održavanjem dinamičke matrice varijabli kontrola koja sadrži pokazivače na svaku kontrolu koja je ispuštena:

```
Private Sub List1_DragDrop(Izvor As VB.Control, _
X As Single, Y As Single)
  Dim intX As Integer
  Static intVeličina As Integer
  Static ctlIspuštena() As Control
  For intX = 1 to intVeličina
    ‘ Ako je ispuštena kontrola u matrici,
    ‘ ve} je jednom ovdje bila ispuštena.
    If ctlIspuštena(intX) Is Izvor Then
      Beep
      Exit Sub
    End If
  Next
  ‘ Povećavanje matrice.
  intVeličina = intVeličina + 1
  ReDim Preserve ctlIspuštena(intVeličina)
  ‘ Spremanje pokazivača na kontrolu koja je ispuštena.
  Set ctlIspuštena(intVeličina) = Izvor
  ‘ Dodavanje imena kontrole u okvir s popisom.
  List1.AddItem.Izvor.Name
End Sub
```

Ovaj primjer koristi operator Is za uspoređivanje varijabli u matrici kontrola s argumentom kontrole. Operator Is može biti upotrijebljen za ispitivanje jednakosti pokazivača objekata Visual Basica: ako usporedite dva različita pokazivača na isti objekt, operator Is vraća True.

Ovaj primjer također koristi naredbu Set kako bi elementu u matrici dodijelio pokazivač objekta u argumentu Izvor.

Za više informacija Pogledajte "Operator If" u Dodatku C "Operatori" dijela *Microsoft Visual Basic 6.0 Language Reference* biblioteke *Microsoft Visual Basic 6.0 Reference Library*.

Matrice su predstavljene u odlomcima "Matrice" i "Dinamičke matrice" u 5. poglavlju "Osnove programiranja".

Stvaranje zbirki objekata

Zbirke pružaju koristan način praćenja objekata. Za razliku od matrica, objekti tipa Collection ne trebaju biti redimenzionirani kako dodajete ili mičete elemente zbirke.

Na primjer, možete poželjeti pratiti svaku kontrolu koja je ispuštena na određenu kontrolu, i ne dozvoliti bilo kojoj kontroli da bude ispuštena više od jednom. Možete to napraviti održavanjem zbirke koja sadrži pokazivače na svaku kontrolu koja je bila ispuštena:

```
Private Sub List1_DragDrop(Izvor As VB.Control, _
X As Single, Y As Single)
  Dim vnt As Variant
  Static colIspušteneKontrole As New Collection
  For Each vnt In colIspušteneKontrole
    ‘ Ako je ispuštena kontrola u zbirci,
    ‘ već je jednom ovdje bila ispuštena.
    If vnt Is Izvor Then
          Beep
          Exit Sub
    End If
  Next
  ‘ Spremanje pokazivača na kontrolu koja je ispuštena.
  colIspušteneKontrole.Add Izvor
  ‘ Dodavanje imena kontrole u okvir s popisom.
  List1.AddItem.Izvor.Name
End Sub
```

Ovaj primjer koristi operator Is za uspoređivanje pokazivača objekta u zbirci colIspušteneKontrole s argumentom događaja koji sadrži pokazivač na ispuštenu kontrolu. Operator Is može biti upotrijebljen za ispitivanje jednakosti pokazivača objekata Visual Basica: ako usporedite dva različita pokazivača na isti objekt, operator Is vraća True.

Ovaj primjer također koristi postupak Add zbirke Collection kako bi u zbirku postavio pokazivač na ispuštenu kontrolu.

Za razliku od matrica, zbirke su objekti sami za sebe. Varijabla colIspušteneKontrole je određena s As New, pa će primjer klase Collection biti stvoren kad ta varijabla prvi put bude pozvana u kodu. Varijabla je također određena kao statična (Static), pa objekt tipa Collection neće biti uništen kad završi potprogram događaja.

Za više informacija Pogledajte “Operator If” u Dodatku C “Operatori” dijela *Microsoft Visual Basic 6.0 Language Reference*.

Svojstva i postupci objekta tipa Collection su raspravljeni u “Objekt Collection Visual Basica” kasnije u ovom poglavlju.

Kako bi usporedili gornji programski kod s kodom potrebnim za korištenje matrica, pogledajte “Stvaranje matrica objekata”, ranije u ovom poglavlju.

Kako bi naučili stvarati snažnije zbirke ubacivanjem objekta tipa Collection u vaše vlastite klase zbirki, pogledajte “Stvaranje vlastitih klasa zbirki” kasnije u ovom poglavlju.

Odlomak “Što trebate znati o objektima u Visual Basicu”, ranije u ovom poglavlju, opisuje kako se objekti stvaraju i uništavaju.

Objekt tipa Collection Visual Basica

Zbirka je način grupiranja niza povezanih predmeta. Zbirke se u Visual Basicu koriste za praćenje raznih stvari, kao što su učitane forme u vašoj aplikaciji (zbirka Forms), ili sve kontrole na formi (zbirka Controls).

Visual Basic pruža opću klasu Collection kako bi vam dao mogućnost određivanja vaših vlastitih zbirki. Možete stvoriti koliko god želite objekata tipa Collection – znači, primjera klase Collection. Objekte tipa Collection možete upotrijebiti kao temelje za vaše vlastite klase zbirki i modela objekata, kao što je raspravljeno u odlomcima “Stvaranje vlastitih klasa zbirki” i “Modeli objekata”, kasnije u ovom poglavlju.

Na primjer, zbirke su dobar način praćenja višestrukih formi. Odlomak “Aplikacije s više dokumenata (MDI)” u 6. poglavlju “Stvaranje korisničkog sučelja” objašnjava aplikacije u kojima korisnik može otvoriti bilo koji broj prozora dokumenata. Sljedeći dio programskog koda pokazuje kako možete koristiti postupak Add objekta zbirke za čuvanje liste MDI prozora koje je stvorio korisnik. Ovaj programski kod pretpostavlja da imate formu imena mdiDokument, kojoj je svojstvo MDIChild postavljeno na True.

```
‘ zbirka na razini modula u roditeljskoj MDI formi.
Public colDokumenti As New Collection
‘ Kod za stvaranje nove MDI forme-djeteta za dokument.
Private Sub mnuNovaDatoteka()
  Dim f As New mdiDokument
  Static intBrojDokumenta As Integer
  intBrojDokumenta = intBrojDokumenta + 1
  ‘ Sljede}a linija stvara formu.
  f.Caption = “Dokument “ & intBrojDokumenta
  ‘ Dodavanje pokazivača objekta u zbirku.
  colDokumenti.Add f
  f.Show
End Sub
```

Zbirka `colDokumenti` djeluje kao podskup ugrađene zbirke Forms, i sadrži samo primjere forme mdiDokument. Veličina zbirke se automatski prilagođuje kad se doda nova forma. Možete upotrijebiti izraz For Each...Next za ponavljanje prolaza kroz zbirku. Ako formi želite dati ključ kojim može biti pronađena, možete dodati tekstualni string kao drugi parametar postupka Add, kao što je opisano kasnije u ovom odlomku.

Ključna riječ New u određivanju varijable `colDokumenti` uzrokuje stvaranje zbirke Collection kad se prvi put pozove ta varijabla u programskom kodu. Budući da je zbirka Collection klasa, prije nego tip podatka, morate stvoriti njezin primjer i čuvati pokazivač na taj primjer (objekt) u varijabli.

Kao i bilo koji objekt, objekt tipa Collection bit će uništen kad zadnja varijabla koja sadrži pokazivač na njega bude postavljena na Nothing ili ode izvan područja. Svi pokazivači objekata koje ova zbirka sadrži bit će oslobođeni. Zbog toa, varijabla colDokumenti je određena u roditeljskoj MDI formi, pa traje cijelo vrijeme izvođenja aplikacije.

> Napomena Ako koristite zbirku za praćenje formi, upotrijebite postupak Remove zbirke za brisanje pokazivača objekta iz zbirke nakon što izbacite formu. Ne možete koristiti memoriju koju je koristila forma sve dok postoji pokazivač na tu formu, a pokazivač u objektu tipa Collection se održava jednako dobro kao i pokazivač u varijabli objekta.

Od čega je napravljen objekt tipa Collection?

Objekt tipa Collection sprema svaku stavku kao tip Variant. Zbog toga je popis stvari koje možete dodati u objekt tipa Collection isti kao i popis stvari koje mogu biti spremljene u varijablu tipa Variant. To uključuje standardne tipove podataka, objekte i matrice -, ali ne i korisnički određene tipove.

Tipovi Variant uvijek troše 16 bajtova, neovisno o tome što je spremljeno u njima, pa korištenje objekta tipa Collection nije tako djelotvorno kao korištenje matrica. Međutim, nikad nećete trebati redimenzionirati objekt tipa Collection, što za posljedicu ima puno čistiji kod koji je lakši za održavanje. Kao dodatak, objekti tipa Collection imaju iznimno brze preglede po ključu, što matrice nemaju.

> Napomena Da budemo precizni, tipovi Variant uvijek troše 16 bajtova *čak i ako su podaci zapravo spremljeni negdje drugdje*. Na primjer, ako tipu Variant dodijelite string ili matricu, tip Variant sadrži pokazivač na kopiju podatka stringa ili matrice. Samo 4 bajta u varijabli tipa Variant koriste se za pokazivač na 32-bitnim sustavima, i nijedan podatak nije zapravo u varijabli tipa Variant.

Ako spremite objekt, tip Variant sadrži pokazivač objekta, kao što bi to sadržavala i varijabla objekta. Kao i kod stringova i matrica, koriste se samo 4 bajta varijable tipa Variant.

Brojčani tipovi podataka spremaju se unutar varijable tipa Variant. Neovisno o tipu podatka, varijabla tipa Variant i dalje troši 16 bajtova.

Unatoč veličini tipova Variant, postoji puno slučajeva gdje ima smisla upotrebljavati objekt tipa Collection za spremanje svih gore navedenih tipova podataka. Samo budite svjesni trgovine koju činite: objekti tipa Collection omogućuju vam pisanje vrlo čistog koda lakog za održavanje – po cijeni spremanja stvari u tipove Variant.

Svojstva i postupci objekta tipa Collection

Svaki objekt tipa Collection dolazi s svojstvima i postupcima koje možete upotrijebiti za ubacivanje, brisanje i dohvaćanje stavki u zbirci.

svojstvo ili postupak	opis
postupak Add	Dodaje stavke u zbirku.
svojstvo Count	Vraća broj stavki u zbirci. Samo za čitanje.
postupak Item	Vraća stavku, po indeksu ili po ključu.
postupak Remove	Briše stavku iz zbirke, po indeksu ili po ključu.

Ova svojstva i postupci pružaju samo najosnovnije usluge za zbirke. Na primjer, postupak Add ne može provjeriti tip podatka koji se dodaje zbirci, kako bi osigurao da zbirka sadrži samo jednu vrstu objekata. Možete pružiti jaču djelotvornost – te dodatna svojstva, postupke i događaje – stvaranjem vaših vlastitih klasa zbirki, kao što je opisano u odlomku "Stvaranje vlastitih klasa zbirki" kasnije u ovom poglavlju.

Osnovne usluge dodavanja, brisanja i dohvaćanja iz zbirke ovise o ključevima i indeksima. *Ključ* je vrijednost tipa String. To može biti ime, broj vozačke dozvole, broj zdravstvenog osiguranja, ili jednostavno cijeli broj pretvoren u string. Postupak Add omogućuje vam udruživanje ključa s stavkom, kao što je opisano kasnije u ovom poglavlju.

Indeks je vrijednost tipa Long između jedan (1) i ukupnog broja stavki u zbirci. Možete nadzirati početnu vrijednost indeksa stavke, korištenjem imenovanih parametara ***before*** i ***after***, ali se njegova vrijednost može promijeniti kad se dodaju ili brišu druge stavke.

> Napomena Zbirka čiji indeks počinje s 1 naziva se *temeljena na jedinici* (*one-based*), kao što je objašnjeno u odlomku "Zbirke u Visual Basicu", u stalnoj pomoći.

Indeks možete upotrijebiti za ponavljanje prolaza kroz stavke u zbirci. Na primjer, sljedeći programski kod pokazuje dva načina za davanje 10-postotne povišice svim djelatnicima u zbirci objekata Djelatnici, s pretpostavkom da varijabla mkolDjelatnici sadrži pokazivač na objekt tipa Collection.

```
Dim lngBroj As Long
For lngBroj = 1 To mkolDjelatnici.Count
  mkolDjelatnici(lngBroj).Rate = _
  mkolDjelatnici(lngBroj).Rate * 1.1
Next
```

```
Dim dje As Djelatnik
For Each dje In mkolDjelatnici
  dje.Rate = dje.Rate * 1.1
Next
```

Savjet Za bolje izvođenje, upotrijebite For Each za ponavljanje prolaza kroz stavke u objektu tipa Collection. Izraz For Each je znatno brži od ponavljanja s indeksom. To nije točno za sve izvedbe s zbirkama – ovisi o unutarnjem načinu na koji zbirka sprema podatke.

Dodavanje stavki u zbirku

Upotrijebite postupak Add za dodavanje stavke u zbirku. Sintaksa je:

Sub Add (*stavka* **As Variant** [, *ključ* **As Variant**] [, ***before*** **As Variant**]
[, ***after*** **As Variant**]

Na primjer, kako bi dodali objekt radnog naloga zbirci radnih naloga koristeći svojstvo ID radnog naloga kao ključ, možete napisati:

```
colRadniNalozi.Add rnNovi, rnNovi.ID
```

Pretpostavljeno je da je svojstvo ID tipa String. Ako je to svojstvo broj (na primjer, tipa Long), upotrijebite funkciju CStr za pretvorbu u vrijednost tipa String potrebnu za ključ:

```
colRadniNalozi.Add rnNovi, CStr(rnNovi.ID)
```

Postupak Add podržava imenovane argumente. Kako bi dodali stavku kao treći element, možete napisati:

```
colRadniNalozi.Add rnNovi, rnNovi.ID, after:=2
```

Možete upotrijebiti imenovane argumente ***before*** i ***after*** kako bi održali poredanu zbirku objekata. Na primjer, before:=1 ubacuje stavku na početak zbirke, pošto su objekti zbirke Collection temeljeni na jedinici.

Brisanje stavki iz zbirke

Upotrijebite postupak Remove za brisanje stavke iz zbirke. Sintaksa je:

objekt.**Remove** *indeks*

Argument indeks može biti ili položaj stavke koju želite obrisati, ili ključ stavke. Ako je ključ treće stavke u zbirci jednak “W017493”, možete upotrijebiti jedan od ova dva izraza kako bi je obrisali:

```
colRadniNalozi.Remove 3
```

- ili -

```
colRadniNalozi.Remove “W017493”
```

Dohvaćanje stavke iz zbirke

Upotrijebite postupak Item za dohvaćanje određenih stavaka iz zbirke. Sintaksa je:

[Set] *varijabla = objekt*.**Item(***indeks***)**

Kao i kod postupka Remove, indeks može biti ili položaj u zbirci, ili ključ stavke. Koristeći isti primjer kao i za postupak Remove, bilo koji od ova dva izraza će dohvatiti treći element u zbirci:

```
Set rnTrenutan = colRadniNalozi.Item(3)
```

- ili -

```
Set rnTrenutan = colRadniNalozi.Item("WO17493")
```

Ako koristite cijele brojeve kao ključeve, morate upotrijebiti funkciju CStr kako bi ih pretvorili u stringove prije nego što ih proslijedite postupcima Item ili Remove. Objekt tipa Collection uvijek pretpostavlja da je cijeli broj indeks.

> **Savjet** Ne dopustite objektima tipa Collection da odlučuju je li vrijednost koju prosljeđujete indeks ili ključ. Ako želite da vrijednost bude protumačena kao ključ, a varijabla koja sadrži vrijednost bude sve osim stringa, upotrijebite funkciju CStr za pretvaranje. Ako želite da vrijednost bude protumačena kao indeks, a varijabla koja sadrži vrijednost nije tipa cjelobrojnih podataka, upotrijebite funkciju CLng za pretvaranje.

Item je podrazumijevan postupak

Postupak Item je podrazumijevan postupak objekta tipa Collection, pa ga možete izostaviti kad pristupate stavci u zbirci. Prema tome bi prethodni primjer programskog koda također mogao biti napisan i ovako:

```
Set rnTrenutan = colRadniNalozi(3)
```

- ili -

```
Set rnTrenutan = colRadniNalozi("WO17493")
```

> **Važno** Objekti zbirke automatski održavaju svoje brojeve numeričkih indeksa kad dodajete ili brišete elemente. Brojčani indeks danog elementa će se zbog toga vremenom promijeniti. Ne spremajte vrijednost brojčanog indeksa očekujući da ćete kasnije dohvatiti isti element u svojoj aplikaciji. Za to iskoristite ključeve.

Korištenje postupka Item za pozivanje svojstava i postupaka

Ne morate dohvatiti pokazivač objekta iz zbirke i postaviti ga u varijablu objekta kako bi ga upotrijebili. Možete iskoristiti pokazivač dok stoji u zbirci.

Na primjer, pretpostavimo da objekt RadniNalog u prethodnom primjeru ima svojstvo Priority. Oba sljedeća izraza će odrediti prednost radnog naloga:

```
colRadniNalozi.Item("WO17493").Priority = 3
colRadniNalozi("WO17493").Priority = 3
```

Razlog zašto ovo radi je što Visual Basic proračunava izraz slijeva nadesno. Kad dođe do postupka Item – izričitog ili podrazumijevanog – Visual Basic uzima pokazivač na označenu stavku (u ovom slučaju, objekt RadniNalog čiji ključ je W017493), i koristi taj pokazivač za proračunavanje ostatka linije.

Savjet Ako namjeravate pozivati više od jednog svojstva ili postupka nekog objekta u zbirci, najprije kopirajte pokazivač objekta u jasno određenu varijablu objekta. Korištenje pokazivača objekta dok je još uvijek u zbirci sporije je od korištenja nakon što ga postavite u jasno određenu varijablu objekta (na primjer, Dim rnTrenutan As RadniNalog), zato što objekt tipa Collection sprema stavke kao tipove Variant. Pokazivači objekta u tipovima Variant su uvijek kasno povezane.

Za više informacija Objekt tipa Collection je također korisna alternativa matricama za puno uobičajenih programskih zadataka. Pogledajte "Korištenje zbirki umjesto matrica" u 8. poglavlju "Više o programiranju".

Zbirke u Visual Basicu

Što je zbirka? U odlomku "Objekt tipa Collection Visual Basica", ranije u ovom poglavlju, zbirka je određena kao način grupiranja povezanih objekata. To ostavlja dosta mjesta raznim tumačenjima; riječ je više o koncepciji nego o definiciji.

Zapravo, kao što ćete vidjeti kad počnete uspoređivati zbirke, postoji puno razlika čak i između vrsti zbirka koje pruža Visual Basic. Na primjer, sljedeći programski kod uzrokovat će pogrešku:

```
Dim col As Collection
Set col = Forms          ‘ Pogreška!
```

Što se ovdje dogodilo? Zbirka Forms je zbirka; varijabla col je određena kao zbirka; zašto ne možete dodijeliti pokazivač na zbirku Forms varijabli col?

Razlog tomu je što klasa Collection i zbirka Forms nisu *polimorfne*; znači, ne možete razmijeniti jednu s drugom, jer su one razvijene iz odvojenih kodnih osnova. Nemaju iste postupke, ne spremaju pokazivače objekta na isti način, ili ne koriste iste vrste vrijednosti indeksa.

To čini ime klase Collection jednim od mnogo izbora, jer ta klasa zapravo predstavlja samo jednu od mnogih izvedbi zbirki. Ova tema istražuje neke od razlika među izvedbama koje ćete otkriti.

Zbirke temeljene na nuli i na jedinici

Zbirka je *temeljena na nuli* ili je *temeljena na jedinici*, ovisno o tome koji je početni indeks. Kao što možete pretpostaviti, prvi izraz znači da je indeks prve stavke u zbirci nula, a drugi izraz znači da je prvi indeks jedan. Primjeri zbirki temeljenih na nuli su zbirke Forms i Controls. Objekt Collection je primjer zbirke temeljene na jedinici.

Starije zbirke u Visual Basicu su vjerojatnije temeljene na nuli, dok je većina novijih dodataka vjerojatnije temeljena na jedinici. Zbirke temeljene na jedinici su ponekad lakše za neposredno korištenje, jer se indeks pruža od jedan do ukupnog broja stavki, gdje je Count svojstvo koje vraća ukupan broj stavki u zbirci.

Za razliku od toga, indeks zbirki temeljenih na nuli pruža se od nule do jedan manje od vrijednosti svojstva Count.

Vrijednosti indeksa i ključa

Većina zbirki u Visual Basicu omogućuje vam da im pristupite korištenjem ili brojčanog indeksa ili stringa kao ključa, kao kod objekta Collection Visual Basica (ipak, objekt Collection Visual Basica omogućuje vam dodavanje stavki bez određivanja ključa).

Zbirka Forms, suprotno tome, dopušta samo brojčani indeks. Razlog tome je što ne postoji jedinstvena vrijednost stringa povezana s formom. Na primjer, možete imati više formi s istim naslovom, ili više učitanih formi s istim svojstvom Name.

Dodavanje i micanje stavki

Zbirke se također razlikuju po tome možete li im dodati stavke ili ne, te ako možete, kako se te stavke dodaju. Na primjer, možete dodati pisač zbirci Printers korištenjem programskog koda Visual Basica.

Budući da je objekt Collection programski alat opće namjene, fleksibilniji je od ostalih zbirki. On ima postupak Add koji možete upotrijebiti za postavljanje stavki u zbirku, te postupak Remove za vađenje stavki.

Suprotno tome, jedini način postavljanja forme u zbirku Forms je da učitate formu. Ako stvorite formu s operatorom New, ili pozivanjem varijable određene s As New, ta forma neće biti dodana u zbirku Forms sve dok ne upotrijebite naredbu Load kako bi je učitali.

Zbirke Forms i Controls nemaju postupke Remove. Možete dodati i ukloniti forme i kontrole iz tih zbirki posredno, korištenjem naredbi Load i Unload.

Što se dobiva u njezinim džepovima?

Kao što je malo prije rečeno, forma se ne dodaje zbirci Forms sve dok se ne učita. Zbog toga je najpreciznije označavanje zbirke Forms da ona sadrži *sve trenutno učitane forme u aplikaciji*.

Čak i to nije potpuno točno. Ako vaš projekt koristi forme tipa Microsoft Forms (uključene za sukladnost s aplikacijom Microsoft Office), pronaći ćete te forme u odijeljenoj zbirci imena UserForms. Prema tome zbirka Forms sadrži *sve trenutno učitane Visual Basic forme u aplikaciji*.

Sadržaj klase Collection je vrlo precizno određen: *sve što može biti spremljeno u tipu Variant*. Zbog toga objekt Collection može sadržavati objekt ili cijeli broj, ali ne može korisnički određen tip.

Na žalost, ovakvo određivanje pokriva veliko područje – neki primjer klase Collection može spremiti bilo koji miješani izbor tipova podataka, matrica i objekata.

> Savjet Jedan od najvažnijih razloga stvaranja vaših vlastitih klasa zbirki, kao što je raspravljeno u odlomku "Stvaranje vlastitih klasa zbirki", kasnije u ovom poglavlju, je što možete kontrolirati sadržaje vaših zbirki – pojam nazvan *sigurnost tipa* (*type safety*).

Nabrajanje zbirke

Možete upotrijebiti izraz For Each...Next za nabrajanje stavki u zbirci, bez obzira je li zbirka temeljena na nuli ili jedinici. Naravno, teško da je to određujuća karakteristika zbirke, jer vam Visual Basic omogućuje korištenje izraza For Each...Next za nabrajanje stavki u matrici.

Izraz For Each...Next radi zahvaljujući malom objektu nazvanom *brojitelj* (*enumerator*). Brojitelj prati gdje se nalazite u zbirci, i vraća iduću stavku kad je potrebno.

Kad nabrajate matricu, Visual Basic stvara objekt brojitelja matrice u letu. Zbirke imaju vlastite objekte brojitelja, koji se također stvaraju kad su potrebni.

Brojitelji ne preskaču stavke

Brojitelji zbirki u Visual Basicu ne preskaču stavke. Na primjer, pretpostavimo da nabrajate zbirku koja sadrži stavke "A", "B" i "C", te tijekom nabrajanja uklonite stavku "B". Zbirka Visual Basica neće preskočiti do stavke "C" kad to napravite.

Brojitelji možda neće uhvatiti dodane stavke

Ako dodate stavke u zbirku dok je nabrajate, neki brojači će uključiti dodane stavke, dok neki neće. Zbirka Forms, na primjer, neće nabrojiti ni jednu formu koju ste učitali tijekom nabrajanja.

Objekt Collection će nabrojiti stavke koje ste dodali tijekom nabrajanja, ako im omogućite da budu dodane na kraj zbirke. Zbog toga se sljedeća petlja nikad neće završiti (odnosno, sve dok ne pritisnete CTRL + BREAK):

```
Dim col As New Collection
Dim vnt as Variant
col.Add "Beskrajno"
col.Add "Beskrajno"
For Each vnt In col
  MsgBox vnt
  col.Add "Beskrajno"
Next
```

S druge strane, stavke koje dodate na početak zbirke neće biti uključene u nabrajanje:

```
Dim col As New Collection
Dim vnt As Variant
col.Add “bit će nabrojeno”
For Each vnt In col
  MsgBox vnt
  ‘ Dodavanje stavke na početak zbirke.
  col.Add “Neće biti nabrojeno”, Before:=1
Next
```

Zašto brojitelji?

Davanjem novog brojitelja svaki put kad započne petlja For Each...Next, zbirka omogućuje ugnjezdena nabrajanja. Na primjer, pretpostavimo da imate pokazivač na objekt Collection u varijabli mcolStringovi, te da ta zbirka sadrži samo stringove. Sljedeći programski kod ispisuje sve kombinacije dva različita stringa:

```
Dim vnt1 As Variant
Dim vnt2 As Variant
For Each vnt1 In mcolStringovi
  For Each vnt2 In mcolStringovi
    If vnt1 <> vnt2 Then
          Debug.Print vnt1 & “ “ & vnt2
    End If
  Next
Next
```

Za više informacija Pogledajte “Stvaranje vlastitih klasa zbirki” kasnije u ovom poglavlju.

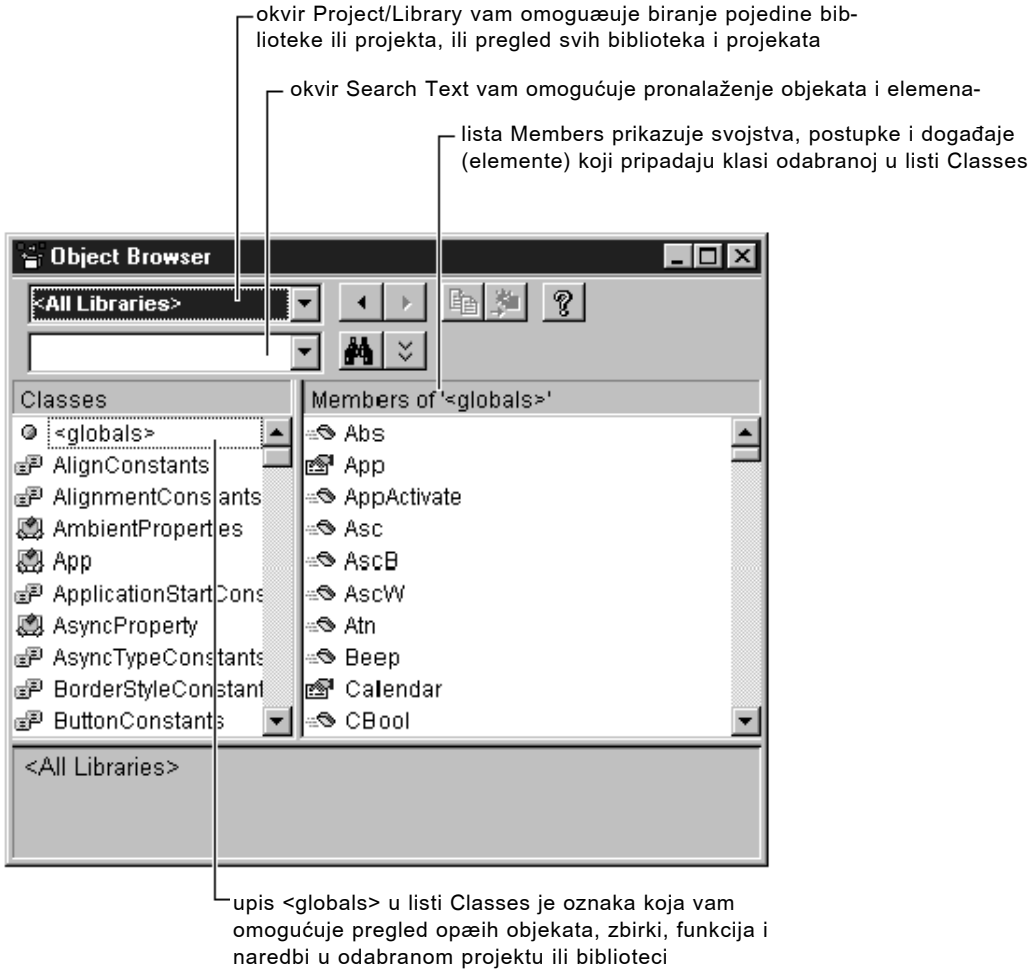
Odgonetavanje objekata

Pretraživač objekata (Object Browser) temeljen je na *tipskim bibliotekama* (*type libraries*), izvorima koji sadržavaju detaljne opise klasa, uključujući svojstva, postupke, događaje, imenovane konstante i drugo.

Visual Basic stvara informaciju tipske biblioteke za klase koje stvarate, pružajući tipske biblioteke za objekte koje klasa sadrži, i dopušta vam pristup tipskim bibliotekama koje pružaju druge aplikacije.

Pretraživač objekata možete upotrijebiti za prikaz klasa dostupnih u projektima i bibliotekama, uključujući klase koje ste vi odredili. Objekti koje stvorite iz tih klasa imat će iste dijelove – svojstva, postupke i događaje – koje vidite u pretraživaču objekata.

Slika 9.1 Pretraživač objekata



Kako prikazati pretraživač objekata

- U izborniku **View**, odaberite **Object Browser**.

  - ili -

  Pritisnite F2.

  - ili -

  Kliknite gumb **Object Browser** na alatnoj traci.

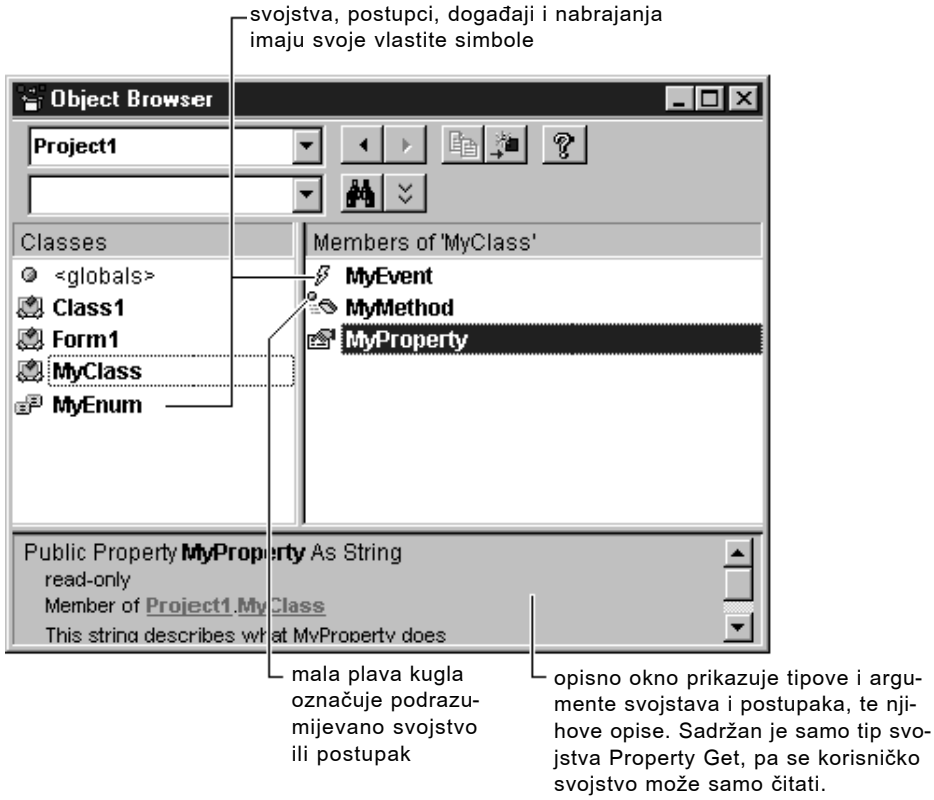
U pravilu, pretraživač objekata ne može biti usidren uz druge prozore. To vam omogućuje prebacivanje između pretraživača objekata i kodnog prozora korištenjem CTRL + TAB. Možete to promijeniti klikom desnom tipkom miša na pretraživač objekata za otvaranje njegovog pomoćnog izbornika, te klikom stavke Dockable.

Napomena Kad je pretraživač objekata usidren, ne možete koristiti CTRL + TAB za pomicanje u taj prozor iz vaših kodnih prozora.

Sadržaji pretraživača objekata

Pretraživač objekata prikazuje informacije u hijerarhiji na tri razine, kao što je prikazano na slici 9.2. Sa početkom od vrha, možete odabrati među dostupnim projektima i bibliotekama, uključujući vaše vlastite projekte Visual Basica, korištenjem okvira Project/Library.

Slika 9.2 Pregled elemenata klase u pretraživaču objekata



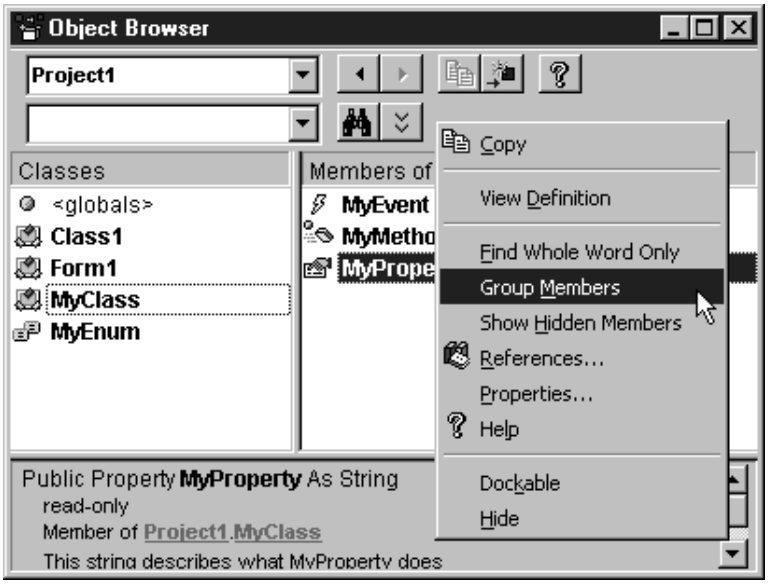
- Kliknite na klasu u listi Classes kako bi vidjeli njezin opis u opisnom oknu na dnu. Svojstva, postupci, događaji i konstante klase pojavit će se u listi Members na desnoj strani. Dostupne klase su izvučene iz projekta ili biblioteke odabrane u okviru Project/Library, ili iz svih projekata i biblioteka ako je odabrano <All Libraries>.
- Možete vidjeti argumente i povratne vrijednosti elemenata odabrane klase, kad kliknete na element u listi Members. Opisno okno na dnu pretraživača objekata prikazuje te informacije.
- Možete skočiti na biblioteku ili projekt koji sadrži element klikom na ime biblioteke ili objekta u upisnom oknu. Možete se vratiti klikom na gumb Go Back na vrhu pretraživača objekata.

  Savjet Kad ste u listi Classes ili listi Members, upis prvog karaktera u imenu pomaknut će listu na prvo iduće ime koje počinje s tim karakterom.

Kontroliranje sadržaja pretraživača objekata

Pomoćni izbornik, prikazan na slici 9.3, pruža alternativu gumbima Copy i View Definition na pretraživaču objekata. On vam također omogućuje otvaranje dijaloškog okvira References, i – ako je odabrana klasa ili element – pregled svojstava odabrane stavke. Možete odrediti opise vaših vlastitih objekata korištenjem te stavke izbornika, kao što je opisano u odlomku "Dodavanje opisa vaših objekata" kasnije u ovom poglavlju.

Slika 9.3 Pomoćni izbornik pretraživača objekata



Klik desnom tipkom miša na pretraživaču objekata poziva pomoćni izbornik. Kao dodatak malo prije spomenutim funkcijama, pomoćni izbornik kontrolira sadržaje lista Classes i Members.

- Kad je potvrđena stavka Group Members, zajedno su grupirana sva svojstva objekta, zajedno su grupirani svi postupci, i tako dalje. Kad stavka Group Members nije potvrđena, lista Members je složena abecednim redom.
- Kad je potvrđena stavka Show Hiddden Members, liste Classes i Members prikazuju informacije označene kao skrivene u tipskoj biblioteci. Obično, nemate potrebe za pregledom tih informacija. Skriveni elementi se prikazuju u svijetlosivoj boji.

  Savjet Kad je potvrđena stavka Group Members, upis prvog karaktera u imenu skočit će na prvo iduće ime koje počinje tim karakterom, čak i ako je ime u drugoj grupi.

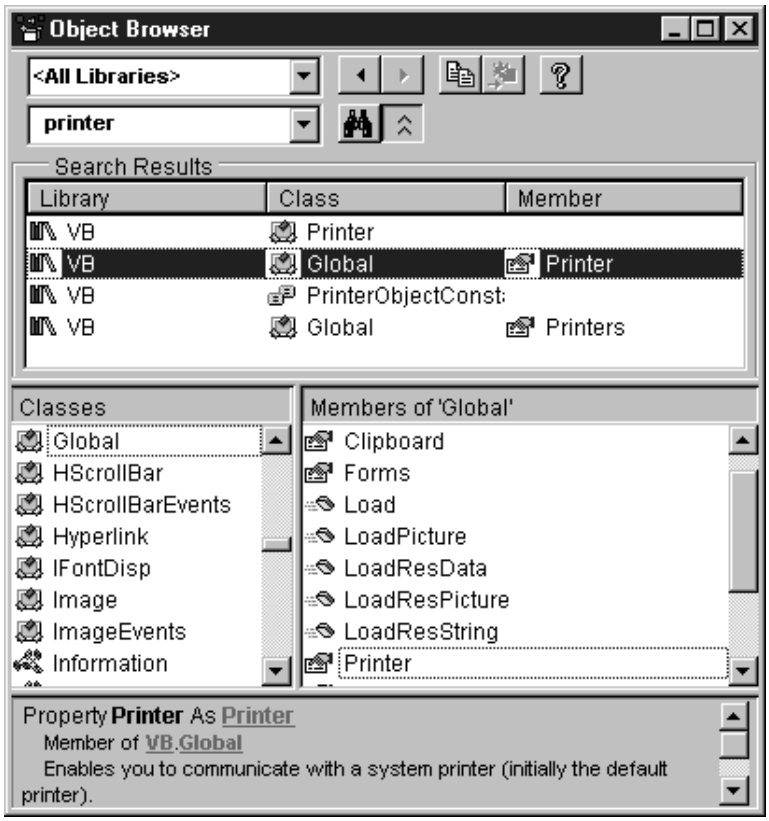
Pronalaženje i pretraživanje objekata

Pretraživač objekata možete upotrijebiti za pronalaženje objekata i njihovih elemenata, te za prepoznavanje projekata ili biblioteka iz kojih su došli.

Upišite tekst u okvir Search Text i kliknite gumb Search (ili pritisnite ENTER). Klase i elementi čija imena sadržavaju tekst koji ste odredili pojavit će se u listi Search Results.

Na primjer, slika 9.4 prikazuje rezultate upisa “printer” u okvir Search Text te klika na gumb Search.

Slika 9.4 Korištenje gumba Search



Možete odabrati stavku u listi Search Results, te pogledati njezin opis u opisnom oknu na dnu pretraživača objekata. Klik na podvučene skokove u opisnom oknu odabire označenu biblioteku ili dovodi do objekta ili elementa.

Možete ograničiti traženje na stavke koje točno odgovaraju tekstu u okviru Search tako da potvrdite stavku Find Whole Word Only u pomoćnom izborniku.

Dodavanje opisa vaših objekata

Pretraživač objekata možete upotrijebiti za dodavanje opisa te brojeva HelpContextID vašim potprogramima, modulima, klasama, svojstvima i postupcima. Takvi opisi mogu vam biti korisni kad radite s vlastitim klasama.

Napomena Opise svojstava, postupaka i događaja možete također dodati korištenjem dijaloškog okvira Procedure Attributes, dostupnog iz izbornika Tools.

Kako unijeti opise te povezati vaše klase i njihove elemente s temama pomoći

1. Pritisnite F2 za otvaranje **pretraživača objekata**. U okviru **Project/Library**, odaberite vaš projekt.
2. U listi **Classes**, kliknite desnom tipkom miša na ime klase za dobivanje pomoćnog izbornika, pa kliknite stavku **Properties** za otvaranje dijaloškog okvira **Member Options**.

   Alternativno, u listi **Members** možete desnom tipkom miša kliknuti svojstvo, postupak ili događaj koji ste dodali klasi. U pomoćnom izborniku kliknite stavku **Properties**. Ako je element tipa Private ili Friend, to će otvoriti dijaloški okvir **Member Options**. Ako je element tipa Public – znači, dio sučelja klase – to će umjesto toga otvoriti dijaloški okvir **Procedure Attributes**.

   Napomena Razlika između ova dva dijaloška okvira je u tome što dijaloški okvir **Procedure Attributes** ima gumb **Advanced** koji može biti upotrijebljen za stvaranje podrazumijevanog elementa klase, kao što je opisano u odlomku "Stvaranje podrazumijevanog svojstva ili postupka" kasnije u ovom poglavlju.
3. U okviru **Help Context ID** upišite kontekstni identifikacijski broj teme pomoći koja će biti prikazana ako kliknete gumb "?" kad je ta klasa ili element odabran u **pretraživaču objekata**.

   Napomena Možete stvoriti datoteku pomoći za vlastitu upotrebu, te povezati teme s svojim klasama i njihovim elementima. Za određivanje datoteke pomoći vašeg projekta, upotrijebite karticu **General** dijaloškog okvira **Project Properties**, dostupnog iz izbornika **Project**.
4. U okviru **Description** upišite kratak opis klase ili elementa.
5. Kliknite **OK** za povratak u **pretraživač objekata**. Tekst opisa kojeg ste upisali trebao bi se pojaviti u opisnom oknu na dnu pretraživača.
1. Ponovite korake od 2 do 5 za svaku klasu i za svaki element svake klase.

   Napomena Pretraživaču ne možete dostaviti tekstove ili teme pomoći za nabrajanja.

Za više informacija Nabrajanja su predstavljena u odlomku "Korištenje nabrajanja za rad s nizovima konstanti" u 8. poglavlju "Više o programiranju".

Kretanje među potprogramima

Pretraživač objekata možete upotrijebiti za brzo pomicanje na programski kod klase, modula ili potprograma u svom projektu.

Kako se pomaknuti na klasu, modul ili potprogram

1. (Neobavezno) Odaberite vaš projekt u okviru **Project/Library**.

   Korak 1 je neobavezan ako u okviru **Project/Library** imate odabrano **<All Libraries>**, jer su tad uključeni svi vaši projekti.

1. Imena klasa, modula i elemenata koji pripadaju vašim projektima prikazana su podebljanim pismom. Dvostruki klik na bilo koje ime ispisano podebljanim pismom dovest će do te klase, modula ili elementa (isti učinak ima i klik desnom tipkom miša na ime te odabir stavke **View Definition** u pomoćnom izborniku).

   Odabrana stavka prikazuje se u kodnom prozoru.

Pretraživanje objekata iz drugih aplikacija

Iz Visual Basica možete pristupiti i kontrolirati objekte isporučene od drugih aplikacija. Na primjer, ako na vašem sustavu imate aplikacije Microsoft Project i Microsoft Excel, možete upotrijebiti objekt Graph iz Microsoft Excela i objekt Calendar iz Microsoft Projecta kao dijelove svoje aplikacije.

Pretraživač objekata možete upotrijebiti za istraživanje tipskih biblioteka drugih aplikacija. *Tipska biblioteka* pruža informacije o objektima koje pružaju druge aplikacije.

Napomena U listi Project/Library, postoje odvojeni unosi za Visual Basic (VB) i Visual Basic za aplikacije (VBA). Iako govorimo o "objektima koje pruža Visual Basic", uočit ćete da je objekt Collection pribavljen od VBA.

Vašem projektu možete dodati biblioteke tako da odaberete stavku References iz pomoćnog izbornika pretraživača objekata, što će otvoriti dijaloški okvir References.

Za više informacija Za više detalja o korištenju automatizacije za kombiniranje i upravljanje objektima iz drugih aplikacija, pogledajte 10. poglavlje "Programiranje sastavnim dijelovima".

Stvaranje vlastitih klasa

Ako ste iskusan programer, već imate biblioteku korisnih funkcija koje ste napisali tijekom godina. Objekti ne zamjenjuju funkcije – i dalje ćete pisati i koristiti uslužne funkcije, ali pružaju prikladan, logičan način organiziranja potprograma i podataka.

Gledajući detaljno, klase iz kojih stvarate objekte spajaju podatke i potprograme u cjelinu.

Klase: zajedničko postavljanje korisnički određenih tipova i potprograma

Korisnički određeni tipovi su moćan alat za grupiranje povezanih dijelova podataka. Uzmite u obzir, na primjer, korisnički određen tip imena udtAccount određen ovdje:

```
Public Type udtAccount
  Number As Long
  Type As Byte
  CustomerName As String
  Balance As Double
End Type
```

Možete odrediti varijablu tipa udtAccount, pojedinačno odrediti vrijednosti njezinih polja, te zatim proslijediti cijeli zapis potprogramu koji ga ispisuje, snima u bazu podataka, izvodi računanja na njemu, provjerava valjanost njegovih polja i tako dalje.

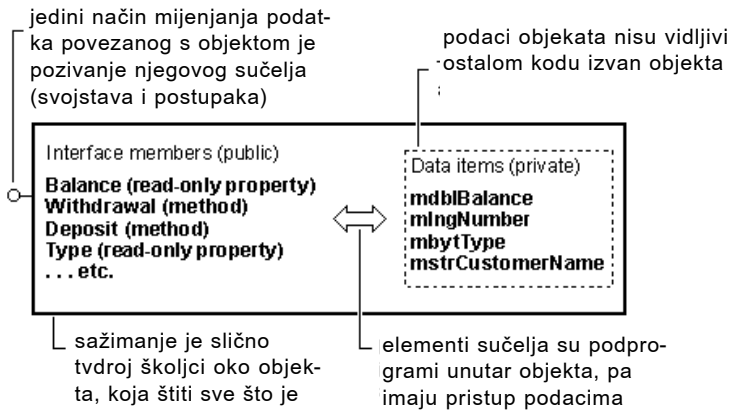
Moćni kakvi jesu, korisnički određeni tipovi predstavljaju programeru neke probleme. Možete stvoriti potprogram Withdrawal koji izaziva pogrešku ako odstupanje prekorači saldo u računu, ali ne postoji ništa što bi spriječilo polje Balance da bude smanjeno iz drugog programskog koda u vašoj aplikaciji.

Drugim riječima, povezanost potprograma i korisnički određenih tipova ovisi o disciplini, pamćenju i znanju programera koji održava programski kod.

Objekti: korisnički određeni tipovi s stavom

Objektno orijentirano programiranje rješava ovaj problem spajanjem podataka i potprograma u jednom unosu, kao što je prikazano na slici 9.5.

Slika 9.5 Objekti spajaju podatke i potprograme



Kad korisnički određen tip udtAccount postane klasa Account, njezini podaci postaju privatni, i potprogrami koji joj pristupaju postavljaju se unutar klase te postaju svojstva i postupci. To je ono što se podrazumijeva pod pojmom *sžimanje* (*encapsulation*) - znači, objekt je cjelina (čahura, ako želite) koja sadrži i programski kod i podatke.

Kad iz klase stvorite objekt Account, jedini način kojim možete pristupiti podacima je kroz svojstva i postupke koji čine njegovo sučelje. Sljedeći dio programskog koda pokazuje kako potprogrami unutar klase Account podržavaju sžimanje:

```
‘ Saldo računa je skriven vanjskom kodu.
Private mdblBalance As Double

‘ Svojstvo Balance samo za čitanje dopu{ta
‘ vanjskom kodu da pronađe saldo računa.
Public Property Get Balance() As Double
  Balance = mdblBalance
End Property
```

```
‘ Postupak Withdrawal mijenja saldo računa,
‘, ali samo ako se ne pojavi pogreška prekoračenja.
Public Sub Withdrawal(ByVal Amount As Double)
  If Amount > Balance Then
    Err.Raise Number:=vbObjectError + 2081, _
    Description:=”Prekoračenje”
  End If
  mdblBalance = mdblBalance - Amount
End Sub
```

Na trenutak se nemojte zabrinjavati kako ćete dohvatiti potprograme unutar klase, ili kako ćete razumjeti sintaksu potprograma svojstava i privatnih varijabli. Važno je zapamtiti da možete odrediti objekt koji sžima i provjerava valjanost svojih vlastitih podataka.

S objektom Account, nikad nećete biti zabrinuti jeste li pozvali prave potprograme za ažuriranje računa, jer su jedini potprogrami koje možete pozvati ugrađeni u objekt.

Za više informacija Sljedeći odlomak “Prilagođivanje klasa forme” postavlja stvaranje postupaka i svojstava u okvir s kojim ste već upoznati. Odlomak “Dodavanje svojstava i postupaka klasi”, kasnije u ovom poglavlju, objasnit će sintaksu.

O korisnički određenim tipovima možete pročitati u odlomku “Stvaranje vlastitih tipova podataka” u 8. poglavlju “Više o programiranju”.

Za detalje o potprogramima tipa Sub i Function, pogledajte “Uvod u potprograme” u 5. poglavlju “Osnove programiranja”.

Prilagođavanje klasa forme

Može vas iznenaditi kad saznate da ste stvarali klase jednako dugo koliko programirate u Visual Basicu. To je istina: Form1, poznati stanovnik svakog projekta kojeg ste ikad započeli, je zapravo – klasa.

Kako bi to vidjeli, otvorite novi standardni izvršni projekt. Dodajte naredbeni gumb na formu Form1, i postavite sljedeći programski kod u njegov događaj Click:

```
Private Sub Command1_Click()
  Dim f As New Form1
  f.Show
End Sub
```

Pritisnite F5 za pokretanje projekta, te kliknite gumb. Tako mu svega, evo još jednog primjera forme Form1! Kliknite njezin gumb. Evo još jedne forme! Svaki primjer koji stvorite izgleda isto, i ima isto ponašanje, jer su sve one primjeri klase Form1.

Što se ovdje događa?

Ako ste pročitali “Rad s objektima” u 5. poglavlju “Osnove programiranja”, znate da varijabla objekta određena s As New sadrži Nothing sve dok je prvi put ne pozovete u programskom kodu. Kad prvi put upotrijebite varijablu, Visual Basic primjećuje da ona sadrži posebnu vrijednost Nothing, i stvara primjer klase (i dobro je što to tako čini, jer bi izraz f.Show izazvao pogrešku).

Ja i moja skrivena opća varijabla

Možda se čudite kako to da možete pozivati formu Form1 u programskom kodu, kao da je ona varijabla objekta. Tu nema nikakve čarolije. Visual Basic stvara skrivenu opću varijablu objekta za svaku klasu forme. To je kao da Visual Basic ima sljedeće određivanje dodano vašem projektu:

```
Public Form1 As New Form1
```

Kad odaberete formu Form1 kao početni objekt, ili upišete Form1.Show u kodu, upućujete se na tu skrivenu opću varijablu objekta. Budući da je ona određena s As New, primjer klase Form1 se stvara kad prvi put upotrijebite tu prije određenu varijablu u kodu.

Razlog zašto je to određivanje skriveno je što ga Visual Basic mijenja svaki put kad promijenite svojstvo Name forme. Na taj način, skrivena varijabla je uvijek istog imena kao i klasa forme.

Vrlo kratak kviz

Koji je od primjera forme Form1 koje ste u prethodnoj vježbi stvorili povezan s skrivenom općom varijablom? Ako je vaš odgovor da je to prvi, u pravu ste. Forma Form1 je podrazumijevan početni objekt projekta, i za Visual Basic je to kao korištenje prije određene opće varijable Form1 u programskom kodu.

> **Savjet** Nakon izbacivanja forme, uvijek bi trebali postaviti vrijednost Nothing u sve pokazivače na formu kako bi oslobodili memoriju i izvore koje je forma koristila. Najčešće neopažen pokazivač je skrivena opća varijabla forme.

Što je s svim ostalim primjerima forme Form1?

U 5. poglavlju “Osnove programiranja”, naučili ste da za upućivanje na objekt, trebate varijablu objekta, te da objekt postoji samo dok postoji bar jedna varijabla objekta koja sadrži pokazivač na njega. Dakle što održava na životu sve te ostale primjere?

Drugi primjer forme Form1, i svi ostali koji slijede, imali su varijablu objekta tako dugo koliko je trebalo za poziv njihovih postupaka Show. Tada je ta varijabla otišla izvan područja, i postavljena je na Nothing. Visual Basic ipak posjeduje posebnu zbirku s imenom Forms, o kojoj možete pročitati u odlomku “Više o formama” u 6. poglavlju “Stvaranje korisničkog sučelja”. Zbirka Forms sadrži pokazivače na svaku učitanu formu u vašem projektu, pa ih tako uvijek možete pronaći i kontrolirati.

Napomena Kao što ćete naučiti, to nije točno za sve klase. Na primjer, klase koje vi oblikujete neće imati skrivene opće varijable ili opće zbirke koje će ih pratiti – to su posebne osobine klasa formi. Međutim, možete odrediti vaše vlastite opće varijable, i možete stvoriti vaše vlastite zbirke – kao što je opisano u odlomku "Stvaranje vlastitih klasa zbirki", kasnije u ovom poglavlju.

Svojstva, postupci i događaji klasa formi

Kad prvi put dodate svojstvo klasi forme, vjerojatno ćete to učiniti vizualno, postavljanjem naredbenog gumba (ili neke druge kontrole) na formu Form1. Kad to napravite, dodali ste klasi forme svojstvo Command1 koje se može samo čitati. Nakon toga, pozivate to svojstvo forme Form1 svaki put kad trebate pozvati postupak ili svojstvo naredbenog gumba:

```
Command1.Caption = "Klikni me"
```

Kad promijenite svojstvo Name bilo koje kontrole na formi, Visual Basic tiho mijenja ime svojstva samo za čitanje, tako da se oni uvijek podudaraju.

Ako još uvijek imate otvoren projekt iz prethodne vježbe, možete vidjeti to svojstvo Command1 pritiskom na F2 za otvaranje pretraživača objekata. U okviru Project/Library, odaberite Project1. Vidjet ćete formu Form1 u oknu Classes. U oknu Members, pomaknite se prema dolje dok ne pronađete Command1, i odaberite ga.

Command1 ima pored sebe simbol svojstva, a ako pogledate u opisno okno, vidjet ćete da je to svojstvo tipa WithEvents. Kao što ćete naučiti u odlomku "Dodavanje događaja klasi", kasnije u ovom poglavlju, to znači da svojstvo (ili varijabla objekta) ima potprograme događaja koji su mu pridruženi. Jedan od tih potprograma događaja, Command1_Click(), možda je bilo prvo mjesto gdje ste napisali neki programski kod Visual Basica.

Sačekajte, ima još

Postavljanje kontrola na formu nije jedini način dodavanja novih elemenata klasi forme. Možete dodati svoja vlastita korisnička svojstva, postupke i događaje, lako kao što stvarate nove varijable i potprograme.

Kako bi to vidjeli, dodajte sljedeći programski kod u odjeljak Declarations forme Form1:

```
‘ Svojstvo Komentar klase Form1.
Public Comentar As String
```

Dodajte sljedeći programski kod događaju Click forme Form1:

```
Private Sub Form1_Click()
  MsgBox Komentar "Moj komentar je:"
End Sub
```

Na kraju, promijenite programski kod u potprogramu događaja Command1_Click() dodavanjem linije, kako slijedi:

```
Private Sub Command1_Click()
  Dim f As New Form1
  f.Komentar = InputBox("Koji je moj komentar?")
  f.Show
End Sub
```

Pritisnite F5 za pokretanje projekta. Kliknite naredbeni gumb Command1, i kad se pojavi okvir za upis podataka, upišite neki sadržajan komentar i kliknite OK. Kad se pojavi novi primjer forme Form1, kliknite na njega za ispis njegovog svojstva Komentar.

Kliknite na prvi primjer forme Form1, i uočite da je njezino svojstvo Komentar prazno. Budući da Visual Basic stvara taj primjer kao početni objekt, nikad nećete imati priliku odrediti njegovo svojstvo Komentar.

Forme mogu međusobno pozivati postupke

Ako ste pažljivo pratili, mogli ste uočiti da programski kod koji ste dodali klasi Form1 nije postavio vlastito svojstvo Komentar – odredio je svojstvo Komentar *novog* primjera forme Form1 koju je stvorio.

Ta sposobnost formi da postave svojstva jedna drugoj i pozivaju jedna drugoj postupke je vrlo korisna tehnika. Na primjer, kad MDI forma otvara novi prozor-dijete, može pokrenuti novi prozor određivanjem njegovih svojstava i pozivanjem njegovih postupaka.

Ovu tehniku možete također upotrijebiti za prosljeđivanje informacija između formi.

Savjet Možete stvoriti korisničke događaje za forme. Odlomak "Dodavanje događaja formi" kasnije u ovom poglavlju, pruža postupak korak po korak.

Ostale vrste modula

Svojstva, postupke i događaje klasama forme dodajete postavljanjem programskog koda u njihove module koda. Na isti način, možete dodati svojstva, postupke i događaje modulima klase i – ako imate Professional ili Enterprise verziju Visual Basica – kodnim modulima UserControl i UserDocument.

Dok čitate odlomke "Dodavanje svojstava i postupaka klasi" i "Dodavanje događaja klasi", kasnije u ovom poglavlju, zapamtite da se sve što pročitate odnosi na klase formi kao i na module klase.

Za više informacija Što je to do vraga modul klase? Odlomak "Moduli klase korak po korak", koji slijedi, pokazuje kako odrediti klasu i prikazuje životni ciklus objekata koje stvarate iz te klase.

Moduli klase korak po korak

Ovaj primjer pokazuje kako možete upotrijebiti module klase za određivanje klasa, iz kojih zatim možete stvarati objekte. Također će vam pokazati kako stvoriti svojstva i postupke za nove klase, i prikazati kako se stvaraju i uništavaju objekti.

Otvorite novi standardni izvršni projekt, i ubacite modul klase odabirom stavke Add Class Module u izborniku Project. Stvorite četiri naredbena gumba na formi. Sljedeća tablica ispisuje vrijednosti svojstava koje trebate postaviti za objekte u ovom primjeru.

objekt	svojstvo	postavka
Modul klase	Name	stvar
Command1	Caption	pokaži stvar
Command2	Caption	okreni ime stvari
Command3	Caption	stvori novu stvar
Command4	Caption	privremena stvar

Napomena Moduli klase snimaju se u datoteke s nastavkom .cls.

U odjeljku Declarations modula klase, dodajte sljedeće:

```
Option Explicit
Public Ime As String
Private mdtmStvoreno As Date
```

Varijabla Ime bit će svojstvo objekta Stvar, jer je određena kao javna.

Napomena Ne miješajte ovo svojstvo Ime s svojstvom Name modula klase, koje ste trebali postaviti prema gornjoj tablici (svojstvo Name modula klase daje klasi Stvar svoje ime). Zašto bi dali klasi Stvar svojstvo Name? Bolje pitanje bi bilo, zašto ne? Možda poželite klasi Stvar dati svojstvo Name zato jer stvari imaju imena! Zapamtite da ovdje ne postoji ništa posebnog o imenima svojstava i postupaka koje koristi Visual Basic. Možete upotrijebiti ta ista imena svojstava i postupaka za vaše klase.

Varijabla mdtmStvoreno je privatni element podataka koji se koristi za spremanje vrijednosti svojstva Stvoreno koje je samo za čitanje. Svojstvo Stvoreno vraća datum i vrijeme kad je objekt Stvar stvoren. Kako bi uključili svojstvo Stvoreno, dodajte sljedeće svojstvo Primi odjeljku Declarations modula klase:

```
Property Primi Stvoreno() As Date
  Stvoreno = mdtmStvoreno
End Property
```

Napomena Ako ste dodali potprogram svojstva korištenjem dijaloškog okvira Add Procedure, u izborniku Tools, obrišite određivanje Property Let koje se automatski dodaje ovim dijalogom. Izraz Property Let je potreban samo za svojstva koja se mogu i čitati i zapisivati, i objašnjen je u odlomku "Postavljanje potprograma svojstava da rade za vas", kasnije u ovom poglavlju.

Objekt Stvar ima jedan postupak, OkreniIme, koji jednostavno okreće redoslijed slova u svojstvu Ime. On ne vraća vrijednost, pa je uključen u potprogram tipa Sub. Dodajte sljedeći potprogram tipa Sub modulu klase.

```
Public Sub OkreniIme()
  Dim intBr As Integer
  Dim strNovi As String
  For intBr = 1 To Len(Ime)
    strNovi = Mid$(Ime, intBr, 1) & strNovi
  Next
  Ime = strNovi
End Sub
```

Moduli klase imaju dva događaja, Initialize i Terminate. U spuštajućem okviru Object modula klase, odaberite Class. Spuštajući okvir Procedure će prikazati događaje. Postavite sljedeći programski kod u potprograme događaja:

```
Private Sub Class_Initialize()
  ‘ Određivanje datuma/vremena kad je objekt stvoren,
  ‘ koji će biti vraćeni svojstvu Stvoreno samo za čitanje.
  mdtmStvoreno = Now
  ‘ Prikaz svojstava objekta.
  MsgBox “Ime: “ & Ime & vbCrLf & “Stvoreno: “ _
  & Stvoreno “Stvar pokrenuta”
  End Sub

Private Sub Class_Terminate()
  ‘ Prikaz svojstava objekta.
  MsgBox “Ime: “ & Ime & vbCrLf & “Stvoreno: “ _
  & Stvoreno “Stvar završena”
End Sub
```

Obično, potprogram događaja Initialize sadrži onaj programski kod koji treba biti izveden u trenutku kad je stvoren objekt, kao što je pružanje obilježavanja vremena za svojstvo Stvoreno. Događaj Terminate sadrži bilo koji programski kod koji se treba izvesti za čišćenje nakon objekta kad je uništen.

U ovom primjeru, ova dva događaja su korištena prvenstveno kako bi vam dali vidljiv znak da je objekt Stvar stvoren ili uništen.

Korištenje objekta Stvar

Dodajte ovo određivanje u odjeljak Declarations modula forme:

```
Option Explicit
Private mth As Stvar
```

Varijabla mth će sadržavati pokazivač na objekt Stvar, koji će biti stvoren u događaju forme Load. Postavite sljedeći programski kod u potprogram događaja Form_Load, te u potprograme događaja Click za četiri naredbena gumba.

```
Private Sub Form_Load()
  Set mth = New Stvar
  mth.Ime = InputBox("Upišite ime stvari")
End Sub

' Gumb "Pokaži stvar"
Private Sub Command1_Click()
  MsgBox "Ime: " & mth.Ime & vbCrLf _
  & "Stvoreno: " & mth.Stvoreno "Oblik stvari"
End Sub

' Gumb "Okreni ime stvari"
Private Sub Command2_Click()
  mth.OkreniIme
  ' Klik gumba "Pokaži stvar"
  Command1.Value = True
End Sub

' Gumb "Stvori novu stvar"
Private Sub Command3_Click()
  Set mth = New Stvar
  mth.Name = InputBox("Upišite ime nove stvari")
End Sub

' Gumb "Privremena stvar"
Private Sub Comand4_Click()
  Dim stPrivremen As New Stvar
  stPrivremen.Ime = InputBox("Upišite ime za privremenu stvar")
End Sub
```

Pokretanje projekta

Pritisnite F5 za pokretanje projekta. Ako pogledate programski kod u potprogramu događaja Form_Load, možete vidjeti da je upotrijebljen operator New za stvaranje objekta Stvar. Pokazivač na taj objekt je dodijeljen varijabli mth.

Vidjet ćete okvir za unos podataka koji od vas traži ime stvari. Kad upišete ime i pritisnete ENTER, povratna vrijednost se dodjeljuje svojstvu Ime objekta Stvar.

Prikaz oblika objekta Stvar

Možete provjeriti da je svojstvo Ime dodijeljeno ako pritisnete prvi naredbeni gumb "Pokaži stvar", koji prikazuje okvir s porukom s svim svojstvima objekta Stvar.

Okretanje imena objekta Stvar

Pritisnite drugi naredbeni gumb "Okreni ime stvari". Ovaj gumb poziva postupak OkreniIme za preokretanje imena objekta Stvar, te zatim klika prvi gumb kako bi prikazao ažurirane vrijednosti svojstava.

Stvaranje novog objekta Stvar

Kliknite gumb "Stvori novu stvar" kako bi uništili postojeći objekt Stvar i stvorili novi (ili, kao preokrenuto, stvorili novi objekt Stvar te zatim uništili stari).

Operator New uzrokuje stvaranje novog objekta Stvar, pa ćete vidjeti okvir s porukom kojeg prikazuje događaj Initialize novog objekta Stvar. Kad kliknete OK, pokazivač na novi objekt Stvar se postavlja u varijablu mth na razini forme.

To briše pokazivač na stari objekt Stvar. Budući da ne postoji više pokazivača na njega, bit će uništen, i vidjet ćete okvir s porukom kojeg prikazuje njegov događaj Terminate. Kad kliknete OK, izraz okvira za unos podataka traži ime za novi objekt Stvar.

> **Napomena** Ako želite uništiti stari objekt Stvar prije stvaranja novog, možete dodati liniju koda Set mth = Nothing na početak potprograma događaja.

Privremeni objekt Stvar

Četvrti gumb prikazuje još jedan izgled postojanja objekta. Kad ga pritisnete, bit ćete upitani za ime privremenog objekta Stvar.

Pričekajte – još ne postoji privremeni objekt Stvar. Niste vidjeli okvir s porukom njegovog događaja Initialize. Kako mu onda možete dodijeliti ime?

Budući da je varijabla stPrivremen određena s As New, objekt Stvar bit će stvoren onog trenutka kad je pozvano jedno od njegovih svojstava ili postupaka. To će se dogoditi kad povratna vrijednost okvira za unos podataka bude dodijeljena svojstvu Ime. Upišite ime i kliknite OK na okviru za unos podataka.

Sada ćete vidjeti okvir s porukom događaja Initialize objekta Stvar, koji vam pokazuje da je svojstvo Ime još uvijek prazno. Kad kliknete OK za otpuštanje okvira s porukom, vrijednost iz okvira za unos podataka se konačno dodjeljuje svojstvu Ime. To je puno aktivnosti za jednu liniju programskog koda.

Naravno, odmah čim ste to napravili, završava se potprogram događaja Click, i varijabla `stPrivremen` odlazi van područja djelovanja. Otpušta se pokazivač objekta na privremeni objekt Stvar, pa ćete vidjeti okvir s porukom događaja Terminate objekta Stvar. Uočite da on sadrži ime koje ste mu dodijelili.

Svaki put kad kliknete ovaj gumb, stvara se idući privremeni objekt Stvar, dobiva ime i uništava se.

Zatvaranje aplikacije

Zatvorite aplikaciju klikom na gumb za zatvaranje forme. *Ne koristite gumb End na alatnoj traci*. Kad se aplikacija zatvori, uništava se forma Form1. Varijabla mth izlazi iz područja djelovanja, pa Visual Basic čisti pokazivače na objekt Stvar. Ne postoje preostali pokazivači na objekt Stvar, pa se i on uništava, i prikazuje se okvir s porukom iz događaja Terminate.

Pokrenite ponovno aplikaciju, ali je ovaj put završite korištenjem gumba End na alatnoj traci. Uočite da se *nije* prikazao okvir s porukom pozvan događajem Terminate objekta Stvar.

Važno je zapamtiti da završavanje vaše aplikacije gumbom End, ili naredbom End u vašem programskom kodu, *trenutno* zaustavlja aplikaciju, bez izvođenja događaja Terminate bilo kojeg objekta. Uvijek je bolje zatvoriti aplikaciju izbacivanjem ili zatvaranjem svih formi.

Može vam biti korisno pokretanje primjera pritiskom na F8 za prolaz kroz programski kod liniju po liniju. To je dobar način za razumijevanje redoslijeda događaja kod stvaranja i uništavanja objekata.

> **Važno** U stvarnoj aplikaciji, događaji Initialize i Terminate ne bi trebali sadržavati okvire s porukama, ili bilo koji drugi programski kod koji dopušta obrađivanje Windows poruka. Općenito, bolje je koristiti izraze Debug.Print kad istražujete trajanje objekata.

Za više informacija Forme i kontrole se ipak malo razlikuju od ostalih objekata, kao što je kasnije raspravljeno u ovom poglavlju u odlomku “Životni ciklus formi Visual Basica”.

Više o tome što možete napraviti s klasama i modulima klase možete pročitati kasnije u ovom poglavlju, u odlomcima “Dodavanje svojstava i postupaka klasi” i “Dodavanje događaja klasi”.

Ispravljanje modula klase

Ispravljanje modula klase neznatno se razlikuje od ispravljanja uobičajenih aplikacija. Razlog tome je što pogreška u svojstvu ili postupku modula klase uvijek djeluje kao obrađena pogreška (znači, uvijek postoji potprogram na stogu poziva koji može obraditi pogrešku – naime potprogram koji je pozvao svojstvo ili postupak modula klase).

Visual Basic nadoknađuje tu razliku pružanjem izbora za hvatanje pogrešaka Prekid u modulu klase (Break in Class Module), kao dodatak starijim izborima Prekid kod neobrađenih pogrešaka (Break on Unhandled Errors) i Prekid kod svih pogrešaka (Break on All Errors).

Napomena Možete odrediti Podrazumijevano stanje hvatanja pogreške (Default Error Trapping State) na kartici General dijaloškog okvira Options, dostupnog iz izbornika Tools. Izbor koji odaberete utječe na trenutan rad, i postaje podrazumijevan za sve sljedeće termine rada s Visual Basicom. Kako bi promijenili ovu postavku samo za trenutan rad, bez mijenjanja podrazumijevanog, odaberite Toggle iz pomoćnog izbornika kodnog prozora (koji je dostupan klikom desnom tipkom miša u kodnom prozoru).

Na primjer, pretpostavimo da modul klase Class1 sadrži sljedeći programski kod:

```
Public Sub Oops()
  Dim intOops As Integer
  intOops = intOops / 0
End Sub
```

Sad pretpostavimo da potprogram u drugom modulu klase, formi, ili standardnom modulu poziva element Oops:

```
Private Sub Command1_Click()
  Dim c1 As New Class1
  c1.Oops
End Sub
```

Ako je izbor hvatanja pogreške postavljen na Prekid kod neobrađenih pogrešaka (Break on Unhandled Errors), izvođenje se neće zaustaviti kod dijeljenja s nulom. Umjesto toga, pogreška će biti izazvana u pozivnom potprogramu, Command1_Click. Izvođenje će se zaustaviti na pozivu postupka Oops.

Možete upotrijebiti Prekid kod svih pogrešaka (Break on All Errors) za zaustavljanje kod dijeljenja s nulom, ali takav prekid je vrlo nezgodan izvor za većinu namjena. On zaustavlja izvođenje na svakoj grešci, čak i kod pogrešaka za koje ste napisali programski kod za obradu pogreške.

Prekid u modulu klase (Break in Class Module) je kompromisna postavka:

- Izvođenje se neće zaustaviti u kodu modula klase za kojeg ste napisali rukovatelja pogreškama.
- Izvođenje se zaustavlja samo kod pogrešaka koje su neobrađene u modulu klase, i zbog toga će biti vraćene pozivatelju postupka.
- Kad se pokrene razvojno okruženje Visual Basica, podrazumijevan je Prekid u modulu klase.
- Ako nema uključenih modula klase, Prekid u modulu klase je potpuno jednak kao Prekid kod neobrađenih pogrešaka.

Savjet Kad pogodite točku prekida korištenjem Prekida u modulu klase ili Prekida kod svih pogrešaka, možete zakoračiti ili proći pored pogreške – u vaš kod obrade pogreške ili u kod koji poziva potprogram u kojem se pojavila pogreška – pritiskom ALT + F8 ili ALT + F5.

Za više informacija Ispravljanje pogrešaka je detaljno raspravljeno u 13. poglavlju “Traženje i obrada pogrešaka”.

Životni ciklus formi Visual Basica

Budući da su vidljive korisniku, forme i kontrole imaju drugačiji životni ciklus od ostalih objekata. Na primjer, forma se neće zatvoriti samo zato što ste otpustili sve svoje pokazivače na nju. Visual Basic održava opću zbirku svih formi u vašem projektu, i uklanja formu iz te kolekcije samo kad izbacite formu.

Na sličan način, Visual Basic održava zbirku kontrola na svakoj formi. Možete učitati i izbaciti forme iz kontrolnih matrica, ali jednostavno otpuštanje svih pokazivača na kontrolu nije dovoljno za njezino uništenje.

Za više informacija Zbirke Forms i Controls su raspravljene u odlomku "Zbirke u Visual Basicu" ranije u ovom poglavlju.

Stanja kroz koja prolaze forme Visual Basica

Forma Visual Basica normalno prolazi kroz četiri stanja u svom životnom ciklusu:

1. Stvorena, ali nije učitana.
2. Učitana, ali nije prikazana.
3. Prikazana.
4. Memorija i izvori potpuno vraćeni.

Postoji i peto stanje u kojem forma može biti pod određenim okolnostima: Izbačena i bez pokazivača dok kontrola još uvijek ima pokazivače.

Ova tema opisuje ta stanja, te prijelaze među njima.

Stvorena, ali nije učitana

Početak ovog stanja je označen događajem Initialize. Programski kod kojeg postavite u potprogram događaja Form_Initialize je zbog toga prvi kod koji će biti izveden kad je forma stvorena.

U ovom stanju, forma postoji kao objekt, ali nema prozor. Još ne postoji ni jedna od njezinih kontrola. *Forma uvijek prolazi kroz ovo stanje*, iako u njemu može biti kratko.

Na primjer, ako izvedete Form1.Show, forma će biti stvorena, i izvest će se potprogram Form1_Initialize; čim je potprogram Form1_Initialize završen, forma će biti učitana, što je iduće stanje.

Ista stvar se događa kad odredite formu kao svoj početni objekt (Startup Object), na kartici General dijaloškog okvira Project Properties (koji je dostupan iz izbornika Project). Forma određena kao početni objekt stvara se odmah čim se pokrene projekt, te se odmah učitava i pokazuje.

Napomena Možete uzrokovati učitavanje vaše forme iz potprograma Form_Initialize, pozivom njezinog postupka Show ili pozivanjem njezinih ugrađenih svojstava i postupaka, kao što je opisano u nastavku.

Ostaje stvorena, ali nije učitana

Suprotno tome, sljedeći programski kod stvara primjer forme Form1 bez napredovanja forme u učitano stanje:

```
Dim frm As Form1
Set frm = New Form1
```

Kad je jednom završen potprogram Form_Initialize, jedini potprogrami koje možete izvesti bez prisiljavanja forme da se učita su potprogrami tipa Sub, Function i Property koje ste dodali u kodni prozor forme. Na primjer, možete dodati sljedeći postupak formi Form1:

```
Public Sub NoviPostupak()
  Debug.Print "Izvodim NoviPostupak"
End Sub
```

Možete pozvati ovaj postupak korištenjem varijable frm (znači, frm.NoviPostupak) bez prisiljavanja forme na iduće stanje. Na sličan način, možete pozvati NoviPostupak kako bi stvorili formu:

```
Dim frm As New Form1
frm.NoviPostupak
```

Budući da je varijabla frm određena s As New, forma neće biti stvorena sve dok varijabla ne bude prvi put iskorištena u kodu – u ovom slučaju, kad je pozvan NoviPostupak. Nakon izvođenja gornjeg koda, forma ostaje stvorena, ali nije učitana.

> **Napomena** Izvođenje Form1.NoviPostupak, bez određivanja varijable forme, ima isti učinak kao i gornji primjer. Kao što je objašnjeno u odlomku "Prilagođivanje klasa forme", Visual Basic stvara skrivenu opću varijablu za svaku klasu forme. Ta varijabla ima isto ime kao i klasa; to je kao da je Visual Basic odredio Public Form1 As New Form1.

Možete izvesti koliko želite korisničkih svojstava i postupaka bez prisiljavanja forme da se učita. Međutim, onog trenutka kad pristupite nekom od ugrađenih svojstava forme, ili bilo kojoj kontroli na formi, forma ulazi u iduće stanje.

> **Napomena** Od velike pomoći može vam biti ako razmišljate o formi kao da ima dva dijela, kodni dio i vidljivi dio. Prije učitavanja forme, u memoriji je samo kodni dio. Možete pozvati proizvoljan broj potprograma iz kodnog dijela bez učitavanja vidljivog dijela forme.

Jedino stanje kroz koje prolaze sve forme

Stvorena, ali nije učitana je jedino stanje kroz koje prolaze *sve* forme. Ako je varijabla frm u gornjem primjeru postavljena na Nothing, kao što je ovdje pokazano, forma će biti uništena prije ulaza u iduće stanje:

```
Dim frm As New Form1
frm.NoviPostupak
Set frm = Nothing                ' Forma je uništena.
```

Forma upotrijebljena na takav način nije bolja od modula klase, pa velika većina formi prolazi u iduće stanje.

Učitana, ali nije prikazana

Događaj koji označuje početak ovog stanja je poznati događaj Load. Kod koji postavite u potprogram događaja Form_Load bit će izveden odmah čim forma uđe u učitano stanje.

Kad započne potprogram događaja Form_Load, stvorene su i učitane sve kontrole na formi, i forma ima prozor – zajedno s rukovanjem prozora (window handle, hWnd) i sadržajem uređaja (device context, hDC) - iako taj prozor još nije prikazan.

Svaka forma koja postaje vidljiva mora prvo biti učitana.

Većina formi automatski prolazi iz stanja Stvorena, ali nije učitana u stanje Učitana, ali nije prikazana. Forma će automatski biti učitana:

- Ako je forma određena kao početni objekt, na kartici General dijaloškog okvira Project Properties.
- Ako je postupak Show prvo svojstvo ili postupak forme koje je pozvano, kao na primjer Form1.Show.
- Ako je prvo svojstvo ili postupak forme koje je pozvano jedno od ugrađenih elemenata forme, kao na primjer postupak Move.

  Napomena Ovaj slučaj uključuje sve kontrole na formi, pošto svaka kontrola određuje svojstvo forme; to znači, ako pristupate svojstvu Caption kontrole Command1, morate proći kroz svojstvo forme Command1: Command1.Caption.
- Ako se koristi naredba Load za učitavanje forme, bez prethodnog korištenja izraza New ili As New za stvaranje forme, kao što je ranije opisano.

Forme koje se nikad ne prikazuju

U prva dva gore navedena slučaja forma će izravno prijeći u vidljivo stanje, čim se završi potprogram Form_Load. U posljednja dva slučaja, forma će ostati učitana, ali se neće prikazati.

Dugo je vremena bila uobičajena praksa programiranja u Visual Basicu učitati formu, ali je nikad prikazati. To se može učiniti zbog više razloga:

- Za korištenje kontrole mjerača vremena za stvaranje vremenskih događaja.
- Za upotrebu kontrola zbog njihove djelotvornosti, radije nego njihovog korisničkog sučelja – na primjer, za serijske komunikacije ili pristup datotečnom sustavu.
- Za izvođenje DDE transakcija.

  Napomena S verzijama Professional i Enterprise, možete stvoriti sastavne dijelove tipa ActiveX (prije zvane OLE poslužitelji), koji su često bolji u pružanju djelotvornosti samo za programski kod nego što su kontrole. Pogledajte “Stvaranje ActiveX sastavnih dijelova” u dijelu *Microsoft Visual Basic 6.0 Component Tools Guide* biblioteke *Microsoft Visual Basic 6.0 Reference Guide*.

Uvijek se vraća kući

Forme se vraćaju iz vidljivog stanja u učitano stanje svaki put kad su sakrivene. Unatoč tome, povrat u učitano stanje ne izvodi ponovno događaj Load. Potprogram Form_Load se izvodi samo jednom u životnom ciklusu forme.

Prikazana

Jednom kad forma postane vidljiva, korisnik s njom može međusobno djelovati. Nakon toga, forma može biti skrivena i prikazana koliko god puta želite prije nego što konačno bude izbačena.

Međuigra: priprema za izbacivanje

Forma može biti ili skrivena ili vidljiva kad se izbacuje. Ako nije izričito skrivena, ostaje vidljiva sve dok ne bude izbačena.

Posljednji događaj koji forma dobiva prije izbacivanja je događaj Unload. Prije pojavljivanja tog događaja, međutim, dobit ćete vrlo važan događaj nazvan QueryUnload. Događaj QueryUnload je vaša prilika da zaustavite izbacivanje forme. Ako postoje podaci koje bi korisnik trebao spremiti, to je vrijeme za upit korisnika želi li spremiti ili odbaciti promjene.

> **Važno** Postavljanje argumenta Cancel događaja QueryUnload na True će spriječiti izbacivanje forme, onemogućujući naredbu Unload.

Jedno od najmoćnijih obilježja ovog događaja je što vam on kazuje kako je neminovno izbacivanje uzrokovano: korisnik je kliknuo gumb Close, vaša aplikacija je izvršila naredbu Unload, aplikacija se zatvara, ili se zatvaraju Windowsi. Zbog toga vam QueryUnload omogućuje da korisniku ponudite šansu za odustajanje od zatvaranja forme, dok vam i dalje dopušta zatvaranje forme iz programskog koda kad to trebate.

> **Važno** Pod određenim okolnostima, forma neće primiti događaj QueryUnload: ako koristite naredbu End za završetak vaše aplikacije, ili ako kliknete gumb End (ili odaberete stavku End iz izbornika Run) u razvojnom okruženju.

Za više informacija Pogledajte “Događaj QueryUnload” u dijelu *Microsoft Visual Basic 6.0 Language Reference* biblioteke *Microsoft Visual Basic 6.0 Langugage Reference Library*.

Povrat u stvoreno, ali ne u učitano stanje

Kad je forma izbačena, Visual Basic je miče iz zbirke Forms. Osim ako negdje niste sčuvali varijablu s pokazivačem na formu, forma će biti uništena, a njezina memorija i izvori će biti vraćeni Visual Basicu.

Ako negdje u varijabli čuvate pokazivač na formu, kao što je skrivena opća varijabla opisana u odlomku “Prilagođavanje klasa forme” ranije u ovom poglavlju, tada se forma vraća u stanje Stvorena, ali nije učitana. Forma više nema prozor, a njezine kontrole više ne postoje.

Objekt forme i dalje zauzima izvore i memoriju. Svi podaci u varijablama na razini modula u kodnom dijelu forme i dalje postoje (međutim, statične varijable u potprogramima događaja su nestale).

Pokazivač koji čuvate možete upotrijebiti za pozivanje postupaka i svojstava koje ste dodali formi, ali ako pozovete ugrađene elemente forme, ili pristupite njezinim kontrolama, forma će se ponovno učitati, i izvest će se potprogram Form_Load.

Memorija i izvori potpuno vraćeni

Jedini način otpuštanja cijele memorije i svih izvora je izbaciti formu i zatim postaviti sve pokazivače na Nothing. Pokazivač koji se najčešće previdi pri takvom djelovanju je ranije spomenuta skrivena opća varijabla. Ako se u bilo koje vrijeme uputite na formu njezinim imenom klase (koje je prikazano u prozoru s svojstvima svojstvom Name), upotrijebili ste skrivenu opću varijablu. Kako bi oslobodili memoriju koju zauzima forma, morate tu varijablu postaviti na Nothing. Na primjer:

```
Set Form1 = Nothing
```

Vaša forma će primiti njezin događaj Terminate upravo prije nego što bude uništena.

Savjet Većina profesionalnih programera izbjegava korištenje skrivene opće varijable, i daje prednost određivanju njihovih vlastitih varijabli (na primjer, Dim dlgOpis As New frmOkvirOpisa) kako bi rukovali formom tijekom njezinog trajanja.

Napomena Izvođenje naredbe End izbacuje sve forme i postavlja sve varijable objekta u vašoj aplikaciji na Nothing. Međutim, to je vrlo iznenadan način ukidanja vaše aplikacije. Nijedna od vaših formi neće dobiti svoje događaje QueryUnload, Unload ili Terminate, a objekti koje ste stvorili neće dobiti svoje događaje Terminate.

Izbačena i bez pokazivača dok kontrola još uvijek ima pokazivače

Kako bi došli u ovo čudno stanje, trebate izbaciti i osloboditi formu dok čuvate pokazivač na jednu od njezinih kontrola. Ako vam to zvuči kao luckasta stvar za napraviti, budite uvjereni da je.

```
Dim frm As New Form1
Dim obj As Object
frm.Show vbModal
‘ Kad je modalna forma otpuštena, sprema se
‘ pokazivač na jednu od njezinih kontrola.
Set obj = frm.Command1
Unload frm
Set frm = Nothing
```

Forma je izbačena, i svi pokazivači na nju su otpušteni. Unatoč tome, i dalje imate pokazivač na jednu od njezinih kontrola, i to će održati kodni dio forme od otpuštanja memorije koju koristi. Ako pozovete bilo koje svojstvo ili postupak te kontrole, forma će biti ponovno učitana:

```
obj.Caption = “Natrag u život”
```

Vrijednosti u varijablama na razini modula će i dalje ostati sčuvane, ali će vrijednosti svojstava svih kontrola opet biti postavljene na podrazumijevane vrijednosti, kao da je forma učitana prvi put. Izvest će se potprogram Form_Load.

Napomena U nekim prethodnim verzijama Visual Basica, forma se nije potpuno ponovno pokrenula, i potprogram Form_Load se nije ponovno izvodio.

Napomena Ne ponašaju se sve forme kao forme Visual Basica. Na primjer, forme tipa Microsoft Forms dane u aplikaciji Microsoft Office nemaju događaje Load i Unload; kad te forme dobiju događaj Initialize, sve njihove kontrole već postoje i spremne su za upotrebu.

Za više informacija Forme su raspravljene u odlomku "Oblikovanje forme" u 3. poglavlju "Forme, kontrole i izbornici", te u odlomku "Više o formama" u 6. poglavlju "Stvaranje korisničkog sučelja".

Moduli klase protiv standardnih modula

Klase se razlikuju od standardnih modula po načinu na koji se spremaju podaci. Nikad ne postoji više od jedne kopije podataka standardnog modula. To znači da kad jedan dio vaše aplikacije promijeni javnu varijablu u standardnom modulu, a drugi dio vaše aplikacije kasnije pročita tu varijablu, dobit će istu vrijednost.

Podaci modula klase, s druge strane, postoje odvojeno od svakog primjera klase (znači, za svaki objekt stvoren iz klase).

Po istim obilježjima, podaci u standardnom modulu imaju kao područje djelovanja cijelu aplikaciju – znači, postoje cijelo vrijeme dok se izvodi aplikacija – dok podaci modula klase za svaki primjer klase postoje samo dok postoji i objekt; stvaraju se kad se stvara objekt, i uništavaju se kad se uništi objekt.

Na kraju, varijable određene tipom Public u standardnom modulu vidljive su bilo gdje iz vašeg projekta, dok se varijablama tipa Public u modulu klase može pristupiti ako imate varijablu objekta koja sadrži pokazivač na određen primjer klase.

Sve gore navedeno je također točno za javne potprograme u standardnim modulima i modulima klase. To je pokazano sljedećim primjerom. Možete pokrenuti taj kod otvaranjem novog standardnog izvršnog projekta te korištenjem izbornika Project za dodavanje standardnog modula i modula klase.

Postavite sljedeći programski kod u modul klase Class1:

```
‘ Slijedi svojstvo objekta Class1.
Public Comment As String

‘ Slijedi postupak objekta Class1.
Public Sub ShowComment()
  MsgBox Comment, gstrVisibleEverywhere
End Sub
```

Postavite sljedeći programski kod u standardni modul Module1:

```
‘ Kod u standardnom modulu je op}i.
Public gstrVisibleEverywhere As String

Public Sub CallableAnywhere(ByVal cl As Class1)
  ‘ Sljedeće linije mijenjaju opću varijablu (svojstvo)
  ‘ primjera Class1. Samo određen objekt proslije|en
  ‘ ovom potprogramu je pogođen.
  cl.Comment = “Touched by, a global Function.”
End Sub
```

Postavite dva naredbena gumba na formu Form1, i dodajte sljedeći kod formi Form1:

```
Private mc1First As Class1
Private mc1Second As Class1

Private Sub Form_Load()
  ‘ Stvaranje dva primjera klase Class1.
  Set mc1First = New Class1
  Set mc1Second = New Class1
  gstrVisibleEverywhere = “Global string data”
End Sub

Private Sub Command1_Click()
  Call CallableAnywhere(mc1First)
  mc1First.ShowComment
End Sub

Private Sub Command2_Click()
  mc1Second.ShowComment
End Sub
```

Pritisnite F5 za pokretanje projekta. Kad se učita forma Form1, stvara dva primjera klase Class1, gdje svaki primjer ima svoje vlastite podatke. Forma Form1 također postavlja vrijednost opće varijable `gstrVisibleEverywhere`.

Pritisnite naredbeni gumb Command1, koji poziva opći potprogram i prosljeđuje pokazivač prvom objektu tipa Class1. Opći potprogram postavlja svojstvo Comment, te gumb Command1 zatim poziva postupak ShowComment za prikaz podatka objekta.

Kao što pokazuje slika 9.6, nastali okvir s porukom prikazuje da je opći potprogram CallableAnywhere postavio svojstvo Comment objekta koji mu je bio proslijeđen, te da je opći string vidljiv iz klase Class1.

Slika 9.6 Okvir s porukom iz prvog objekta tipa Class1

Pritisnite gumb Command2, koji jednostavno poziva postupak ShowComment drugog primjera klase Class1.

Kao što pokazuje slika 9.7, oba objekta imaju pristup varijabli općeg stringa; ali svojstvo Comment drugog objekta je prazno, budući da je poziv općeg potprograma CallableAnywhere promijenio samo svojstvo Comment prvog objekta.

Slika 9.7 Okvir s porukom iz drugog objekta tipa Class1

Važno Izbjegavajte stvaranje koda u vašim klasama koji je ovisan o općim podacima – znači, javne varijable u standardnim modulima. Istovremeno može postojati puno primjera klase, i svi ti objekti dijele opće podatke u vašoj aplikaciji.

Korištenje općih varijabli u kodu modula klase također narušava objektno orijentirani programski koncept sžimanja, zato što objekti stvoreni iz takvih klasa ne sadrže sve njezine podatke.

Statični podatak klase

Mogu se pojaviti situacije kad trebate dijeliti jedan podatak među svim objektima stvorenim iz modula klase. Ponekad se to naziva kao *statični podatak klase* (*static class data*).

Ne možete ostvariti stvarni statični podatak klase u modulu klase Visual Basica. Međutim, možete ga oponašati korištenjem potprograma Property za postavljanje i vraćanje vrijednosti javnog podatkovnog elementa u standardnom modulu, kao u sljedećem dijelu programskog koda:

```
‘ Svojstvo samo za čitanje vra}a ime aplikacije.
Property Get CommonString() As String
  ‘ Varijabla gstrVisibleEverywhere je spremljena u
  ‘ standardnom modulu, i odre|ena kao Public.
  CommonString = gstrVisibleEverywhere
End Property
```

Napomena Ne možete upotrijebiti ključnu riječ Static za varijable na razini modula u modulu klase. Ključna riječ Static može se upotrijebiti samo unutar potprograma.

Moguće je oponašati statični podatak klase koji nije samo za čitanje pružanjem pripadajućeg potprograma Property Let – ili Property Set za svojstvo koje sadrži pokazivač objekta – za dodjelu nove vrijednosti podatkovnom elementu standardnog modula. Međutim, korištenje općih varijabli na taj način narušava pojam sžimanja, i nije preporučeno.

Na primjer, varijabla gstrVisibleEverywhere može biti postavljena bilo gdje iz vašeg projekta, čak i iz koda koji ne pripada klasi koja ima svojstvo CommonString. To može dovesti do tajanstvenih pogrešaka u vašoj aplikaciji.

Za više informacija Opći podaci u ActiveX sastavnim dijelovima traže drugačije rukovanje nego u uobičajenim aplikacijama. Ako imate Professional ili Enterprise verzije Visual Basica, pogledajte odlomak "Standardni moduli protiv modula klase" u 6. poglavlju "Opća načela oblikovanja sastavnih dijelova", u poglavlju "Stvaranje ActiveX sastavnih dijelova" dijela *Microsoft Visual Basic 6.0 Component Tools Guide* biblioteke *Microsoft Visual Basic 6.0 Reference Library*.

Dodavanje svojstava i postupaka klasi

Svojstva i postupci klase čine njezino podrazumijevano sučelje. Podrazumijevano sučelje je najuobičajeniji način upravljanja objektom.

Općenito, svojstva predstavljaju *podatke o objektu*, dok postupci predstavljaju *akcije koje objekt može poduzeti*. Recimo to na drugi način, svojstva pružaju opis objekta, dok su postupci njegovo ponašanje.

> **Važno** Sljedeća imena ne mogu biti upotrijebljena kao imena svojstava ili postupaka, jer pripadaju podređenim sučeljima IUnknown i IDispatch: QueryInterface, AddRef, Release, GetTypeInfoCount, GetTypeInfo, GetIDsOfNames i Invoke. Ta imena će izazvati pogrešku kod prevođenja.

Za više informacija Događaji su raspravljeni u odlomku "Dodavanje događaja klasi" kasnije u ovom poglavlju.

Dodavanje svojstava klasi

Najlakši način određivanja svojstava klase je dodavanje javnih varijabli modulu klase. Na primjer, mogli bi vrlo lako stvoriti klasu Račun određivanjem dvije javne varijable u modulu klase imena Račun:

```
Public Saldo As Double
Public Ime As String
```

To je prilično jednostavno. To je jednostavno kao i stvaranje privatnih podataka za klasu; jednostavno odredite varijablu kao Private, i ona će biti pristupačna samo iz programskog koda unutar modula klase:

```
Private mstrDjevojačkoPrezimeMajke As String
Private mintMjesečniOpozivDoDanas As Integer
```

Skrivanje podataka

Sposobnost zaštite dijela podataka nekog objekta, dok se ostatak izlaže kao svojstva, naziva se *skrivanje podataka* (*data hiding*). To je jedan oblik objektno orijentiranog načela sžimanja, kao što je objašnjeno u odlomku “Klase: Zajedničko postavljanje korisnički određenih tipova i potprograma”.

Skrivanje podataka znači da možete napraviti promjene u ostvarenju klase – na primjer, povećanje privatne varijable klase Račun mintMjesečniOpozivDoDanas s tipa Integer na tip Long – bez utjecaja na postojeći kod koji koristi objekt Račun.

Skrivanje podataka vam također omogućuje određivanje svojstava koja su samo za čitanje. Na primjer, možete upotrijebiti potprogram Property Get za povrat vrijednosti privatne varijable koja sadrži broj opoziva u mjesecu, dok povećavate samo varijablu iz unutrašnjosti koda objekta Račun. To nas dovodi do potprograma svojstava.

Potprogrami svojstava

Skrivanje podataka ne bi bilo od velike koristi kad bi jedini način stvaranja svojstava bio određivanjem javnih varijabli. Koliko bi vam bilo korisno dati svojstvo Tip klasi Račun, kad svaki programski kod koji ima pokazivač na objekt Račun može bez problema postaviti tip računa na bilo koju vrijednost?

Potprogrami svojstava vam omogućuju izvođenje programskog koda kad je vrijednost svojstva postavljena ili dohvaćena. Na primjer, možete ostvariti svojstvo Tip objekta Račun kao par potprograma tipa Property:

```
Public Enum TipoviRačuna
  trUšteđevina = 1
  trProvjera
  trVrstaKredita
End Enum

‘ Privatno spremanje podataka za svojstvo Tip.
Private mbytTip As TipoviRačuna

Public Property Get Tip() As TipoviRačuna
  Tip = mbytTip
End Property

Public Property Let Tip(ByVal NoviTip As TipoviRačuna)
  Select Case NoviTip
    Case trProvjera, trUšte|evina, trVrstaKredita
          ‘ Ne treba raditi ništa ako je NoviTip valjan.
    Case Else
          Err.Raise Number:=vbObjectError + 32112, _
          Description:=”Pogrešan tip računa”
  End Select
```

```
  If mbytTip > NoviTip Then
    Err.Raise Number:=vbObjectError + 32113, _
    Description:="Ne mogu smanjiti tip ra~una"
  Else
    mbytTip = NoviTip
  End If
End Property
```

Sad pretpostavite da imate varijablu imena rčn koja sadrži pokazivač na objekt Račun. Kad se izvede kod `x = rčn.Tip`, poziva se potprogram Procedure Get kako bi vratio vrijednost spremljenu u privatnom podatkovnom elementu `mbytTip` modula klase.

Kad je izveden programski kod `rčn.Tip = trProvjera`, poziva se potprogram Property Let. Ako je objekt Račun potpuno nov, varijabla `mbytTip` će biti nula, i može biti dodijeljen bilo koji valjani tip računa. Ako je trenutni tip računa trUštеđevina, račun će biti nadograđen.

Međutim, ako je trenutni tip računa trVrstaKredita, potprogram Property Let će oživjeti pogrešku, sprječavajući vraćanje u niži razred. Slično tome, ako se izvede kod `rčn.Tip = 0`, naredba Select u potprogramu Property Let će otkriti neispravan tip računa i oživjet će pogrešku.

Ukratko, *potprogrami svojstava omogućuju objektu da zaštiti i provjerava valjanost svojih vlastitih podataka*.

Za više informacija Jesu li onda javne varijable dobre za bilo što? Idući odlomak "Potprogrami svojstava protiv javnih varijabli" ističe prikladnu upotrebu obje vrste.

Sposobnosti potprograma svojstava su detaljnije istraženi u odlomku "Postavljanje potprograma svojstava da rade za vas", kasnije u ovom poglavlju.

Potprogrami svojstava protiv javnih varijabli

Potprogrami svojstava su očigledno tako moćna sredstva za omogućavanje sžimanja da ćete biti začuđeni ako se ikad budete gnjavili s javnim varijablama. Odgovor je, kao i uvijek u programiranju "Naravno – ponekad". Ovo su neka temeljna pravila:

Upotrijebite potprograme svojstava:

- Kad je svojstvo samo za čitanje, ili ne može biti promijenjeno jednom kad je određeno.
- Kad svojstvo ima prikladno određen skup vrijednosti kojima treba provjeriti valjanost.
- Kad su vrijednosti izvan određenog opsega – na primjer, negativne vrijednosti – valjane za tip podatka svojstva, ali uzrokuju pogreške u aplikaciji ako je svojstvu omogućeno da preuzme te vrijednosti.
- Kad postavljanje svojstva uzrokuje neke osjetne promjene u stanju objekta, kao na primjer svojstvo Visible.
- Kad postavljanje svojstva uzrokuje promjene ostalih unutarnjih varijabli ili vrijednosti ostalih svojstava.

Upotrijebite javne varijable za svojstva koja se mogu čitati i zapisivati:

- Kad samo svojstvo provjerava valjanost svoje vrijednosti. Na primjer, pojavit će se pogreška ili automatska pretvorba podatka ako vrijednost različita od True ili False bude dodijeljena varijabli tipa Boolean.
- Kad je valjana svaka vrijednost u opsegu podržana tipom podatka. To će biti točno za puno svojstava tipa Single ili Double.
- Kad je svojstvo tipa podatka String, a ne postoji ograničenje veličine ili vrijednosti stringa.

Napomena Ne ostvarujte svojstvo kao javnu varijablu samo da bi zaobišli nadgradnju poziva funkcije. Iza scene, Visual Basic će u svakom slučaju ostvariti javne varijable u vašim modulima klase kao parove potprograma svojstava, pošto su potrebni tipskim bibliotekama.

Postavljanje potprograma svojstava da rade za vas

Visual Basic pruža tri vrste potprograma svojstava, kao što je opisano u sljedećoj tablici.

potprogram	namjena
Property Get	Vraća vrijednost svojstva.
Property Let	Postavlja vrijednost svojstva.
Property Set	Postavlja vrijednost svojstva objekta (znači, svojstvo koje sadrži pokazivač na objekt).

Kao što možete vidjeti iz tablice, svaki od ovih potprograma svojstava ima posebnu ulogu u određivanju svojstva. Tipično svojstvo će biti sčinjeno od para potprograma svojstava: potprograma Property Get za dohvaćanje vrijednosti svojstva i potprograma Property Let ili Property Set za dodjelu nove vrijednosti.

Ove uloge se mogu preklapati u nekim slučajevima. Razlog zašto postoje dvije vrste potprograma svojstva za dodjelu vrijednosti je što Visual Basic ima posebnu sintaksu za dodjelu pokazivača objekta varijablama objekta:

```
Dim spr As Spravica
Set spr = New Spravica
```

Pravilo je jednostavno: Visual Basic poziva potprogram Property Set ako je upotrijebljena naredba Set, a potprogram Property Let ako nije.

Savjet Kako bi održali ispravnim potprograme Property Let i Property Set, poslušajte Basic programske jezike prošlosti, kad ste umjesto `x = 4` morali pisati `Let x = 4` (sintaksa koju Visual Basic podržava i danas). Visual Basic uvijek poziva potprogram svojstva koji odgovara tipu dodjeljivanja – Property Let za `Let x = 4`, i Property Set za `Set c1 = New Class1` (znači, svojstva objekata).

Za više informacija Odlomak "Rad s objektima" u 5. poglavlju "Osnove programiranja", objašnjava korištenje naredbe Set s varijablama objekata.

Svojstva za čitanje i pisanje

Sljedeći dio programskog koda pokazuje tipično svojstvo za čitanje i pisanje:

```
‘ Privatna pohrana za vrijednost svojstva.
Private mintBrojZuba As Integer

Public Property Get BrojZuba() As Integer
  BrojZuba = mintBrojZuba
End Property

Public Property Let BrojZuba(ByVal NovaVrijednost As Integer)
  ‘ (Kod za provjeru vrijednosti svojstva.)
  mintBrojZuba = NovaVrijednost
End Property
```

Ime privatne varijable koja sprema vrijednost svojstva sčinjeno je od prefiksa područja (`m`) koje je označava kao varijablu na razini modula; prefiksa tipa (`int`); te imena (`BrojZuba`). Korištenje istog imena kao i svojstvo služi kao podsjetnik da su varijabla i svojstvo povezani.

Kao što ste bez sumnje uočili, ovdje i u ranijim primjerima, imena potprograma svojstava koja uređuju svojstvo za čitanje i brisanje moraju biti ista.

> **Napomena** Potprogrami svojstava su u pravilu javni, pa ako izostavite ključnu riječ Public, i dalje će biti javni. Ako zbog nekog razloga želite da svojstvo bude privatno (znači, dostupno samo iz objekta), morate ga odrediti s ključnom riječi Private. Dobra je praksa koristiti ključnu riječ Public, čak iako nije potrebna, jer čini jasnim vaše namjere.

Potprogrami svojstava u radu i igri

Poučno je koračati kroz kodove nekih potprograma svojstava. Otvorite novi standardni izvršni projekt i dodajte modul klase, korištenjem izbornika Project. Kopirajte kod za svojstvo BrojZuba, prikazan iznad, u klasu Class1.

Prebacite se na formu Form1, i dodajte sljedeći programski kod u događaj Load:

```
Private Sub Form_Load()
  Dim c1 As Class1
  Set c1 = New Class1
  ‘ Dodjela nove vrijednosti svojstva.
  c1.BrojZuba = 42
  ‘ Prikaz vrijednosti svojstva.
  MsgBox c1.BrojZuba
End Sub
```

Pritisnite F8 za prolaz kroz programski kod jednu po jednu liniju. Uočite da kad je vrijednost svojstva dodijeljena, ulazite u potprogram Property Let, a kad je dohvaćena, ulazite u potprogram Property Get. Kopiranje ove vježbe s ostalim kombinacijama potprograma svojstava moglo bi vam biti korisno.

Argumenti sparenih potprograma svojstava moraju se podudarati

Primjeri potprograma svojstava koje ste dosad vidjeli bili su jednostavni, kao što će i biti za većinu svojstava. Međutim, potprogrami svojstava mogu imati višestruke argumente - čak i neobavezne argumente. Višestruki argumenti su korisni za svojstva koja djeluju poput matrica, kao što je raspravljeno u nastavku.

Kad koristite višestruke argumente, argumenti para potprograma svojstava moraju se podudarati. Sljedeća tablica prikazuje zahtjeve za argumente u određivanjima potprograma svojstava.

potprogram	sintaksa određivanja
Property Get	Property Get *imesvojstva*(1,..., *n*) As *tip*
Property Let	Property Let *imesvojstva*(1,..., *n*, *n+1*)
Property Set	Property Set *imesvojstva*(1,..., *n*, *n+1*)

Argumenti od prvog do pretposljednjeg (1,...,n) moraju dijeliti ista imena i tipove podataka u svim potprogramima tipa Property s istim imenom. Kao i kod ostalih tipova potprograma, svi traženi argumenti u toj listi moraju se nalaziti ispred prvog neobaveznog argumenta.

Vjerojatno ste uočili da određivanje potprograma Property Get uzima jedan argument manje od povezanog potprograma tipa Property Let ili Property Set. Tip podatka u potprogramu Property Get mora biti isti kao i tip podatka u posljenjem argumentu (*n*+1) povezanog potprograma Property Let ili Property Set.

Na primjer, razmotrite ovo određivanje potprograma Property Let, za svojstvo koje djeluje kao dvodimenzionalna matrica:

```
Public Property Let Stvari(ByVal X As Integer, _
ByVal Y As Integer, ByVal Stvar As Variant
  ‘ (Kod za dodjelu elementa matrice.)
End Property
```

Određivanje Property Get mora koristiti argumente s istim imenom i tipom podataka kao i argumenti u potprogramu Property Let:

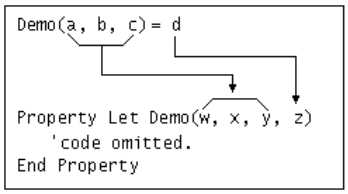
```
Public Property Get Stvari(ByVal X As Integer, _
ByVal Y As Integer) As Variant
  ‘ (Kod za dohvaćanje iz matrice)
End Property
```

Tip podatka zadnjeg argumenta u određivanju potprograma Property Set mora biti ili tip objekta ili tip Variant.

Podudaranje argumenata

Razlog za takva podudaranja argumenata prikazan je na slici 9.8, koja pokazuje kako Visual Basic uspoređuje dijelove naredbe dodjeljivanja s argumentima potprograma Property Let.

Slika 9.8 Pozivanje potprograma Property Let



Najuobičajenija upotreba potprograma svojstava s višestrukim argumentima je za stvaranje matrica svojstava.

Svojstva samo za čitanje

Kako bi stvorili svojstvo samo za čitanje, jednostavno izostavite Property Let ili (za svojstva objekta) Property Set.

Svojstva objekata

Ako stvarate svojstvo objekta koje je za čitanje i pisanje, upotrijebite Property Get i Property Set, kao ovdje:

```
Private msprSpravica As Spravica

Public Property Get Spravica() As Spravica
  ‘ Naredba Set mora biti upotrijebljena
  ‘ za vraćanje pokazivača objekta.
  Set Spravica = msprSpravica
End Property

Public Property Set Spravica(ByVal NovaSpravica As Spravica)
  Set msprSpravica = NovaSpravica
End Property
```

Svojstva tipa Variant

Svojstva za čitanje i pisanje tipa podatka Variant su najzamršenija. Ona koriste sva tri tipa potprograma svojstva, kao što je ovdje prikazano:

```
Private mvntBiloŠto As Variant

Public Property Get BiloŠto() As Variant
  ‘ Naredba Set se koristi samo ako svojstvo BiloŠto
  ‘ sadrži pokazivač objekta.
  If IsObject(mvntBiloŠto) Then
    Set BiloŠto = mvntBiloŠto
  Else
    BiloŠto = mvntBiloŠto
  End If
End Property

Public Property Let BiloŠto(ByVal NovaVrijednost As Variant)
  ‘ (Kod za provjeru valjanosti.)
  mvntBiloŠto = NovaSpravica
End Property

Public Property Set BiloŠto(ByVal NovaVrijednost As Variant)
  ‘ (Kod za provjeru valjanosti.)
  Set mvntBiloŠto = NovaSpravica
End Property
```

Potprogrami Property Set i Property Let su jednostavni, jer se uvijek pozivaju u ispravnim okolnostima. Međutim, potprogram Property Get mora obraditi oba sljedeća slučaja:

```
strNekiString = objvar1.BiloŠto
Set objvar2 = objvar1.BiloŠto
```

U prvom slučaju, svojstvo BiloŠto sadrži string, koji je dodijeljen varijabli tipa String. U drugom slučaju, svojstvo BiloŠto sadrži pokazivač objekta, koji je dodijeljen varijabli objekta.

Potprogram Property Get može biti kodiran da obradi takve slučajeve, korištenjem funkcije IsObject za ispitivanje privatne varijable tipa Variant prije vraćanja vrijednosti.

Naravno, ako je prva linija koda pozvana kad svojstvo BiloŠto sadrži pokazivač objekta, pojavit će se pogreška, ali to nije problem potprograma Property Get – to je problem s korištenjem svojstava tipa Variant.

Svojstva u koja se zapisuje samo jednom

Postoji puno mogućih kombinacija potprograma svojstava. Sve one su valjane, ali neke su razmjerno nepoznate, kao svojstva u koja se može zapisati samo jednom (samo Property Let, bez Property Get). Neka od njih ovise o drugim čimbenicima osim vrste potprograma svojstava koje kombinirate.

Na primjer, kad u vašoj aplikaciji organizirate objekte stvaranjem modela objekta, kao što je opisano u odlomku "Modeli objekata" kasnije u ovom poglavlju, možete poželjeti da objekt bude sposoban upućivati se natrag na objekte koji ga sadrže. Možete to napraviti ostvarivanjem svojstva Roditelj.

Trebate odrediti to svojstvo Roditelj kad je objekt stvoren, ali nakon toga možete ga poželjeti spriječiti da bude promijenjeno – slučajno ili namjerno. Sljedeći primjer pokazuje kako objekt Račun može ostvariti svojstvo Roditelj koje pokazuje na objekt Odjel koji sadrži račun.

```
‘ Spremanje privatnog podatka za svojstvo Roditelj.
Private modjRoditelj As Odjel

Property Get Roditelj() As Odjel
  ‘ Upotreba naredbe Set za pokazivač objekta.
  Set Roditelj = modjRoditelj
End Property

‘ Vrijednost svojstva može biti postavljena samo jednom.
Public Property Set Roditelj(ByVal NoviRoditelj As Odjel)
  If modjRoditelj Is Nothing Then
    ‘ Dodjela početne vrijednosti.
    Set modjRoditelj = NoviRoditelj
  Else
    Err.Raise Number:=vbObjectError + 32144, _
    Description:=”Svojstvo Roditelj je samo za čitanje”
  End If
End Property
```

Kad pristupite roditelju objekta Račun, na primjer programskim kodom strX = rčnNew.Roditelj.Name kako bi dobili ime odjela, poziva se potprogram Property Get za vraćanje pokazivača na roditeljski objekt.

Potprogram Property Set u ovom primjeru je programiran tako da svojstvo Roditelj može biti postavljeno samo jednom. Na primjer, kad objekt Odjel stvori novi račun, mogao bi izvesti programski kod `Set rčnNew.Roditelj = Me` za postavljanje svojstva. Zbog toga je ovo svojstvo samo za čitanje.

Za više informacija Budući da su forme u Visual Basicu forme, možete dodati korisnička svojstva formama. Pogledajte "Prilagođivanje klasa forme" ranije u ovom poglavlju.

Dodavanje postupaka klasi

Postupci klase su samo javni potprogrami tipa Sub ili Function koje ste odredili. Budući da su potprogrami tipa Sub i Function u pravilu javni, ne trebate čak ni izričito odrediti ključnu riječ Public za stvaranje postupka.

Na primjer, kako bi stvorili postupak Opoziv klase Račun, možete dodati ovaj potprogram tipa Public Function modulu klase:

```
Public Function Opoziv(ByVal Račun As Currency, _
ByVal KodTransakcije As Byte) As Double
   ‘ (Kod za izvođenje opoziva i vraćanje novog salda,
   ‘ ili za oživljavanje pogreške prekoračenja.)
End Function
```

Savjet Iako ne trebate pisati ključnu riječ Public, dobra je programerska praksa raditi to, jer to vaše namjere čini jasnim ljudima koji će kasnije održavati vaš programski kod.

Određivanje postupaka s Public Sub

Vraćanje novog salda je neobavezno, jer bi lako mogli pozvati svojstvo Saldo objekta Račun nakon pozivanja postupka Opoziv. Zbog toga možete kodirati potprogram Opoziv kao potprogram tipa Public Sub.

Savjet Ako pozivate svojstvo Saldo gotovo svaki put kad pozivate postupak Opoziv, vraćanje novog salda bit će neznatno učinkovitije. To je zato što svaki pristup svojstvu, čak i čitanje javne varijable, znači poziv funkcije – izričit ili posredan Property Get.

Važno Sljedeća imena ne mogu biti upotrijebljena kao imena svojstava ili postupaka, jer pripadaju podređenim sučeljima IUnknown i IDispatch: QueryInterface, AddRef, Release, GetTypeInfoCount, GetTypeInfo, GetIDsOfNames i Invoke. Ta imena će izazvati pogrešku kod prevođenja.

Za više informacija Za više informacija o potprogramima tipa Sub i Function, pogledajte “Uvod u potprograme” u 5. poglavlju “Osnove programiranja”.

Zaštita ostvarivanja detalja

Javno sučelje klase je definirano određivanjima svojstava i postupaka u modulu klase. Kao i kod skrivanja podataka, potprogrami koje odredite kao tip Private nisu dio sučelja. To znači da možete promijeniti uslužne potprograme koji se koriste unutar modula klase, bez utjecaja na programski kod koji koristi objekte.

Još važnije, također možete mijenjati programski kod unutar javnih potprograma tipa Sub ili Function koji ostvaruju postupak, bez utjecaja na kod koji koristi postupak. Sve dok ne promijenite tipove podataka za argumente potprograma, ili tipove podataka koji se vraćaju potprogramom tipa Function, sučelje je nepromijenjeno.

Skrivanje detalja ostvarivanja objekta iza sučelja je drugi vid sžimanja. Sažimanje vam omogućuje poboljšanje izvođenja postupaka, ili potpuno mijenjanje načina na koji se ostvaruje postupak, bez potrebe za mijenjanjem programskog koda koji koristi postupak.

Napomena Smjernice za imenovanje elemenata sučelja – raspravljene u odlomku "Imenovanje svojstava, postupaka i događaja" kasnije u ovom poglavlju – primjenjive su ne samo za imena svojstava i postupaka, već i za imena parametara u potprogramima tipa Sub i Function koji određuju vaše postupke. Ta imena parametara su vidljiva kad pregledavate postupke u pretraživaču objekata, i mogu biti korištena kao imenovani parametri (znači, *imeparametra:=vrijednost*) kad se pozivaju postupci.

Za više informacija Imenovani argumenti su predstavljeni u odlomku "Prosljeđivanje argumenata potprogramima" u 5. poglavlju "Osnove programiranja".

Dodavanje postupaka klasama forme je moćna programerska tehnika, raspravljena u odlomku "Prilagođavanje klasa forme" ranije u ovom poglavlju.

Ponekad nije jasno treba li element biti svojstvo ili postupak. Sljedeći odlomak "Je li to svojstvo ili postupak?" nudi neke smjernice.

Je li to svojstvo ili postupak?

Općenito, svojstvo je podatak o objektu, dok je postupak akcija koju objekt može izvesti ako to zatražite. Neke stvari su očigledno svojstva, kao Color i Name, a neke su očigledno postupci, kao Move i Show.

Međutim, kao i kod puno aspekata ljudskog ponašanja, postoji sivo područje u kojem argument može biti jedno ili drugo.

Na primjer, zašto je postupak Item klase Collection Visual Basica postupak, a ne indeksirano svojstvo? Nisu li stavke u zbirci samo podaci? Postupak Item zamišljene klase zbirke Spravice mogao bi biti ostvaren na jedan od dva načina, kao što je ovdje prikazano:

```
‘ Privatno spremanje objekata u zbirci Spravice
‘ (isto za oba ostvarivanja).
Private mcol As New Collection

Public Property Get Item(Index As Variant) As Spravica
  Set Item = mcol.Item(Index)
End Property
```

- ili -

```
Public Function Item(Index As Variant) As Spravica
  Set Item = mcol.Item(Index)
End Function
```

Ne postoji puno razlika između ova dva ostvarivanja. Oba su samo za čitanje, pa oba ovise o postupku Add klase Spravice za dodavanje objekata Spravica u zbirku. Oba načina delegiraju sve u objekt Collection - čak i njihove pogreške su stvorene objektom Collection!

Za više informacija Delegiranje je objašnjeno u odlomcima "Puno sučelja (lica) ponovnog korištenja koda" i "Stvaranje vlastitih klasa zbirki" kasnije u ovom poglavlju.

Mogli bi biti vrlo sitničavi pokušavajući odlučiti je li element zapravo podatak o objektu ili ponašanje objekta. Na primjer, mogli bi uvjeravati da je Item postupak jer zbirka radi nešto za vas – traži spravicu koju želite. Međutim, ta vrsta razloga može se obično koristiti s jednakom valjanošću na obje strane.

Možda bi korisnije bilo gledati na razlog od početka, pa se pitati kako *želite* razmišljati o elementu. Ako želite da ljudi o njemu razmišljaju kao o podatku o objektu, učinite ga svojstvom. Ako želite da o njemu razmišljaju kao o nečemu što objekt radi, napravite ga postupkom.

Razlog sintakse

Jak razlog ostvarivanja elementa koji koristi potprograme svojstva ovisi o načinu na koji želite koristiti element u programskom kodu. Znači, hoće li korisniku zbirke Spravice biti omogućeno da kodira sljedeće?

```
Set Spravice.Item(4) = stvMojaNovaSpravica
```

Ako hoće, ostvarite element kao svojstvo za čitanje i pisanje, korištenjem potprograma Property Get i Property Set, budući da postupci ne podržavaju takvu sintaksu.

Napomena U većini ostvarivanja zbirki koje ste otkrili, takva sintaksa nije dozvoljena. Ostvarivanje potprograma Property Set za zbirku nije tako jednostavno kako izgleda.

Razlog prozora s svojstvima

Na trenutak možete također pretpostaviti da je vaš objekt sličan kontroli. Možete li zamisliti da se element pojavljuje u prozoru s svojstvima, ili na stranici s svojstvima? Ako to nema smisla, ne ostvarujte element kao svojstvo.

Razborit razlog pogreške

Ako ste zaboravili da ste svojstvo Item napravili samo za čitanje te mu pokušali dodijeliti vrijednost, vjerojatno ćete lakše razumjeti poruku pogreške koju Visual Basic oživljava za potprogram Property Get - "Nemoguće dodjeljivanje svojstvu samo za čitanje" (Can't assign to read-only property) - od pogreške koja se pojavljuje kod potprograma Function - "Poziv funkcije na lijevoj strani dodjeljivanja mora vratiti tip Variant ili tip Object" (Function call on left side of assignment must return Variant or Object).

Razlog preostalog sredstva

Kao posljednje preostalo sredstvo, bacite novčić. Ako vas ni jedan od razloga u ovoj temi nije nagnao na odluku, vjerojatno ni nema neke razlike.

Za više informacija Potprogrami svojstava predstavljeni su u odlomku "Dodavanje svojstava klasi" ranije u ovom poglavlju. Postupci su raspravljeni u odlomku "Dodavanje postupaka klasi" u stalnoj pomoći.

Stvaranje podrazumijevanog svojstva ili postupka

Objektima stvorenim iz vaših klasa možete dati podrazumijevana svojstva, kao podrazumijevana svojstva objekata koje pruža Visual Basic. Najbolji kandidat za podrazumijevanog elementa je onaj kojeg najčešće koristite.

Kako odrediti svojstvo ili postupak kao podrazumijevan

1. U izborniku **Tools** odaberite naredbu **Procedure Attributes** za otvaranje dijaloškog okvira **Procedure Attributes**.
2. Kliknite **Advanced** za proširivanje dijaloškog okvira **Procedure Attributes**.
3. U okviru **Name**, odaberite svojstvo ili postupak koji je trenutno podrazumijevan za klasu. Ako klasa trenutno nema podrazumijevanog elementa, prijeđite na 5. korak.

   Napomena Možete upotrijebiti pretraživač objekata za pronalaženje trenutnog podrazumijevanog elementa klase. Kad u listi Classes odaberete klasu, možete se pomicati kroz elemente u listi Members; podrazumijevani element će biti označen malom plavom kuglom pored ikone.

4. U okviru **Procedure ID**, odaberite **None** za uklanjanje podrazumijevanog statusa svojstva ili postupka.
5. U okviru **Name**, odaberite svojstvo ili postupak za koji želite da bude podrazumijevan.

61. U okviru **Procedure ID**, odaberite **(Default)**, zatim kliknite **OK**.

   Važno Klasa može imati samo jedan podrazumijevan element. Ako je svojstvo ili postupak već označeno kao podrazumijevano, morate postaviti njezin Procedure ID na None prije nego što drugo svojstvo ili postupak učinite podrazumijevanim. Neće se pojaviti pogreške kod prevođenja ako su dva elementa označena kao podrazumijevani, ali nema načina za predviđanje koje će od njih Visual Basic odabrati kao podrazumijevano.

Možete također otvoriti dijaloški okvir Procedure Attributes iz pretraživača objekata. To je prikladno kad mijenjate podrazumijevani element klase, jer vam omogućuje brzo pronalaženje postojećeg podrazumijevanog elementa.

Kako promijeniti podrazumijevano svojstvo korištenjem pretraživača objekata

1. Pritisnite F2 za otvaranje **pretraživača objekata**.
2. U listi **Classes**, odaberite klasu kojoj želite promijeniti podrazumijevano svojstvo.
3. U listi **Members**, kliknite desnom tipkom miša na element s malom plavom kuglom pored ikone za otvaranje pomoćnog izbornika. Kliknite **Properties** za prikaz dijaloškog okvira **Procedure Attributes**.
4. Kliknite **Advanced** za proširivanje dijaloškog okvira **Procedure Attributes**.
5. U okviru **Procedure ID**, odaberite **None** za uklanjanje podrazumijevanog statusa svojstva ili postupka, te kliknite **OK**.

6. U listi **Members**, kliknite desnom tipkom miša na element za kojeg želite da bude novi podrazumijevani element kako bi otvorili pomoćni izbornik. Kliknite **Properties** za prikaz dijaloškog okvira **Procedure Attributes**.
7. Kliknite **Advanced** za proširivanje dijaloškog okvira **Procedure Attributes**.
8. U okviru **Procedure ID**, odaberite **(Default)**, zatim kliknite **OK**.

Napomena Ne možete upotrijebiti dijaloški okvir **Procedure Attributes** za promjenu podrazumijevanog elementa u klasama koje pruža Visual Basic.

Popravljanje podrazumijevanog koje ste slučajno učinili tipom Private ili Friend

Dijaloški okvir Procedure Attributes dopušta vam samo određivanje javnih svojstava i postupaka kao podrazumijevanih za klasu. Ako javno svojstvo ili postupak učinite podrazumijevanim za klasu, i kasnije promijenite određivanje u tip Private ili Friend, svojstvo ili postupak se može nastaviti ponašati kao da je i dalje određeno tipom Public.

Kako bi ispravili taj problem, morate ponovno odrediti svojstvo ili postupak kao tip Public, budući da dijaloški okvir Procedure Attributes neće pokazivati potprograme određene kao tip Private ili Friend. Jednom kad promijenite određivanje u tip Public, možete upotrijebiti dijalog Procedure Attributes za micanje obilježja Default. Nakon toga možete promijeniti određivanje natrag u tip Friend ili Private.

Svojstva i postupci tipa Friend

Kao dodatak određivanju svojstava i postupaka tipom Public i Private, možete ih odrediti i kao tip Friend. Elementi tipa Friend izgledaju baš kao i elementi tipa Public ostalim objektima u vašem projektu. Znači, izgleda da bi trebali biti dio sučelja klase. Oni to nisu.

U ActiveX sastavnim dijelovima koje možete stvoriti s Professional i Enterprise verzijama Visual Basica, elementi tipa Friend imaju značajnu ulogu. Budući da oni nisu dio sučelja objekta, aplikacije koje koriste objekte sastavnog dijela im ne mogu pristupati. Unatoč tome, vidljivi su svim ostalim objektima unutar sastavnog dijela, pa omogućuju sigurnu unutarnju komunikaciju unutar sastavnog dijela.

Važno Budući da elementi tipa Friend nisu dio javnog sučelja objekta, ne može im se pristupati kasnim povezivanjem – znači, kroz varijable određene s As Object. Za korištenje elemenata tipa Friend, morate odrediti varijable s ranim povezivanjem – znači, s As *imeklase*.

Standardni izvršni projekti ne mogu biti ActiveX sastavni dijelovi, budući da njihovi moduli klase ne mogu biti tipa Public, te ih zbog toga ne mogu koristiti druge aplikacije. Sva komunikacija između objekata u standardnom izvršnom projektu je zbog toga privatna, i nema potrebe za elementima tipa Friend.

Međutim, elementi tipa Friend imaju jedno posebno korisnu osobinu. Budući da nisu dio ActiveX sučelja, mogu se upotrijebiti za prosljeđivanje korisnički određenih tipova

između objekata bez njihovog javnog izlaganja. Na primjer, pretpostavimo da imate sljedeći korisnički određen tip u standardnom modulu:

```
Public Type udtDemo

  intA As Integer
  lngB As Long
  strC As String
End Type
```

Možete odrediti sljedeću privatnu varijablu te elemente tipa Friend u klasi Class1:

```
Private mDemo As udtDemo

Friend Property Get Demo() As udtDemo
  Demo = mDemo
End Property

‘ Uočite da udtDemo mora biti proslijeđen pokazivačem.
Friend Property Let Demo(NoviDemo As udtDemo)
  mDemo = NoviDemo
End Property

Friend Sub PostaviDijeloveDemoa(ByVal, a As Integer, _
ByVal B As Long, ByVal C As String)
  mDemo.intA = A
  mDemo.lngB = B
  mDemo.strC = C
End Sub

Public Sub PrikažiDemo()
  MsgBox mDemo.intA & vbCrLf & mDemo.intB & vbCrLf & mDemo.intC
End Sub
```

> **Napomena** Kad prosljeđujete korisnički određene tipove kao argumente potprograma tipa Sub, Function, ili potprograma svojstava, morate ih proslijediti pokazivačem (ByRef je podrazumijevano za argumente potprograma).

Zatim možete napisati sljedeći programski kod za korištenje klase Class1:

```
Private Sub Command1_Click()
  Dim c1A As New Class1
  Dim c1B As New Class1
  c1A.PostaviDijeloveDemoa 42, 1138 “Bok”
  c1B.Demo = c1A.Demo
  c1B.PrikažiDemo
End Sub
```

Okvir s porukom će pokazati 42, 1138 i “Bok”.

Napomena Budući da potprogrami tipa Friend nisu dio sučelja klase, nisu uračunati kad upotrijebite naredbu Implements za ostvarivanje višestrukih sučelja, kao što je opisano u odlomku "Polimorfizam" kasnije u ovom poglavlju.

Za više informacija Korištenje elemenata tipa Friend u sastavnim dijelovima je raspravljeno u odlomku "Privatne komunikacije između objekata" u 6. poglavlju "Opća načela oblikovanja sastavnih dijelova", u vodiču *Microsoft Visual Basic 6.0 Component Tools Guide*.

Dodavanje događaja klasi

U redu, recimo da ste stvorili simulaciju dinosaura, zajedno s klasama Stegosaur, Triceratops i Tiranosaur. Kao konačni potez, želite da Tiranosaur zareži, a kad on to napravi želite da svi ostali dinosauri u vašoj simulaciji sjednu i obrate pažnju.

Ako klasa Tiranosaur ima događaj Zareži, mogli bi rukovati tim događajem u svim ostalim dinosaurskim klasama. Ova tema raspravlja određivanje i rukovanje događajima u vašim modulima klase.

Napomena Djeco, ne pokušavajte ovo kod kuće, barem ne s više od nekoliko dinosaura. Povezivanje svakog dinosaura s svim ostalim dinosaurima korištenjem događaja moglo bi tako usporiti vaše dinosaure da bi ti objekti sisavaca preuzeli simulaciju.

Za svojstva i postupke se kaže da pripadaju *dolazećim sučeljima* (*incoming interface*), jer se pozivaju izvan objekta. Suprotno tome, događaji se nazivaju *odlazećim sučeljem* (*outgoing interface*), jer su pokrenuti unutar objekta, i obrađuju se drugdje.

Za više informacija Drugi dio "Stvaranje ActiveX sastavnih dijelova", vodiča *Microsoft Visual Basic 6.0 Component Tools Guide* pribavljenog s Professional i Enterprise verzijama, raspravlja o korištenju događaja u oblikovanju vaših softverskih sastavnih dijelova.

Za raspravu o boljem načinu rukovanja dinosaurima, pogledajte "Polimorfizam", kasnije u ovom poglavlju.

Određivanje i izazivanje događaja

Pretpostavimo na trenutak da imate klasu Widget (Spravica). Vaša klasa Widget ima postupak koji može trebati puno vremena za izvođenje, a vi bi željeli da vaša aplikacija bude sposobna dati neku vrstu pokazatelja izvršenja.

Naravno, mogli bi to napraviti tako da objekt Widget pokazuje dijaloški okvir s postotkom izvršenja, ali tada bi taj dijaloški okvir morali prikazivati u svakom projektu u kojem koristite klasu Widget. Dobro načelo oblikovanja objekata je prepuštanje rukovanja korisničkim sučeljem aplikaciji koja koristi objekt – osim ako je cijela namjena objekta upravljanje formom ili dijaloškim okvirom.

Namjena klase Widget je izvođenje drugih zadataka, pa je razumno dati joj događaj PercentDone (napravljeno u postocima), te prepustiti potprogramu koji poziva postupke klase Widget da rukuje tim događajem. Događaj PercentDone može također pružiti mehanizam za odustajanje od zadatka.

Primjer programskog koda ove teme možete početi graditi otvaranjem standardnog izvršnog projekta, te dodavanjem dva gumba i kontrole natpisa formi Form1. U izborniku Project, odaberite Add Class Module za dodavanje modula klase projektu. Imenujte objekte prema sljedećoj tablici.

objekt	svojstvo	postavka
Modul klase	Name	Widget
Prvi gumb	Caption	Start Task
Drugi gumb	Caption	Cancel
Kontrola natpisa	Name Caption	lblPercentDone “0%”

Klasa Widget

Događaj određujete u odjeljku Declarations modula klase, korištenjem ključne riječi Event. Događaj može imati argumente tipa ByVal ili ByRef, kao što to pokazuje događaj PercentDone klase Widget:

```
Option Explicit
Public Event PercentDone(ByVal Percent As Single, _
ByRef Cancel As Boolean)
```

Kad pozivajući objekt primi događaj PercentDone, argument Percent sadrži postotak obavljenog zadatka. Argument ByRef Cancel može biti postavljen na True kako bi poništio postupak koji je izazvao događaj.

Napomena Možete odrediti argumente događaja na isti način kao i argumente potprograma, uz sljedeće iznimke: događaji ne smiju imati imenovane, neobavezne ili argumente tipa ParamArray. Događaji nemaju povratne vrijednosti.

Izazivanje događaja PercentDone

Događaj PercentDone izaziva se postupkom LongTask klase Widget. Postupak LongTask ima dva argumenta: koliko dugo će se postupak praviti da nešto radi, te najmanji vremenski razmak prije nego što postupak LongTask prekine rad kako bi izazvao događaj PercentDone.

```
Public Sub LongTask(ByVal Duration As Single, _
ByVal MinimumInterval As Single)
  Dim sngThreshold As Single
  Dim sngStart As Single
  Dim blnCancel As Boolean

  ‘ Funkcija Timer vraća broj sekundi
  ‘ od ponoći, kao Single.
  sngStart = Timer
  sngThreshold = MinimumInterval
```

```
    Do While Timer < (sngStart + Duration)
      ‘ U stvarnoj aplikaciji, ovdje bi kroz petlju
      ‘ mogao biti obavljen neki dio posla.

      If Timer > (sngStart + sngThreshold) Then
            RaiseEvent PercentDone( _
            sngThreshold / Duration, blnCancel)
            ‘ Provjera je li operacija poništena.
            If blnCancel Then Exit Sub
            sngThreshold = sngThreshold ‘ MinimumInterval
      End If
    Loop
End Sub
```

Nakon broja sekundi određenog u MinimumInterval, izaziva se događaj PercentDone. Kad se izvođenje vrati s događaja, postupak LongTask provjerava je li argument Cancel postavljen na True.

> **Napomena** Zbog jednostavnosti, postupak LongTask pretpostavlja da unaprijed znate koliko će zadatak trajati. To gotovo nikad nije slučaj. Rastavljanje zadataka u dijelove ili čak veličina mogu biti problematični, i često je korisnicima najznačajniji iznos vremena koje protekne prije nego što dobiju pokazatelj da se nešto događa.

Rukovanje događajima objekata

Objekt koji izaziva događaje naziva se *izvor događaja* (*event source*). Kako bi rukovali događajima izazvanim izvorom događaja, možete odrediti varijablu klase objekta korištenjem ključne riječi WithEvents.

Za rukovanje događajem PercentDone klase Widget, postavite sljedeći programski kod u odjeljak Declarations forme Form1:

```
Option Explicit
Private WithEvents mWidget As Widget
Private mblnCancel As Boolean
```

Ključna riječ WithEvents označava da će varijabla mWidget biti korištena za rukovanje događajima objekta. Vrstu objekta određujete davanjem imena klase iz koje će objekt biti stvoren.

Varijabla mWidget određena je u odjeljku Declarations forme Form1 zato što varijable tipa WithEvents moraju biti varijable na razini modula. To je točno neovisno o tipu modula u koji ih postavljate.

Varijabla `mblnCancel` bit će upotrijebljena za odustajanje od postupka LongTask.

Ograničenja varijabli tipa WithEvents

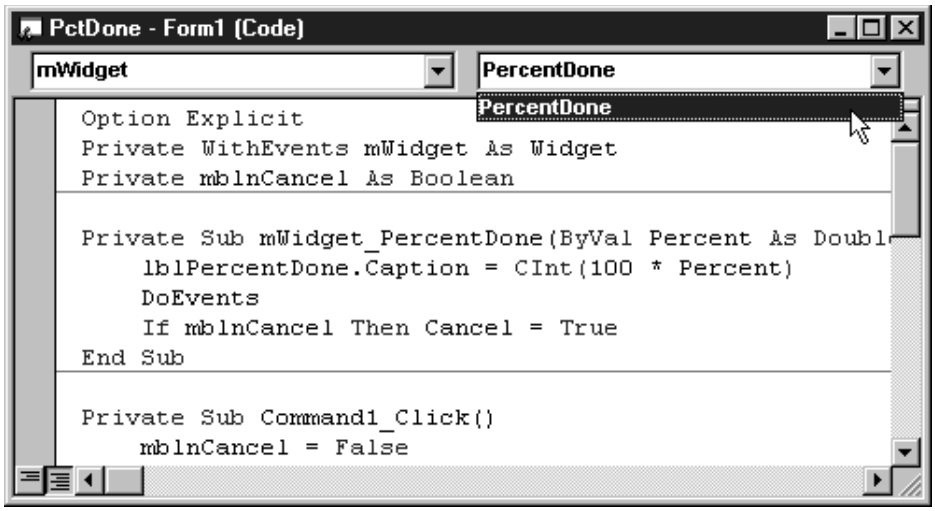
Trebate biti svjesni sljedećih ograničenja za korištenje varijabli tipa WithEvents:

- Varijabla tipa WithEvents ne može biti opća varijabla objekta. Znači, ne možete je odrediti s As Object – morate odrediti ime klase kad određujete varijablu.
- Ne možete odrediti varijablu tipa WithEvents s As New. Objekt izvora događaja mora biti izričito stvoren i dodijeljen varijabli tipa WithEvents.
- Ne možete odrediti varijable tipa WithEvents u standardnom modulu. Možete ih odrediti samo u modulima klase, modulima forme, te ostalim modulima koji određuju klase.
- Ne možete stvarati matrice s varijablama tipa WithEvents.

Pisanje programskog koda za rukovanje događajem

Odmah čim odredite varijablu tipa WithEvents, ime varijable pojavljuje se u lijevom spuštajućem okviru kodnog prozora tog modula. Kad odaberete mWidget, događaji klase Widget će se pojaviti u desnom spuštajućem okviru, kao što je prikazano na slici 9.9.

Slika 9.9 Događaj pridružen varijabli tipa WithEvents



Odabir događaja će prikazati odgovarajući potprogram događaja, s prefiksom mWidget_. Svi potprogrami događaja pridruženi varijabli tipa WithEvents će imati ime varijable kao prefiks. Dodajte sljedeći programski kod potprogramu događaja mWidget_PercentDone.

```
Private Sub mWidget_PercentDone(ByVal Percent As Single, _
Cancel As Boolean)
  lblPercentDone.Caption = CInt(100 * Percent) & "%"
  DoEvents
  If mblnCancel Then Cancel = True
End Sub
```

Uvijek kad je izazvan događaj PercentDone, potprogram događaja prikazuje postotak obavljenog zadatka u kontroli natpisa. Naredba DoEvents omogućuje obnavljanje kontrole natpisa, i također daje korisniku priliku da klikne gumb Cancel. Dodajte sljedeći programski kod događaju Click gumba kojem je natpis Cancel.

```
Private Sub Command2_Click()
  mblnCancel = True
End Sub
```

Ako korisnik klikne gumb Cancel dok se izvodi postupak LongTask, događaj Command2_Click će biti izveden odmah čim naredba DoEvents omogući pojavljivanje obrade događaja. Varijabla mblnCancel na razini modula je postavljena na True, a događaj mWidget_PercentDone je zatim ispituje i postavlja argument ByRef Cancel na True.

Povezivanje varijable tipa WithEvents s objektom

Forma Form1 je sad postavljena tako da rukovodi događajima objekta Widget. Sve što je preostalo je da negdje pronađe objekt Widget.

Kad odredite varijablu tipa WithEvents tijekom izrade aplikacije, ne postoji objekt koji joj je pridružen. Varijabla tipa WithEvents je kao i bilo koja varijabla objekta. Trebate stvoriti objekt i dodijeliti pokazivač na objekt varijabli tipa WithEvents. Dodajte sljedeći programski kod potprogramu događaja Form_Load kako bi stvorili objekt Widget.

```
Private Sub Form_Load()
  Set mWidget = New Widget
End Sub
```

Kad se gornji programski kod izvede, Visual Basic stvara objekt Widget i povezuje njegove događaje s potprogramima događaja pridruženim varijabli mWidget. Od tog trenutka nadalje, uvijek kad objekt Widget izazove događaj PercentDone, bit će izveden potprogram događaja mWidget_PercentDone.

Kako bi pozvali postupak LongTask, dodajte sljedeći programski kod u događaj Click gumba kojem je natpis Start Task.

```
‘ Gumb Start Task.
Private Sub Command1_Click()
  mblnCancel = False
  lblPercentDone.Caption = “0%”
  lblPercentDone.Refresh

  Call mWidget.LongTask(14.4, 0.66)

  If Not mblnCancel Then lblPercentDone.Caption = 100
End Sub
```

Prije poziva postupka LongTask, kontrola natpisa koja prikazuje postotak obavljenog zadatka mora biti pokrenuta, a zastavica tipa Boolean na razini modula za odustajanje od postupka mora biti postavljena na False.

Postupak LongTask se poziva s trajanjem zadatka od 14.4 sekunde. Događaj PercentDone bit će izazivan svake dvije trećine sekunde. Svaki put kad je događaj izazvan, bit će izveden potprogram događaja mWidget_PercentDone.

Kad je postupak LongTask gotov, varijablom mblnCancel se ispituje je li postupak LongTask završio normalno, ili je zaustavljen zato što je varijabla mblnCancel postavljena na True. Postotak obavljenog zadatka se ažurira samo u prvom slučaju.

Pokretanje aplikacije

Pritisnite F5 za postavljanje projekta u mod rada. Kliknite gumb s natpisom Start Task. Svaki put kad se izazove događaj PercentDone, kontrola natpisa se ažurira postotkom obavljenog zadatka. Kliknite gumb Cancel za zaustavljanje zadatka. Uočite da se izgled gumba Cancel ne mijenja odmah kad ga kliknete. Događaj Click ne može se dogoditi sve dok naredba DoEvents ne omogući obradu događaja.

Možete naći poučnim pokretanje aplikacije s F8, te prolaz kroz programski kod jednu po jednu liniju. Možete jasno vidjeti kako izvođenje ulazi u postupak LongTask, te ponovno kratko ulazi u formu Form1 svaki put kad se izazove događaj PercentDone.

Što bi se dogodilo ako bi, dok je izvođenje bilo natrag u programskom kodu forme Form1, postupak LongTask bio ponovno pozvan? Pometnja, zbrka, te na kraju (ako bi se to dogodilo svaki put kad je događaj izazvan) prekoračenje stoga (stack overflow).

Rukovanje događajima za drugačiji objekt Widget

Možete potaknuti varijablu mWidget da rukuje događajima različitih objekata Widget dodjeljivanjem pokazivača na novi objekt Widget varijabli mWidget. Zapravo, možete napraviti programski kod u gumbu Command1 za to svaki put kad kliknete gumb, dodavanjem sljedeće dvije linije koda:

```
Set mWidget = New Windget         ‘ <- Nova linija.
Call mWidget.LongTask(14.4, 0.66)
Set mWidget = Nothing             ‘ <- Nova linija.
```

Gornji programski kod stvara novi objekt Widget svaki put kad je pritisnut gumb. Odmah čim se završi postupak LongTask, pokazivač na objekt Widget se otpušta postavljanjem varijable mWidget na Nothing, i objekt Widget je uništen.

Varijabla tipa WithEvents može istovremeno sadržavati samo jedan pokazivač objekta, pa ako varijabli mWidget dodijelite drugi objekt Widget, događaji prethodnog objekta Widget neće biti obrađivani. Ako je mWidget jedina varijabla objekta koja sadrži pokazivač na stari objekt Widget, taj objekt će biti uništen.

Napomena Možete odrediti željeni broj varijabli tipa WithEvents, ali matrice varijabli tipa WithEvents nisu podržane.

Završavanje rukovanja događajima za varijablu tipa WithEvents

Sve dok je objekt Widget dodijeljen varijabli mWidget, potprogrami događaja pridruženi varijabli mWidget bit će pozivani svaki put kad objekt Widget izazove događaj. Kako bi završili rukovanjem događajima, možete postaviti mWidget na Nothing, kao što je pokazano u sljedećem dijelu programskog koda.

```
‘ Završavanje rukovanja događajima za mWidget.
Set mWidget = Nothing
```

Kad je varijabla tipa WithEvents postavljena na Nothing, Visual Basic prekida vezu između događaja objekta i potprograma događaja povezanih s varijablom.

> **Važno** Varijabla tipa WithEvents sadrži pokazivač objekta, kao i bilo koja varijabla objekta. Postojanje pokazivača objekta održava objekt živim. Kad postavite sve pokazivače prema objektu na Nothing kako bi ih uništili, ne zaboravite varijable koje ste odredili kao tip WithEvents.

Usporedba tipa WithEvents s događajima kontrola na formi

Vjerojatno ste uočili neke sličnosti između načina na koje koristite varijable tipa WithEvents i načina rukovanja događajima koje su izazvale kontrole na formi. U oba slučaja, kad odaberete događaj u desnom spuštajućem okviru kodnog prozora, dobit ćete potprogram događaja koji sadrži ispravne argumente za događaj.

Zapravo, mehanizam je potpuno isti. Kontrola se obrađuje kao *svojstvo* klase forme, i ime tog svojstva je vrijednost koju ste dodijelili svojstvu Name kontrole u prozoru s svojstvima.

To je kao da postoji javna varijabla na razini modula s istim imenom kao i kontrola, i sva imena potprograma događaja te kontrole počinju s imenom te varijable, kao što bi počinjali s varijablom tipa WithEvents.

Možete to jednostavno vidjeti ako odredite varijablu mWidget kao tip Public umjesto Private. Onog trenutka kad to učinite, mWidget će se pojaviti u pretraživaču objekata kao svojstvo forme Form1, kao i kontrole na formi.

Razlika između ova dva slučaja je u tome što Visual Basic automatski stvara primjere svih kontrola na formi kad se stvori forma, dok vi morate stvoriti vaše vlastite primjere klasa čijim događajima želite rukovati, te dodijeliti pokazivače za te objekte varijablama tipa WithEvents.

Dodavanje događaja formi

Sljedeći primjer pokazuje vam korak po korak kako možete stvoriti korisničke događaje za forme. Kako bi isprobali ovu vježbu, otvorite standardni izvršni projekt i napravite sljedeće:

Kako dodati događaj formi Form1

1. U izborniku **Project**, odaberite **Add Class Module** za dodavanje modula klase projektu. Postavite sljedeći programski kod u odjeljak **Declarations** klase Class1:

   ```
   Public Property Get Form1() As Form1
     Set Form1 = mForm1
   End Property
   Public Property Set Form1(ByVal NewForm1 As Form1)
     Set mForm1 = NewForm1
   End Property
   ```

   Ako koristite pregled po potprogramima, potprogrami svojstava ne mogu se vidjeti u isto vrijeme. Kliknite gumb **Full Module View** u donjem lijevom kutu kodnog prozora za prebacivanje na pregled koda cijelog modula. Možete se vratiti na pregled po potprogramima klikom na gumb **Procedure View** odmah do njega (držite pokazivač miša iznad gumbâ kako bi vidjeli koji je koji).

2. Dodajte sljedeći programski kod u odjeljak **Declarations** forme Form1:

   ```
   Event Gong
     Private mc1 As Class1
   ```

   Sad kad je stvorena klasa Class1, moguće je stvoriti varijablu tipa Class1. Ovaj potprogram izmjenjuje nekoliko puta formu Form1 i klasu Class1, jer korak u jednom modulu prvo zahtjeva dodavanje programskog koda drugom.

3. Vratite se u klasu Class1 i dodajte sljedeći kod u odjeljak **Declarations**.

   ```
   Private WithEvents mForm1 As Form1
   ```

   Kao što je raspravljeno u odlomku “Dodavanje događaja klasi”, ključna riječ WithEvents znači da je primjer forme Form1 povezan *s događajima*. Uočite da ovaj korak nije bio moguć sve dok događaj Gong nije bio stvoren.

4. U lijevom spuštajućem okviru (**Object**) kodnog prozora klase Class1, odaberite **mForm1** kako bi dobili potprogram događaja za događaj Gong. Dodajte sljedeći kod u potprogram događaja:

   ```
   Private Sub mForm1_Gong()
     MsgBox “Gong!”
   End Sub
   ```

5. Vratite se u formu Form1. U spuštajućem okviru s listom **objekata**, odaberite **Form**. U desnom spuštajućem okviru (**Procedure**), odaberite **Load**. Dodajte sljedeći kod u potprogram događaja:

```
Private Sub Form_Load()
  Set mc1 = New Class1
  Set mc1.Form1 = Me
End Sub
```

   Prva linija stvara objekt Class1, a druga dodjeljuje njegovom svojstvu **Form1** (stvorenom u koraku 1) pokazivač na formu Form1 (znači, Me – kad ste u kodnom prozoru forme Form1, Me označava formu Form1; kad ste u kodnom prozoru klase Class1, Me označava klasu Class1).

6. Postavite tri okvira s tekstom na formu Form1. Upotrijebite redom spuštajuće okvire **Object** i **Procedure** za odabir potprograma događaja Change za svaku kontrolu, i postavite istu liniju koda u svaku od njih:

```
Private Sub Text1_Change()
  RaiseEvent Gong
End Sub
```

   Svaki put kad se promijeni sadržaj teksta u okvirima s tekstom, bit će izazvan događaj Gong forme.

7. Pritisnite F5 za pokretanje projekta. Svaki put kad upišete karakter u jedan od okvira s tekstom, pojavit će se okvir s porukom. To je vrlo dosadno, ali pokazuje kako možete dodati događaj formi, i na taj način dobiti upozorenja od više kontrola.

Kao što je pokazano u odlomku “Određivanje i izazivanje događaja”, događajima možete dodati argumente. Na primjer, možete proslijediti ime kontrole – ili još bolje, pokazivač na kontrolu – primatelju događaja.

Sažetak određivanja, izazivanja i rukovanja događaja

Kako bi dodali događaj klasi te zatim upotrijebili događaj, morate napraviti sljedeće:

- U odjeljku Declarations modula klase koji određuje klasu, morate upotrijebiti naredbu Event za određivanje događaja s bilo kojim argumentima koje mu želite dati. Događaji su uvijek javni (tipa Public).

  Napomena Događaji ne mogu imati imenovane, neobavezne ili argumente tipa ParamArray. Događaji nemaju povratne vrijednosti.

- Na prikladnim mjestima u kodu modula klase, morate upotrijebiti naredbu RaiseEvent za izazivanje događaja, opskrbljenu potrebnim argumentima.

- U odjeljku Declarations modula koji će rukovati događajem, trebate dodati varijablu tipa klase, korištenjem ključne riječi WithEvents. Ta varijabla mora biti na razini modula.

- U lijevom spuštajućem okviru kodnog prozora, trebate odabrati varijablu koju ste odredili tipom WithEvents.
- U desnom spuštajućem okviru, trebate odabrati događaj kojim želite rukovati (možete odrediti više događaja za klasu).
- Trebate dodati programski kod za potprogram događaja, korištenjem pribavljenih argumenata.

Stvaranje klasa svjesnih podataka

Ako ste pročitali prethodni materijal o stvaranju klasa, sad znate da je klasa objekt koji sžima podatke i programski kod, te da su svojstva klase podaci koji opisuju objekt. Također znate da možete upotrijebiti potprograme svojstava ili javna svojstva za izlaganje podataka predstavljenih tim svojstvima.

Za sada, dobro je – svi primjeri do sada radili su s kratkotrajnim podacima, znači, podacima koji su stvoreni i trošeni tijekom izvođenja aplikacije. Za većinu aplikacija, to bi moglo biti sve što trebate, ali što ako trebate spremati podatke nakon rada, ili prilagoditi podatke koji već postoje izvan vaše aplikacije? Kako bi radili s vanjskim izvorima podataka, trebate stvoriti klase svjesne podataka.

Klase svjesne podataka mogu biti podijeljene u dvije vrste – potrošači podataka i izvori podataka. Moduli klase imaju dva svojstva za vrijeme izrade, DataBindingBehavior i DataSourceBehavior, koja određuju kako će klasa surađivati s vanjskim podacima. Objekt BindingCollection se koristi za međusobno povezivanje klasa svjesnih podataka i kontrola.

Izvori podataka

Izvor podataka je klasa koja pruža podatke iz vanjskog izbora koji će biti upotrijebljeni od ostalih objekata. Kontrola podataka (Data control) je u stvarnosti primjer klase koja je izvor podataka, ali klase koje su postavljene da djeluju kao izvori podataka mogu biti puno moćnije od kontrole podataka. Za razliku od kontrole podataka, klase svjesne podataka ne trebaju imati vidljivi dio, i nisu ograničene na određeno podatkovno sučelje kao objekti za pristup podacima (Data Access Objects, DAO) ili objekti udaljenih podataka (Remote Data Objects, RDO). Zapravo, klase svjesne podataka mogu djelovati kao izvor podataka za bilo koji tip podataka, uključujući uobičajene izvore tipa ODBC, ActiveX objekte podataka (ActiveX Data Objects, ADO), ili bilo koji OLE DB davatelj.

Svojstvo DataSourceBehavior određuje hoće li ili ne klasa djelovati kao izvor podataka. Postavljanjem svojstva DataSourceBehavior na 1 (vbDataSource), vaša klasa može djelovati kao izvor podataka za druge objekte.

Potrošači podataka

Jednostavno rečeno, potrošač podataka je klasa koja može biti povezana s vanjskim izvorom podataka, otprilike kao što s kontrola okvira s tekstom može biti povezana s kontrolom podataka. U ranijim verzijama Visual Basica, kontrole su bile jedini objekti koji su mogli biti povezani s izvorom podataka. Klase svjesne podataka postavljene

kao potrošači podataka omogućuju vam povezivanje bilo kojeg objekta s bilo kojim objektom stvorenim iz klase koja je napravljena kao izvor podataka.

Svojstvo DataBindingBehavior omogućuje klasi da se poveže s vanjskim podacima. Postavljanjem ovog svojstva na 1 (vbSimpleBound), objekt stvoren iz vaše klase bit će povezan jednim podatkovnim poljem u vanjskom izvoru podataka. Postavljanjem svojstva DataBindingBehavior na 2 (vbComplexBound), vaša klasa bit će povezana s redom podataka u vanjskom izvoru podataka. Razmišljajte o tome na ovaj način – da su vaši objekti kontrole, kontrola okvira s tekstom bila bi povezana na jednostavan način, dok bi kontrola mreže bila povezana na složen način.

Objekt BindingCollection

Upravo kako bi povezali kontrolu s bazom podataka kroz kontrolu podataka, klase svjesne podataka trebaju središnji objekt s kojim će zajedno biti povezane. Taj objekt je objekt BindingCollection. Baš kako i zvuči, to je zbirka povezivanja između izvora podataka i jednog ili više potrošača podataka.

Kako bi mogli koristiti objekt BindingCollection najprije morate dodati pokazivač na objekt Microsoft Data Binding Collection tako da ga odaberete u dijalogu References, dostupnom preko izbornika Project. Kao i kod bilo kojeg objekta, trebate stvoriti primjer objekta BindingCollection tijekom izvođenja aplikacije.

Svojstvo DataSource objekta BindingCollection koristi se za određivanje objekta koji će pružati podatke. Objekt mora biti klasa ili korisnička kontrola (dakle, tipa UserControl) s svojim svojstvom DataSourceBehavior postavljenim na vbDataSource.

Jednom kad je stvoren primjer objekta BindingCollection i određeno njegovo svojstvo DataSource, možete upotrijebiti postupak Add za određivanje odnosa povezivanja. Postupak Add treba tri obavezna argumenta: ime objekta potrošača, svojstvo tog objekta koje će biti povezano s izvorom, i polje iz izvora koje će biti povezano s svojstvom. Objektu BindingCollection možete dodati višestruka povezivanja ponavljanjem postupka Add; možete upotrijebiti postupak Remove za brisanje povezivanja.

Za više informacija Za primjere korak po korak stvaranja klasa svjesnih podataka, pogledajte idući odlomak “Stvaranje izvora podataka”, i odlomak “Stvaranje potrošača podataka”, kasnije u ovom poglavlju.

Stvaranje izvora podataka

U ovom dijelu, prošetat ćemo se korak po korak kroz postupak stvaranja klase svjesne podataka koja djeluje kao izvor podataka. Ovaj primjer će povezati kontrolu okvira s tekstom s našom klasom koja je izvor podataka, kako bi se podaci mogli prikazati. Idući odlomak “Stvaranje potrošača podataka”, pokazuje kako povezati našu klasu izvora podataka s klasom potrošača podataka.

Primjeri programskog koda u ovom dijelu uzeti su iz primjera Data-aware Classes (Dataware.vbp). Ovu aplikaciju možete pronaći u direktoriju Samples.

Stvaranje izvora podataka je postupak od dvije faze. U prvoj fazi ćemo stvoriti klasu izvora podataka; u drugoj fazi ćemo je zakvačiti za kontrolu okvira s tekstom kako bi mogli prikazati izlazne rezultate.

Stvaranje klase izvora

Prvi korak u stvaranju klase izvora je određivanje nove klase te davanje svojstava i postupaka potrebnih za pružanje podataka.

1. Otvorite novi standardni izvršni projekt, i ubacite modul klase odabirom naredbe **Add Class Module** u izborniku **Project**.
2. U prozoru s svojstvima, odredite svojstva klase kako slijedi:

svojstvo	postavka
Name	MySource
DataSourceBehavior	vbDataSource

   Kad je svojstvo DataSourceBehavior postavljeno na vbDataSource, modulu klase će biti dodan novi potprogram tipa Sub s imenom GetDataMember. Možete to vidjeti odabirom stavke **Class** u listi Objects kodnog prozora, te odabirom liste **Event**.
3. Odaberite stavku **References** iz izbornika **Project**, i dodajte pokazivač na biblioteku Microsoft ActiveX Data Objects 2.0 Library.
4. Dodajte sljedeće u odjeljak **Declarations** modula klase:

```
Option Explicit
Private rs As ADODB.Recordset
```

   Tako određujete varijablu objekta za objekt ADO Recordset.
5. Dodajte sljedeći kod u potprogram događaja Initialize modula klase:

```
Private Sub Class_Initialize()
  Dim strPath As String, strName As String
  Dim i As Integer

  ‘ Stvaranje primjera objekta Recordset.
  Set rs = New ADODB.Recordset

  ‘ Određivanje svojstava objekta Recordset.
  With rs
     .Fields.Append “DirID”, adInteger
     .Fields.Append “Directory”, adBSTR, 255
     .CursorType = adOpenStatic
     .LockType = adLockOptimistic
     .Open
  End With

  ‘ Petlja kroz direktorije i popunjavanje objekta Recordset.
  strPath = “C:\”
```

```
  strName = Dir(strPath, strDirectory)
  i = 0
  Do While strName <> “”
     If strName <> “.” And strName <> “..” Then
       If (GetAttr(strPath & strName) And _
       vbDirectory) = vbDirectory Then
          i = i + 1
          With rs
               .AddNew
               .Fields.Item(“DirID”) = i
               .Fields.Item(“Directory”) = strName
               .Update
          End With
       End If
     End If
     strName = Dir
  Loop
  ‘ Povratak na prvi zapis.
  rs.MoveFirst
End Sub
```

U ovom primjeru smo stvorili objekt ADO Recordset u letu te ga napunili listom direktorija. Alternativno, možete upotrijebiti postojeći skup zapisa dodjeljivanjem svojstvu Connect objekta ADO Recordset u događaju Initialize.

6. Odaberite **Class** na listi **Objects** u **kodnom prozoru**, te odaberite GetDataMember s liste **Events**. Dodajte sljedeći kod u potprogram GetDataMember tipa Sub:

```
Private Sub Class_GetDataMember(DataMember As String, _
Data As Object)
  ‘ Dodjela objekta Recordset objektu Data.
  Set Data = rs
End Sub
```

Potprogram GetDataMember određuje izvor podataka za klasu. Vaša klasa izvora podataka može pružiti višestruke izvore podataka dodavanjem izraza Select Case u potprogram GetDataMember te prosljeđivanjem imena izvora argumentom DataMember.

7. Dodajte novi potprogram tipa Sub za pružanje javnog postupka petlje kroz objekt Recordset:

```
Public Sub Cycle()
  ‘ Kruženje kroz Recordset.
  rs.MoveNext
  If rs.EOF = True Then
     rs.MoveFirst
  End If
End Sub
```

Kako bi se pomicali kroz skup zapisa, trebate pokazati postupke kretanja za našu klasu. Zbog jednostavnosti, ovaj primjer može se kretati samo prema naprijed kroz skup zapisa. Kako bi klasu učinili korisnijom, možete poželjeti dodati postupke kao što su MoveFirst, MoveNext, Add i Delete.

Korištenje klase izvora

Kad je klasa određena, s njom možemo napraviti nešto korisno. U ovom primjeru ćemo je povezati s kontrolom okvira s tekstom tako da možemo vidjeti njezine izlazne rezultate; također ćemo upotrijebiti naredbeni gumb za izvođenje postupka Cycle.

1. Odaberite formu **Form1** te dodajte kontrolu okvira s tekstom i kontrolu naredbenog gumba na formu.
2. U prozoru s svojstvima, odredite svojstva okvira s tekstom kako slijedi:

svojstvo	postavka
Name	txtConsumer
Text	(prazno)

3. U prozoru s svojstvima, odredite svojstva naredbenog gumba kako slijedi:

svojstvo	postavka
Name	cmdCycle
Text	Cycle

4. Odaberite **References** iz izbornika **Project**, te dodajte pokazivač na Microsoft Data Binding Collection.

   Objekt DataBinding kojeg pruža zbirka Data Binding Collection je "ljepilo" koje povezuje izvor podataka s potrošačem podataka.

5. Dodajte sljedeće u odjeljak **Declarations** modula klase:

```
Option Explicit
Private objSource As MySource
Private objBindingCollection As BindingCollection
```

   Trebamo odrediti našu klasu izvora (MySource) i objekt BindingCollection koji koristi rano povezivanje.

6. Dodajte sljedeći kod u potprogram događaja Form_Load:

```
Private Sub Form_Load()
  Set objSource = New MySource
  Set objBindingCollection = New BindingCollection
```

```
‘ Dodjela klase izvora svojstvu DataSource
‘ objekta BindingCollection.
  Set objBindingCollection.DataSource = objSource
  ‘ Dodavanje povezivanja.
  objBindingCollection.Add txtConsumer “Text”, “Directory”
End Sub
```

U događaju Load stvaramo primjere klase izvora i objekta BindingCollection, te dodjeljujemo objekt izvora svojstvu DataSource objekta BindingCollection. Na kraju, dodajemo povezivanje određivanjem imena potrošača (txtConsumer), svojstva potrošača koje će biti povezano (svojstvo Text), te svojstva Field objekta izvora s kojim se povezujemo (Directory).

7. Dodajte sljedeći kod u potprogram događaja cmdCycle_Click:

```
Private cmdCycle_Click()
  ‘ Poziv postupka Cycle u izvoru podataka.
  objSource.Cycle
End Sub
```

To će izvesti postupak Cycle naše klase izvora.

8. Pritisnite F8 za pokretanje projekta.

Kad kliknete gumb Cycle, imena direktorija iz skupa zapisa stvorenog u našoj klasi izvora pojavit će se u okviru s tekstom. Čestitamo – upravo ste povezali kontrolu s klasom izvora podataka bez korištenja kontrole podataka!

9. Snimite projekt. Kad ste upitani za imena datoteka, upotrijebite sljedeća imena.

Snimite klasu izvora kao “MySource.cls”.

Snimite formu kao “Dataform.frm”.

Snimite projekt kao “Dataware.vbp”.

U sljedećem odlomku “Stvaranje potrošača podataka”, pogledat ćemo postupak stvaranja klase svjesne podataka koja djeluje kao potrošač podataka.

Stvaranje potrošača podataka

U ovom dijelu, prošetat ćemo korak po korak kroz postupak stvaranja klase svjesne podataka koja djeluje kao potrošač podataka. Prethodni dio “Stvaranje izvora podataka”, prikazuje kako stvoriti izvor podataka s kojim potrošač podataka može biti povezan. Ovaj primjer pokazuje kako stvoriti klasu potrošača podataka i povezati je s izvorom podataka stvorenim u prethodnom dijelu.

Primjeri programskog koda u ovom dijelu uzeti su iz primjera Data-aware Classes (Dataware.vbp). Ovu aplikaciju možete pronaći u direktoriju Samples.

Povezivanje potrošača podataka s objektom izvora podataka

Ovaj primjer pokazuje kako stvoriti klasu potrošača podataka i povezati je s klasom izvora podataka. Primjer koristi klasu MySource stvorenu u dijelu "Stvaranje izvora podataka".

1. Otvorite projekt Dataware.vbp (odaberite **Open Project** iz izbornika **File**).

   Napomena Ako prethodno niste završili primjer iz odlomka "Stvaranje izvora podataka", taj projekt neće postojati. Kompletnu verziju projekta Dataware.vbp možete također pronaći u direktoriju Samples.

2. Umetnite novi modul klase odabirom naredbe **Add Class Module** u izborniku **Project**.

3. U prozoru s svojstvima, odredite svojstva nove klase kako slijedi:

svojstvo	postavka
Name	MyConsumer
DataBindingBehavior	vbSimpleBound

4. Dodajte sljedeće odjeljku **Declarations** modula klase:

```
Option Explicit
Private mDirectory As String
```

5. Dodajte par potprograma tipa Property Get / Property Let javnom svojstvu **DirName**:

```
Public Property Get DirName() As String
  DirName = mDirectory
End Property

Public Property Let DirName(mNewDir As String)
  mDirectory = mNewDir
  ' Prikaz nove vrijednosti u prozoru za neposredan upis naredbi.
  Debug.Print mDirectory
End Property
```

   Budući da je klasa MySource nevidljiva, trebamo upotrijebiti naredbu Debug.Print u potprogramu Property Let za dokazivanje da dohvaća nove vrijednosti iz izvora podataka.

6. Odaberite formu **Form1** i dodajte sljedeći kod u odjeljak **Declarations**:

```
Option Explicit
Private objSource As MySource
Private objBindingCollection As BindingCollection
Private objConsumer As MyConsumer
```

Nova linija koda dodaje pokazivač na našu klasu potrošača.

7. Dodajte sljedeći kod u potprogram događaja Form_Load:

```
Private Sub Form_Load()
  Set objSource = New MySource
  Set objBindingCollection = New BindingCollection
  Set objConsumer = New MyConsumer

  ‘ Dodjela klase izvora svojstvu DataSource
  ‘ objekta BindingCollection.
  Set objBindingCollection.DataSource = objSource
  ‘ Dodavanje povezivanja.
  objBindingCollection.Add txtConsumer “Text”, “Directory”
  objBindingCollection.Add objConsumer “DirName”, “Directory”
End Sub
```

   Novi programski kod stvara primjer klase potrošača i dodaje ga objektu BindingCollection, povezujući svojstvo DirName potrošača s poljem Directory izvora podataka.

8. Pritisnite F8 za pokretanje projekta. Osigurajte vidljivost prozora za neposredan upis naredbi (Immediate window).

   Kad kliknete gumb Cycle, imena direktorija koje pruža klasa MySource pojavit će se i u okviru s tekstom i u prozoru za neposredan upis naredbi, dokazujući da je klasa MyConsumer povezana s klasom MySource.

Imenovanje svojstava, postupaka i događaja

Svojstva, postupci i događaji koje dodajete modulima klase određuju sučelje koje će biti korišteno za upravljanje objektima stvorenim iz klase. Kad imenujete te elemente, i njihove argumente, može vam biti korisno slijediti nekoliko jednostavnih pravila.

- Koristite cijele riječi uvijek kad je to moguće, kao na primjer ProvjeraSricanja. Skraćenice mogu imati puno oblika, i zbog toga mogu biti zbunjujuće. Ako su cijele riječi preduge, upotrijebite cijele prve slogove.
- Koristite velika i mala slova za svoje identifikatore, pišući svaku riječ ili slog velikim slovom, kao na primjer, PomoćniIzbornici ili AsinČitanjeGotovo.
- Koristite ispravne množine za imena klasa zbirki, kao na primjer RadniListovi, Forme ili Spravice. Ako zbirka sadrži objekte čije je ime u množini, dodajte riječ “Zbirka”, kao na primjer “SerijeZbirka”.
- Koristite dosljedan red glagol/objekt ili objekt/glagol za imena svojih postupaka. Znači, koristite UbaciSpravicu, UbaciZupčanik, i tako dalje, ili uvijek postavljajte najprije objekt, kao u SpravicaUbaci i ZupčanikUbaci.

  Napomena Iako je moguće koristiti karakter podvučene linije (_) u imenu svojstva, podvlaka u imenu događaja će uzrokovati pogrešku. Karakter podvučene linije koristi se u potprogramima događaja za odvajanje imena događaja od imena objekta. Zbog toga, najbolje je u potpunosti izbjegavati karakter podvučene linije kod imenovanja svojstava, postupaka i događaja.

Jedna od glavnih prednosti programiranja objektima je ponovno korištenje koda. Poštovanje gornjih pravila, koja su dio ActiveX smjernica za sučelja, olakšava pamćenje imena i namjenu svojstava, postupaka i događaja.

Za više informacija Ako imate Professional ili Enterprise verziju biblioteke *Visual Basic 6.0 Language Reference Library*, pogledajte proširenu listu u odlomku "Što je u imenu?" u 6. poglavlju "Opća načela oblikovanja sastavnih dijelova", u 2. dijelu "Stvaranje ActiveX sastavnih dijelova", vodiča *Microsoft Visual Basic 6.0 Component Tools Guide*.

Polimorfizam

Polimorfizam (mnogoobličnost) znači da mnogo klasa može pružiti isto svojstvo ili postupak, a pozivatelj ne mora znati kojoj klasi objekt pripada prije nego što pozove svojstvo ili postupak.

Na primjer, i klasa Buha i klasa Tiranosaur mogu imati postupak Ugrizi. Polimorfizam znači da možete pozvati postupak Ugrizi bez znanja je li objekt Buha ili Tiranosaur – iako ćete to sigurno znati nakon ugriza.

Za više informacija S Professional i Enterprise verzijama Visual Basica, polimorfizam postaje moćan mehanizam nizanja sustava softverskih sastavnih dijelova. To je raspravljeno u 6. poglavlju "Opća načela oblikovanja sastavnih dijelova", u 2. dijelu "Stvaranje ActiveX sastavnih dijelova", vodiča *Microsoft Visual Basic 6.0 Component Tools Guide*.

Kako Visual Basic pruža polimorfizam

Većina objektno orijentiranih sustava programiranja pruža polimorfizam kroz *naslijeđe* (*inheritance*). Znači, obje zamišljene klase Buha i Tiranosaur mogu biti naslijeđene od klase Životinje. Svaka klasa nadjačava postupak Ugrizi klase Životinje, kako bi pružila svoje vlastite značajke ugriza.

Polimorfizam proizlazi iz činjenice da možete pozvati postupak Ugrizi objekta koji pripada bilo kojoj klasi koja je izvedena iz klase Životinje, bez znanja kojoj klasi objekt stvarno pripada.

Pružanje polimorfizma s sučeljima

Visual Basic ne koristi nasljeđivanje za pružanje polimorfizma. Visual Basic pruža polimorfizam kroz višestruka ActiveX sučelja. U Modelu objekta kao sastavnog dijela (Component Object Model, COM) koji oblikuje infrastrukturu ActiveX određivanja, višestruka sučelja omogućuju sustavima softverskih sastavnih dijelova da se nižu bez prekidanja postojećeg koda.

Sučelje (*interface*) je skup povezanih svojstava i postupaka. Velik dio ActiveX specifikacija brine se o provođenju uobičajenih sučelja za dobivanje sistemskih usluga ili za pružanje djelotvornosti ostalim aplikacijama.

U Visual Basicu, stvorit ćete sučelje Životinje i provesti ga u svojim klasama Buha i Tiranosaur. Nakon toga ćete moći pozvati postupak Ugrizi jedne od tih vrsti objekta, a da ne znate koje je vrste.

Polimorfizam i izvođenje

Polimorfizam je važan za tijek izvođenja. Kako bi to vidjeli, razmotrite sljedeću funkciju:

```
Public Sub UzmiHranu(ByVal Stvorenje As Object, _
ByVal Hrana As Object)
  Dim dblUdaljenost As Double
  ‘ (Kod za izračunavanje udaljenosti od hrane).
  Stvorenje.Pomakni dblUdaljenost       ‘ Kasno povezivanje.
  Stvorenje.Ugrizi Hrana                ‘ Kasno povezivanje.
End Sub
```

Postupci Pomakni i Ugrizi su *kasno povezani* s objektom Stvorenje. Kasno povezivanje se događa kad Visual Basic tijekom prevođenja ne može ustvrditi koju vrstu objekta će sadržavati varijabla. U ovom primjeru, argument Stvorenje je određen s As Object, pa tijekom izvođenja može sadržavati pokazivač na bilo koju vrstu objekta – kao Auto ili Kamen.

Budući da ne može ustanoviti koji će to biti objekt, Visual Basic prevodi nešto dodatnog koda za upit objektu podržava li postupak koji je pozvan. Ako objekt podržava postupak, taj dodatni kod ga poziva; ako ne podržava, dodatni kod izaziva pogrešku. Svaki poziv postupka ili svojstva izložen je takvoj dodatnoj nadgradnji.

Suprotno tome, sučelja omogućuju *rano povezivanje*. Kad Visual Basic tijekom prevođenja zna koje će sučelje biti pozvano, može provjeriti tip biblioteke kako bi vidio podržava li to sučelje pozvani postupak. Visual Basic zatim može prevesti izravan skok na postupak, koristeći tablicu prividnih funkcija (virtual function table, vtable). To je puno puta brže od kasnog povezivanja.

Sad pretpostavite da postupci Pomakni i Ugrizi pripadaju sučelju Životinje, te da sve životinjske klase ostvaruju to sučelje. Argument Stvorenje sad može biti određen s As Životinje, a postupci Pomakni i Ugrizi će biti rano povezani:

```
Public Sub UzmiHranu(ByVal Stvorenje As Životinje, _
ByVal Hrana As Object)
  Dim dblUdaljenost As Double
  ‘ (Kod za izračunavanje udaljenosti od hrane).
  Stvorenje.Pomakni dblUdaljenost       ‘ Rano povezivanje (vtable).
  Stvorenje.Ugrizi Hrana                ‘ Rano povezivanje (vtable).
End Sub
```

Stvaranje i ostvarivanje sučelja

Kao što je objašnjeno u odlomku "Kako Visual Basic pruža polimorfizam", ranije u ovom poglavlju, sučelje je skup svojstava i postupaka. U sljedećem primjeru programskog koda, stvorit ćete sučelje Animal i ostvariti ga u dvije klase, Flea i Tyrannosaur.

Sučelje Animal možete stvoriti dodavanjem modula klase svom projektu, dajući modulu ime Animal, te ubacivanjem sljedećeg programskog koda:

```
Public Sub Move(ByVal Distance As Double)

End Sub

Public Sub Bite(ByVal What As Object)

End Sub
```

Uočite da u ovim postupcima nema koda. Klasa Animal je *apstraktna klasa*, koja ne sadrži kod za ostvarivanje. Apstraktna klasa nije namijenjena stvaranju objekata – njezina namjena je pružanje predloška za sučelje koje dodajete drugim klasama (iako je, kako se pokazuje, ponekad korisno ostvariti sučelje klase koja nije apstraktna; to je raspravljeno kasnije u ovoj temi).

> **Napomena** Ispravno govoreći, apstraktna klasa je ona iz koje ne možete stvarati objekte. Uvijek možete stvoriti objekte iz klasa Visual Basica, čak i ako one ne sadrže kod; zbog toga one nisu zaista apstraktne.

Sad možete dodati još dva modula klase, dajući jednom ime Flea, a drugom Tyrannosaur. Kako bi ostvarili sučelje klase Animal u klasi Flea, upotrijebite naredbu Implements:

```
Option Explicit
Implements Animal
```

Odmah čim dodate ovu liniju programskog koda, možete kliknuti lijevi spuštajući okvir (Object) u kodnom prozoru. Jedna od stavki će biti Animal. Kad je odaberete, desni spuštajući okvir (Procedure) će prikazati postupke sučelja Animal.

Odaberite redom svaki postupak, kako bi stvorili prazne predloške potprograma za sve postupke. Predlošci će imati ispravne argumente i tipove podataka, kao što je određeno u klasi Animal. Svako ime potprograma će imati prefiks Animal_ za identificiranje sučelja.

> **Važno** Sučelje je kao ugovor. Ostvarivanjem sučelja, klasa pristaje odgovoriti kad je pozvano bilo koje svojstvo ili postupak sučelja. Zbog toga, morate ostvariti *sva* svojstva i postupke sučelja.

Sad možete dodati sljedeći programski kod klasi Flea:

```
Private Sub Animal_Move(ByVal Distance As Double)
  ‘ (Kod za skok nekog broja centimetara.)
  Debug.print “Flea moved”
End Sub

Private Sub Animal_Bite(ByVal What As Object)
  ‘ (Kod za isisavanje krvi.)
  Debug.Print “Flea bit, a “ & TypeName(What)
End Sub
```

Može vas čuditi zašto su potprogrami određeni tipom Private. Kad bi bili javni, potprogrami Animal_Jump i Animal_Bite bili bi dio sučelja Flea, i mi bi zapeli u istom povezivanju u kojem smo izvorno bili, određujući argument Critter (Stvorenje) s As Object kako bi mogao sadržavati ili klasu Flea ili klasu Tyrannosaur.

Višestruka sučelja

Klasa Flea sad ima dva sučelja: sučelje Animal koje ste upravo ostvarili, koje ima dva elementa, te podrazumijevano sučelje Flea, koje nema elemenata. Kasnije ćete u ovom primjeru dodati element jednom od podrazumijevanih sučelja.

Sučelje Animal možete ostvariti na sličan način za klasu Tyrannosaur:

```
Option Explicit
Implement Animal

Private Sub Animal_Move(ByVal Distance As Double)
  ‘ (Kod za skok nekog broja metara.)
  Debug.Print “Tyrannosaur moved”
End Sub

Private Sub Animal_Bite(ByVal What As Object)
  ‘ (Kod za uzimanje kilograma mesa.)
  Debug.Print “Tyrannosaur bit, a “ & TypeName(What)
End Sub
```

Vježbanje tiranosaura i buhe

Dodajte sljedeći programski kod događaju Load forme Form1:

```
Private Sub Form_Load()
  Dim fl As Flea
  Dim ty As Tyrannosaur
  Dim anim As Animal
```

```
  Set fl = New Flea
  Set ty = New Tyrannosaur
  ‘ Prvo dajemo domet buhi.
  Set anim = fl
  Call anim.Bite(ty)      ‘ Buha grize tiranosaura.
  ‘ Sad je tiranosaur na potezu.
  Set anim = ty
  Call anim.Bite(fl)      ‘ Tiranosaur grize buhu.
End Sub
```

Pritisnite F8 za prolaz kroz kod. Uočite poruke u prozoru za neposredan upis naredbi. Kad varijabla anim sadrži pokazivač na klasu Flea, pozvano je ostvarivanje postupka Bite klase Flea, jednako je i za klasu Tyrannosaur.

Varijabla anim može sadržavati pokazivač na svaki objekt koji ostvaruje sučelje Animal. Zapravo, ona može sadržavati *samo* pokazivače na takve objekte. Ako varijabli anim pokušate dodijeliti objekt forme ili okvira za sliku, pojavit će se pogreška.

Postupak Bite je rano povezan kad ga pozivate kroz varijablu anim, budući da Visual Basic tijekom prevođenja zna da će, bez obzira koji, objekt dodijeljen varijabli anim imati postupak Bite.

Prosljeđivanje tiranosaura i buha potprogramima

Sjetite se potprograma UzmiHranu (GetFood) iz odlomka “Kako Visual Basic pruža polimorfizam”. Možete dodati *drugu* verziju potprograma GetFood – onu koja ilustrira polimorfizam – formi Form1, te zamijeniti programski kod u događaju Load sljedećim:

```
Private Sub Form_Load()
  Dim fl As Flea
  Dim ty As Tyrannosaur

  Set fl = New Flea
  Set ty = New Tyrannosaur
  ‘ Buha objeduje na tiranosauru.
  Call GetFood(fl, ty)
  ‘ I obratno.
  Call GetFood(ty, fl)
End Sub
```

Prolaz korak po korak kroz ovaj kod pokazuje kako se pokazivač objekta kojeg prosljeđujete argumentu drugog tipa sučelja pretvara u pokazivač na drugo sučelje (u ovom slučaju, Animal). Događa se da Visual Basic istražuje objekt kako bi pronašao podržava li on drugo sučelje. Ako objekt to podržava, vraća pokazivač na sučelje, a Visual Basic postavlja taj pokazivač u varijablu argumenta. Ako objekt ne podržava drugo sučelje, pojavljuje se pogreška.

Ostvarivanje postupaka koji vraćaju vrijednosti

Pretpostavimo da je postupak Move vratio vrijednost. Nakon svega, znate koliko daleko želite pomaknuti životinju, ali pojedini primjerak možda nije sposoban pomaknuti se tako daleko. Povratna vrijednost postupka Move mogla bi biti upotrijebljena da vam kaže kako se daleko životinja zapravo pomaknula.

```
Public Function Move(ByVal Distance As Double) As Double
End Function
```

Kad ostvarite ovaj postupak u klasi Tyrannosaur, dodjeljujete povratnu vrijednost imenu potprograma, baš kao što bi to napravili za bilo koji potprogram tipa Function:

```
Private Function Animal_Move(ByVal Distance As Double) As Double
  Dim dblDistanceMoved As Double
  ‘ (Kod za izračunavanje kako daleko skočiti, temeljen
  ‘ na starosti, zdravstvenom stanju i preprekama.)
  ‘ Ovaj primjer pretpostavlja da je rezultat postavljen
  ‘ u varijablu dblDistanceMoved.
  Debug.Print “Tyrannosaur moved”; dblDistanceMoved
  Animal_Move = dblDistanceMoved
End Function
```

Kako bi dodijelili povratnu vrijednost, upotrijebite puno ime potprograma, uključujući prefiks sučelja.

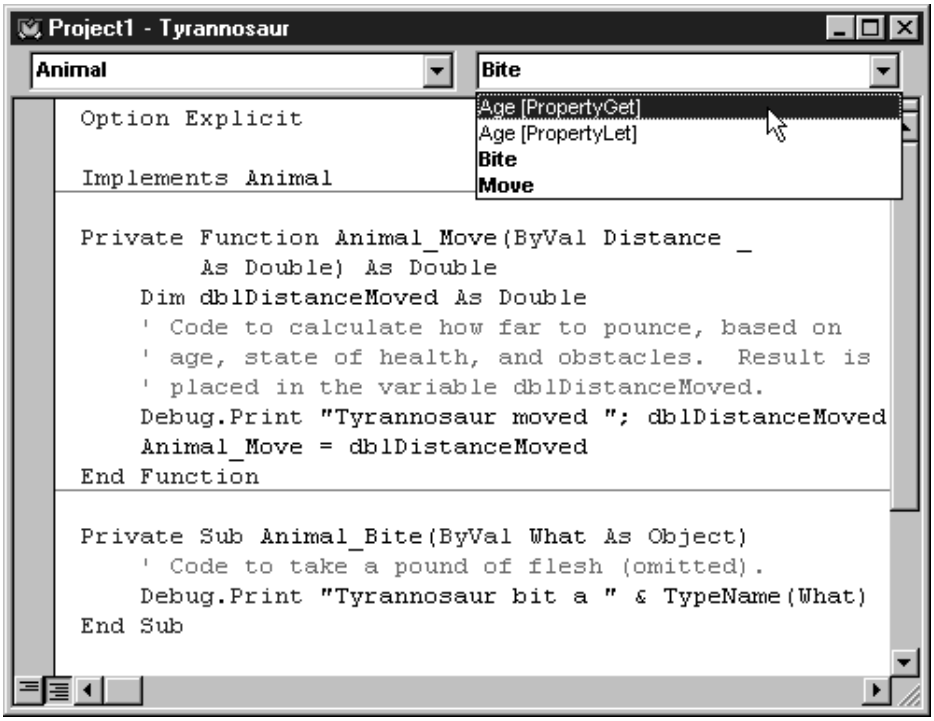
Ostvarivanje svojstava

Pretpostavimo da smo klasi Animal dali svojstvo Age, dodavanjem javne varijable u odjeljak Declarations:

```
Option Explicit
Public Age As Double
```

Spuštajući okviri Procedure u modulu koda klasa Tyrannosaur i Flea sad sadrže potprograme svojstava za ostvarivanje svojstva Age, kao što je prikazano na slici 9.10.

Slika 9.10 Ostvarivanje potprograma svojstava



Ovo prikazuje trenutak napravljen u odlomku “Dodavanje svojstava klasi”, ranije u ovom poglavlju. Upotreba javne varijable za ostvarivanje svojstva je isključivo prikladna za programera. Iza scene, Visual Basic ostvaruje svojstvo kao par potprograma svojstava.

Morate ostvariti oba potprograma. Potprogrami svojstva se lako ostvaruju spremanjem vrijednosti u elementu privatnog podatka, kao što je ovdje pokazano:

```
Private mdblAge As Double

Private Property Get Animal_Age() As Double
  Animal_Age = mdblAge
End Property

Private Property Let Animal_Age(ByVal RHS As Double)
  mdblAge = RHS
End Property
```

Element privatnog podatka je detalj ostvarivanja, pa ga morate sami dodati.

Napomena Kad naredba Implements pruža predložak za potprograme Property Set ili Property Let, nema načina za određivanje imena posljednjeg argumenta, pa je on zamijenjen imenom RHS, kao što je prikazano u gornjem primjeru.

Ne postoji provjera valjanosti podatka u svojstvu ostvarenom kao element javnog podatka, ali to ne znači da ne možete dodati kod za provjeru valjanosti potprogramu Property Let svojstva Animal_Age. Na primjer, možete poželjeti ograničiti vrijednosti godina prikladnih za tiranosaura i buhu.

Zapravo, ovo prikazuje neovisnost sučelja i ostvarivanja. Sve dok se sučelje podudara s opisom u tipskoj biblioteci, ostvarivanje može biti bilo što.

Prije nego što pređete na idući korak, maknite ostvarivanje svojstva Age samo za čitanje iz oba modula klase.

Ostvarivanje svojstva samo za čitanje

Naravno, dopuštanje određivanja starosti životinje proizvoljnom vrijednošću je loše oblikovanje objekta. Objekt treba znati svoju vlastitu starost, i pružiti je korisniku kao svojstvo samo za čitanje. Maknite javnu varijablu Age iz klase Animal, i dodajte predložak za svojstvo starosti samo za čitanje, poput ovog:

```
Public Property Get Age() As Double
End Property
```

Sad spuštajući okvir Procedure u kodnim prozorima klasa Tyrannosaur i Flea sadrži samo jedan unos, Age [Property Get]. Mogli bi to ostvariti za klasu Tyrannosaur na sljedeći način:

```
Private mdblBirth As Double

Private Property Get Animal_Age() As Double
  Animal_Age = Now - mdblBirth
End Property
```

Gornji programski kod vraća starost tiranosaura u danima. Mogli bi postaviti varijablu mdblBirth u događaju Initialize klase Tyrannosaur, ovako:

```
Private Sub Class_Initialize()
  mdblBirth = Now
End Sub
```

Naravno, mogli bi vratiti vrijednost svojstva u uobičajenijim mjernim jedinicama, kao godine starosti psa.

Prekid za kratku raspravu o objektima i sučeljima

Primjeri koda klasa Tyrannosaur i Flea vidljivo rade brzo i nesputano s sučeljima i objektima. Pokazivači na objekte su dodijeljeni jednoj varijabli objekta, a pokazivači na sučelja drugoj.

Zapravo, *svi pokazivači su pokazivači objekata*. Pokazivač na sučelje je također pokazivač na objekt koji ostvaruje sučelje. Nadalje, objekt može imati više sučelja, ali je ispod toga i dalje isti objekt.

U Visual Basicu, svaka klasa ima podrazumijevano sučelje koje ima isto ime kao i klasa. Zapravo, gotovo isto. Po dogovoru, podvlaka prethodi imenu klase. Podvlaka ukazuje da je to sučelje skriveno u tipskoj biblioteci.

Prema tome klasa Tyrannosaur ima podrazumijevano sučelje imena _Tyrannosaur. Budući da klasa Tyrannosaur također ostvaruje klasu Animal, ima drugo sučelje imena _Animal.

Međutim, ispod svega toga, objekt je i dalje klase Tyrannosaur. Postavite naredbeni gumb na formu Form1, i dodajte sljedeći kod:

```
Private Sub Command1_Click()
  Dim ty As Tyrannosaur
  Dim anim As Animal

  Set ty = New Tyrannosaur
  Set anim = ty
  MsgBox TypeName(anim)
End Sub
```

Vjerojatno pretpostavljate da će se u okviru s porukom ispisati “Animal”, međutim zapravo će se ispisati “Tyrannosaur”.

Ispitivanje sučelja

Kad objekt Tyrannosaur dodijelite varijabli tipa Animal, Visual Basic pita objekt Tyrannosaur podržava li sučelje klase Animal (postupak korišten za to naziva se QueryInterface, ili kratko QI; ponekad možete vidjeti korištenje kratice QI kao glagola). Ako je odgovor ne, pojavljuje se pogreška.

Ako je odgovor da, objekt se dodjeljuje varijabli. Kroz tu varijablu može se pristupiti samo postupcima i svojstvima sučelja Animal.

Opće varijable objekta i sučelje

Što se događa kad pokazivač objekta dodijelite općoj varijabli objekta, kao u sljedećem programskom kodu?

```
Private Sub Command1_Click()
  Dim ty As Tyrannosaur
  Dim anim As Animal
  Dim obj As Object

  Set ty = New Tyrannosaur
  Set anim = ty
  Set obj = anim
  MsgBox TypeName(obj)
End Sub
```

Rezultat je ponovno Tyrannosaur. Sad, koje sučelje dobivate kad pozivate svojstva i postupke kroz varijablu obj? Dodajte sljedeći postupak klasi Tyrannosaur:

```
Public Sub Growl()
  Debug.Print "Rrrrr"
End Sub
```

Postupak Growl pripada podrazumijevanom sučelju objekta Tyrannosaur. U programskom kodu događaja Click naredbenog gumba, zamijenite naredbu MsgBox s sljedeće dvije linije koda:

```
obj.Move 42
obj.Growl
```

Kad pokrenete projekt i kliknete gumb, izvođenje se zaustavlja na postupku Growl, s pogreškom "Objekt ne podržava to svojstvo ili postupak" (Object does not support this property or method). Jasno, sučelje je i dalje Animal.

Ovo je nešto što treba imati na umu kod korištenja varijabli tipa Object s objektima koji imaju višestruka sučelja. Sučelje kojem će varijabla pristupiti je *zadnje dodijeljeno sučelje*. Na primjer:

```
Private Sub Command1_Click()
  Dim ty As Tyrannosaur
  Dim anim As Animal
  Dim obj As Object

  Set ty = New Tyrannosaur
  Set anim = ty
  Set obj = anim
  obj.Move 42                    ' Uspješno
  obj.Growl                      ' Neuspješno

  Set obj = ty
  obj.Move 42                    ' Neuspješno
  obj.Growl                      ' Uspješno
End Sub
```

Na sreću, postoji vrlo malo razloga za korištenje sporijeg, kasnog povezivanja podatka tipa Object s objektima koji imaju višestruka sučelja. Jedan od glavnih razloga za korištenje višestrukih sučelja je dobivanje prednosti ranog povezivanja kroz polimorfizam.

Ostali izvori sučelja

Moduli klase Visual Basica nisu vaš jedini izvor sučelja za ostvarivanje. Možete ostvariti svako sučelje sadržano u tipskoj biblioteci, sve dok to sučelje podržava automatizaciju.

Ako imate Professional ili Enterprise verzije Visual Basica, možete stvoriti svoje vlastite tipove biblioteka apstraktnih klasa. Takvi tipovi biblioteka mogu biti upotrijebljeni u puno projekata, kao što je opisano u 6. poglavlju "Opća načela oblikovanja sastavnih dijelova", u 2. dijelu "Stvaranje ActiveX sastavnih dijelova" vodiča *Microsoft Visual Basic 6.0 Component Tools Guide*.

Verzije Professional i Enterprise također uključuju uslužnu aplikaciju MkTypLib (Make Type Library, stvaranje tipske biblioteke) u direktoriju Tools. Ako ste koristili ovu uslužnu aplikaciju s aplikacijom Microsoft C++, možete je smatrati primjerenijim načinom stvaranja sučelja.

Korištenje sučelja u vašem projektu

Kako bi upotrijebili sučelje u svom projektu, kliknite naredbu References u izborniku Project za otvaranje dijaloškog okvira References. Ako je tipska biblioteka registrirana, pojavit će se u listi pokazivača, i možete je odabrati. Ako tipska biblioteka nije na listi, možete upotrijebiti gumb Browse kako bi je pronašli.

Jednom kad imate pokazivač na tipsku biblioteku, možete upotrijebiti naredbu Implements za ostvarivanje svakog sučelja koje podržava automatizaciju i nalazi se u tipskoj biblioteci.

Puno sučelja (lica) ponovnog korištenja koda

Postoje dva glavna oblika ponovnog korištenja koda – binarni i strojni. Ponovno korištenje binarnog koda ostvaruje se stvaranjem i korištenjem objekta, dok se ponovno korištenje strojnog koda postiže nasljeđivanjem, koje ne podržava Visual Basic (ponovno korištenje strojnog koda može se također postići kopiranjem i mijenjanjem strojnog koda, ali ta tehnika nije ništa novo, i ima puno dobro poznatih problema).

Visual Basic je bio predvodnik ponovnog korištenja binarnog koda – kontrole su klasičan postojeći primjer. Programski kod u kontrolama možete ponovno koristiti postavljanjem primjera kontrole na svoju formu. To je poznato kao odnos *sadržavanja* ili odnos *ima*; znači, forma *sadrži* ili *ima* kontrolu CommandButton.

Za više informacija Odnosi sadržavanja raspravljeni su u odlomku "Modeli objekata" kasnije u ovom poglavlju.

Delegiranje u ostvareni objekt

Ostvarivanja pružaju moćan nov način ponovnog korištenja koda. Možete ostvariti apstraktnu klasu (kao što je raspravljeno u odlomku "Stvaranje i ostvarivanje sučelja"), ili možete ostvariti sučelje potpuno djelotvorne klase. Možete stvoriti *unutarnji objekt* (znači, ostvaren objekt) u događaju Initialize *vanjskog objekta* (znači, onog koji ostvaruje sučelje unutarnjeg objekta).

Kao što je naznačeno u odlomku “Stvaranje i ostvarivanje sučelja”, ranije u ovom poglavlju, sučelje je kao ugovor – morate ostvariti sve elemente sučelja unutarnjeg objekta u modulu klase vanjskog objekta. Unatoč tome, možete biti vrlo izbirljivi u načinu delegiranja prema svojstvima i postupcima unutarnjeg objekta. U jednom postupku mogli bi delegirati izravno unutarnjem objektu, prosljeđujući nepromijenjene argumente, dok u drugom postupku možete izvesti neki vlastiti kod prije pozivanja unutarnjeg objekta, a u trećem postupku mogli bi izvesti samo svoj vlastiti kod, potpuno zanemarujući unutarnji objekt!

Na primjer, pretpostavimo da imate klasu JedanSvirač u klasi Kakofonija, gdje obje stvaraju zvukove. Željeli bi dodati djelotvornost klase Kakofonija klasi JedanSvirač, te ponovno upotrijebiti neka od ostvarivanja postupaka klase Kakofonija.

```
‘ Klasa JedanSvirač ostvaruje sučelje klase Kakofonija.
Implements Kakofonija

‘ Varijabla objekta za čuvanje pokazivača.
Private mkak As Kakofonija

Private Sub Class_Initialize()
  ‘ Stvaranje objekta.
  Set mkak = New Kakofonija
End Sub
```

Sad možete otići u spuštajući izbornik Object i odabrati klasu Kakofonija, te dobiti predloške potprograma za postupke sučelja klase Kakofonija. Kako bi ostvarili te postupke, možete ih delegirati objektu Kakofonija. Na primjer, postupak Pisak mogao bi izgledati ovako:

```
Private Sub Kakofonija_Pisak(ByVal Frekvencija As Double, _
ByVal Trajanje As Double)
  ‘ Delegiranje unutarnjem objektu Kakofonija.
  Call mkak.Pisak(Frekvencija, Trajanje)
End Sub
```

Gornje ostvarivanje je vrlo jednostavno. Vanjski objekt (JedanSvirač) delegira izravno unutarnjem (Kakofonija), ponovno koristeći postupak Pisak objekta Kakofonija bez ikakvih promjena. To je dobra stvar, ali je samo početak.

Naredba Implement je vrlo moćan alat za ponovno korištenje koda, budući da vam daje ogromnu fleksibilnost. Možete odlučiti promijeniti djelovanje postupka Pisak klase JedanSvirač, ubacivanjem vašeg vlastitog koda prije (ili nakon) poziva unutarnjeg objekta Kakofonija:

```
Private Sub Kakofonija_Pisak(ByVal Frekvencija As Double, _
ByVal Trajanje As Double)
  ‘ Podizanje svega za oktavu.
  Frekvencija = Frekvencija * 2
  ‘ Temeljeno na drugom svojstvu klase JedanSvirač,
  ‘ Staccato, rezanje trajanja svakog piska.
```

```
  If Staccato Then Trajanje = Trajanje *7 / 8
  Call mkak.Pisak(Frekvencija, Trajanje)
  ' Mo`ete ~ak pozvati druge postupke klase JedanSvira~.
  If Staccato Then Pauza(Trajanje * 1 / 8)
End Sub
```

Za neke postupke, vaše ostvarivanje može delegirati izravno unutarnjem objektu Kakofonija, dok za druge možete umetnuti svoj vlastiti programski kod prije i nakon delegiranja – ili čak izostaviti delegiranje u potpunosti, koristeći potpuno svoj vlastiti kod za ostvarivanje postupka.

Budući da klasa JedanSvirač ostvaruje sučelje klase Kakofonija, možete je upotrijebiti s bilo kojom glazbenom aplikacijom koja poziva to sučelje. Detalji vašeg ostvarivanja su skriveni od pozivajuće aplikacije, ali rezultirajući zvukovi su potpuno vaši.

Napomena Model objekta kao sastavnog dijela (COM) pruža drugi mehanizam za ponovno korištenje binarnog koda, nazvano *gomilanje* (*aggregation*). U gomilanju, ponovno se koristi cijelo sučelje, bez ikakvih promjena, a ostvarivanje se pruža primjerom klase koja je gomilana. Visual Basic ne podržava takav oblik ponovnog korištenja koda.

Nije li ovo postalo zamorno?

Pisanje koda delegiranja može zaista postati dosadno, posebno ako većina svojstava i postupaka vanjskih objekata jednostavno delegira izravno pripadajućim svojstvima i postupcima unutarnjeg objekta.

Ako imate Professional ili Enterprise verzije Visual Basica, možete upotrijebiti model proširivanja Visual Basica (Visual Basic Extensibility) za stvaranje vaših vlastitih čarobnjaka delegiranja kako bi automatizirali zadatak, slično čarobnjaku klasa (Class Wizard) koji je uključen u Professional i Enterprise verzije.

Za više informacija Korištenje polimorfizma i višestrukih sučelja u softveru sastavnih dijelova raspravljeno je u 6. poglavlju "Opća načela oblikovanja sastavnih dijelova", u 2. dijelu "Stvaranje ActiveX sastavnih dijelova", vodiča *Microsoft Visual Basic 6.0 Component Tools Guide*.

Upotreba modela proširivanja dokumentirana je u 3. dijelu "Proširivanje okoline Visual Basica dodacima", u vodiču *Microsoft Visual Basic 6.0 Component Tools Guide*.

Programiranje vašim vlastitim objektima

Objekte možete početi koristiti postupno, pronalazeći upotrijebljive zadatke za koje je udruživanje programskog koda i podataka prednost. Djelotvornost tih objekata možete iskoristiti određivanjem varijabli objekta, dodjeljivanjem novih objekata tim varijablama, te pozivanjem svojstava i postupaka tih objekata.

Kako ćete svojim aplikacijama dodavati sve više i više objekata, početi ćete uočavati odnose među njima. Možete početi stvarati dizajn aplikacije ovisniji o objektima i

njihovim odnosima, te možete početi koristiti snažnije tehnike – kao stvaranje korisničkih klasi zbirke – za iskazivanje tih odnosa u programskom kodu.

U nekom trenutku, iznenada ćete vidjeti kako međusobno povezivanje objekata mijenja samu narav vaše aplikacije, i bit ćete spremni započeti oblikovanje objektno te–meljnih aplikacija iz temelja.

Sljedeće teme pružaju pregled takvih razvojnih promjena u vašem stilu kodiranja. Pročitajte ih sad, kako bi dobili grubu sliku kamo stremite, i pročitajte ih ponovno kad se vaše ideje objektno temeljenog programiranja počnu zgušnjavati.

Za više informacija Sastavni dijelovi tipa ActiveX otvaraju još jednu dimenziju ponovnog korištenja koda i objektno temeljenog programiranja. Ako imate Professional ili Enterprise verzije Visual Basica, možete početi istraživati tu dimenziju kroz 2. dio "Stvaranje ActiveX sastavnih dijelova", u vodiču *Microsoft Visual Basic 6.0 Component Tools Guide*.

Pokazivači objekata i brojanje pokazivača

Prvo pravilo za životni vijek objekta je vrlo jednostavno: objekt je uništen kad je posljednji pokazivač na njega otpušten. Međutim, kao i u stvarnom životu, jednostavno ne znači uvijek i lako.

Kad koristite više objekata, i čuvate više varijabli koje sadrže pokazivače na te objekte, možete proći kroz razdoblja kad izgleda nemoguće pustiti vaše objekte da odu kad to poželite.

U nekom trenutku, past će vam na pamet da Visual Basic mora pratiti pokazivače objekata – kako bi inače znao kad je otpušten posljednji pokazivač na objekt? Možete početi misliti da bi ispravljanje bilo puno lakše, samo kad bi mogli dobiti pristup broju pokazivača Visual Basica.

Na žalost, to nije točno. Kako bi korištenje objekata napravio djelotvornijim, model objekta kao sastavnog dijela (Component Object Model, COM) određuje broj složenih prečica za svoja pravila brojanja pokazivača. Čisti rezultat je da ne bi mogli vjerovati vrijednosti broja pokazivača sve i da mu možete pristupiti.

Prema pravilima COM-a, jedina informacija na koju se možete osloniti je *je li broj pokazivača nula ili nije*. Znate kad broj pokazivača postane nula, jer se pojavi događaj Terminate vašeg objekta. Iza toga, ne postoje pouzdane informacije koje bi mogle biti skupljene iz broja pokazivača.

Napomena Činjenica da ne trebate zapamtiti pravila COM-a o brojanju pokazivača nije mala stvar. Osobno upravljanje brojanjem pokazivača je puno teže od praćenja koje varijable objekta u vašoj aplikaciji sadrže pokazivače na objekte.

Savjet Odredite svoje varijable objekta tipom klase, umjesto s As Object. Na taj način, ako imate objekt Spravica koje se ne ukida, jedine varijable o kojima se trebate brinuti su one određene s As Spravica.

Za zbirke pokazivača objekta, ne upotrebljavajte objekt Collection Visual Basica sam za sebe. Pokazivači objekta su u objektu Collection Visual Basica spremljeni s tipom Variant – koji, kao varijable određene s As Object, može sadržavati pokazivače na objekte bilo koje klase. Umjesto toga stvorite svoje vlastite klase zbirke koje prihvaćaju objekte samo jedne klase, kao što je opisano u odlomku "Stvaranje vlastitih klasa zbirki", kasnije u ovom poglavlju. Na taj način, jedine zbirke u kojima trebate tražiti svoj objekt Spravica su one tipa Spravica.

Organizirajte svoj objekt u hijerarhiju, kao što je opisano u idućem odlomku "Modeli objekata". Ako su svi vaši objekti povezani, jednostavno je napisati potprogram koji će prošetati kroz cijeli model i izvijestiti o postojećim objektima.

Ne određujte varijable s As New. Takve varijable su kao one rođendanske svijećice koje se ponovno zapale kad ih ugasite: ako upotrijebite takvu varijablu nakon što ste je postavili na Nothing, Visual Basic uslužno stvara drugi objekt.

Za više informacija Kružni pokazivači su vrsta koju je najteže potpuno ukinuti. Pogledajte sljedeći odlomak "Modeli objekata".

Modeli objekata

Jednom kad odredite klasu stvaranjem modula klase i dodavanjem svojstava i postupaka, iz te klase možete stvoriti bilo koji broj objekata. Kako ćete pratiti trag objekata koje stvarate?

Najjednostavniji način praćenja objekata je određivanje varijable objekta za svaki objekt koji namjeravate stvoriti. Naravno, takav način postavlja ograničenje broja objekata koje možete stvoriti.

Pokazivače na više objekata možete držati u matrici ili zbirci, kao što je raspravljeno u odlomcima "Stvaranje matrica objekata" i "Stvaranje zbirki objekata", ranije u ovom poglavlju.

U početku, vjerojatno ćete smještati varijable objekta, matrice i zbirke u forme ili standardne module, kao što to radite s uobičajenim varijablama. Ipak, kako budete dodavali više klasa, vjerojatno ćete otkriti da objekti koje koristite imaju jasne međusobne odnose.

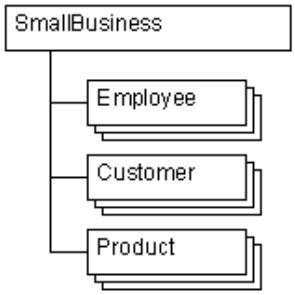
Modeli objekata pokazuju odnose povezivanja

Modeli objekata daju ustroj objektno temeljnoj aplikaciji. Određivanjem odnosa među objektima koje koristite u svojoj aplikaciji, model objekata organizira vaše objekte na način koji programiranje čini lakšim.

Tipično, model objekta pokazuje činjenicu da su neki objekti "veći", ili važniji od ostalih – o tim objektima može se razmišljati kao o spremnicima drugih objekata, ili kao da su sčinjeni od drugih objekata.

Na primjer, možete stvoriti objekt SmallBusiness kao jezgru svoje aplikacije. Možete poželjeti da objekt SmallBusiness ima druge tipove objekata koji su mu pridruženi, kao što su objekt Employee i objekt Customer. Vjerojatno ćete poželjeti da sadrži i objekt Product. Model objekta za takvu aplikaciju prikazan je na slici 9.11.

Slika 9.11 Model objekta



Možete odrediti četiri modula klase, s imenima SmallBusiness, Employee, Customer i Product, i dati svakom prikladna svojstva i postupke, ali kako ćete napraviti veze među objektima? Za tu namjenu imate dva alata: svojstva Object i objekt Collection. Sljedeći dio programskog koda pokazuje jedan način ostvarivanja hijerarhije s slike 9.11.

```
‘ Kod za odjeljak Declarations
‘ modula klase SmallBusiness.
Public Name As String
Public Product As New Product
Public Employees As New Collection
Public Customers As New Collection
```

Prvi put kad se uputite na svojstvo Product, objekt će biti stvoren, budući da je određen s As New. Na primjer, sljedeći programski kod mogao bi stvoriti i odrediti ime i cijenu objekta Product u objektu SmallBusiness.

```
‘ Kod za standardni modul.
Public sbMain As New SmallBusiness
Sub Main
  sbMain.Name = “Poduzeće Velociraptor, d.d.”
  ‘ Prvi put kad se varijabla Product upotrijebi
  ‘ u kodu, bit će stvoren objekt Product.
  sbMain.Product.Name = “Velociraptor za napuhavanje”
  sbMain.Product.Price = 1.98
  .
  . ‘ Kod za pokretanje i prikaz glavne forme.
  .
End Sub
```

Napomena Ostvarivanje svojstva objekta s javnim varijablama je površno. Mogli bi nepažnjom uništiti objekt Product postavljanjem svojstva na Nothing negdje u svom kodu. Bolje je stvoriti svojstva objekta kao svojstva samo za čitanje, kao što je pokazano u sljedećem dijelu programskog koda.

```
‘ Kod za snažnije svojstvo objekta. Spremanje
‘ svojstva je privatno, pa ne može biti
‘ postavljeno na Nothing izvan objekta.
Private mProduct As New Product
```

```
Property Get Product() As Product
  ‘ Prvi put kad se pozove ovo svojstvo, mProduct
  ‘ sadržava Nothing, pa će Visual Basic
  ‘ stvoriti objekt Product.
  Set Product = mProduct
End Property
```

Odnosi objekata jedan-za-puno

Svojstva objekata dobro rade kad su odnosi među objektima jedan-za-jedan. Međutim, često se događa da objekt jednog tipa sadrži više objekata drugog tipa. U modelu objekta SmallBusiness, svojstvo Employees je ostvareno kao objekt tipa Collection, pa objekt SmallBusiness može sadržavati više objekata tipa Employee. Sljedeći dio programskog koda pokazuje kako novi objekti tipa Employee mogu biti dodani toj zbirci.

```
Public Function NewEmployee(Name, Salary, _
HireDate, ID) As Employee
  Dim empNew As New Employee
  empNew.Name = Name             ‘ Bezuvjetno stvaranje objekta.
  empNew.Salary = Salary
  empNew.HireDate = HireDate
  ‘ Dodavanje zbirci, korištenjem ID kao ključa.
  sbMain.Employees.Add empNew, CStr(ID)
  ‘ Vraćanje pokazivača na novog djelatnika.
  Set NewEmployee = empNew
End Function
```

Funkcija NewEmployee može biti pozivana potreban broj puta kako bi stvorila djelatnike za posao predstavljen objektom SmallBusiness. Postojeći djelatnici mogu biti ispisani u svako vrijeme prolazom kroz zbirku Employees.

Napomena Još jednom, ovo nije snažno ostvarivanje. Bolja praksa je stvaranje vaših vlastitih klasi zbirki, te njihovo izlaganje kao svojstva samo za čitanje. To je raspravljeno u odlomku “Stvaranje vlastitih klasa zbirki”, kasnije u ovom poglavlju.

Savjet Uslužni dodatak za gradnju klasa, Class Builder, uključen u Professional i Enterprise verzije Visual Basica, može stvoriti većinu koda kojeg trebate za ostvarivanje modela objekta. Dodatak Class Builder stvara snažna svojstva objekta i klase zbirki, te vam omogućuje jednostavno preuređivanje vašeg modela.

Svojstva Parent

Kad imate pokazivač na objekt, možete pristupiti objektima koje sadrži korištenjem njegovih svojstava i zbirki. Također je vrlo korisno omogućiti kretanje prema gore u hijerarhiji, kako bi došli do objekta koji sadrži objekt na kojeg imate pokazivač.

Kretanje prema gore se obično radi s svojstvima Parent. Svojstvo Parent vraća pokazivač na spremnik objekta. Za raspravu o kretanju kroz model objekta, pogledajte "Kretanje kroz modele objekata" u 10. poglavlju "Programiranje sastavnim dijelovima".

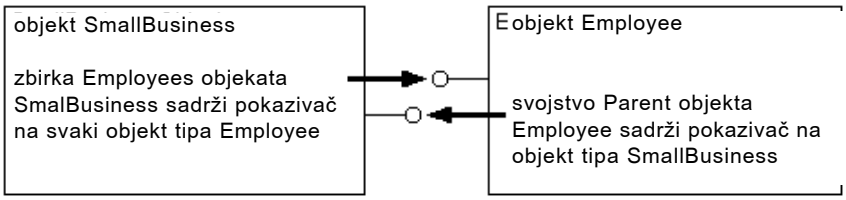
Primjer svojstva Parent možete pronaći u odlomku "Dodavanje svojstava klasi" ranije u ovom poglavlju.

Savjet Kad dodijelite svojstvo Parent objektu u zbirci, ne upotrijebljavajte pokazivač na objekt Collection. Stvarni roditelj objekta je objekt koji sadrži zbirku. Ako svojstvo Parent pokazuje na zbirku, morat ćete upotrijebiti dvije razine posredovanja kako bi došli do stvarnog roditelja – znači, obj.Parent.Parent umjesto obj.Parent.

Svojstva Parent, kružni pokazivači i razaranje objekata

Jedan od najvećih problema s svojstvima Parent je što ona stvaraju kružne pokazivače. Objekt koji je "veći" ima pokazivač na objekt kojeg sadrži, a sadržani objekt ima pokazivač kroz svoje svojstvo Parent, stvarajući petlju kao što je prikazano na slici 9.12.

Slika 9.12 Slučaj kružnih pokazivača



Što nije u redu s ovom slikom? Način na koji ćete se riješiti objekata kad ste s njima gotovi je otpuštanje svih pokazivača na te objekte. Pretpostavljajući da je pokazivač na objekt SmallBusiness u varijabli imena sbMain, kao ranije u ovoj temi, mogli bi napisati sljedeći kod:

```
Set sbMain = Nothing
```

Na žalost, i dalje postoji pokazivač na objekt SmallBusiness – zapravo, može biti puno pokazivača, jer će svojstvo Parent svakog objekta Employee sadržavati pokazivač na objekt SmallBusiness.

Budući da zbirka Employees objekta SmallBusiness sadržava pokazivače na svaki objekt Employee, ni jedan od objekata nikad neće biti uništen.

Postupci razaranja

Jedno rješenje je dati postupak Razaranje objektu SmallBusiness. Taj postupak bi mogao postaviti sva svojstva objekata u objektu SmallBusiness na Nothing, te također postaviti sve objekte tipa Collection (Employees, Customers) na Nothing.

Kad je uništen objekt tipa Collection, Visual Basic postavlja sve pokazivače objekata koje on sadrži na Nothing. Ako više nema drugih pokazivača na objekte Employee i Customer koji su bili sadržani u zbirkama Employees i Customers, oni će biti uništeni.

Naravno, ako je objekt Employee sčinjen od drugih objekata, imat će isti problem kružnih pokazivača kao i njegov roditelj. U tom slučaju, klasi Employee ćete morati dati postupak Razaranje. Umjesto da samo postavi sve pokazivače objekata u zbirci Employees na Nothing, objekt SmallBusiness će najprije morati proći kroz zbirku, pozivajući postupak Razaranje svakog objekta Employee.

Još nije gotovo

Čak i tada, možda neće biti uništeni svi objekti. Ako bilo gdje u vašoj aplikaciji postoje varijable koje i dalje sadrže pokazivače na objekt SmallBusiness, ili na bilo koji od objekata koji on sadržava, ti objekti neće biti uništeni. Dio pospremanja vaše aplikacije mora osigurati da su sve varijable objekata svugdje postavljene na Nothing.

Kako bi ispitali događa li se to, možete poželjeti dodati nekakav kod za ispravljanje svojim objektima. Na primjer, možete dodati sljedeći kod standardnom modulu:

```
‘ Opća zbirka ispravljanja
Public gcolDebug As New Collection

‘ Op}a funkcija za davanje jedinstvenog ID-a svakom objektu.
Public Function DebugSerial() As Long
  Static lngSerial As Long
  lngSerial = lngSerial + 1
  DebugSerial = lngSerial
End Function
```

U svakom modulu klase, možete postaviti kod sličan sljedećem. Tamo gdje se pojavljuje “Product” treba napisati ime svake klase.

```
‘ Spremanje ID-a za ispravljanje.
Private mlngDebugID As Long

Propety Get DebugID() As Long
  DebugID = mlngDebugID
End Property

Private Sub Class_Initialize()
  mlngDebugID = DebugSerial
  ‘ Dodavanje stringa općoj zbirci.
  gcolDebug.Add “Product Initialize: DebugID=” _
  & DebugID, CStr(DebugID)
End Sub

Private Sub Class_Terminate()
  ‘ Micanje stringa, kako bi znali
  ‘ da više nema objekta u blizini.
  gcolDebug.Remove CStr(DebugID)
End Sub
```

Kako se svaki objekt stvara, postavlja string u opću zbirku; kako se uništava, miče svoj string. U svako vrijeme, možete proći kroz opću zbirku kako bi vidjeli koji objekti nisu uništeni.

Za više informacija Modeli objekata preuzimaju nov značaj, i drugačiji skup problema, kad koristite Professional ili Enterprise verzije Visual Basica za stvaranje ActiveX sastavnih dijelova. Pogledajte 6. poglavlje "Opća načela oblikovanja sastavnih dijelova", u 2. dijelu "Stvaranje ActiveX sastavnih dijelova", vodiča *Microsoft Visual Basic 6.0 Component Tools Guide*.

Stvaranje vlastitih klasa zbirki

Postoje tri uobičajena pristupa koje možete poduzeti za ostvarivanje sadržavanja objekata korištenjem zbirki. Uzmite u obzir zbirku Employees objekta SmallBusiness raspravljenog u odlomku "Modeli objekata". Kako bi ostvarili tu zbirku možete:

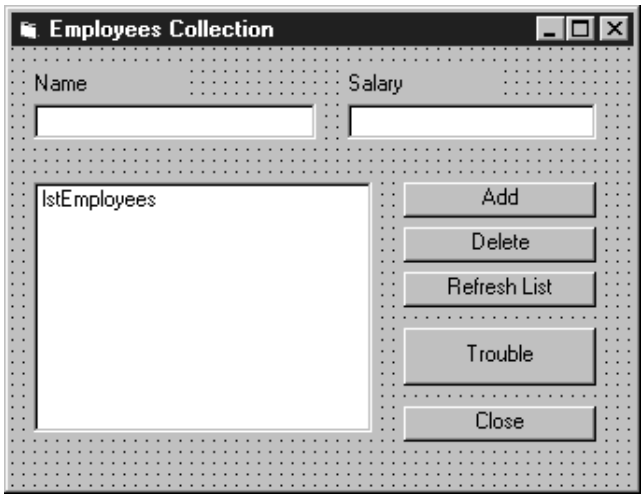
- U modulu klase SmallBusiness odrediti varijablu Employees s As Collection, i učiniti je javnom (Public). To je banalno rješenje.
- U modulu klase SmallBusiness odrediti varijablu mcolEmployees s As Collection, i učiniti je privatnom (Private). Dajte objektu SmallBusiness skup postupaka za dodavanje i brisanje objekata. To je najslabiji objektno orijentirano oblikovanje od ova tri.
- Ostvariti svoju vlastitu klasu zbirke, stvaranjem modula klase zbirke s imenom Employees, kao što je opisano kasnije u ovom poglavlju. Dajte objektu SmallBusiness svojstvo samo za čitanje klase Employees.

Strategije su izlistane po rastućem redoslijedu snage. Mogu biti obilježene kao pristupi slični kući od slame, kući od šiblja i kući od cigli.

Primjer javne zbirke: kuća od slame

Kako bi stvorili ovaj primjer, otvorite nov projekt i dodajte dva modula klase. Na formi stvorite pet naredbenih gumbâ, okvir s popisom, dva okvira s tekstom, i dvije kontrole natpisa, kao što je prikazano na slici 9.13.

Slika 9.13 Primjer zbirke Employees



Sljedeća tablica ispisuje vrijednosti svojstava koje trebate postaviti za ovaj primjer:

objekt	svojstvo	postavka
modul klase	Name	Employee
modul klase	Name	SmallBusiness
forma	Caption	Employees Collection
prvi naredbeni gumb	Caption Name	Add cmdAddEmployee
drugi naredbeni gumb	Caption Name	Delete cmdDeleteEmployee
treći naredbeni gumb	Caption Name	Refresh List cmdListEmployees
četvrti naredbeni gumb	Caption Name	Trouble cmdTrouble
peti naredbeni gumb	Caption Name	Close cmdClose
prva kontrola natpisa	Caption	Name
druga kontrola natpisa	Caption	Salary
prvi okvir s tekstom	Name Text	txtName (prazno)
drugi okvir s tekstom	Name Text	txtSalary (prazno)
okvir s popisom	Name	lstEmployees

U modulu klase Employee, dodajte sljedeća određivanja i potprograme svojstava:

```
Option Explicit
‘ Svojstva klase Employee.
Public Name As String
Public Salary As Long

‘ Privatni podatak za svojstvo ID samo jednom za pisanje.
Private mstrID As String

Property Get ID() As String
  ID = mstrID
End Property

‘ Kad se prvi put postavi svojstvo ID, također se postavlja
‘ statični Boolean. Sljedeći pozivi ne čine ništa.
‘ (Umjesto toga, bolje bi bilo izazvati pogrešku.)
Property Let ID(strNew As String)
  Static blnAlreadySet As Boolean
  If Not blnAlreadySet Then
    blnAlreadySet = True
    mstrID = strNew
  End If
End Property
```

Svojstvo ID je ključ za dohvaćanje ili brisanje objekta Employee iz zbirke, pa mora biti postavljeno jednom i nikad se ne mijenja. To se ostvaruje statičkom varijablom tipa Boolean koja se postavlja na True prvi put kad se postavlja svojstvo. Svojstvo se uvijek može čitati, pošto postoji potprogram Property Get.

U modulu klase SmallBusiness, dodajte sljedeće određivanje. Objekt zbirke bit će stvoren kad se prvi put pozove varijabla Employees u programskom kodu.

```
Option Explicit
Public Employees As New Collection
```

Forma radi cijeli posao

Sav preostali programski kod ide u modul forme. Dodajte sljedeće određivanje u odjeljak Declarations.

```
Option Explicit
Public sbMain As New SmallBusiness
```

Programski kod u događaju cmdAddEmployee_Click dodaje element zbirci.

```
Private Sub cmdAddEmployee_Click()
  Dim empNew As New Employee
  Static intEmpNum As Integer
  ‘ Upotreba izraza With čini vaš kod bržim i
  ‘ kraćim (.ID umjesto empNew.ID)
  With empNew
    ‘ Stvaranje jedinstvenog ID-a za novog djelatnika.
    intEmpNum = intEmpNum + 1
    .ID = “E” & Format$(intEmpNum “00000”)
    .Name = txtName.Text
    .Salary = CDbl(txtSalary.Text)
    ‘ Dodavanje pokazivača objekta Employee zbirci,
    ‘ korištenjem svojstva ID kao ključa.
    sbMain.Employees.Add empNew, .ID
  End With
  txtName.Text = “”
  txtSalary.Text = “”
  ‘ Klik gumba Refresh List.
  cmdListEmployees.Value = True
End Sub
```

Programski kod u potprogramu događaja cmdListEmployees_Click koristi izraz For Each...Next za dodavanje informacije o djelatniku u kontrolu okvira s popisom.

```
Private Sub cmdListEmployees_Click()
  Dim emp As Employee
  lstEmployees.Clear
  For Each emp In sbMain.Employees
    lstEmployees.AddItem emp.ID & “, “ & emp.Name _
    & “, “ & emp.Salary
  Next
End Sub
```

Događaj cmdDeleteEmployee_Click koristi postupak Remove objekta Collection za brisanje elementa zbirke koji je trenutno odabran u kontroli okvira s popisom.

```
Private Sub cmdDeleteEmployee_Click()
  ‘ Provjera je li odabran djelatnik.
  If lstEmployees.ListIndex      > -1 Then
    ‘ Prvih šest karaktera su ID.
    sbMain.Employees.Remove Left(lstEmployees.Text, 6)
  End If
  ‘ Klik gumba Refresh List.
  cmdListEmployees.Value = True
End Sub
```

Dodajte sljedeći programski kod gumbu Trouble.

```
Private Sub cmdTrouble_Click()
  ‘ Što si rekao?
  sbMain.Employees.Add Me
End Sub
```

Događaj cmdClose_Click zatvara aplikaciju. Kad zatvorite projekte koji koriste objekte, napravite to izbacivanjem svih formi, kao bi osigurali izvođenje svih potprograma događaja Terminate u vašim modulima klase. Suprotno tome, upotreba naredbe End zaustavlja aplikaciju iznenadno, bez izvođenja događaja Terminate.

```
Private Sub cmdClose_Click()
  Unload Me
End Sub
```

Kako bi dodali djelatnike u primjeru, pokrenite aplikaciju, unesite vrijednosti u dva okvira s tekstom, te odaberite gumb Add. Dodajte nekoliko djelatnika, te eksperimentirajte s gumbima za brisanje i izlistavanje.

Čvrsto kao kuća od slame

Ovakvo jednostavno ostvarivanje nije jako čvrsto. Budući da je svojstvo Employee samo javni objekt tipa Collection, možete mu slučajno pristupiti s bilo kojeg mjesta u svojoj aplikaciji. Nadalje, postupak Add objekta Collection ne radi nikakav oblik provjere. Na primjer, programski kod u događaju Click gumba Trouble veselo ubacuje pokazivač objekta na formu u zbirku djelatnika.

Kliknite gumb Trouble, i uočite da se nije pojavila pogreška. Sad kliknite gumb Refresh List. Kad petlja For Each...Next otkrije neočekivan tip objekta, uzrokovat će pogrešku 13, Pogrešan tip podatka (Type mismatch).

To je primjer vrste pogreške kojoj ste izloženi kad gradite model objekta s javnim objektima tipa Collection. Objekti mogu biti dodani bilo gdje iz vašeg projekta, i ne postoji jamstvo da će biti ispravno pokrenuti. Ako programer klonira programski kod za dodavanje djelatnika, a strojni kod se kasnije promijeni, bit će vrlo teško uhvatiti izvor rezultirajućih pogrešaka.

Primjer privatne zbirke: kuća od šiblja

Donekle snažniji način povezivanja objekata Employee s objektom SmallBusiness je stvaranje privatnog objekta tipa Collection. Za ovaj primjer, ponovno ćemo upotrijebiti formu i većinu koda iz primjera javne zbirke.

Modul klase Employee ostaje nepromijenjen. Međutim, modul klase SmallBusiness dobiva potpuno obnavljanje. Zamijenite određivanje javnog objekta Collection sljedećim određivanjem, te dodajte potprograme tipa Sub i Function opisane u sljedećim odlomcima.

```
Option Explicit
Private mcolEmployees As New Collection
```

Kao i prije, programski kod koji dodaje djelatnika čini većinu posla (blok koda između linija s crticama možete izvući iz potprograma događaja cmdAddEmployee_Click u prethodnom primjeru).

Važna promjena je što se postupak Add objekta Collection više ne poziva iz bilo kojeg modula u vašoj aplikaciji, pošto je varijabla mcolEmployees privatna. Objekt Employee možete pozvati samo korištenjem postupka EmployeeAdd, koji ispravno pokreće novi objekt:

```
‘ Postupak klase SmallBusiness.
Public Function EmployeeAdd(ByVal Name As String, _
ByVal Salary As Double) As Employee
‘ - - - - - - - - - - - - - - - - -
  Dim empNew As New Employee
  Static intEmpNum As Integer
  ‘ Upotreba izraza With čini vaš kod bržim i
  ‘ kraćim (.ID umjesto empNew.ID)
  With empNew
    ‘ Stvaranje jedinstvenog ID-a za novog djelatnika.
    intEmpNum = intEmpNum + 1
    .ID = “E” & Format$(intEmpNum “00000”)
    .Name = txtName.Text
    .Salary = CDbl(txtSalary.Text)
    ‘ Dodavanje pokazivača objekta Employee zbirci,
    ‘ korištenjem svojstva ID kao klju~a.
    ‘ - - - - - - - - - - - - - - - - -
    mcolEmployee.Add empNew, .ID
  End With
  ‘ Vraćanje pokazivača na novog djelatnika.
  Set EmployeeAdd = empNew
End Function
```

Postupak EmployeeAdd vraća pokazivač na novo dodani objekt Employee. Ovo je dobra praksa, jer ćete odmah čim stvorite objekt najvjerojatnije željeti i nešto napraviti s njim.

Postupci EmployeeCount, EmployeeDelete i Employees *delegiraju* pripadajućim postupcima objekta Collection. Delegiranje znači da objekt Collection radi sav posao.

```
‘ Postupci klase SmallBusiness.
Public Function EmployeeCount() As Long
  EmployeeCount = mcolEmployees.Count
End Function
```

```
Public Sub EmployeeDelete(ByVal Index As Variant)
  mcolEmployees.Remove Index
End Sub

Public Function Employees(ByVal Index As Variant) As Employee
  Set Employees = mcolEmployees.Item(Index)
End Function
```

Napomena Ovim postupcima možete dodati posebne djelotvornosti. Na primjer, možete izazvati vaše vlastite pogreške ako je indeks neispravan.

Posljednji postupak je Trouble. Taj postupak pokušava dodati nepokrenuti objekt Employee zbirci. Pogađate li što će se dogoditi?

```
‘ Postupak klase SmallBusiness.
Public Sub Trouble()
  Dim x As New Employee
  mcolEmployees.Add x
End Sub
```

Promjene forme

Trebat ćete napraviti nekoliko promjena u modulu klase. Možete upotrijebiti ista određivanja na razini modula korištena za prethodni primjer, događaj Click gumba Close ostaje isti, ali mijenjaju se ostali potprogrami događaja – programski kod gumba Add je puno kraći, dok programski kod za gumbe Delete i List Employees ima male, ali značajne promjene:

```
Private Sub cmdAddEmployee_Click()
  sbMain.EmployeeAdd txtName.Text, txtSalary.Text
  txtName.Text = “”
  txtSalary.Text = “”
  cmdListEmployees.Value = True
End Sub

Private Sub cmdDeleteEmployee_Click()
  ‘ Provjera je li odabran djelatnik.
  If lstEmployees.ListIndex > -1 Then
    ‘ Prvih šest karaktera su ID.
    sbMain.EmployeeDelete Left(lstEmployees.Text, 6)
  End If
  cmdListEmployees.Value = True
End Sub
```

```
Private Sub cmdListEmployees_Click()
  Dim lngCt As Long
  lstEmployees.Clear
  For lngCt = 1 To sbMain.EmployeeCount
    With sbMain.Employees(lngCt)
          lstEmployees.AddItem .ID & “, “ & .Name _
          & “, “ & .Salary
    End With
  Next
End Sub
```

Ali što je s svim dodatnim kodom u događaju cmdListEmployees_Click? Na žalost, u potrazi za snagom morate odustati od mogućnosti korištenja petlje For Each...Next za prolaz kroz stavke zbirke, budući da je objekt Collection sad određen kao privatni. Ako pokušate izvesti sljedeći kod, dobit ćete samo pogrešku:

```
‘ Neće raditi, pošto Employees nije stvarno zbirka.
For Each emp In sbMain.Employees
```

Na sreću, postupak EmployeeCount može se upotrijebiti za ograničavanje opsega prolaza.

Gumb Trouble također se malo promijenio, ali i dalje mu je namjena izazvati probleme.

```
Private Sub cmdTrouble_Click()
  sbMain.Trouble
End Sub
```

Pokrenite projekt i eksperimentirajte s gumbima Add, Delete i Refresh List. Sve radi isto kao i prije.

Kad kliknete gumb Trouble, još jednom neće biti stvorena pogreška. Međutim, ako nakon toga kliknete gumb Refresh List, možete vidjeti da je nepokrenuti objekt Employee nekako dodan zbirci.

Kako se to može dogoditi? Stvaranjem objekta Collection kao privatnog, zaštitili ste ga od svog koda u svojoj aplikaciji koji je *izvan* objekta SmallBusiness, ali ne i od koda *unutar*. Objekt SmallBusiness može biti velik i složen, s priličnom količinom koda u njemu. Na primjer, on će vrlo vjerojatno imati postupke kao AddCustomer, AddProduct i tako dalje.

Pogreška u kodiranju, ili stvaranje duplikata postupka EmployeeAdd, i dalje može kao rezultat imati pogrešne podatke - čak i neispravne objekte – ubačene u zbirku, budući da je privatna varijabla vidljiva preko modula klase.

Stvaranje vlastite klase zbirke: kuća od cigli

Najsnažniji način ostvarivanja zbirke je da je stvorite kao modul klase. Suprotno prethodnim primjerima, postavljanje cijelog koda za stvaranje objekta u klasu zbirke slijedi dobra načela oblikovanja objekata.

Ovaj primjer koristi istu formu i isti modul klase Employee kao u prethodnim primjerima. Ubacite novi modul klase, i postavite njegovo svojstvo Name na “Employees”. Ubacite sljedeća određivanja i kod u novi modul klase.

```
Option Explicit
Private mcolEmployees As New Collection
```

Postupci Add, Count i Delete klase Employees su u biti isti kao oni iz stare klase SmallBusiness. Možete ih jednostavno prebaciti iz modula klase SmallBusiness, ulijepiti ih u modul klase Employees, i promijeniti njihova imena.

Imena se mogu promijeniti jer više nije potrebno razlikovati EmployeeAdd od, recimo, CustomerAdd. Svaka klasa zbirke koju ostvarujete ima svoj vlastiti postupak Add.

```
‘ Postupci klase zbirke Employees.
Public Function Add(ByVal Name As String, _
ByVal Salary As Double) As Employee
  Dim empNew As New Employee
  Static intEmpNum As Integer
  ‘ Upotreba izraza With ~ini vaš kod bržim i
  ‘ kraćim (.ID umjesto empNew.ID)
  With empNew
    ‘ Stvaranje jedinstvenog ID-a za novog djelatnika.
    intEmpNum = intEmpNum + 1
    .ID = “E” & Format$(intEmpNum “00000”)
    .Name = Name
    .Salary = Salary
    ‘ Dodavanje pokazivača objekta Employee zbirci,
    ‘ korištenjem svojstva ID kao ključa.
    mcolEmployees.Add empNew, .ID
  End With
  ‘ Vraćanje pokazivača na novog djelatnika.
  Set Add = empNew
End Function

Public Funtion Count() As Long
  Count = mcolEmployees.Count
End Function
```

```
Public Sub Delete(ByVal Index As Variant)
  mcolEmployees.Remove Index
End Sub
```

Postupak Employees objekta SmallBusiness postaje postupak Item klase zbirke. On i dalje delegira objektu Collection, kako bi dohvatio elemente indeksom ili ključem.

```
‘ Postupak klase zbirke Employees
Public Function Item(ByVal Index As Variant)
As Employee
  Set Item = mcolEmployees.Item(Index)
End Function
```

Ovdje postoji zgodan potez kojeg možete dodati. Ako postupak Item napravite podrazumijevanim svojstvom klase Employees, dobivate mogućnost kodiranja poput Employees(“E00001”), kao što to možete s objektom Collection.

Kako napraviti Item podrazumijevanim svojstvom

1. U izborniku **Tools**, kliknite **Procedure Attributes** za otvaranje dijaloškog okvira **Procedure Attributes**. U okviru **Name** odaberite postupak Item.
2. Kliknite **Advanced** za prikaz naprednih osobina. U okviru **Procedure ID** odaberite (**Default**) kako bi učinili svojstvo Item podrazumijevanim. Kliknite **OK**.

   Napomena Klasa može imati samo jedan podrazumijevani element (svojstvo ili postupak).

Osposobljavanje za For Each...Next

Zajedno s snagom, natrag dobivate i petlju For Each...Next. Ponovno možete delegirati sav posao objektu Collection, dodavanjem sljedećeg postupka:

```
‘ NewEnum mora vratiti sučelje IUnknown
‘ za brojitelja zbirke.
Public Function NewEnum() As IUnknown
  Set NewEnum = mcolEmployees.[_NewEnum]
End Function
```

Važna stvar koju delegirate objektu Collection je *brojitelj* (*enumerator*). Brojitelj je mali objekt koji zna kako proći kroz stavke u zbirci. Ne možete napisati objekt brojitelja s Visual Basicom, ali budući da je klasa Employees temeljena na objektu Collection, možete vratiti brojitelj objekta Collection – koji prirodno dovoljno zna kako nabrajati stavke koje sadrži objekt Collection.

Uglate zagrade oko postupka _NewEnum objekta Collection potrebne su zbog početne podvlake u imenu postupka. Početna podvlaka je dogovor koji pokazuje da je postupak sakriven u tipskoj biblioteci. Ne možete svom postupku dati ime _NewEnum, ali ga možete sakriti u tipskoj biblioteci i dati mu identifikacijski broj potprograma kojeg zahtijeva petlja For Each...Next.

Kako sakriti postupak NewEnum i dati mu potreban ID potprograma

1. U izborniku **Tools** kliknite **Procedure Attributes** za otvaranje dijaloškog okvira **Procedure Attributes**. U okviru **Name** odaberite postupak NewEnum.
2. Kliknite **Advanced** za prikaz naprednih osobina. Potvrdite **Hide this member** kako bi postupak NewEnum učinili skrivenim u tipskoj biblioteci.
3. U okviru **Procedure ID**, upišite **-4** (minus četiri) kako bi postupku NewEnum dali ID potprograma tražen od petlje For Each...Next. Kliknite **OK**.

   Važno Kako bi vaše klase zbirki mogle raditi s petljama tipa For Each...Next, skrivenom postupku NewEnum morate osigurati ispravan ID potprograma.

Od klase SmallBusiness nije puno ostalo

Klasa SmallBusiness će sada sadržavati znatno manje koda. Kako bi zamijenili objekt Collection i sve postupke koje ste maknuli, postoji novo određivanje i svojstvo samo za čitanje:

```
Option Explicit
Private mEmployees As New Employees

Public Property Get Employees() As Employees
  Set Employees = mEmployees
End Property
```

Ovo završeđuje par riječi objašnjenja. Pretpostavimo na trenutak da ste izostavili Property Get, i jednostavno odredili Public Employees As New Employees.

Sve bi lijepo radilo dok netko ne bi napravio neku popogrešku, ali što ako pri programiranju nehotično napišete Set sbMain.Employees = Nothing? Točno, zbirka Employees bila bi uništena. Stvaranjem svojstva Employees kao svojstva samo za čitanje, sprječavate takvu mogućnost.

Promjene u formi

Programski kod za modul forme je vrlo sličan prethodnom primjeru. Možete upotrijebiti ista određivanja na razini modula, a i događaj Click gumba Close je isti.

Jedina promjena u većini potprograma događaja je zamjena starih postupaka klase SmallBusiness novim postupcima objekta zbirke Employees:

```
Private Sub cmdAddEmployee_Click()
  sbMain.Employees.Add txtName.Text, txtSalary.Text
  txtName.Text = ""
  txtSalary.Text = ""
  cmdListEmployees.Value = True
End Sub
```

```
Private Sub cmdDeleteEmployee_Click()
  ‘ Provjera je li odabran djelatnik.
  If lstEmployees.ListIndex      > -1 Then
    ‘ Prvih šest karaktera su ID.
    sbMain.Employees.Delete Left(lstEmployees.Text, 6)
  End If
  cmdListEmployees.Value = True
End Sub

Private Sub cmdListEmployees_Click()
  Dim emp As Employee
  lstEmployees.Clear
  For Each emp In sbMain.Employees
    lstEmployees.AddItem emp.ID & “, “ & emp.Name _
    & “, “ & emp.Salary
  Next
End Sub
```

Uočite da ponovno možete upotrijebiti petlju For Each...Next za ispis djelatnika.

Pokrenite projekt i provjerite radi li sve. Ovaj put nema programskog koda za gumb Trouble, budući da je sžimanje otjeralo probleme.

Za više informacija Pročitajte odlomke “Objekt zbirke u Visual Basicu” i “Zbirke u Visual Basicu”, u stalnoj pomoći, za pozadinu priče o zbirkama.

Uslužni dodatak Class Builder uključen u Professional i Enterprise verzije će stvoriti klase zbirki za vas.

Prednosti dobrog objektno orijentiranog oblikovanja

Stvaranje klase zbirke Employees kao rezultat daje vrlo jasan, modularan stil programskog koda. Cijeli kod za zbirku je u klasi zbirke (sžimanje), smanjujući veličinu modula klase SmallBusiness. Ako se zbirke objekata Employee pojavljuju na više mjesta u hijerarhiji vašeg objekta, ponovno korištenje klase zbirke ne zahtijeva dupliciranje koda.

Poboljšavanje klasa zbirki

Možete ostvariti dodatne postupke i svojstva za vaše klase zbirki. Na primjer, mogli bi ostvariti postupke Copy i Move, ili svojstvo Parent samo za čitanje koje sadrži pokazivač na objekt SmallBusiness.

Mogli bi također dodati događaj. Na primjer, svaki put kad postupak Add ili Remove mijenja broj stavki u vašoj zbirci, mogli bi izazvati događaj CountChanged.

Snaga, snaga, snaga

Ne morate uvijek ostvarivati zbirke na najsnažniji mogući način. Međutim, jedno od prednosti programiranja objektima je ponovna upotreba koda; puno je lakše ponovno upotrijebiti objekte nego kopirati strojni kod, i puno je sigurnije upotrebljavati snažan, sžiman kod.

Neki mudrac je jednom rekao "Ako želiš napisati stvarno snažan kod, trebaš pretpostaviti da će se dogoditi stvarno loše stvari".

Klase zbirki i sastavni dijelovi

Ako koristite Professional ili Enterprise verziju Visual Basica, možete pretvoriti svoj projekt u ActiveX sastavni dio, tako da ostali programeri u vašoj organizaciji mogu koristiti objekte koje ste stvorili.

Koraci k ostvarivanju klase zbirke

Sljedeća lista sžima korake potrebne za stvaranje klase zbirke.

1. Dodajte modul klase svom projektu, i dajte mu ime – obično množinu imena objekta kojeg će klasa zbirke sadržavati (pogledajte "Imenovanje svojstava, postupaka i događaja", ranije u ovom poglavlju).
2. Dodajte privatnu varijablu koja će sadržavati pokazivač na objekt tipa Collection kojem će delegirati vaši postupci i svojstva.
3. U potprogramu događaja Class_Initialize, stvorite objekt Collection (ako želite odgoditi stvaranje tog objekta sve dok nije potreban, možete odrediti privatnu varijablu u 2. koraku s As New Collection. To će dodati malu količinu nadgradnje svaki put kad se pristupi objektu Collection).
4. Dodajte svojstvo Count i postupke Add, Item i Remove svom modulu klase; u svakom slučaju, delegirajte privatnom objektu Collection pozivom njegovog pripadajućeg elementa.
5. Kad ostvarite postupak Add, možete nadjačati ponašanje nekritičnog postupka Add objekta Collection prihvaćanjem objekata samo jednog tipa. Možete čak učiniti nemogućim dodavanje vanjski stvorenih objekata svojoj zbirci, tako da vaš postupak Add u potpunosti nadzire stvaranje i *pokretanje* objekata.
6. Upotrijebite dijaloški okvir **Procedure Attributes** kako bi postupak Item napravili podrazumijevanim za vašu klasu zbirke.
7. Dodajte postupak NewEnum, kao što je prije pokazano. Upotrijebite dijaloški okvir **Procedure Attributes** kako bi ga označili kao skrivenog, te kako bi postupku dali identifikacijski broj potprograma (Procedure ID) od -4 tako da će moći raditi s petljama For Each...Next.

```
Public Function NewEnum() As IUnknown
  Set NewEnum = mcol.[_NewEnum]
End Function
```

Napomena Gornji kod podrazumijeva da privatna varijabla iz 2. koraka ima ime mcol.

8. Dodajte korisnička svojstva, postupke i događaje klasi zbirke.

Napomena Uslužni dodatak Class Builder, uključen u Professional i Enterprise verzije Visual Basica, će stvoriti klase zbirki za vas. Tako dobijeni strojni kod možete prilagođivati.

Za više informacija Više o softverskim sastavnim dijelovima možete pročitati u 2. dijelu "Stvaranje ActiveX sastavnih dijelova", vodiča *Microsoft Visual Basic 6.0 Component Tools Guide*.

ActiveX kreatori

Kreator (*designer*) pruža prozor za vidljivo oblikovanje u razvojnom okruženju Visual Basica. Možete upotrijebiti taj prozor za oblikovanje vidljivog dijela novih klasa. Visual Basic ima ugrađene kreatore za forme i – u Professional i Enterprise verzijama – ActiveX kontrole i dokumente.

Objekti stvoreni iz klasa koje oblikujete na ovaj način imaju različito ponašanje i izgled tijekom izrade i tijekom izvođenja aplikacije, iako većina objekata – kao što su forme i kontrole – izgledaju vrlo slično u ova dva načina.

Kao dodatak ugrađenim kreatorima, Visual Basic omogućuje trećoj strani razvijanje kreatora za upotrebu u razvojnom okruženju Visual Basica. Takvi *ActiveX kreatori* rade jednako kao i kreatori ugrađeni u Visual Basic, što ih čini lakim za učenje i korištenje.

Što su ActiveX kreatori?

ActiveX kreatori mogu pružiti vidljiva sučelja za zadatke koji bi inače mogli zahtijevati vrlo mnogo koda. Na primjer, kreator UserConnection uključen u Enterprise verziju Visual Basica pruža vidljive alate za određivanje složenih upita bazi podataka. Tijekom izvođenja aplikacije, ti upiti mogu biti pozvani s vrlo malo koda.

Sličnosti između ActiveX kreatora i ugrađenih kreatora

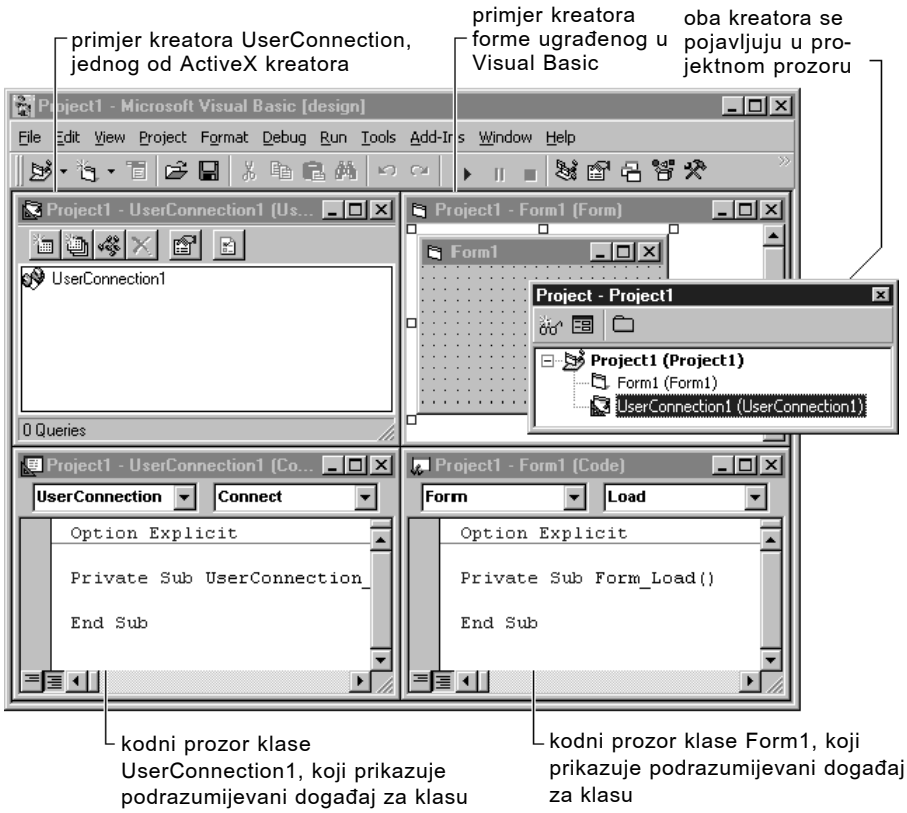
ActiveX kreatori slični su kreatorima forme na sljedeće načine:

- ActiveX kreatori proizvode klase iz kojih možete stvarati objekte. Te klase se pojavljuju u projektnom prozoru, baš kao i klase forme.
- Svaka klasa koju stvorite s ActiveX kreatorom ima svoj vlastiti modul koda, u kojem možete pisati kod za potprograme događaja koje je proizveo kreator.
- ∨Možete prilagoditi klasu, dodavanjem vaših vlastitih svojstava, postupaka i događaja onima koje je proizveo ActiveX kreator.
- Objekti stvoreni iz klasa koje oblikujete imaju različite karakteristike tijekom izrade i tijekom rada aplikacije.

- Prozor za oblikovanje u ActiveX kreatoru je potpuno uklopljen u razvojnu okolinu. Može mu se mijenjati veličina i raspored kao i kod prozora ugrađenih kreatora.
- Možete dodati potreban broj primjera ActiveX kreatora vašem projektu, baš kao što možete dodati koliko želite kreatora forme.

Slika 9.14 uspoređuje kreator forme ugrađen u Visual Basic s kreatorom UserConnection, ActiveX kreatorom koji je uključen u Enterprise verziju Visual Basica.

Slika 9.14 ActiveX kreator i kreator ugrađen u Visual Basic



Usporedba klasa ActiveX kreatora i ostalih vizualno oblikovanih klasa

ActiveX kreatori su iznimno fleksibilni. Neki, kao kreator UserConnection, stvaraju klase čiji su primjeri tijekom izvođenja programabilni, ali nisu vidljivi. Drugi, kao kreator Microsoft Forms kojeg koristi Microsoft Office, proizvode vidljive objekte slične formama Visual Basica.

ActiveX kreatori koji imaju vidljive dijelove tijekom izvođenja aplikacije mogu biti sposobni primiti ActiveX kontrole. Po učinku, oni postaju zamjenski paketi formi, koji mogu biti upotrijebljeni kao dodatak prirodnim formama Visual Basica.

Sljedeća lista uspoređuje klase proizvedene ActiveX kreatorima s onima proizvedenima s kreatorima ugrađenim u Visual Basic.

- Ako je objekt stvoren iz klase ActiveX kreatora vidljiv tijekom izvođenja, ima svoj vlastiti prozor. Nije sadržan unutar druge forme, kao što jesu ActiveX kontrole.
- Poput klasa forme, ali za razliku od ActiveX kontrola, klase proizvedene ActiveX kreatorima su privatne klase. Ako upotrebljavate Professional ili Enterprise verzije Visual Basica za stvaranje ActiveX sastavnih dijelova, ne možete odrediti javne postupke koji koriste te klase kao tipove argumenta ili povratne tipove.

Na primjer, sljedeća određivanja postupaka proizvode pogreške kod prevođenja ako se pojave u javnoj klasi:

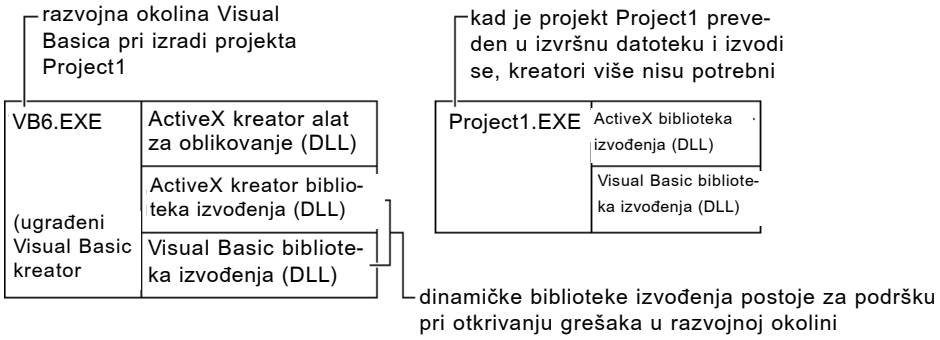
```
Public Function A() As UseConnection1          ' Pogreška
Public Sub B(CallBack As UseConnection1)       ' Pogreška
```

Oprez Iako je moguće proslijediti pokazivače na privatne objekte izvan vašeg projekta, određivanjem povratnih vrijednosti s As Object, to je vrlo loša praksa, i može narušiti ravnotežu vaše aplikacije. Za više informacija, pogledajte 2. dio "Stvaranje ActiveX sastavnih dijelova", u vodiču *Microsoft Visual Basic 6.0 Component Tools Guide*.

Korištenje objekata ActiveX kreatora tijekom izvođenja

Poput ugrađenog kreatora forme, ActiveX kreatori su dostupni samo u razvojnom okruženju. Jednom kad vaš projekt načinite izvršnim, on koristi samo biblioteku ActiveX kreatora (.dll) za izvođenje. Ona može biti puno manja od biblioteke za oblikovanje, pošto ne uključuje alate za vizualno oblikovanje. Slika 9.15 prikazuje takav koncept.

Slika 9.15 Sastavni dijelovi kreatora u memoriji



Kao što je ranije napomenuto, ActiveX kreatori mogu proizvesti klase čiji objekti nisu vidljivi tijekom izvođenja. Kreator UserConnection prikazan na slici 9.14 je primjer. Kreator UserConnection proizvodi klase čiji objekti upravljaju vezama s SQL bazama podataka tijekom izvođenja aplikacije. Nema razloga zašto bi ti objekti bili vidljivi tijekom izvođenja.

Kako bi upotrijebili klasu stvorenu kreatorom UserConnection, odredite varijablu tipa klase i stvorite primjer klase. Na primjer, ako ste dodali kreator UserConnection i postavili njegovo svojstvo Name na OpćiSaldokonto, možete stvoriti objekt OpćiSaldokonto kao što je prikazano u sljedećem dijelu programskog koda:

```
‘ Opća varijabla u standardnom modulu, koja
‘ čuva pokazivač na objekt OpćiSaldokonto.
Public gOpćiSaldokonto As OpćiSaldokonto

‘ Kod u modulu Form za stvaranje objekta OpćiSaldokonto
‘ i uspostavljanje veze s bazom podataka.
Private Sub Command1_Click()
Set gOpćiSaldokonto = New OpćiSaldokonto
gOpćiSaldokonto.UspostavljanjeVeze
‘ (Kod koji koristi objekt.)
End Sub
```

Stvaranje ActiveX kreatora

Možete upotrijebiti aplikaciju ActiveX Designer Software Development Kit (SDK) kako bi stvorili nove ActiveX kreatore za korištenje s Visual Basicom. Aplikacija SDK uključuje pune upute i primjere koda. Možete je pronaći na mreži Microsoft Development Network (MSDN) ispod naslova “SDK dokumentacija”.

Napomena Aplikacija ActiveX Designer SDK zahtijeva C++ prevoditelja, kao što je Microsoft Visual C++. ActiveX kreatori ne mogu biti napisani korištenjem Visual Basica.

Za više informacija Potprogrami za pripajanje ActiveX kreatora vašem projektu su dani u sljedeća dva odlomka “Dodavanje ActiveX kreatora izborniku Project” i “Ubacivanje novog primjera ActiveX kreatora”.

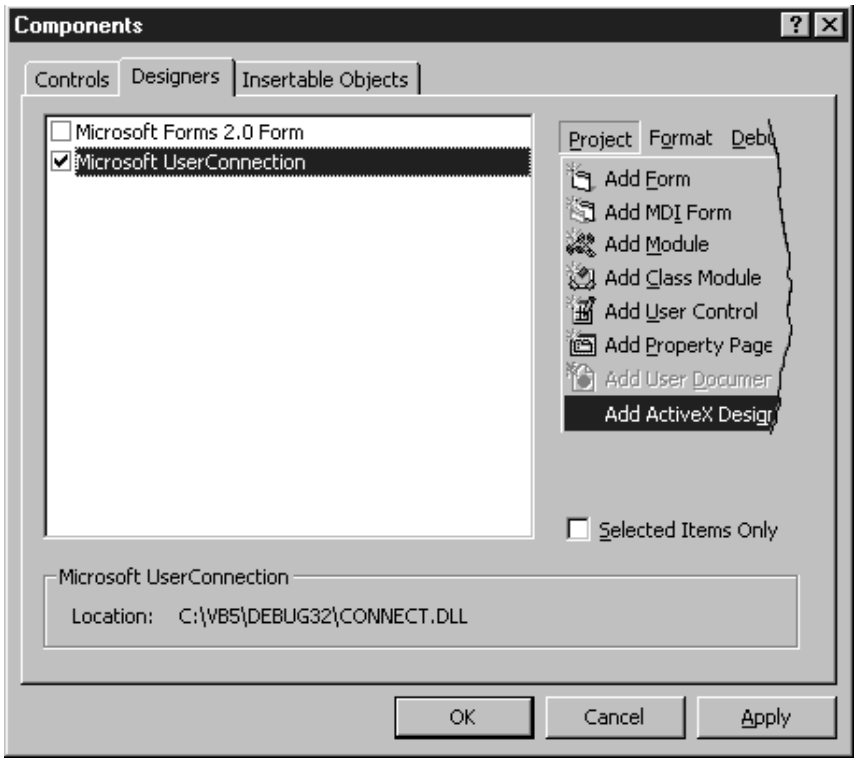
Dodavanje ActiveX kreatora izborniku Project

Nakon što ugradite novi ActiveX kreator, korištenjem aplikacije Setup koju osigurava prodavač, morate napraviti kreatora dostupnim vašim projektima tako da ga dodate u izbornik Project.

Ugrađivanje ActiveX kreatora će biti zabilježeno u registrima Windowsa, ispod odgovarajuće kategorije sastavnog dijela. Nakon toga će kreator biti dostupan s kartice Designers dijaloškog okvira Components.

Kako dodati ActiveX kreator izborniku Project

1. U izborniku **Project**, kliknite **Components** za otvaranje dijaloškog okvira **Components**.
2. Kliknite karticu **Designers** i odaberite kreatora kojeg želite upotrijebiti, kao što je prikazano na sljedećoj slici, i kliknite **OK**.



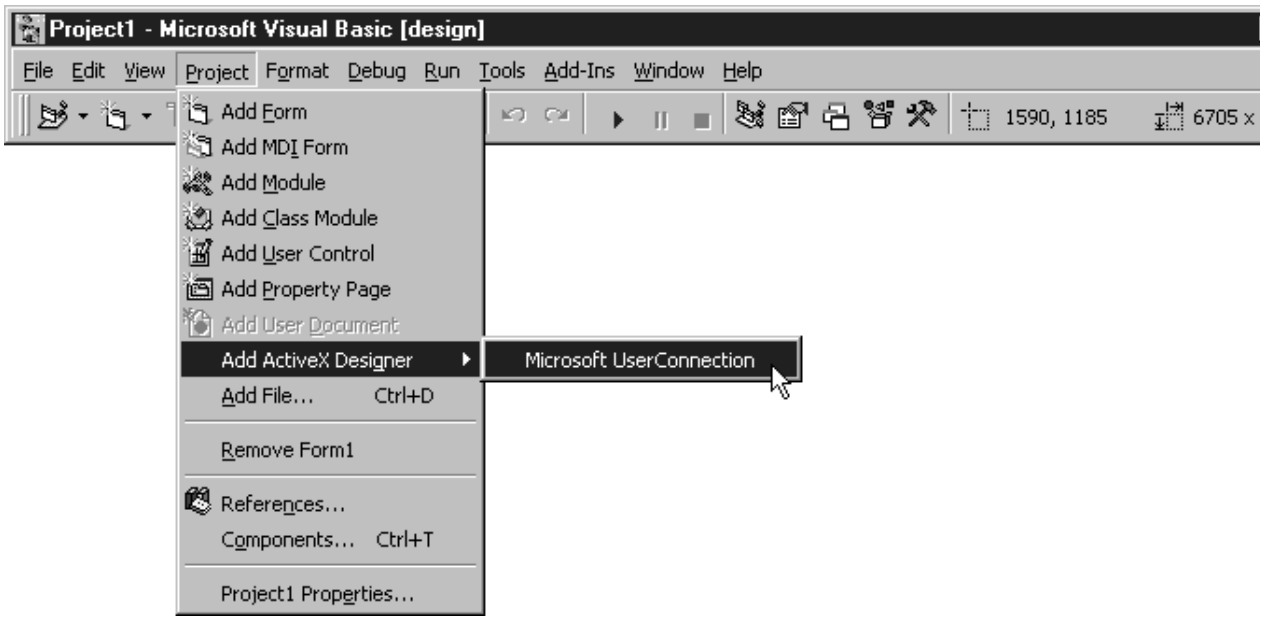
Napomena Kreator Microsoft Forms je uključen u sve verzije Visual Basica, kako bi omogućio lagano prenošenje formi stvorenih u aplikacijama Microsoft Officea. Poput svih kreatora, kreator Microsoft Forms ima svoju vlastitu .dll biblioteku za izvođenje. Upotreba ovog kreatora u projektu Visual Basica će zbog toga povećati zahtjeve za memorijom rezultirajuće izvršne datoteke.

Ubacivanje novog primjera ActiveX kreatora

Kad ste jednom upotrijebili dijaloški okvir Components za dodavanje kreatora u izbornik Project, kao što je pokazano u odlomku "Dodavanje ActiveX kreatora izborniku Project", možete umetnuti potreban broj primjera kreatora.

Kako umetnuti primjer OLE kreatora

- U izborniku **Project** kliknite **Add ActiveX Designer** za prikaz liste ugrađenih kreatora. Odaberite s liste kreatora kojeg želite, kao što je prikazano na sljedećoj slici:



Kad ste jednom dodali primjer ActiveX kreatora svom projektu, možete upotrijebiti njegovo vidljivo sučelje za oblikovanje klase. Možete kliknuti gumb Code u projektnom prozoru za otvaranje modula koda tog kreatora, i dodati kod potprogramima događaja. Možete osim toga prilagoditi klasu dodavanjem svojih vlastitih svojstava, postupaka i događaja, kao što to možete s modulom klase Visual Basica.

Za više informacija Kreatori su predstavljeni u odlomku "ActiveX kreatori", ranije u ovom poglavlju. Kako bi dodali kreatora izborniku Project, pogledajte "Dodavanje ActiveX kreatora izborniku Project", ranije u ovom poglavlju.

Programiranje sastavnim dijelovima

Trebate li ponekad u vašoj Visual Basic aplikaciji pružiti istu raščlambu i sposobnosti proračunavanja kakve pruža Microsoft Excel? Možda bi voljeli oblikovati dokumente korištenjem alata za oblikovanje Microsoft Worda, ili spremati podatke i upravljati njima korištenjem Microsoft Jet mehanizma za baze podataka. Još bolje, biste li voljeli biti u mogućnosti stvarati ili kupovati standardne sastavne dijelove, pa ih upotrebljavati u mnogim aplikacijama bez potrebe da ih mijenjate?

Sve to i više može se ostvariti gradnjom vaših aplikacija korištenjem ActiveX sastavnih dijelova. *Sastavni dio tipa ActiveX* (*ActiveX component*) je dio programskog koda i podataka koji se može koristiti više puta i sčinjen je od jednog ili više objekata stvorenih korištenjem ActiveX tehnologije. Vaše aplikacije mogu upotrebljavati postojeće sastavne dijelove, kao što su oni koji su uključeni u aplikacije Microsoft Office-a, sastavne dijelove koda, ActiveX dokumente, ili ActiveX kontrole (prije zvane OLE kontrole) koje pružaju razni prodavači. Ili, ako imate Visual Basic, verzije Professional ili Enterprise, možete stvoriti svoje vlastite ActiveX kontrole.

Za sastavne dijelove koji podržavaju povezivanje i umetanje objekata, možete ubaciti objekte u vaše aplikacije bez pisanja ikakvog koda korištenjem vidljivog sučelja sastavnog dijela. U vašu aplikaciju možete ubaciti objekt koji podržava OLE korištenjem kontrole OLE spremnika ili dodavanjem klase objekta u alatni okvir.

Kako bi potpuno shvatili ActiveX sastavne dijelove, prvo bi se trebali upoznati s načinom rada s klasama, objektima, svojstvima i postupcima, što je objašnjeno u 9. poglavlju “Programiranje objektima”.

Sadržaj

- Tipovi ActiveX sastavnih dijelova.
- Poslužitelji vrste u-procesu i izvan-procesa.
- Rad s ActiveX sastavnim dijelovima.
- Stvaranje pokazivača na objekt.
- Korištenje svojstava, postupaka i događaja objekta.
- Otpuštanje ActiveX sastavnog dijela.
- Kretanje kroz modele objekata.

- Rukovanje pogreškama tijekom rada u ActiveX sastavnim dijelovima.
- Rukovanje zahtjevima u tijeku za ActiveX sastavni dio.
- Korištenje vidljivog sučelja sastavnih dijelova.

Sučelja: Geofact.vbp i Olecont.vbp

Većina pojmova u ovom poglavlju prikazana je primjerima aplikacija Geofact.vbp i Olecont.vbp. Primjeri aplikacija nalaze se u direktoriju Samples.

Tipovi ActiveX sastavnih dijelova

Sastavni dijelovi tipa ActiveX daju vam snagu za sastavljanje usavršenih aplikacija iz dijelova koji već postoje. Vaše Visual Basic aplikacije mogu uključivati nekoliko tipova ActiveX sastavnih dijelova:

- Aplikacije koje podržavaju ActiveX tehnologiju, kao Microsoft Excel, Microsoft Word i Microsoft Access, pribavljaju objekte kojima možete programski upravljati iz svoje Visual Basic aplikacije. Na primjer, u vašoj aplikaciji možete upotrijebiti svojstva, postupke i događaje proračunske tablice Microsoft Excela, dokumenta Microsoft Worda ili baze podataka Microsoft Accessa.
- Sastavni dijelovi koda pružaju biblioteke programabilnih objekata. Na primjer, sastavni dio koda može uključivati biblioteku specijaliziranih financijskih funkcija za korisnike proračunske tablice, ili elemente korisničkog sučelja, kao dijaloški okviri, koji su zajednički mnogim aplikacijama. Za razliku od objekta u aplikaciji koja omogućuje ActiveX tehnologiju, objekt u sastavnom dijelu koda može raditi u istom procesu kao vaša aplikacija, omogućujući brži pristup objektu.
- Možete dodati osobine bez potrebe da ih sami stvarate, korištenjem ActiveX kontrola kao sastavnih dijelova. ActiveX kontrole su dostupne od velikog broja proizvođača i pružaju puno specijaliziranih osobina, kao prikazivanje kalendara na formi ili čitanje podataka posebnog oblika.
- ActiveX dokumenti vam omogućuju stvaranje interaktivnih Internet aplikacija. Možete stvoriti forme koje će biti sadržane unutar Internet Explorera. ActiveX dokumenti mogu prikazivati okvire s porukama i podređene forme te sadržavati ActiveX kontrole. ActiveX dokumenti mogu također djelovati kao sastavni dijelovi koda. Za uvod korak po korak u ActiveX dokumente, pogledajte 2. dio "Stvaranje ActiveX dokumenta" u vodiču *Microsoft Visual Basic 6.0 Component Tools Guide* biblioteke *Microsoft Visual Basic 6.0 Reference Library*, dostupne u Professional i Enterprise verzijama.

  Napomena Kao dodatak Internet aplikacijama koje koriste ActiveX dokumente, možete stvoriti klijentske i poslužiteljske Internet aplikacije korištenjem spajanja koda Visual Basica i HTML stranica. Pogledajte "Uvod u Internet aplikacije" u 5. dijelu "Gradnja Internet aplikacija" u vodiču *Microsoft Visual Basic 6.0 Component Tools Guide* za više informacija o vašim mogućnostima kod stvaranja Internet i intranet aplikacija.

Neki ActiveX sastavni dijelovi izvode se u istom procesu kao i vaša aplikacija, dok se drugi izvode u odvojenom procesu. Za više informacija, pogledajte "Poslužitelji vrste u-procesu i izvan-procesa", u stalnoj pomoći.

Kao dodatak sastavnim dijelovima u postojećim aplikacijama koje omogućuju ActiveX tehnologiju, bibliotekama sastavnih dijelova koda, ActiveX kontrolama i ActiveX dokumentima možete stvoriti svoje vlastite sastavne dijelove. Za više informacija o stvaranju vaših vlastitih ActiveX sastavnih dijelova, pogledajte "Stvaranje ActiveX sastavnih dijelova" u vodiču *Microsoft Visual Basic 6.0 Component Tools Guide* biblioteke *Microsoft Visual Basic 6.0 Reference Library*, dostupne u Professional i Enterprise verzijama.

Poslužitelji vrste u-procesu i izvan-procesa

ActiveX sastavni dijelovi surađuju s vašom aplikacijom – i međusobno – kroz odnos *klijent/poslužitelj*. *Klijent* je programski kod aplikacije ili sastavni dio koji koristi osobine sastavnog dijela. *Poslužitelj* je sastavni dio i njegovi pridruženi objekti. Na primjer, pretpostavimo da vaša aplikacija koristi ActiveX kontrolu za pružanje standardne forme Djelatnik za više aplikacija u vašoj tvrtki. ActiveX kontrola koja pruža formu Djelatnik je poslužitelj; aplikacije koje koriste kontrolu su njezini klijenti.

Ovisno o tome kako je ActiveX sastavni dio ostvaren, on se može izvoditi u istom procesu kao i njegova klijentska aplikacija, ili u drugom procesu. Na primjer, ako vaša aplikacija koristi sastavni dio koji je dio aplikacije koja podržava ActiveX tehnologiju, izvodi se u odvojenom procesu. Ako je sastavni dio bio ostvaren kao programabilan objekt u dinamički povezivoj biblioteci (datoteka .dll), izvodit će se u istom procesu kao vaša aplikacija.

Općenito, ako je ActiveX sastavni dio bio ostvaren kao dio izvršne datoteke (datoteka .exe), on je poslužitelj vrste *izvan-procesa* i izvodi se u svom vlastitom procesu. Ako je bio ostvaren kao dinamički poveziva datoteka, on je poslužitelj *u-procesu* i izvodi se u istom procesu kao i klijentska aplikacija. Aplikacije koje koriste poslužitelje vrste u-procesu obično se brže izvode nego one koje koriste poslužitelje vrste izvan-procesa budući da aplikacije ne trebaju prelaziti granice procesa kako bi upotrijebile svojstva, postupke i događaje objekta.

Sljedeća tablica prikazuje kako možete ostvariti različite tipove sastavnih dijelova:

sastavni dio	vrsta poslužitelja
Aplikacija koja podržava ActiveX	izvan-procesa
Sastavni dio koda	ili u-procesu ili izvan-procesa
ActiveX kontrola	u-procesu
ActiveX dokument	ili u-procesu ili izvan-procesa

Upotreba sastavnih dijelova vrste u-procesu je jedan način poboljšavanja izvođenja vaše aplikacije. Drugi način poboljšavanja izvođenja je korištenje ranog povezivanja. Za više informacija, pogledajte "Ubrzavanje pokazivača objekata" kasnije u ovom poglavlju.

Rad s ActiveX sastavnim dijelovima

S objektima koje pružaju ActiveX sastavni dijelovi radite na uglavnom isti način na koji radite i s ostalim objektima. Pokazivač objekta dodjeljujete varijabli, te zatim pišete kod koji koristi postupke, svojstva i događaje objekta. Međutim, postoje neke stvari kojih trebate biti svjesni kad radite s objektima koje pružaju sastavni dijelovi.

Ova tema pruža pregled najvažnijih poslova za rad s objektima koje pružaju sastavni dijelovi i primjer je korištenja objekata u aplikaciji koja podržava ActiveX tehnologiju. Za detalje o svakom poslu, pogledajte pripadajuću temu opisanu ispod svake stavke posla.

Kako upotrijebiti većinu objekata koje pružaju ActiveX sastavni dijelovi

1. Stvorite pokazivač na objekt kojeg želite upotrijebiti. Kako ćete to napraviti ovisi o tipu objekta te isporučuje li ActiveX sastavni dio tipsku biblioteku.

   Za više informacija Pogledajte "Stvaranje pokazivača na objekt", kasnije u ovom poglavlju.

2. Napišite programski kod korištenjem postupaka, svojstava i događaja objekta.

   Za više informacija Pogledajte "Korištenje svojstava, postupaka i događaja objekta", kasnije u ovom poglavlju.

3. Otpustite objekt kad ste gotovi s njegovim korištenjem.

   Za više informacija Pogledajte "Otpuštanje ActiveX sastavnog dijela", kasnije u ovom poglavlju.

4. Stvorite rukovatelje pogreškama.

   Za više informacija Pogledajte "Rukovanje pogreškama tijekom rada u ActiveX sastavnim dijelovima", kasnije u ovom poglavlju.

Na primjer, pretpostavimo da ste stvorili formu s tri okvira s tekstom (Text1, Text2 i Text3) i naredbenim gumbom (Command1), te u svom projektu dodali pokazivač na biblioteku Microsoft Excel 8.0 Object Library. Nakon toga možete dodati kod potprogramu događaja Command1_Click naredbenog gumba koji koristi postupak Microsoft Excel Formula za zbrajanje dva broja upisana u okvirima Text1 i Text2, prikazujući rezultat u okviru Text3. Kako bi izbjegli pogrešku pogrešnog tipa podatka, možete maknuti podrazumijevani tekst u svakom okviru s tekstom tako da njihova svojstva Text postavite na prazan string:

```
Private Sub Command1_Click()
  ‘ Određivanje varijabli objekta za Microsoft Excel,
  ‘ radnu knjigu te aplikacije i objekte radnog lista.
  Dim xlApp As Excel.Applicaton
  Dim xlBook As Excel.Workbook
  Dim xlSheet As Excel.Worksheet

  ‘ Dodjela pokazivača objekta varijablama.
  ‘ Upotrijebite postupke Add za stvaranje
```

```
  ‘ novih objekata radne knjige i radnog lista.
  Set xlApp = New Excel.Application
  Set xlBook = xlApp.Workbooks.Add
  Set xlSheet = xlBook.Worksheets.Add

  ‘ Dodjela vrijednosti unesenih u okvire
  ‘ s tekstom ćelijama Microsoft Excela.
  xlSheet.Cells(1, 1).Value = Text1.Text
  xlSheet.Cells(2, 1).Value = Text2.Text

  ‘ Upotreba postupka Formula za zbrajanje
  ‘ vrijednosti u Microsoft Excelu.
  xlSheet.Cells(3, 1).Formula = “=R1C1 + R2C1”
  Text3.Text = xlSheet.Ceels(3, 1)

  ‘ Snimanje radnog lista.
  xlSheet.SaveAs “c:\Temp.xls”

  ‘ Zatvaranje radne knjige.
  xlBook.Close

  ‘ Zatvaranje Microsoft Excela postupkom Quit.
  xlApp.Quit

  ‘ Otpuštanje objekata.
  Set xlApp = Nothing
  Set xlBook = Nothing
  Set xlSheet = Nothing
End Sub
```

Zbog jednostavnosti, ovaj primjer ne uključuje rukovanje pogreškama. Međutim, vrlo je preporučljivo da uključite rukovanje pogreškama u aplikacije koje koriste objekte pružene od ActiveX sastavnih dijelova.

Stvaranje pokazivača na objekt

Prije nego što u svojoj aplikaciji možete koristiti svojstva, postupke i događaje objekta, najprije morate odrediti varijablu objekta, te zatim varijabli dodijeliti pokazivač objekta. Kako ćete dodijeliti pokazivač objekta ovisi o dva čimbenika:

- Isporučuje li ActiveX sastavni dio tipsku biblioteku. Tipska biblioteka ActiveX sastavnog dijela sadrži definicije svih objekata koje pruža sastavni dio, uključujući definicije svih dostupnih postupaka, svojstava i događaja. Ako ActiveX sastavni dio pruža tipsku biblioteku, u svoj projekt Visual Basica trebate dodati pokazivač na tipsku biblioteku prije nego što možete upotrebljavati objekte biblioteke.
- vJe li riječ o objektu najviše razine, *objektu stvorenom izvana* (*externally creatable object*) ili *ovisnom objektu* (*dependent object*). Pokazivač vanjski stvorenom objektu možete dodijeliti izravno, dok se pokazivači na ovisne objekte dodjeljuju posredno.

Ako je objekt stvoren izvana, pokazivač objekta možete dodijeliti varijabli korištenjem ključne riječi New, te funkcija CreateObject ili GetObject u izrazu Set izvan sastavnog

dijela. Ako je objekt ovisan objekt, pokazivač objekta dodjeljujete korištenjem postupka iz objekta više razine u izrazu Set.

U Microsoft Excelu, na primjer, objekt Application je objekt stvoren izvana – možete mu dodijeliti pokazivač izravno iz vaše Visual Basic aplikacije korištenjem ključne riječi New, te funkcija CreateObject ili GetObject u izrazu Set. Objekt Range, suprotno tome, je ovisan objekt – dodjeljujete mu pokazivač korištenjem postupka Cells objekta Worksheet u izrazu Set. Za više informacija o objektima stvorenim izvana i ovisnim objektima, pogledajte "Kretanje kroz modele objekata", kasnije u ovom poglavlju.

Ako je klasa objekta uključena u tipsku biblioteku, brže izvođenje vaše aplikacije možete postići tako da stvorite pokazivač objekta upotrebom varijable te određene klase. Inače, morate upotrijebiti varijablu opće klase Object, rezultat čega je kasno povezivanje. Za više informacija, pogledajte "Ubrzavanje pokazivača objekata", kasnije u ovom poglavlju.

Kako stvoriti pokazivač na objekt definiran u tipskoj biblioteci

1. U izborniku **Project** odaberite **References**.
2. U dijaloškom okviru **References**, odaberite ime ActiveX sastavnog dijela koji sadržava objekte koje želite upotrijebiti u svojoj aplikaciji.
3. Možete upotrijebiti gumb **Browse** za traženje datoteke tipske biblioteke koja sadrži potreban objekt. Tipske biblioteke imaju datoteke s nastavcima imena .tlb ili .olb. Izvršne datoteke (.exe) i dinamički povezive biblioteke (.dll) također mogu pružiti tipske biblioteke, pa možete potražiti i datoteke s tim nastavcima imena.

   Ako niste sigurni podržava li aplikacija ActiveX tehnologiju i pruža li tipsku biblioteku, pokušajte dodati pokazivač na nju korištenjem gumba **Browse**. Ako pokazivač ne uspije, Visual Basic će prikazati poruku pogreške "Ne mogu dodati pokazivač na označenu datoteku" (Can't add, a reference to the specified file), ukazujući da tipska biblioteka ne postoji. Za više informacija o radu s objektima koji nisu udruženi s tipskim bibliotekama, pogledajte "Stvaranje pokazivača na objekt", ranije.
4. U izborniku **View**, odaberite **Object Browser** kako bi vidjeli pokazanu tipsku biblioteku. Odaberite odgovarajuću tipsku biblioteku s liste **Project/Library**. U svojoj aplikaciji možete upotrijebiti sve objekte, postupke i svojstva ispisana u **pretraživaču objekata**.

   Za više informacija Pogledajte "Pretraživanje tipskih biblioteka ActiveX sastavnih dijelova", kasnije u ovom poglavlju.
5. Odredite varijablu objekta klase objekta. Na primjer, mogli bi odrediti varijablu klase Excel.Chart za upućivanje na objekt Microsoft Excel Chart.

   ```
   Dim xlChart As Excel.Chart
   ```

   Za više informacija Pogledajte "Određivanje varijable objekta", kasnije u ovom poglavlju.
6. Dodijelite pokazivač objekta varijabli korištenjem ključne riječi New, funkcija CreateObject ili GetObject u izrazu Set. Za više informacija, pogledajte "Dodjela pokazivača objekta varijabli", kasnije u ovom poglavlju.

Ako je objekt ovisan objekt, dodijelite pokazivač objekta korištenjem postupka objekta više razine u izrazu Set.

Kako stvoriti pokazivač na objekt koji nije definiran u tipskoj biblioteci

1. Odredite varijablu objekta tipom podatka Object.

   Budući da objekt nije povezan s tipskom bibliotekom, nećete mogući koristiti **preglednik objekata** kako bi vidjeli svojstva, postupke i događaje objekta. Trebate znati koja svojstva, postupke i događaje pruža objekt, uključujući sve postupke za stvaranje pokazivača na ovisan objekt.

   Za više informacija Pogledajte "Određivanje varijable objekta", kasnije u ovom poglavlju.

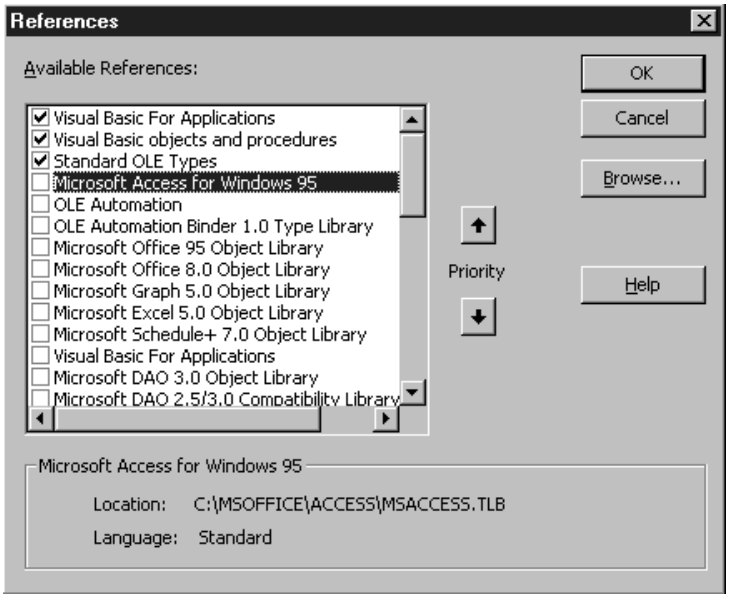
2. Dodijelite pokazivač objekta varijabli korištenjem funkcija CreateObject ili GetObject u izrazu Set. Za više informacija, pogledajte "Dodjela pokazivača objekta varijabli", kasnije u ovom poglavlju.

Ako je objekt ovisan objekt, dodijelite pokazivač objekta korištenjem postupka objekta više razine u izrazu Set.

Nepouzdani pokazivači i prednost pokazivača

Kad se u programskom kodu upućujete na konstantu ili objekt, Visual Basic traži konstantu ili klasu objekta u svakoj tipskoj biblioteci odabranoj u dijaloškom okviru References redoslijedom po kojem su tipske biblioteke prikazane. Ako dvije tipske biblioteke sadrže konstante ili klase s istim imenima, Visual Basic upotrebljava određivanje pruženo od tipske biblioteke koja je prije izlistana u okviru Available References.

Slika 10.1 Dijaloški okvir References



Najbolji način rukovanja mogućim nepouzdanim pokazivačima je izričito određivanje tipske biblioteke koja pruža konstantu ili klasu kad je upotrebljavate. Na primjer, konstanta vbCancel ima različite vrijednosti za tipske biblioteke Visual Basica i Visual Basica za aplikacije. Sljedeći programski kod prikazuje potpuno označene i nepouzdane pokazivače na konstantu vbCancel:

```
‘ Ispis konstante vbCancel Visual Basica.
Debug.Print “VB.vbCancel = “; VB.vbCancel
‘ Ispis konstante vbCancel Visual Basica za aplikacije.
Debug.Print “VBA.vbCancel = “; VBA.vbCancel
‘ Nepouzdan pokazivač ispisuje vrijednost konstante
‘ vbCancel koja se prva pojavljuje u najgornjoj
‘ tipskoj biblioteci liste Available References.
Debug.Print “vbCancel = “; vbCancel
```

Sljedeći primjer programskog koda prikazuje potpuno označena i nepouzdana određivanja za varijablu objekta Application. Ako se u okviru Available References prije pojavi Microsoft Word nego Microsoft Excel, xlApp2 će biti određena korištenjem klase Microsoft Word Application prije nego korištenjem klase Microsoft Excel Application.

```
‘ Potpuno označeno odre|ivanje varijable objekta.
Dim xlApp1 As Excel.Application
‘ Nepouzdano određivanje varijable objekta.
Dim xlApp2 As Application

‘ Dodjela pokazivača objekta.
Set xlApp1 = New Excel.Application
‘ Sljedeća linija stvara pogrešku pogrešnog tipa podatka.
Set xlApp2 = xlApp1
```

Možda vam je primamljivo rukovanje mogućim nepouzdanim pokazivačima promjenom redoslijeda po kojem Visual Basic traži pokazivače. Dijaloški okvir References uključuje dva gumba Priority koja vam omogućuju pomicanje tipske biblioteke prema gore u listi, tako da će njene konstante i klase biti pronađene prije nego konstante i klase istih imena niže u listi. Međutim, promjena redoslijeda prednosti može uzrokovati neočekivane probleme u vašim aplikacijama ako postoje drugi nepouzdani pokazivači. Općenito, bolje je izričito odrediti tipsku biblioteku u svim pokazivačima.

Napomena Sintaksa Excel.Application za ukazivanje na klasu Microsoft Excel Application nije podržana u verzijama ranijim od Microsoft Excela 97. Kako bi ukazali na klasu Microsoft Excel Application u Microsoft Excelu 5.0 i Microsoft Excelu 95, upotrijebite umjesto toga sintaksu Š_ExcelApplicationĆ. Na primjer:

```
Set xlApp = New [_ExcelAplication]
```

Pretraživanje tipskih biblioteka ActiveX sastavnih dijelova

Ako ActiveX sastavni dio pruža tipsku biblioteku, možete upotrijebiti pretraživač objekata kako bi vidjeli klase sastavnog dijela, kao i svojstva, postupke, događaje i konstante pridružene s objektima svake klase.

Kako vidjeti klase dostupne u tipskoj biblioteci ActiveX sastavnog dijela

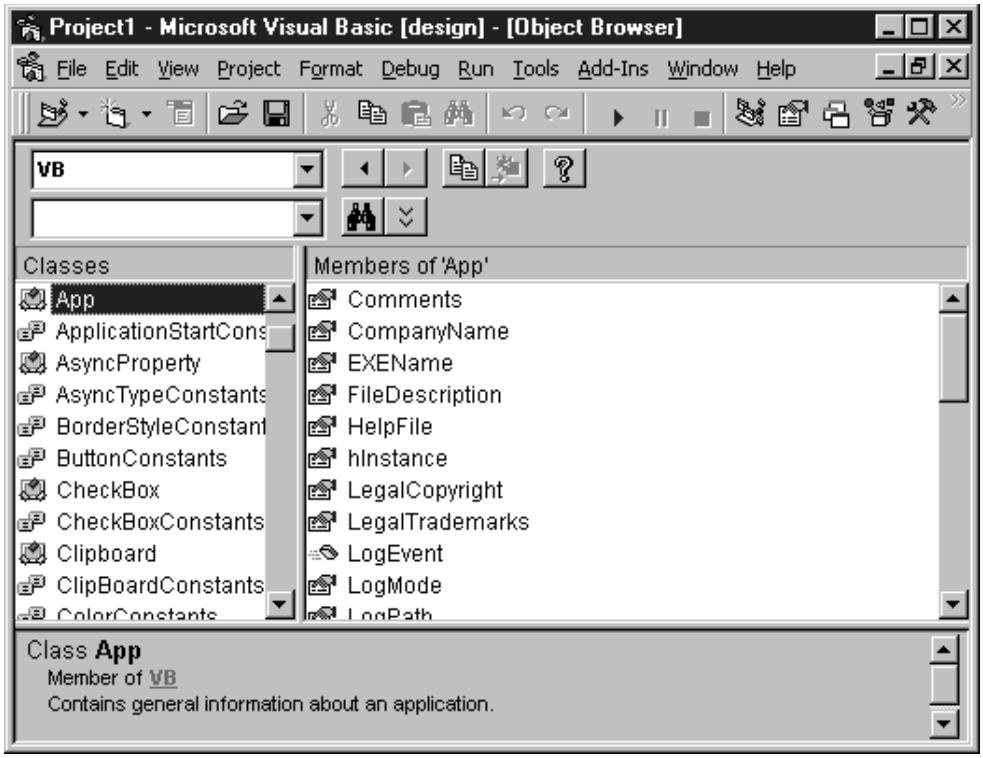
1. Ako to do sada niste napravili, dodajte svom projektu Visual Basica pokazivač na tipsku biblioteku.

   Za više informacija Pogledajte "Stvaranje pokazivača na objekt", ranije u ovom poglavlju.

2. Otvorite **pretraživač objekata** i odaberite ime tipske biblioteke iz liste **Project/Library**.

   Pretraživač objekata će prikazati dostupne klase u listi **Classes**.

Slika 10.2 Pretraživač objekata



Na primjer, kako bi vidjeli klase dostupne u tipskoj biblioteci Data Access Object (DAO), dodajte pokazivač na biblioteku u dijaloškom okviru References, te odaberite DAO u listi Project/Library pretraživača objekata.

Kako vidjeti elemente klase

- Odaberite ime klase iz liste **Classes** u **pretraživaču objekata**.

  Pretraživač objekata će prikazati elemente klase u listi **Member of**.

Ako tražite informaciju o pojedinoj klasi ili elementu u tipskoj biblioteci, upotrijebite osobinu Search pretraživača objekata.

Kako upotrijebiti osobinu Search

- Upišite što tražite u okvir **Search Text**, pa zatim kliknite gumb **Search**.

  Pretraživač objekata će prikazati okvir **Search Results** pokazujući biblioteke, klase i elemente pronađene traženjem.

Određivanje varijable objekta

Prije nego što možete koristiti svojstva, postupke i događaje objekta kojeg pruža ActiveX sastavni dio, najprije morate odrediti varijablu objekta. Način na koji odre–đujete varijablu objekta ovisi o tome isporučuje li ActiveX sastavni dio tipsku biblioteku ili ne.

Kako odrediti varijablu za objekt definiran u tipskoj biblioteci

1. Dodajte vašem Visual Basic projektu pokazivač na tipsku biblioteku. Za više informacija o dodavanju pokazivača na tipsku biblioteku, pogledajte “Stvaranje pokazivača na objekt”, ranije u ovom poglavlju.
2. Odredite ime klase koju pruža ta tipska biblioteka u vašem određivanju varijable. Određivanje varijable objekta specifične klase može ubrzati pokazivače objekta. Upotrijebite sljedeću sintaksu:

   ```
   Dim varijabla As [New] klasa
   ```

   Argument *klasa* može biti sastavljen od dva dijela, u obliku *sastavnidio.klasa*.

dio	opis
sastavnidio	Ime sastavnog dijela koji dobavlja objekt. Izbori su prikazani u listi Project/Library pretraživača objekata.
klasa	Ime klase objekta (pruža ga tipska biblioteka sastavnog dijela). Izbori su prikazani u okviru Classes/Modules pretraživača objekata.

Na primjer, možete odrediti varijablu za objekt Microsoft Excel Chart na jedan od sljedeća sva načina:

```
Dim xlChart As Chart
Dim xlChart As Excel.Chart
```

Ako varijablu objekta odredite korištenjem ključne riječi New, Visual Basic će automatski stvoriti objekt i dodijeliti pokazivač objekta kad prvi put upotrijebite varijablu. Na primjer, sljedeći izrazi dodjeljuju pokazivač na novi objekt DAO tablice varijabli txtNarudžbe, postavljajući svojstvo tablice Name na "Narudžbe":

```
Dim tdfNarudžbe As New TableDef
tdfNarud`be.Name = "Narudžbe"
```

Napomena Upotreba varijabli određenih korištenjem ključne riječi New može usporiti vašu aplikaciju. Svaki put kad Visual Basic otkrije varijablu određenu korištenjem ključne riječi New, mora ispitati je li pokazivač objekta već bio dodijeljen varijabli ili nije.

Kako odrediti varijablu objekta za objekt koji nije definiran u tipskoj biblioteci

- Odredite varijablu objekta općom klasom Object, kako slijedi:

  Dim *varijabla* **As Object**

Na primjer, varijabla objSvaki u sljedećem određivanju može biti upotrijebljena za objekt Microsoft Excel Chart ili svaki drugi objekt kojeg pruža ActiveX sastavni dio:

```
Dim objSvaki As Object
```

Glavna razlika između određivanja varijable specifičnom klasom i određivanja varijable općom klasom Object je u tome kako ActiveX povezuje varijablu s objektom. Kad odredite varijablu opće klase Object, ActiveX mora upotrijebiti kasno povezivanje. Kad odredite varijablu objekta s specifičnom klasom, ActiveX koristi rano povezivanje, koje može ubrzati izvođenje objekta. Za više informacija, pogledajte "Ubrzavanje pokazivača objekata", kasnije u ovom poglavlju.

Za više informacija Za više informacija o određivanju varijabli objekata, pogledajte "Naredba Dim" u dijelu *Microsoft Visual Basic 6.0 Language Reference* biblioteke *Microsoft Visual Basic 6.0 Reference Library*. Za više informacija o dodjeljivanju pokazivača objekta varijabli, pogledajte idući odlomak "Dodjela pokazivača objekta varijabli".

Dodjela pokazivača objekta varijabli

Nakon što odredite varijablu objekta, morate dodijeliti pokazivač objekta varijabli prije nego što možete upotrebljavati svojstva, postupke i događaje objekta. Novi pokazivač objekta možete dodijeliti na nekoliko načina:

- Ako odredite varijablu korištenjem ključne riječi New, Visual Basic će automatski stvoriti novi pokazivač objekta kad prvi put upotrijebite varijablu.
- Pokazivač na novi objekt možete dodijeliti u izrazu Set korištenjem ključne riječi New ili funkcije CreateObject.
- Pokazivač na novi ili postojeći objekt možete dodijeliti u izrazu Set korištenjem funkcije GetObject.

Dodjela pokazivača objekta korištenjem ključne riječi New

Ako ActiveX sastavni dio dobavlja tipsku biblioteku, možete upotrijebiti ključnu riječ New u određivanju varijable ili izraz Set za stvaranje novog objekta i dodjelu pokazivača objekta varijabli objekta.

Ako odredite varijablu objekta s ključnom riječi New, Visual Basic će automatski stvoriti novi objekt kad prvi put upotrijebite varijablu. Za više informacija pogledajte "Određivanje varijable objekta", u stalnoj pomoći.

U izrazu Set možete također upotrijebiti ključnu riječ New za dodjelu pokazivača novom objektu specifične klase. Na primjer, sljedeći izrazi dodjeljuju pokazivač na novi objekt DAO table varijabli tdfNarudžbe, postavljajući svojstvo Name table na "Narudžbe":

```
Dim tdfNarudžbe As DAO.TableDef
Set tdfNarudžbe = New DAO.TableDef
tdfNarud`be.Name = "Narudžbe"
```

Za više informacija Pogledajte odlomke "Naredba Dim" i "Naredba Set" u priručniku *Microsoft Visual Basic 6.0 Langugage Reference*.

Dodjela pokazivača objekta korištenjem funkcije CreateObject

Neovisno o tome dobavlja li ActiveX sastavni dio tipsku biblioteku ili ne, možete upotrijebiti funkciju CreateObject u izrazu Set za stvaranje novog objekta i dodjelu pokazivača na objekt varijabli objekta. Kao argument funkcije morate odrediti programski identifikator objekta, a objekt kojem želite pristupiti mora biti stvoren izvana.

Kako dodijeliti pokazivač objekta korištenjem funkcije CreateObject

- Upotrijebite sljedeću sintaksu za funkciju CreateObject.

  Set *varijablaobjekta* = **CreateObject**(*"progID"*, *["imeposlužitelja"]*)

Argument *progID* je obično potpuno označeno ime klase iz koje je stvoren objekt; na primjer, Word.Document. Međutim, argument progID može biti različit od imena klase. Na primjer, progID za objekt Microsoft Excela je "Sheet" prije nego "Worksheet". Neobavezni argument *imeposlužitelja* može biti određen za stvaranje objekta na udaljenom računalu putem mreže. On je dio imena računala u imenu dijeljenja. Na primjer, s mrežnim dijeljenim imenom \\MojPoslužitelj\Javno, argument *imeposlužitelja* bio bi "MojPoslužitelj".

Sljedeći primjer programskog koda pokreće Microsoft Excel (ako se Microsoft Excel već ne izvodi) i uspostavlja varijablu xlApp za upućivanje na objekt klase Application. Argument "Excel.Application" potpuno označuje klasu Application kao klasu koju određuje Microsoft Excel:

```
Dim xlApp As Excel.Application
Set xlApp = CreateObject("Excel.Application")
```

Za više informacija Pogledajte "Funkcija CreateObject" u priručniku *Microsoft Visual Basic 6.0 Langugage Reference*.

Dodjela pokazivača objekta korištenjem funkcije GetObject

Funkcija GetObject se najčešće koristi za dodjeljivanje pokazivača na postojeći objekt, iako ju također možete upotrijebiti i za dodjeljivanje pokazivača na novi objekt.

Kako bi dodijelili pokazivač na postojeći objekt, upotrijebite sljedeću sintaksu.

Set *varijablaobjekta* = **GetObject**([*imestaze*] [, *progID*])

Argument *imestaze* može biti staza do postojeće datoteke, prazan string ili se može potpuno izostaviti. Ako je izostavljena, argument *progID* je obavezan. Određivanje staze do postojeće datoteke uzrokovat će stvaranje objekta funkcijom GetObject uz korištenje informacija spremljenih u datoteci. Korištenje praznog stringa kao prvog argumenta uzrokovat će djelovanje funkcije GetObject kao funkcije CreateObject – stvorit će novi objekt klase čiji je programski identifikator jednak argumentu progID. Sljedeća tablica opisuje rezultate korištenja funkcije GetObject.

Ako se izvodi ActiveX sastavni dio	rezultat
`Set X = GetObject( "MojPos.Application")`	X upućuje na postojeći objekt Application.
`Set X = GetObject("" "MojPos.Object")`	X upućuje na novi, izvana stvoren objekt.

Ako se ne izvodi ActiveX sastavni dio	rezultat
`Set X = GetObject( "MojPos.Object")`	Vraća se pogreška.
`Set X = GetObject("" "MojPos.Object")`	Pokreće se ActiveX sastavni dio (MojPos), te X ukazuje na novi objekt.

Na primjer, varijabla wdApp ukazuje na aplikaciju Microsoft Word koja se izvodi:

```
Dim wdApp As Word.Application
Set wdApp = GetObject("" "Word.Application")
```

Isto kao kod funkcije CreateObject, argument "Word.Application" je programski identifikator za klasu Application određen Microsoft Wordom. Ako se izvodi više primjera Microsoft Worda, ne možete predskazati na koji će primjer ukazivati varijabla wdApp.

Važno Objekt GetObject možete također upotrijebiti za dodjeljivanje pokazivača na objekt u složenoj datoteci dokumenta. *Složena datoteka dokumenta* sadrži pokazivače na više tipova objekata. Na primjer, složena datoteka dokumenta može sadržavati proračunsku tablicu, tekst i slike.

Sljedeći primjer pokreće aplikaciju proračunske tablice, ako se već ne izvodi, i otvara datoteku Prihod.xls:

```
Dim xlBook As Excel.Workbook
Set xlBook = GetObject("C:\Ra~uni\Prihod.xls")
```

Za više informacija Pogledajte "Funkcija GetObject" u priručniku *Microsoft Visual Basic 6.0 Langugage Reference*.

Ubrzavanje pokazivača objekata

Izvođenje svojih Visual Basic aplikacija možete učiniti bržim optimiziranjem načina na koji Visual Basic rješava pokazivače objekata. Na brzinu kojom Visual Basic rukuje pokazivačima objekata utječe sljedeće:

- Je li ActiveX sastavni dio ostvaren kao poslužitelj u-procesu ili poslužitelj izvan-procesa ili nije.
- Je li pokazivač objekta rano povezan ili kasno povezan.

Općenito, ako je sastavni dio bio ostvaren kao dio izvršne datoteke (datoteke .exe), on je poslužitelj *izvan-procesa* i izvodi svoj vlastiti proces. Ako je bio ostvaren kao dinamički poveziva biblioteka, on je poslužitelj *u-procesu* i izvodi se u istom procesu kao klijentska aplikacija.

Aplikacije koje koriste poslužitelje u-procesu obično se izvode brže od onih koje koriste poslužitelje izvan-procesa zato što aplikacija ne treba prelaziti granice procesa kako bi koristila svojstva, postupke i događaje objekta. Za više informacija u poslužiteljima u-procesu i izvan-procesa, pogledajte “Poslužitelji vrste u-procesu i izvan-procesa”, ranije u ovom poglavlju.

Pokazivači objekata su *rano povezani* ako koriste varijable objekta određene kao varijable specifične klase. Pokazivači objekata su *kasno povezani* ako koriste varijable objekata koje su određene kao varijable opće klase Object. Pokazivači objekata koji koriste varijable za rano povezivanje izvode se brže od onih koji koriste varijable za kasno povezivanje.

Na primjer, mogli bi dodijeliti pokazivač na objekt Excel jednoj od sljedećih varijabli:

```
Dim xlApp1 As Excel.Application
Set xlApp1 = New Excel.Application

Sim xlApp2 As Object
Set xlApp2 = CreateObject(“Excel.Application”)
```

Programski kod koji koristi varijablu xlApp1 je rano povezan i izvodit će se brže od koda koji koristi varijablu xlApp2, koja je kasno povezana.

Kasno povezivanje

Kad varijablu odredite s As Object, Visual Basic tijekom prevođenja ne može ustanoviti koju vrstu pokazivača objekta će varijabla sadržavati. U takvim okolnostima, Visual Basic mora upotrijebiti *kasno povezivanje* – znači, Visual Basic mora tijekom izvođenja odrediti hoće li ili neće taj objekt stvarno imati svojstva i postupke koje ste koristili u svom kodu.

Na primjer, Visual Basic će prevesti sljedeći programski kod bez stvaranja pogrešaka, čak iako on upućuje na postupak koji ne postoji, zato što koristi varijablu objekta s kasnim povezivanjem. Varijabla ne provjerava postojanje postupka sve do izvođenja, kad će proizvesti pogrešku:

```
Dim xlApp As Object
Set xlApp = CreateObject("Excel.Application")
xlApp.Nemogu}iPostupak            ' Postupak koji ne postoji.
```

Ovaj kod se izvodi sporije od koda koji koristi varijablu objekta s ranim povezivanjem zato što Visual Basic u prevedenu izvršnu datoteku mora uključiti dodatni kod koji će tijekom izvođenja utvrđivati sadrži li objekt Microsoft Excel Application postupak NemogućiPostupak ili ne.

Iako je kasno povezivanje najsporiji način pozivanja svojstava i postupaka objekta, postoje trenuci kad je neophodno. Na primjer, možete napisati funkciju koja koristi varijablu objekta kako bi djelovala na svaku od različitih klasa objekata. Budući da unaprijed ne znate koja će klasa objekta biti dodijeljena varijabli, trebate ju odrediti kao varijablu s kasnim povezivanjem korištenjem izraza As Object.

Rano povezivanje

Ako Visual Basic tijekom prevođenja može odrediti kojem objektu pripada svojstvo ili postupak, može riješiti pokazivač na objekt tijekom prevođenja. Prevedena izvršna datoteka sadrži samo programski kod za pozivanje svojstava, postupaka i događaja objekta. To se naziva *rano povezivanje*.

Kad varijablu objekta odredite korištenjem klase koja definira objekt, varijabla može sadržavati samo pokazivač na objekt te klase. Visual Basic može upotrijebiti rano povezivanje za svaki kod koji koristi takvu varijablu.

Rano povezivanje dramatično smanjuje vrijeme potrebno za postavljanje ili dohvaćanje vrijednosti svojstva, budući da poziv nadgradnje može činiti značajan dio ukupnog vremena. Kod poziva postupaka poboljšanje ovisi o količini posla koji čini postupak. Kratki postupci, gdje je poziv nadgradnje usporediv s vremenom potrebnim za obavljanje posla, imaju najveću korist.

Korištenje svojstava, postupaka i događaja objekta

Nakon što varijabli objekta dodijelite pokazivač objekta, možete upotrijebiti varijablu za upravljanje svojstvima i postupcima objekta. Varijablu objekta možete također odrediti korištenjem ključne riječi WithEvents te ju upotrijebiti kako bi vaša aplikacija odgovarala na događaje objekta.

Korištenje svojstava i postupaka objekta

Možete upotrijebiti sintaksu *objekt.svojstvo* za postavljanje i vraćanje vrijednosti svojstava ili sintaksu *objekt.postupak* za korištenje postupaka objekta. Na primjer, mogli bi postaviti svojstvo Caption objekta Application kako slijedi:

```
Dim xlApp As Excel.Application
Set xlApp = New Excel.Application
xlApp.Caption = "MojPrviObjekt"
```

Napomena Sintaksa Excel.Application za ukazivanje na klasu Microsoft Excel Application nije podržana u verzijama ranijim od Microsoft Excela 97. Kako bi ukazali na klasu Microsoft Excel Application u Microsoft Excelu 5.0 i Microsoft Excelu 95, upotrijebite umjesto toga sintaksu Š_ExcelApplicationĆ. Na primjer:

```
Set xlApp = New [_ExcelAplication]
```

Možete pozvati postupak Quit objekta Microsoft Excel Application na ovaj način:

```
xlApp.Quit
```

Općenito, dobra je ideja biti specifičan koliko je moguće kod upućivanja na postupke ili svojstva objekata definiranih od ostalih aplikacija ili projekata. Na primjer:

```
‘ Potpuno označeno ime svojstva postavlja
‘ naslov prozora Microsoft Projecta.
Dim pjWindow As Project.Window
‘ Dobivanje pokazivača na prvi objekt prozora.
Set pjWindow = ActiveProject.Windows(1)
pjWindow.Caption = “Naslov projekta”

‘ Upotrebom neodređenog imena Visual Basic će
‘ upotrijebiti prvi objekt koji pronađe s svojstvom
‘ imena Caption - u ovom slučaju, Form1.
Caption = “Naslov forme Microsoft Form1”
```

Napomena Ako u vašu Visual Basic aplikaciju trebate uvesti binarne podatke i namjeravate ih dijeliti između aplikacija korištenjem ActiveX tehnologije, upotrijebite matricu tipa Byte za spremanje podataka. Ako binarne podatke dodijelite stringu i zatim pokušate proslijediti te podatke objektu tipa Automation koji prihvaća string, podaci možda neće biti ispravno pretvoreni. Za više informacija o tipovima podataka, pogledajte 5. poglavlje “Osnove programiranja”.

Za više informacija Za više informacija o radu s svojstvima i postupcima objekta, pogledajte 9. poglavlje “Programiranje objektima”.

Odgovaranje na događaje objekta

Kao dodatak odgovaranju na događaje koji se pojavljuju kod objekata Visual Basica, vaša aplikacija može odgovoriti na događaje u objektu kojeg pruža ActiveX sastavni dio. Na primjer, vaša Visual Basic aplikacija može prikazati okvir s porukom ako se pojavi događaj u radnoj knjizi Microsoft Excela.

Kako bi vaša aplikacija odgovarala na događaje objekta, morate dodati programski kod u potprogram događaja za objekt. Međutim, potprogrami događaja za objekte koje pružaju sastavni dijelovi nisu automatski dostupni u Visual Basicu. Najprije morate odrediti varijablu objekta korištenjem ključne riječi WithEvents.

Nakon što odredite varijablu objekta korištenjem ključne riječi WithEvents, kodni prozor Visual Basica koristi varijablu za prikaz potprograma događaja objekta. Nakon toga možete dodati kod u te potprograme događaja za odgovor na događaje objekta. Kad varijabli dodijelite pokazivač objekta, uspostavljate vezu između varijable i objekta tijekom izvođenja aplikacije.

Kako stvoriti potprogram događaja za objekt kojeg pruža sastavni dio

1. Dodajte pokazivač na tipsku biblioteku objekta vašem projektu Visual Basica. Za više informacija o dodavanju pokazivača na tipsku biblioteku, pogledajte “Stvaranje pokazivača na objekt”, ranije u ovom poglavlju.
2. U odjeljku Declarations forme ili modula klase, odredite varijablu objekta korištenjem ključne riječi WithEvents. Na primjer:

   ```
   Dim WithEvents xlBook As Excel.Workbook
   ```

   Visual Basic dodaje ime varijable objekta okviru Objects u kodnom prozoru. Kad odaberete ime varijable, Visual Basic prikazuje potprograme događaja objekta u okviru s popisom Procedure.
3. Odaberite potprogram događaja, te zatim dodajte potprogramu kod kojeg želite izvesti svojom aplikacijom kad se događaj pojavi.

   Na primjer, pretpostavimo da se vaša Visual Basic aplikacija oslanja na podatke prikazane u radnoj knjizi Microsoft Excela i da ste već s ključnom riječi WithEvents odredili varijablu `xlBook` za radnu knjigu. Kad korisnik pokuša zatvoriti radnu knjigu, možete prikazati poruku i spriječiti zatvaranje radne knjige dodavanjem sljedećeg programskog koda potprogramu događaja xlBook_BeforeClose u vašoj aplikaciji:

```
Private Sub xlBook_BeforeClose(Cancel As Boolean)
  ‘ Skrivanje prozora Microsoft Excela
  ‘ tako da poruka bude vidljiva.
  xlBook.Application.Visible = False
  ‘ Prikaz poruke.
  MsgBox “Ova radna knjiga mora ostati otvorena.”
  ‘ Pokazivanje prozora Microsoft Excela.
  xlBook.Application.Visible = True
  ‘ Postavljanje argumenta Cancel potprograma
  ‘ događaja na True, za poništavanje događaja.
  Cancel = True
End Sub
```

4. Dodijelite pokazivač objekta varijabli objekta određenoj ključnom riječi WithEvents.

   Na primjer, možete dodati sljedeći kod potprogramu događaja Form_Load forme Visual Basica za dodjelu pokazivača na radnu knjigu Prodaja.xls Microsoft Excela varijabli `xlBook`:

```
Private Sub Form_Load()
  Set xlBook = GetObject(“Prodaja.xls”)
  ‘ Prikaza prozora Microsoft Excela
  ‘ i prozora radnog lista.
  xlBook.Application.Visible = True
  xlBook.Windows(1).Visible = True
End Sub
```

Za više informacija Pogledajte "Naredba Dim" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Otpuštanje ActiveX sastavnog dijela

Kad ste gotovi s korištenjem objekta, očistite sve varijable koje pokazuju na objekt kako bi objekt mogao biti otpušten iz memorije. Kako bi očistili varijablu objekta, postavite ju na Nothing. Na primjer:

```
Dim acApp As Access.Application
Set acApp = New Access.Application
MsgBox acApp.SysCmd(acSysCmdAccessVer)
Set AcApp = Nothing
```

Varijable objekta se automatski čiste kad izađu iz područja. Ako želite da varijabla zadrži svoju vrijednost kroz potprograme, upotrijebite javne varijable ili varijable na razini forme, ili stvorite potprograme koji vraćaju objekt. Sljedeći kod pokazuje kako bi mogli upotrijebiti javnu varijablu:

```
Public wdApp As Word.Application
.
.
.
‘ Stvaranje objekta Word i pokretanje Microsoft Worda.
Set wdApp = New Word.Application
.
.
.
‘ Microsoft Word se neće zatvoriti sve dok se aplikacija
‘ ne završi ili dok pokazivač ne bude postavljen na Nothing.
Set wdApp = Nothing
```

Također, pripazite da postavite sve pokazivače objekata na Nothing kad završavate, čak i za neovisne objekte. Na primjer:

```
Dim xlApp As Excel.Application
Dim xlBook As Excel.Workbook
Set xlApp = New Excel.Application
Set xlBook = xlApp.Workbooks.Add
Set xlApp = Nothing       ‘ Pripazite! xlBook bi još uvijek
                          ‘ mogao sadržavati pokazivač objekta.
Set xlBook = Nothing      ‘ Sad su očišćeni svi pokazivači.
```

Kretanje kroz modele objekata

Kad jednom shvatite kako koristiti objekte koje pružaju sastavni dijelovi, možete upotrebljavati svaki objekt koji vam sastavni dio izloži. Sastavni dijelovi se kreću u području od jednostavnih sastavnih dijelova koda ili ActiveX kontrola do velikih sastavnih dijelova, kao programska sučelja aplikacija Microsoft Excel i Microsoft Data Access Object (DAO), koja izlažu puno objekata.

Svaki objekt postoji negdje u hijerarhiji objekata sastavnog dijela, i možete pristupiti objektima na dva načina:

- Izravno, ako je objekt stvoren izvana.
- Posredno, ako je objekt ovisan objekt. Pokazivač na njega možete dobiti iz drugog objekta, višeg u hijerarhiji sastavnog dijela.

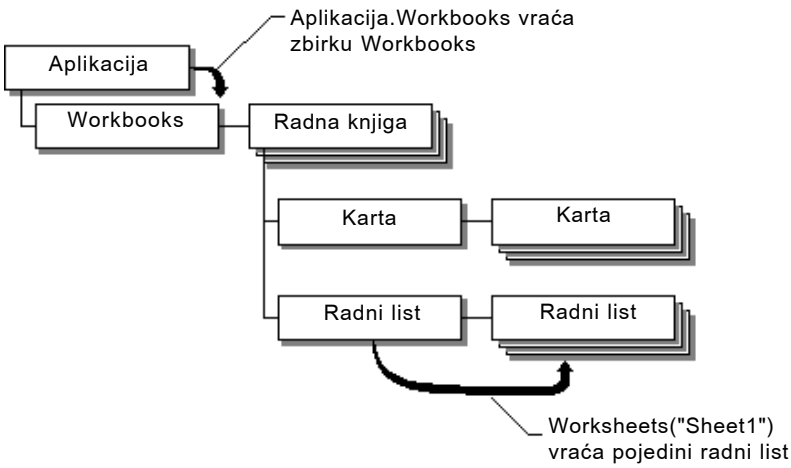
Najbolji način kretanja kroz hijerarhiju objekta je korištenje pretraživača objekata (ako sastavni dio pruža biblioteku objekata).

Kretanje kroz hijerarhiju objekta

Kao što ste vidjeli, kroz hijerarhiju objekta krećete se prema dolje postavljanjem pokazivača na ovisne objekte kroz objekte stvorene izvana. Možete također upotrijebiti postupak na objekt zbirke kako bi natrag dobili pojedini objekt. Za više informacija, pogledajte “Rad s objektima stvorenim izvana i ovisnim objektima”, kasnije u ovom poglavlju.

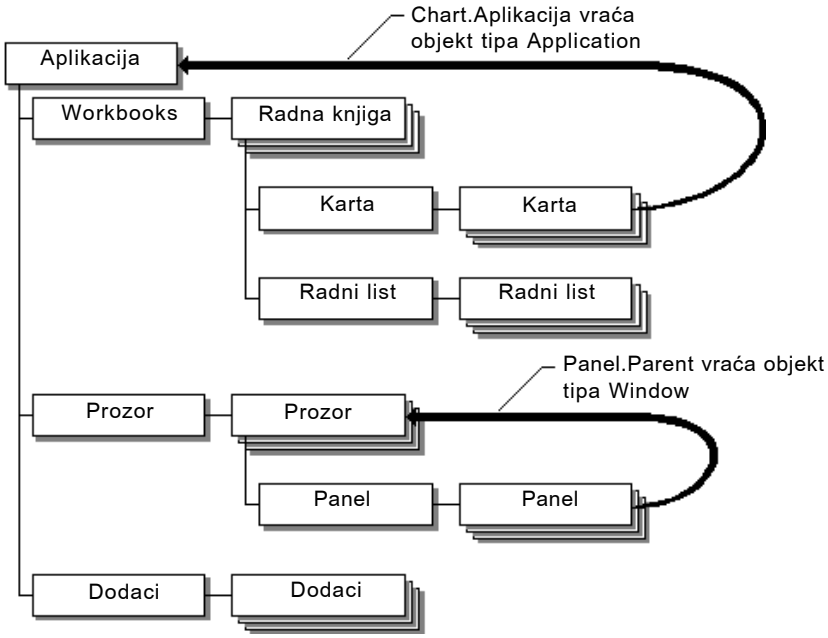
Slika 10.3 pokazuje stazu kretanja objekta u aplikaciji Microsoft Excel.

Slika 10.3 Kretanje nadolje u hijerarhiji objekta Microsoft Excel korištenjem zbirki



Za kretanje natrag prema gore, većina aplikacija koristi svojstva Parent i Application, kao što je prikazano na slici 10.4

Slika 10.4 Kretanje prema gore u hijerarhiji objekta Microsoft Excel korištenjem svojstava Parent i Application



Objekti zbirke

Objekti zbirke su spremnici za grupe drugih objekata. Ti objekti pružaju jednostavan način praćenja niza objekata koji su istog tipa. Na primjer, zbirci svih objekata tipa Menu u aplikaciji može se pristupiti korištenjem objekta zbirke Menus. Sljedeći programski kod možete upotrijebiti za upućivanje na *sve* radne knjige koje su trenutno učitane u aplikaciju Microsoft Excel:

```
Application.Workbooks
```

Uočite da je riječ Workbooks u množini. Standardan dogovor za davanje imena objektima zbirke je množina imena tipa objekta koji čini zbirku. Kroz objekte u zbirci možete prolaziti koristeći izraz For Each, kako slijedi:

```
Dim xlBook As Excel.Workbook
.
.
.
For Each xlBook In Application.Workbooks
  ‘ Prikaz imena svake radne knjige.
  MsgBox xlBook.FullName
Next xlBook
```

Na pojedine objekte u većini zbirki može se također upućivati imenom ili njihovim indeksnim redoslijedom u zbirci. Sljedeći primjer pokazuje kako se možete uputiti na objekte zbirke Styles s imenima “Normal” “Example” i “Heading”.

```
xlBook.Styles(“Normal”)
xlBook.Styles(“Example”)
xlBook.Styles(“Heading”)
```

Uz pretpostavku da su ovi objekti prva tri objekta u zbirci Styles, te da je zbirka temeljena na nuli, možete se također uputiti na njih kako slijedi:

```
xlBook.Styles(1)   ‘ Upućivanje na objekt Normal Style.
xlBook.Styles(2)   ‘ Upućivanje na objekt Example Style.
xlBook.Styles(3)   ‘ Upućivanje na objekt Heading Style.
```

Za više informacija Za više informacija o radu s objektima zbirke, pogledajte 9. poglavlje “Programiranje objektima”.

Rad s objektima stvorenim izvana i ovisnim objektima

Kako ćete stvoriti pokazivač na objekt kojeg pruža sastavni dio ovisi o tome je li objekt stvoren izvana ili je ovisan objekt. Možete izravno stvoriti pokazivač na objekt stvoren izvana; pokazivač na ovisan objekt stvarate posredno korištenjem postupka objekta više razine u hijerarhiji objekata sastavnog dijela.

Objekti stvoreni izvana

Većina velikih aplikacija koje podržavaju ActiveX tehnologiju te drugih ActiveX sastavnih dijelova pruža najviše rangiran objekt stvoren izvana u njihovoj hijerarhiji objekata koji:

- Pruža pristup ostalim objektima u hijerarhiji.
- Pruža postupke i svojstva koja utječu na cijelu aplikaciju.

Na primjer, sve aplikacije Microsoft Officea pružaju najviše rangiran objekt Application. Sljedeći primjer pokazuje kako možete dodijeliti pokazivače objektima Application Microsoft Excela, Microsoft Worda i Microsoft Accessa:

```
Dim xlApp As Excel.Application
Dim wdApp As Word.Application
Dim acApp As Access.Application

Set xlApp = New Excel.Application
Set wdApp = New Word.Application
Set acApp = New Access.Application
```

Nakon toga možete upotrijebiti te varijable za pristup ovisnim objektima svake aplikacije te svojstvima i postupcima tih objekata. Za više informacija pogledajte "Stvaranje pokazivača na objekt", ranije u ovom poglavlju.

> **Napomena** Sintaksa Excel.Application za ukazivanje na klasu Microsoft Excel Application nije podržana u verzijama ranijim od Microsoft Excela 97. Kako bi ukazali na klasu Microsoft Excel Application u Microsoft Excelu 5.0 i Microsoft Excelu 95, upotrijebite umjesto toga sintaksu [_ExcelApplication]. Na primjer:
>
> ```
> Set xlApp = New [_ExcelApplication]
> ```

Kao dodatak ovim najviše rangiranim objektima stvorenim izvana, ActiveX sastavni dijelovi mogu također pružiti objekte stvorene izvana koji su niže rangirani u hijerarhiji objekata sastavnog dijela. Tim objektima možete pristupiti ili izravno kao objektima stvorenima izvana ili posredno kao objektima ovisnim o objektu najviše razine stvorenom izvana. Na primjer, možete stvoriti pokazivač na objekt DAO TableDef ili izravno ili posredno:

```
‘ Izravno stvaranje pokazivača na daoTable1.
Dim daoTable1 As DAO.TableDef
Set daoTable1 = New DAO.TableDef
daoTable1.Name = “Table1”

‘ Posredno stvaranje pokazivača na daoTable2,
‘ kao objekta ovisnog o objektu DAO DBEngine.
Dim daoDBE As DAO.DBEngine
Dim daoWs As DAO.Workspace
Dim daoDb As DAO.Database
Dim daoTable2 As DAO.TableDef

Set daoDBE = DAO.DBEngine
Set daoWs = daoDBE.Workspaces(0)
Set daoDb = daoWs.CreateDatabase(“db1.mdb”, dbLangGeneral)
Set daoTable2 = daoDb.CreateTableDef(“Table2”)
```

Neki objekti pružaju objekt tipa Application, ali mu daju različito ime. Na primjer, Microsoft Jet mehanizam baza podataka u Microsoft Accessu naziva svoj objekt najviše razine imenom DBEngine.

Ovisni objekti

Pokazivač na ovisan objekt možete dobiti samo na jedan način – korištenjem svojstva ili postupka objekta stvorenog izvana za vraćanje pokazivača na ovisan objekt. Ovisni objekti su niže rangirani u hijerarhiji objekata, i može im se pristupati samo korištenjem postupka objekta stvorenog izvana. Na primjer, pretpostavimo da želite pokazivač na objekt Button iz Microsoft Excela. Ne možete dobiti pokazivač na taj objekt korištenjem sljedećeg programskog koda (pojavit će se pogreška):

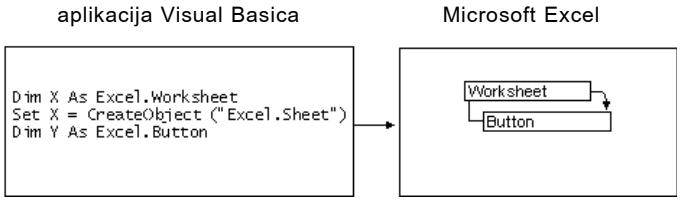
```
Dim xlButton As Excel.Button
Set xlButton = New Excel.Button
```

Umjesto toga, upotrijebite sljedeći kod za dobivanje pokazivača na objekt Button:

```
Dim xlApp As Excel.Application
Dim xlBook As Excel.Workbook
Dim xlSheet As Excel.Worksheet
Dim xlButton As Excel.Button

Set xlApp = New Excel.Application
Set xlBook = xlApp.Workbooks.Add
Set xlSheet = xlBook.Worksheets.Add
Set xlButton = xlSheet.Buttons.Add(44, 100, 100, 44)

‘ Sad možete koristiti svojstvo objekta Button.
xlButton.Caption = “Prvi gumb”
```

Slika 10.5 prikazuje kako aplikacija Visual Basica dobiva pokazivač na objekt Button.

Slika 10.5 Pristupanje ovisnim objektima



Rukovanje pogreškama tijekom rada u ActiveX sastavnim dijelovima

Programski kod za rukovanje pogreškama je posebno važan kad radite s ActiveX sastavnim dijelovima, jer se kod u sastavnom dijelu koristi iz vaše Visual Basic aplikacije. Gdje je to moguće, trebali bi uključiti programski kod za rukovanje pogreškama koje može izazvati sastavni dio. Na primjer, dobra je praksa potražiti pogrešku koja se pojavljuje kad korisnik neočekivano zatvori aplikaciju sastavnog dijela:

```
Function PokreniWord()
‘ Pokretanje Microsoft Worda.
On Error Goto ZamkaZaPogrešku

   ‘ Određivanje varijable tipa Application za Microsoft Word
   ‘ te cjelobrojne varijable za zamku pogreške.
   Dim wdApp As Word.Application
   Dim iPokušaja As Integer

   ‘ Dodjela pokazivača objekta.
   Set wdApp = New Word.Application

   ‘ Otpuštanje varijable objekta.
   Set wdApp = Nothing

   Exit Function
```

```
ZamkaZaPogrešku:
  ‘ Zamka za pogrešku koja se pojavljuje ako
  ‘ Microsoft Word ne može biti pokrenut.
  Select Case Err.Number
    Case 440              ‘ Pogreška automatizacije.
          iPokušaja = iPokušaja + 1
          ‘ Dozvoljavanje najviše 6 pokušaja pokretanja Worda.
          If iPokušaja < 5 Then
                Set wdApp = New Word.Application
                Resume
          Else
                Err.Raise Number:=VBObjectError + 28765, _
                Description:= “Ne mogu pokrenuti Word”
          End If
    Case Else
          Err.Raise Number:= Err.Number
  End Select
End Function
```

Ako se u prethodnom primjeru pojavi bilo koja osim pogreške 440, potprogram prikazuje pogrešku i oživljava pogrešku. Aplikacija koja pruža objekt može natrag proslijediti svoju vlastitu pogrešku. U nekim slučajevima, aplikacija može upotrijebiti isti kod pogreške koji Visual Basic koristi za drugačiju pogrešku. U takvim slučajevima, trebate upotrijebiti izraz On Error Resume Next te provjeriti pogreške odmah nakon svake linije koja može uzrokovati pogrešku. Takav tip provjeravanja pogreške naziva se *ugrađeno rukovanje pogreškom* (*inline error-handling*).

Ispitivanje pokazivača objekata

Prije nego što upotrijebite varijablu objekta u svom kodu, možete poželjeti provjeriti sadržava li varijabla valjan pokazivač objekta. Možete utvrditi je li pokazivač objekta dodijeljen varijabli ili nije upotrebom izraza Is Nothing. Na primjer, sljedeći kod provjerava je li pokazivač objekta dodijeljen varijabli wdDoc:

```
If wdDoc Is Nothing Then MsgBox “Nema pokazivača objekta.”
```

Međutim, izraz Is Nothing neće odrediti je li valjan pokazivač objekta postao nedostupan. Na primjer, ako pokazivač objekta Microsoft Word dodijelite varijabli objekta i Microsoft Word postane nedostupan, varijabla će i dalje sadržavati valjan pokazivač objekta. U takvoj situaciji, upotrijebite svog rukovatelja pogreškama za hvatanje pogreške koja će se pojaviti kad vaš kod pokuša upotrijebiti varijablu.

Za više informacija Za informacije o pogreškama koje određena aplikacija može proslijediti natrag, pogledajte dokumentaciju te aplikacije. Za više informacija o hvatanju pogrešaka, pogledajte 13. poglavlje “Traženje i obrada pogrešaka”.

Rukovanje zahtjevima u tijeku za ActiveX sastavni dio

Obično je potreban samo djelić sekunde za postavljanje svojstva ili poziv postupka objekta. Ponekad, međutim, vaš zahtjev možda neće biti odmah završen.

- Ako pozivate postupak Close radne knjige u Microsoft Excelu dok korisnik ima otvoren dijaloški okvir, Microsoft Excel signalizira da je zaposlen i ne može izvesti vaš zahtjev. To može voditi u stanje *sastavni dio zaposlen* (*component busy*).
- Ako pozivate postupak koji izvodi operaciju duljeg trajanja, kao što je velika količina rada baze podataka dok je baza podataka vrlo aktivna, možete pokušati izvesti drugu operaciju dok prva operacija još uvijek nije riješena. To može voditi u stanje *zahtjev u tijeku* (*request pending*).
- Ako imate dvije ili više aplikacija koje pozivaju zajednički sastavni dio, jedan poziv mora biti završen prije nego što drugi može započeti. Sastavni dijelovi rukuju takvim sukobima nizanjem zahtjeva, znači, postavljajući ih da čekaju u redu. To može također voditi u stanje zahtjeva u tijeku.

Stanje *sastavni dio zaposlen* sliči dobivanju signala zauzete linije kod telefonskog poziva. Znate da se kroz to nećete probiti, pa možete spustiti slušalicu i probati ponovno kasnije.

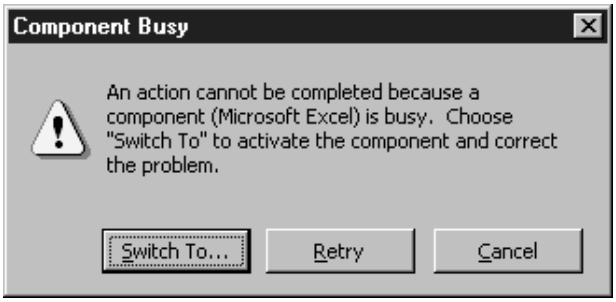
Stanje *zahtjev u tijeku* je slično situaciji kad uspijete dobiti vezu, ali vas onda osoba koju ste nazvali zadrži u razgovoru dulje nego što ste namjeravali. Ako je vaš zahtjev stavljen u niz, tada je zahtjev u tijeku sličan situaciji kad druga strana podigne slušalicu i odmah kaže “Sačekajte trenutak, molim”.

Stanje sastavni dio zaposlen

Sastavni dio može odbiti vaš zahtjev zato što ima otvoren modalni dijaloški okvir, ili zato što je u tijeku operacija editiranja od korisnika. Na primjer, aplikacija Microsoft Excel odbija zahtjeve klijentske aplikacije dok se editira ćelija proračunske tablice.

Visual Basic pretpostavlja da je stanje zauzeća privremeno pa i dalje pokušava poslati zahtjev u određenom vremenskom trajanju. Kad to vrijeme istekne, Visual Basic prikazuje dijaloški okvir Component Busy, kao što je prikazano na slici 10.6.

Slika 10.5 Dijaloški okvir Component Busy



Korisnik može ponoviti zahtjev, odustati od zahtjeva ili se prebaciti na sastavni dio i ispraviti problem (na primjer, otpuštanjem dijaloškog okvira). Ako korisnik odabere Cancel, izaziva se pogreška &h80010001 (RPC_E_CALL_REJECTED) u potprogramu koji je podnio zahtjev.

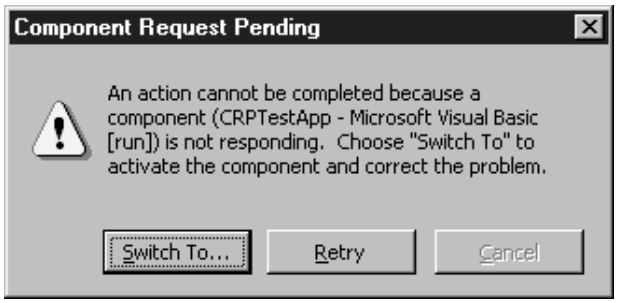
Stanje zahtjev u tijeku

Jednom kad je sastavni dio prihvatio zahtjev vaše aplikacije, vaša aplikacija mora čekati sve dok zahtjev ne bude ispunjen. Ako ispunjenje zahtjeva traje dulje vrijeme, korisnik može pokušati smanjiti vašu aplikaciju ili joj promijeniti veličinu tako da je makne s puta.

Nakon isteka kratkog vremenskog razmaka, Visual Basic odgovara na takve zahtjeve prikazivanjem dijaloga Component Request Pending.

Izgled dijaloškog okvira Component Request Pending je neznatno drugačiji od dijaloga Component Busy. Gumb Cancel je onemogućen, kao što je prikazano na slici 10.7, budući da zahtjev u tijeku ne može biti poništen.

Slika 10.7 Dijaloški okvir Component Request Pending



Prebacivanje na sastavni dio korisno je samo ako je on zaustavljen zbog prikazivanja poruke pogreške kao rezultata vašeg zahtjeva. To ne bi trebalo biti uobičajeno zbivanje, zato što je prikladno ponašanje sastavnog dijela vraćanje stanja pogreške aplikaciji koja ga je pozvala.

Za više informacija Za više informacija, pogledajte odlomke “Promjena poruka Component Busy ili Request Pending” “Nadzor vremenskih razmaka za prekid” i “Izazivanje pogreške nakon isteka vremena”, kasnije u ovom poglavlju.

Promjena poruka Component Busy ili Request Pending

Visual Basic pruža dijaloške okvire Component Busy i Request Pending kao jednostavne unaprijed određene poruke. Postoji puno situacija kad ovi dijaloški okviri možda neće odgovarati vašim potrebama.

- Vaša aplikacija može pozivati postupak objekta kojeg pruža sastavni dio koji nema korisničkog sučelja. Sastavni dijelovi stvoreni korištenjem Visual Basica, verzije Professional ili Enterprise, na primjer, mogu se izvoditi u pozadini bez vidljivih formi.
- Sastavni dio kojeg pozivate možda je stvoren korištenjem osobina udaljene automatizacije Visual Basica, verzija Enterprise, i može se izvoditi na drugom računalu koje se nalazi na nekoj udaljenosti od korisnika.
- Ako je vaša aplikacija učitala radnu knjigu Microsoft Excela korištenjem funkcije GetObject, radna knjiga neće biti vidljiva kad se korisnik prebaci na Microsoft Excel. Zapravo, sam Microsoft Excel možda neće biti vidljiv, i u takvom slučaju gumb Switch To ne radi ništa.

U takvim situacijama, gumb Switch To je neprikladan i može zbuniti korisnika vaše aplikacije. Možete odrediti zamjensku poruku za svako ili oba prekida. Vaše poruke biti će prikazane u jednostavnom okviru s porukom, bez gumba Switch To.

Za stanje zahtjeva u tijeku, okvir s porukom ima samo gumb OK. Za stanje zaposlenosti sastavnog dijela, pruženi su gumb OK i gumb Cancel. Ako korisnik pritisne gumb Cancel, biti će izazvana pogreška -2147418111 (&h80010001) u potprogramu koji je podnio zahtjev.

Sljedeća svojstva objekta App utvrđuju kad će dijaloški okviri Component Busy ili Component Request Pending biti zamijenjeni okvirom s porukom i omogućiti vam određivanje teksta i naslova okvira s porukom.

Svojstvo OLEServerBusyMsgText

Određuje tekst poruke koji će biti prikazan kad se pojavi stanje zaposlenosti sastavnog dijela. Određivanje ovog svojstva uzrokuje korištenje zamjenskog okvira s porukom umjesto uobičajenog dijaloškog okvira Component Busy.

Svojstvo OLEServerBusyMsgTitle

Određuje naslov koji će biti upotrijebljen ako je pribavljena zamjenska poruka za stanje zaposlenosti sastavnog dijela (određivanje samo ovog svojstva neće uzrokovati korištenje zamjenskog okvira s porukom).

Svojstvo OLERequestPendingMsgText

Određuje tekst poruke koji će biti prikazan kad se pojavi stanje zahtjeva u tijeku. Određivanje ovog svojstva uzrokuje korištenje zamjenskog okvira s porukom umjesto uobičajenog dijaloškog okvira Component Request Pending.

Svojstvo OLERequestPendingMsgTitle

Određuje naslov koji će biti upotrijebljen ako je pribavljena zamjenska poruka za stanje zahtjeva u tijeku (određivanje samo ovog svojstva neće uzrokovati korištenje zamjenskog okvira s porukom).

Sljedeći primjer postavlja naslove i tekstove poruka za oba stanja, sastavni dio zaposlen i zahtjev u tijeku, potpuno nadjačavajući dijaloške okvire Component Busy i Component Request Pending.

```
Public Const APP_NASLOV = “Aplikacija Demo”

Private Sub cmdDugiPrijenos_Click()
  On Error Goto DugiPrijenos_Pogreška
  ‘ možete jednom postaviti naslov, u Sub Main.
  App.OLEServerBusyMsgTitle = APP_NASLOV
  App.OLERequestPendingMsgTitle = APP_NASLOV

  ‘ Tekstovi poruke specifični za taj zahtjev.
  App.OLEServerBusyMsgText = “Sastavni dio za dugi _
    prijenos” & “nije odgovorio. Ako ste čekali” _
    & “više od pet minuta, možete” _
    & “odustati od ovog zahtjeva” _
    & “i pokušati ponovno kasnije.” & vbCrLf _
    & “Pozovite mrežnog administratora za provjeru” _
    & “rada sastavnog dijela, ili za prijavu problema.”
  App.OLERequestPendingMsgText = “Vaš zahtjev “ _
    & “se još uvijek izvodi.” & vbCrLf _
    & “Pozovite mrežnog administratora za provjeru” _
    & “rada sastavnog dijela, ili za prijavu problema.”
  ‘ Kod za stvaranje zahtjeva i korištenje rezultata...
  ‘ ...
DugiPrijenos_Čišćenje:
  ‘ Kod za izvođenje potrebnog čišćenja...
  ‘ ...
  Exit Sub
```

```
DugiPrijenos_Pogreška:
  If Err.Number = &h80010001 Then
    MsgBox “Prijenos poništen”
  Else
    ‘ Kod za rukovanje drugim pogreškama.
  End If
  Resume DugiPrijenos_Čišćenje
End Sub
```

Važno Duljina vaših poruka može biti ograničena operativnim sustavom. Poruke s više od tisuću karaktera duljine mogu se koristiti kad je ciljni operativni sustav Windows NT ili Windows 95/98.

Nadzor vremenskih razmaka za prekid

Možete postaviti vremenske razmake za prekid koji određuju kad će Visual Basic prikazati dijaloške okvire Component Busy i Component Request Pending, korištenjem dvaju svojstava objekta App.

Svojstvo OLEServerBusyTimeout

Određuje koliko će dugo Visual Basic pokušavati vaše zahtjeve automatizacije prije prikazivanja dijaloga Component Busy. Podrazumijevana vrijednost je 10 000 milisekundi (10 sekundi).

Svojstvo OLERequestPendingTimeout

Određuje koliko će dugo Visual Basic čekati prije odgovora na klikove mišem, događaje pritiska tipke, te ostale događaje prikazivanjem dijaloga Component Request Pending. Podrazumijevana vrijednost je 5000 milisekundi (5 sekundi).

Sljedeći primjer pokazuje kako vrijednosti prekida mogu biti prilagođene i obnovljene za poziv postupka AnalizaZalihe u zamišljenom objektu VođenjePosla.

```
Public Sub Odre|ivanjePrekida(ByVal lngSasDioZaposlen As _
Long, ByVal lngZahtjevUTijeku As Long)
  App.OLEServerBusyTimeout = lngSasDioZaposlen
  App.OLERequestPendingTimeout = lngZahtjevUTijeku
End Sub

Public Sub ObnovaPrekida()
  App.OLEServerBusyTimeout = 10000

  App.OLERequestPendingTimeout = 5000
End Sub
```

```
Private Sub cmdPunaAnaliza_Click()
  On Error Goto PunaAnaliza_Pogreška
  ‘ Postavljanje vrlo kratkih prekida. Nakon 2 sekunde,
  ‘ korisnik će biti obaviješten i pritisci na tipku ili
  ‘ klikovi mišem će prikazati dijaloge Component Busy
  ‘ i Component Request Pending.
  Set Timeouts 2, 2
  Me.MousePointer = vbHourglass
  gobjVoljenjePosla.AnalizaZalihe txtZGSjeCode.Text, ATIP_PUN
PunaAnaliza_Čišćenje:
  Me.MousePointer = vbDefault
  ResetTimeouts
  Exit Sub

PunaAnaliza_Pogreška:
  If Err.Number = &h80010001 Then
    MsgBox “Analiza poništena”
  Else
    ‘ Kod za rukovanje drugim pogreškama...
  End If
  Resume PunaAnaliza_Čišćenje
End Sub
```

Svaki od ovih prekida možete postaviti na vrlo velike vrijednosti, jer se one spremaju kao tipovi Long. Na primjer, 86 400 000 milisekundi je dan, koji je jednak neograničenom vremenu do prekida. Međutim, kada to učinite, riskirate zaglavljivanje svoje aplikacije sve dok sastavni dio ne bude slobodan, ili dok zahtjev u tijeku ne bude završen.

> Važno Budući da su vrijednosti prekida svojstva objekta App, one također utječu na dokumente koje povezujete ili umećete korištenjem kontrole OLE spremnika ili alatnog okvira. Ako koristite povezane ili umetnute dokumente i promijenite ova svojstva za zahtjev automatizacije, dobra je ideja obnoviti vrijednosti nakon toga.

Izazivanje pogreške nakon isteka vremena

Kod stanja sastavni dio zaposlen, možete zaobići i dijaloški okvir Component Busy i zamjensku poruku postavljanjem svojstva OLEServerBusyRaiseError tipa Boolean objekta App na True. Visual Basic će pokušati izvesti vaš zahtjev u trajanju vremena određenog svojstvom OLEServerBusyTimeout, a zatim će izazvati pogrešku u potprogramu koji je podnio zahtjev automatizacije, kao da je korisnik pritisnuo gumb Cancel na dijaloškom okviru Component Busy.

Pogreška koja se vraća je -2147418111 (&h80010001). U rukovatelju pogreškom za potprogram možete zatim poduzeti najprikladniju akciju. Na primjer, mogli bi prikazati složen dijaloški okvir koji korisniku nudi više izbora za ponovni pokušaj ili drugih izbora.

Ovo svojstvo bit će posebno korisno za sastavne dijelove oblikovane za izvođenje na udaljenim mrežnim računalima, korištenjem osobine udaljene automatizacije Visual Basica, verzija Enterprise. Takav sastavni dio može pozivati druge sastavne dijelove, i mora rukovati pogreškama u tim pozivima bez prikazivanja bilo kakve forme.

Ne postoji odgovarajuće svojstvo za stanje zahtjeva u tijeku. Kad je jednom sastavni dio prihvatio zahtjev automatizacije, klijentska aplikacija mora čekati sve dok zahtjev ne bude završen.

Korištenje vidljivog sučelja sastavnih dijelova

Ako sastavni dio podržava povezivanje i umetanje objekata (objekt linking and embedding, OLE), možete povezati ili umetnuti objekt u svoju aplikaciju bez pisanja bilo kakvog koda korištenjem vidljivog sučelja sastavnog dijela. Vidljivo sučelje sastavnog dijela možete upotrijebiti na jedan od dva načina:

- Dodavanjem kontrole OLE spremnika vašoj aplikaciji, te zatim ubacivanjem objekta u ovu kontrolu.
- Dodavanjem klase objekta alatnom okviru, te zatim dodavanjem objekta te klase vašoj aplikaciji isto kao što bi dodali kontrolu na formu.

Ubacivanje objekta s kontrolom OLE spremnika

Kontrola OLE spremnika daje vam najveću fleksibilnost u korištenju vidljivog sučelja objekta. S kontrolom OLE spremnika, možete:

- Stvoriti oznaku mjesta za objekt u svojoj aplikaciji. Objekt koji se pojavljuje unutar kontrole OLE spremnika možete stvoriti tijekom izvođenja, ili tijekom izrade aplikacije možete mijenjati objekt koji ste postavili u kontrolu OLE spremnika.
- Stvoriti povezani objekt u vašoj aplikaciji.
- Povezati kontrolu OLE spremnika s bazom podataka.
- Izvesti akciju ako korisnik pomakne, promijeni veličinu ili ažurira objekt u kontroli OLE spremnika.
- Stvoriti objekte iz podataka koji su kopirani u odlagalište.
- Prikazati objekte kao ikone.

Kontrola OLE spremnika može istovremeno sadržavati samo jedan objekt. Postoji nekoliko načina za stvaranje povezanih ili umetnutih objekata u kontroli OLE spremnika – vaš odabir načina ovisi o tome stvarate li povezan ili umetnut objekt tijekom izrade ili tijekom izvođenja aplikacije. Jednom kad ste na svojoj formi stvorili kontrolu OLE spremnika, objekt u kontrolu spremnika možete ubaciti:

- Korištenjem dijaloškog okvira Insert Object ili Paste Special. Pogledajte odlomke “Ubacivanje objekata tijekom izrade s kontrolom OLE spremnika” i “Stvaranje objekata tijekom izvođenja s kontrolom OLE spremnika”, kasnije u ovom poglavlju.

- Određivanjem svojstava Class, SourceDoc i SourceItem u prozoru s svojstvima. Pogledajte “Stvaranje objekata tijekom izvođenja s kontrolom OLE spremnika”, kasnije u ovom poglavlju.
- Pozivanjem postupaka CreateEmbed ili CreateLink. Pogledajte “Stvaranje objekata tijekom izvođenja s kontrolom OLE spremnika”, kasnije u ovom poglavlju.

Za više informacija Za više informacija o korištenju kontrole OLE spremnika, pogledajte “Korištenje kontrole OLE spremnika” u 7. poglavlju “Korištenje standardnih kontrola Visual Basica”.

Ubacivanje objekta dodavanjem njegove klase alatnom okviru

Na isti način kako koristite alatni okvir za dodavanje aplikaciji neke od kontrola ugrađenih u Visual Basic, možete upotrijebiti alatni okvir za dodavanje objekta. Prvo, dodajte klasu objekta alatnom okviru, pa zatim dodajte objekt formi.

Kako alatnom okviru dodati klasu objekta

1. U izborniku Project odaberite **Components**.
2. U dijaloškom okviru **Components** kliknite karticu **Insertable Objects**.
3. Odaberite klasu koju želite dodati alatnom okviru, pa kliknite **OK**. Visual Basic će alatnom okviru dodati gumb za tu klasu.

Na primjer, kako bi alatnom okviru dodali gumb za radni list Microsoft Excela, odaberite Microsoft Excel Worksheet.

Jednom kad ste dodali klasu objekta alatnom okviru, možete je potegnuti na formu kako bi stvorili objekt te klase. Na primjer, nakon što ste alatnom okviru dodali gumb Microsoft Excel Worksheet, možete ga potegnuti na formu kako bi stvorili objekt radnog lista na formi.

Razlike među povezanim i umetnutim objektima

Vidljivo sučelje sastavnog dijela koristite za spremanje podataka iz druge aplikacije povezivanjem ili umetanjem tih podataka u svoju Visual Basic aplikaciju. Temeljna razlika između povezanog i umetnutog objekta je u mjestu spremanja podataka. Na primjer, podacima pridruženima *povezanom objektu* upravlja aplikacija koja ih je stvorila i spremila izvan kontrole OLE spremnika. Podaci pridruženi *umetnutom objektu* sadržani su u kontroli OLE spremnika i mogu biti snimljeni zajedno s vašom Visual Basic aplikacijom.

Kad je stvoren povezan ili umetnut objekt, sadrži *ime* aplikacije koja pribavlja objekt, njegove podatke (ili, u slučaju povezanog objekta, *pokazivač* na podatke) i *sliku* podataka.

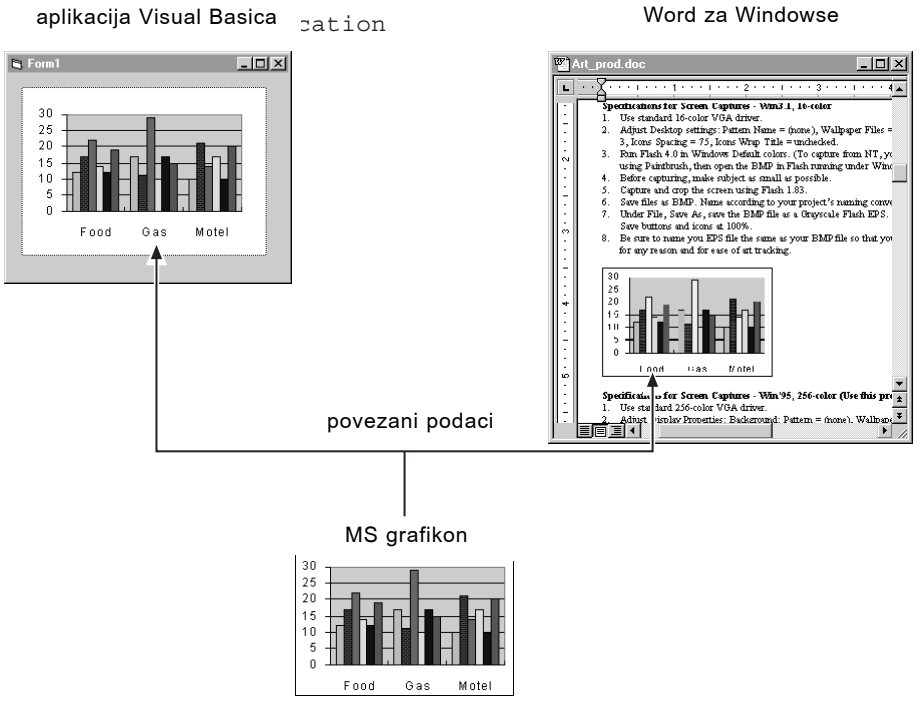
> Napomena Kako bi postavili objekt u kontrolu OLE spremnika, sastavni dio koji pruža objekt mora biti registriran u vašim sistemskim registrima. Kad instalirate aplikaciju koja pribavlja objekte koje želite koristiti u svom projektu, ta aplikacija bi na vašem sustavu trebala registrirati svoju biblioteku objekata tako da se objekti te aplikacije pojavljuju u dijaloškom okviru Insert Object. Možete upotrijebiti aplikaciju Regedit.exe za traženje objekta u sistemskim registrima, ali pripazite da ne mijenjate sadržaje registara.

Povezani objekti

Kad povežete objekt, u svoju aplikaciju ubacujete *oznaku mjesta* (a ne same stvarne podatke) za *povezan objekt*. Na primjer, kad povežete niz ćelija proračunske tablice s Visual Basic aplikacijom, podaci pridruženi s ćelijama spremljeni su u drugoj datoteci; u kontroli OLE spremnika nalaze se samo veza prema podacima i slika podataka. Tijekom rada s vašom Visual Basic aplikacijom, korisnik može aktivirati povezani objekt (dvoklikom na objekt, na primjer), a aplikacija proračunske tablice će se automatski pokrenuti. Korisnik zatim može editirati te ćelije proračunske tablice koristeći aplikaciju proračunske tablice. Kod editiranja povezanog objekta, editiranje se obavlja u odvojenom prozoru izvan kontrole OLE spremnika.

Kad je objekt povezan s Visual Basic aplikacijom, trenutni podaci objekta mogu se vidjeti iz bilo koje aplikacije koja sadrži veze s tim podacima. Podaci postoje samo na jednom mjestu – ActiveX sastavnom dijelu – koji je izvorna aplikacija koja pruža objekt. Na primjer, na slici 10.8, Visual Basic sadrži vezu na aplikaciju Graph. Aplikacija Microsoft Word također sadrži vezu na grafikon. Ako su podaci grafikona promijenjeni bilo kojom od ove dvije aplikacije, promijenjeni grafikon će se pojaviti u *obje* aplikacije – u Visual Basic aplikaciji i u dokumentu Microsoft Worda.

Slika 10.8 Podacima objekta može se pristupati iz puno različitih aplikacija koje sadrže veze na te podatke



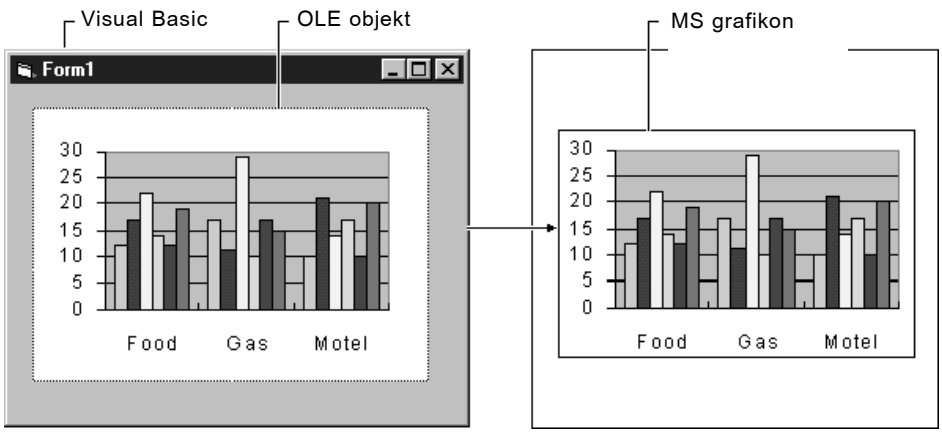
Kao što možete vidjeti, povezivanje čini laganim praćenje pojedinih informacija koje se pojavljuju u više od jedne aplikacije. Povezivanje je korisno kad želite održavati jedan skup podataka kojima se pristupa iz više aplikacija.

Umetnuti objekti

Kako bi stvorili umetnuti objekt, možete upotrijebiti kontrolu OLE spremnika ili dodati klasu objekta alatnom okviru. Sa umetnutim objektima, svi podaci pridruženi s objektom su kopirani i sadržani u kontroli OLE spremnika. Kad snimite sadržaj kontrole u datoteku, ta datoteka sadrži ime aplikacije koja je proizvela objekt, podatke objekta, te sliku objekta u metadatoteci. Zbog toga, umetnuti objekti mogu znatno povećati veličinu datoteke.

Za razliku od povezanih objekata, nijedna druga aplikacija nema pristup podacima u umetnutom objektu. Umetanje je korisno kad želite da vaša aplikacija održava podatke koji su proizvedeni i editirani u drugoj aplikaciji, kao što je prikazano na slici 10.9.

Slika 10.9 Vaša aplikacija održava podatke za umetnuti objekt

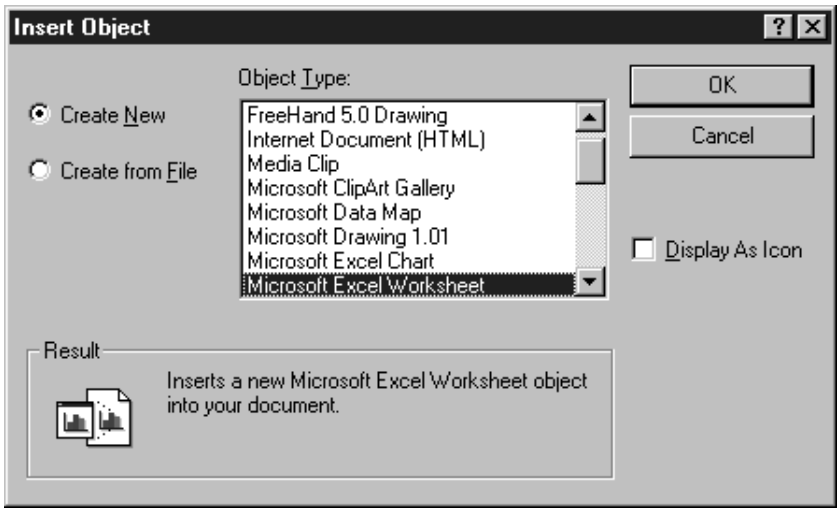


Kad korisnik aktivira objekt (grafikon), aplikacije spremnika poziva ActiveX sastavni dio koji je stvorio objekt (Microsoft Graph), a podaci objekta se otvaraju za editiranje. Kao dodatak, u aplikaciji spremnika prikazuje se korisničko sučelje i sustav izbornika objekta tako da korisnik može nadzirati objekt na mjestu. Za više informacija o aktiviranju na mjestu, pogledajte “Aktiviranje objekta u kontroli OLE spremnika”, kasnije u ovom poglavlju.

Ubacivanje objekata tijekom izrade s kontrolom OLE spremnika

Svaki put kad na formi stvorite kontrolu OLE spremnika, Visual Basic prikazuje dijaloški okvir Insert Object. Taj dijaloški okvir, prikazan na slici 10.10, upotrebljavate za ubacivanje povezanih ili umetnutih objekata tijekom izrade aplikacije. Dijaloški okvir Insert Object predstavlja listu dostupnih objekata koje možete povezati ili umetnuti u svoju aplikaciju.

Slika 10.10 Dijaloški okvir Insert Object



Kad tijekom izrade aplikacije ubacite objekt u kontrolu OLE spremnika, automatski se postavljaju svojstva Class, SourceDoc i SourceItem. Ova svojstva označuju aplikaciju koja pribavlja objekt, ime izvorne datoteke te sve specifične podatke koji su povezani iz te datoteke.

Ubacivanje povezanih objekata tijekom izrade aplikacije

Kad ubacite povezan objekt, podaci prikazani u kontroli OLE spremnika postoje na jednom mjestu – izvornoj datoteci. Trenutni podaci objekta mogu se vidjeti iz bilo koje aplikacije koja ima vezu s tim podacima. Kontrola OLE spremnika održava informacije o vezi s objektom, kao što su ime aplikacije koja je pribavila objekt, ime povezane datoteke te slika povezanih podataka.

Kako ubaciti povezan objekt korištenjem dijaloškog okvira Insert Object

1. Stvorite **kontrolu OLE spremnika** na formi.

   Prikazat će se dijaloški okvir **Insert Object**. Taj dijaloški okvir možete također prikazati u svako vrijeme klikom desnom tipkom miša na **kontrolu OLE spremnika** te odabirom naredbe **Insert Object**.

2. Odaberite gumb izbora **Create from File**.

3. Odaberite gumb **Browse**.

   Prikazat će se dijaloški okvir **Browse**.

4. Odaberite datoteku s kojom se želite povezati.

5. Kliknite **Insert** za povratak u dijaloški okvir **Insert Object**.
6. Odaberite kontrolnu kućicu **Link** u dijaloškom okviru **Insert Object**, te odaberite **OK** za stvaranje povezanog objekta.

Kad koristite povezani objekt, svaki korisnik koji pokrene vašu aplikaciju mora imati pristup (valjanu stazu) povezanoj datoteci i kopiju aplikacije koja je stvorila datoteku. Inače, kad se pokrene vaša aplikacija, prikazat će se slika originalnih podataka, ali korisnik neće biti u mogućnosti mijenjati podatke, niti će vidjeti promjene koje su napravili drugi u povezanim podacima. To može biti važno ako se vaša aplikacija izvodi u mreži.

Ako vaša aplikacija sadrži povezani objekt, moguće je da će podaci tog objekta biti promijenjeni u drugoj aplikaciji dok se vaša aplikacija ne izvodi. Idući put kad se vaša aplikacija pokrene, promjene u izvoru neće se automatski pojaviti u kontroli OLE spremnika. Kako bi prikazali trenutne podatke u kontroli OLE spremnika, upotrijebite postupak Update kontrole:

```
oleObj.Update
```

Za više informacija Pogledajte "Postupak Update (OLE spremnik)" u biblioteci *Microsoft Visual Basic 6.0 Reference Library*.

Ako korisnik želi snimiti promjene u povezanom objektu, mora ih snimiti iz izbornika ActiveX sastavnog dijela. postupak SaveToFile kontrole OLE spremnika primjenjiv je samo za umetnute objekte.

Stvaranje umetnutih objekata tijekom izrade aplikacije

Kad stvarate umetnuti objekt, možete umetnuti podatke iz datoteke ili stvoriti nov, prazan objekt koje će kasnije biti popunjen podacima. Kad umetnete podatke iz datoteke, u kontroli OLE spremnika prikazuje se kopija podataka određenog objekta. Kad stvorite nov objekt, poziva se aplikacija koja je stvorila taj objekt i možete unijeti podatke u objekt.

U pravilu, umetnute objekte koji prikazuju postojeće podatke stvarate tijekom izrade aplikacije. To vam omogućuje pregled podataka objekta onako kako će biti predstavljeni korisniku. Možete pomicati kontrolu OLE spremnika i ostale kontrole na formi te im mijenjati veličinu kako bi stvorili korisničko sučelje vaše aplikacije.

Kako bi prikazali postojeće podatke u umetnutom objektu, stvorite objekt korištenjem postojeće datoteke kao predloška. Kontrola OLE spremnika tad sadrži sliku podataka u datoteci. Aplikacija koja prikazuje podatke korištenjem umetnutog objekta bit će veća od aplikacije koja prikazuje iste podatke korištenjem povezanog objekta, zato jer aplikacija s umetnutim objektom stvarno sadrži podatke izvorne datoteke.

Kako stvoriti umetnuti objekt korištenjem postojeće datoteke

1. Stvorite **kontrolu OLE spremnika** na vašoj formi.

   Automatski se prikazuje dijaloški okvir **Insert Object**.

2. Odaberite gumb izbora **Create from File**.
3. Odaberite gumb **Browse**.

   Prikazati će se dijaloški okvir **Browse**.

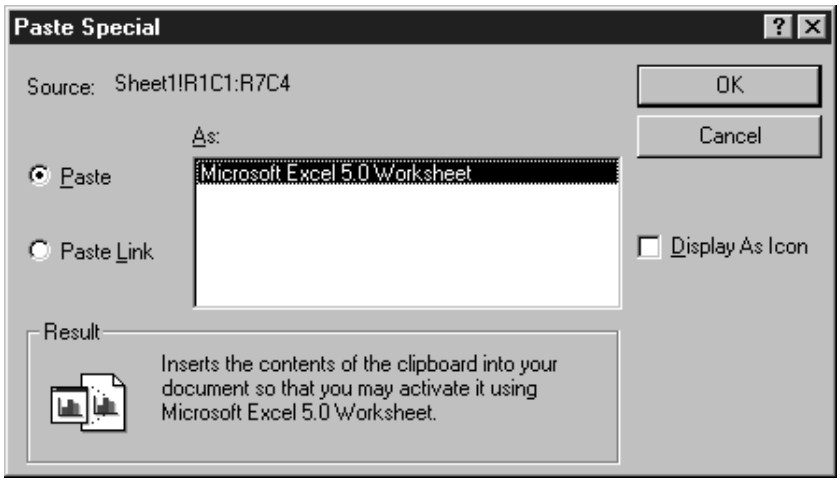
4. Odaberite datoteku koju želite umetnuti.
5. Odaberite **Insert** za povratak u dijaloški okvir **Insert Object**.
6. U dijaloškom okviru **Insert Object**, odaberite **OK** za stvaranje umetnutog objekta.

Za razliku od podataka u povezanom objektu, podaci u umetnutom objektu nisu postojani. Drugim riječima, ako želite da se promjene koje je unio korisnik pojave kad se idući put pokrene vaša aplikacija, morate upotrijebiti postupak SaveToFile za snimanje podataka. Za više informacija o snimanju umetnutih podataka u datoteku, pogledajte “Snimanje i dohvaćanje umetnutih podataka”, kasnije u ovom poglavlju.

Stvaranje objekata korištenjem dijaloškog okvira Paste Special

Drugi način stvaranja objekta tijekom izrade aplikacije je korištenje dijaloškog okvira Paste Special (prikazanog na slici 10.11). Ovaj dijaloški okvir je koristan ako samo želite upotrijebiti dio datoteke – na primjer, skup ćelija iz proračunske tablice, ili odlomak iz Word dokumenta.

Slika 10.11 Dijaloški okvir Paste Special



Kako stvoriti objekt korištenjem dijaloškog okvira Paste Special

1. Pokrenite aplikaciju koja sadrži podatke koje želite povezati ili umetnuti.
2. Odaberite podatke koje želite povezati ili umetnuti.
3. U izborniku **Edit** ActiveX sastavnog dijela, odaberite **Copy**.

   Podaci će se kopirati u odlagalište.
4. U Visual Basicu, kliknite **kontrolu OLE spremnika** s desnom tipkom miša, i odaberite naredbu **Paste Special** iz pomoćnog izbornika.
5. Odaberite gumb izbora **Paste** ako želite stvoriti umetnuti objekt.

   - ili -

   Odaberite gumb izbora **Paste Link** ako želite stvoriti povezan objekt.

   Ako u kontroli već postoji objekt koji je umetnut ili povezan, poruka će vas upitati želite li obrisati taj postojeći objekt i stvoriti novi na njegovom mjestu.

1. Odaberite **OK** za stvaranje objekta.

Stvaranje objekata tijekom izvođenja s kontrolom OLE spremnika

Kako bi tijekom izvođenja aplikacije stvorili povezan ili umetnut objekt, upotrijebite postupke i svojstva u programskom kodu. Kontrola OLE spremnika ima niz svojstava i postupaka koje možete upotrijebiti za upravljanje povezanim ili umetnutim objektima. Za potpunu listu svojstava i postupaka koji se primjenjuju s kontrolom OLE spremnika, pogledajte “Kontrola OLE spremnika”, u stalnoj pomoći.

Korištenje svojstva Object

Korištenjem svojstva Object kontrole OLE spremnika, također možete koristiti svojstva i postupke povezanog ili umetnutog objekta. Svojstvo Object dostupno je samo tijekom izvođenja, samo je za čitanje i sadrži pokazivač na objekt u kontroli OLE spremnika. Upotrijebite to svojstvo za izvođenje zadataka automatizacije s kontrolom OLE spremnika, uključujući programsko upravljanje svojstvima i postupcima koje podržava objekt:

```
strObjIme = oleObj1.Object.Name
```

Kako bi mogli upotrijebiti ovo svojstvo, kontrola OLE spremnika mora sadržavati objekt koji je programabilan. Za više informacija o programabilnim objektima, pogledajte “Tipovi ActiveX sastavnih dijelova”, u stalnoj pomoći.

Stvaranje povezanih objekata tijekom izvođenja

Tijekom izvođenja aplikacije povezani objekt možete stvoriti iz datoteke korištenjem postupka CreateLink kontrole OLE spremnika. Ovaj postupak traži jedan argument, *dokumentizvora*, koji je datoteka iz koje je objekt stvoren, i neobavezni argument *stavkaizvora*, koji određuje podatak iz izvorne datoteke s kojim se želite povezati. Sljedeći dio programskog koda stvara povezan objekt tijekom izvođenja:

```
oleObj1.CreateLink "C:\Excel\Test.xls"
```

Napomena Ako upotrijebite postupak CreateLink za stvaranje povezanog objekta, ne trebate odrediti svojstva Class, SourceDoc i SourceItem u prozoru s svojstvima.

Za više informacija Pogledajte "Postupak CreateLink" u priručniku *Microsoft Visual Basic 6.0 Langugage Reference*.

Stvaranje umetnutih objekata tijekom izvođenja

Kako bi stvorili umetnuti objekt iz datoteke tijekom izvođenja aplikacije, možete upotrijebiti postupak CreateEmbed. Ovaj postupak ima dva argumenta, *dokumentizvora* i *klasa* (koji je neobavezan ako je postavljeno svojstvo SourceDoc). *Dokumentizvora* određuje predložak za objekt, a *klasa* određuje tip objekta. Kad koristite postupak CreateEmbed, ne trebate određivati svojstva SourceDoc i Class.

Sljedeći dio programskog koda stvara umetnut objekt korištenjem postojeće datoteke kao predloška za objekt.

```
oleObj1.CreateEmbed "Q1profit.xls"
```

Za više informacija Pogledajte "Postupak CreateEmbed", u priručniku *Microsoft Visual Basic 6.0 Langugage Reference*.

Kad stvorite prazan umetnut objekt, dobra je ideja pokrenuti ActiveX sastavni dio koji će pružiti podatke za objekt. Možete to napraviti s postupkom DoVerb. To omogućuje korisniku da tijekom izvođenja aplikacije unese bilo kakve podatke. Nakon toga korisnik može prikazati te nedavno unesene podatke u kontroli OLE spremnika odabirom naredbe Update ActiveX sastavnog dijela (ta naredba izbornika trebala bi se pojaviti u izborniku File sastavnog dijela).

Kako stvoriti prazan umetnut objekt tijekom izvođenja aplikacije

1. Upotrijebite postupak CreateEmbed bez određivanja izvornog dokumenta kako bi stvorili prazan umetnut objekt. Na primjer, ovaj dio koda ubacuje predložak za radni list Microsoft Excela u kontrolu OLE spremnika:

   ```
   oleObj1.CreateEmbed "" "Excel.Sheet"
   ```

2. Upotrijebite postupak DoVerb. Podrazumijevan glagol (verb) za postupak DoVerb ovisi o aplikaciji. Kod Microsoft Excela, podrazumijevan glagol je Edit.

Na primjer, sljedeći dio programskog koda stvara prazan umetnut objekt te zatim pokreće aplikaciju koja ga je stvorila korištenjem podrazumijevane akcije DoVerb.

```
  oleObj1.CreateEmbed "" "Excel.Sheet"
oleObj1.DoVerb -5                ' Pokretanje
```

Pružanje praznih umetnutih objekata korisno je kod stvaranja aplikacija temeljenih na dokumentima koje koriste raznolike informacije iz različitih aplikacija. Za više informacija, pogledajte "Dopuštanje korisniku da odredi objekte tijekom izvođenja", u nastavku.

Povezivanje baze podataka s kontrolom OLE spremnika

Kontrolu OLE spremnika možete povezati s podacima spremljenim u Microsoft Jet mehanizmu za baze podataka ili u bazi podataka Microsoft Access. Možete to poželjeti napraviti, na primjer, ako imate bazu podataka s tablicom slika djelatnika. Ako su slike spremljene kao objekti, možete ih povezati s kontrolom OLE spremnika i prikazati ih na formi kad se pristupi svakom zapisu s kontrolom podataka. Kako bi povezali podatke s jednom od tih baza podataka, odredite izvor podataka (ime skupa slogova) u svojstvu DataSource te ime polja iz tog izvora podataka u svojstvu DataField kontrole OLE spremnika. Kod prikazivanja objekta iz baze podataka, kontrola OLE spremnika omogućuje korisniku pokretanje, editiranje i ažuriranje objekta. Kao i kod svih kontrola povezivanja, ažurirani objekt se automatski zapisuje natrag u bazu podataka kad se promijeni položaj zapisa.

Dopuštanje korisniku da odredi objekte tijekom izvođenja

Prikazivanjem dijaloških okvira Paste Special i Insert Object tijekom izvođenja, možete omogućiti korisniku da stvori razne objekte. Možete to napraviti kad stvarate aplikaciju temeljenu na dokumentima. U takvoj aplikaciji, korisnik spaja podatke iz raznih aplikacija kako bi stvorio jedan dokument. Na primjer, ta aplikacija mogla bi biti obradnik teksta u kojem korisnik može unijeti neki tekst te zatim umetnuti proračunsku tablicu i grafikon korištenjem dijaloških okvira Insert Object ili Paste Special.

Postupak InsertObjDlg kontrole OLE spremnika koristite za prikaz dijaloškog okvira Insert Object, a postupak PasteSpecialDlg možete upotrijebiti za prikaz dijaloga Paste Special. Ova dva dijaloga omogućuju korisniku da odluči što će biti postavljeno u kontrolu OLE spremnika.

- Dijaloški okvir Insert Object predstavlja listu dostupnih objekata i stvara objekt temeljen na odabiru korisnika.
- Dijaloški okvir Paste Special omogućuje korisniku da ulijepi objekt iz sistemskog odlagališta (Clipboard) u kontrolu OLE spremnika.

Ove dijaloške okvire možete prikazati tijekom izvođenja pozivom odgovarajućeg postupka ili događaja – na primjer, događaja Click izbornika:

```
Private Sub cmdUbaci_Click()
  ‘ Prikaz dijaloškog okvira Insert Object.
  oleObj1.InsertObjDlg
  ‘ Provjera svojstvom OLEType je li objekt stvoren.
  If oleObj1.OLEType = vbOLENone Then
    MsgBox “Objekt nije stvoren.”
  End If
End Sub

Private Sub oleObj1_Click()
  ‘ Utvrđivanje može li podatak koji je u odlagalištu
  ‘ biti ulijepljen u kontrolu OLE spremnika.
  If oleObj1.PasteOK Then
    ‘ Prikaz dijaloškog okvira Paste Special.
    oleObj1.PasteSpecialDlg
    ‘ Provjera je li objekt stvoren.
    If oleObj1.OLEType = vbOLENone Then
          MsgBox “Objekt nije stvoren.”
    End If
  End If
End Sub
```

Kad se dijaloški okvir jednom prikaže, ne trebate pisati dodatni programski kod za stvaranje objekta. Korisnik čini odabire u dijaloškom okviru i odabire OK za stvaranje objekta. Ako korisnik poništi dijalog, objekt neće biti stvoren.

Napomena Prije prikazivanja dijaloških okvira Insert Object ili Paste Special, možete poželjeti ustvrditi vrijednost svojstva OLEType kako bi vidjeli sadržava li kontrola OLE spremnika povezan objekt, umetnut objekt ili ne sadržava ništa, kao što je prikazano u prethodnom primjeru programskog koda.

Konstanta vbOLENone i ostale ugrađene konstante izlistane su objektnoj biblioteci Visual Basic (VB) u pretraživaču objekata.

Određivanje kako se objekt prikazuje u kontroli OLE spremnika

Svojstvo DisplayType kontrole OLE spremnika možete upotrijebiti za naznačivanje hoće li se objekt pojaviti kao ikona (postavite DisplayType = 1), ili će podaci objekta biti prikazani kao kontrola (postavite DisplayType = 0). Ovo svojstvo također određuje standardnu postavku za kontrolnu kućicu Display As Icon kad se prikažu dijaloški okviri Insert Object i Paste Special tijekom izvođenja i izrade aplikacije.

Napomena Kad kontrola OLE spremnika jednom sadržava objekt, ne možete promijeniti njezin tip prikazivanja. Možete, međutim, obrisati povezan ili umetnut objekt, odrediti svojstvo DisplayType, te ubaciti nov objekt.

Svojstvo SizeMode upotrebljavate za određivanje kako će ikona objekta ili slika podatka biti prikazana u kontroli OLE spremnika kad kontrola nije aktivna u korisničkom sučelju. Postavke 0 – Clip ili 3 – Zoom reže višak slike tako da odgovara veličini kontrole, ali ne mijenja stvarnu veličinu slike (možda nećete vidjeti cijelu sliku kad je editirate). Objekt koji je manji od kontrole editira se u području manjem od kontrole. Objekt veći od kontrole popunjava cijelo područje spremnika kontrole i može biti odrezan ako je veći od tog područja. Suprotno tome, postavljanje svojstva SizeMode na 2 – AutoSize mijenja veličinu kontrole tako da odgovara veličini slike.

Aktiviranje objekta u kontroli OLE spremnika

Dok postupak DoVerb kontrole OLE spremnika aktivira objekt tijekom izvođenja aplikacije, svojstvo AppIsRunning možete upotrijebiti za određivanje je li aplikacija koja pribavlja objekt aktivna i izvodi li se. Svojstvo AppIsRunning možete postaviti na True kako bi pokrenuli ActiveX sastavni dio, što će uzrokovati brže aktiviranje objekata. Ovo svojstvo možete također postaviti na False kako bi zatvorili aplikaciju ili poduzeli drugu prikladnu akciju kad objekt izgubi fokus.

Zamjensko aktiviranje

Neki umetnuti objekti mogu biti editirani (aktivirani) iz kontrole OLE spremnika. To se naziva *zamjensko aktiviranje* (*in-place activation*), Budući da korisnik može dva puta kliknuti na objekt u vašoj aplikaciji i surađivati s aplikacijom koja pribavlja objekt, bez prebacivanja na drugu aplikaciju ili prozor.

Za objekte koji podržavaju zamjensko aktiviranje, možete odrediti svojstvo AutoActivate tako da korisnik može aktivirati objekt u svako vrijeme. Znači, kad je svojstvo AutoActivate kontrole OLE spremnika postavljeno na Double-Click, korisnik može dva puta kliknuti kontrolu kako bi je aktivirao. Važno je zapamtiti da aktiviranje objekta pokreće aplikaciju tog objekta ako se ona već ne izvodi.

> Napomena Ako tijekom izvođenja želite prikazati izbornike ActiveX sastavnog dijela kad korisnik klikne kontrolu OLE spremnika, morate odrediti bar jednu stavku izbornika za formu i postaviti njeno svojstvo Visible na False. To može biti nevidljivi izbornik ako ne želite prikazati ni jedan izbornik. Pogledajte 6. poglavlje “Stvaranje korisničkog sučelja”, za više informacija o prikazivanju izbornika i alatnih traka ActiveX sastavnog dijela u aplikaciji spremnika kad je objekt aktiviran tijekom izvođenja te aplikacije.

Odgovaranje na pomicanje i promjenu veličine spremnika

Kontrola OLE spremnika ima događaj ObjectMove, koji se izaziva kad se objekt pridružen s kontrolom OLE spremnika pomakne ili mu se promijeni veličina. Argumenti događaja ObjectMove predstavljaju koordinate objekta (bez njegovog okvira) unutar spremnika objekta. Ako je objekt maknut s forme, argumenti imaju vrijednosti koje

predstavljaju položaje relativne u odnosu na gornji lijevi kut forme. Te vrijednosti mogu biti pozitivne ili negativne. Ako su promijenjena svojstva Width ili Height ActiveX sastavnog dijela, o tome će biti obaviještena kontrola OLE spremnika.

Događaj ObjectMove je jedini način na koji kontrola OLE spremnika može ustvrditi da je objekt pomican ili mu je promijenjena veličina. Događaj ObjectMove pojavljuje se kad korisnik pomakne ili promijeni veličinu objektu sadržanom u kontroli OLE spremnika. Na primjer:

```
Private Sub oleObj1_ObjectMove(Left As Single, _
Top As Single, Width As Single, Height As Single)
  ‘ Ovaj postupak mijenja veličinu kontrole OLE
  ‘ spremnika na veličinu novog objekta.
  oleObj1.Move oleObj1.Left, oleObj1.Top, Width, Height
  ‘ Ovaj postupak pomiče kontrolu OLE spremnika
  ‘ na položaj novog objekta.
  oleObj1.Move Left, Top, _
  oleObj1.Width, oleObj1.Height
  ‘ Ponovno iscrtavanje forme.
  Me.Refresh
End Sub
```

Snimanje i dohvaćanje umetnutih podataka

Podaci pridruženi s umetnutim objektom nisu postojani; znači, kad se zatvori forma koja sadrži kontrolu OLE spremnika, sve promjene u podacima pridruženim tom kontrolom su izgubljene. Kako bi snimili ažurirane podatke iz objekta u datoteku, upotrijebite postupak SaveToFile kontrole OLE spremnika. Kad su jednom podaci snimljeni u datoteku, možete otvoriti tu datoteku i obnoviti objekt.

Ako je objekt povezan (OLEType = 0 – Linked), tada se u određenu datoteku snimaju samo informacija o vezi i slika podataka. Podatke objekta održava aplikacija koja je stvorila objekt. Ako korisnik želi snimiti promjene u povezanu datoteku, mora odabrati naredbu Save iz izbornika File ActiveX sastavnog dijela jer je postupak SaveToFile primjenjiv samo za umetnute objekte.

Ako je objekt umetnut (OLEType = 1 – Embedded), podatke objekta održava kontrola OLE spremnika i mogu biti snimljeni iz vaše Visual Basic aplikacije.

Objekti u kontroli OLE spremnika mogu biti snimljeni samo u otvorene, binarne datoteke.

Kako snimiti podatke iz objekta u datoteku

1. Otvorite datoteku u binarnom modu.
2. Upotrijebite postupak SaveToFile.

Potprogram događaja cmdSnimiObjekt_Click prikazuje ove korake:

```
Private Sub cmdSnimiObjekt_Click()
  Dim BrojDat as Integer
```

```
    ‘ Dobivanje broja datoteke.
    BrojDat = FreeFile
    ‘ Otvaranje datoteke u koju se snima.

    Open “TEST.OLE” For Binary As #BrojDat
    ‘ Snimanje datoteke.
    oleObj1.SaveToFile BrojDat
    ‘ Zatvaranje datoteke.
    Close #BrojDat
End Sub
```

Kad je objekt jednom snimljen u datoteku, može biti otvoren i prikazan u kontroli OLE spremnika.

Napomena Kad koristite postupke SaveToFile ili ReadFromFile, položaj datoteke se smješta odmah iza objekta. Zbog toga, ako snimate više objekata u datoteku, trebate ih čitati po istom redu kako ste ih snimali.

Kako učitati podatke iz datoteke u kontrolu OLE spremnika

1. Otvorite datoteku u binarnom modu.
2. Upotrijebite postupak ReadFromFile na objektu.

Potprogram događaja cmdOtvoriObjekt_Click prikazuje ove korake:

```
Private Sub cmdOtvoriObjekt_Click()
    Dim BrojDat As Integer
    ‘ Dobivanje broja datoteke.
    BrojDat = FreeFile
    ‘ Otvaranje datoteke.
    Open “TEST.OLE” For Binary As #BrojDat
    ‘ Čitanje datoteke.
    oleObj1.ReadFromFile BrojDat
    ‘ Zatvaranje binarne datoteke.
    Close #BrojDat
End Sub
```

Događaj Updated poziva se svaki put kad je promijenjen sadržaj objekta. Ovaj događaj je koristan za utvrđivanje jesu li podaci objekta bili mijenjani nakon što su posljednji put snimljeni. Kako bi to napravili, postavite opću varijablu u događaju Updated koja će pokazivati da objekt treba biti snimljen. Kad snimite objekt, obnovite Varijablu.

Odgovaranje na događaje miša i tipkovnice

Vaše Visual Basic aplikacije mogu odgovarati na niz događaja miša i događaja tipkovnice. Na primjer, forme, okviri za sliku te kontrole slike mogu otkriti položaj pokazivača miša, mogu otkriti je li pritisnuta lijeva ili desna tipka miša, te mogu odgovoriti na različite kombinacije tipki miša te tipki SHIFT, CTRL ili ALT. Korištenjem događaja tipaka, možete programirati kontrole i forme tako da odgovaraju na razne akcije tipaka ili tumače i obrađuju ASCII karaktere.

Kao dodatak, aplikacije Visual Basica mogu podržati osobine OLE povlačenja i ispuštanja te povlačenja i ispuštanja upravljanog događajima. Postupak Drag možete upotrijebiti s nekim svojstvima i događajima kako bi omogućili operacije poput povlačenja i ispuštanja kontrola. OLE povlačenje i ispuštanje daje vašim aplikacijama svu snagu koju trebate za razmjenu podataka kroz Windows okruženje, a velik dio te tehnologije je dostupan vašim aplikacijama bez pisanja programskog koda.

Možete također upotrijebiti miša ili tipkovnicu za upravljanje obradom dugotrajnih pozadinskih zadataka, što vašim korisnicima omogućuje prebacivanje na druge aplikacije ili prekid pozadinske obrade.

Ostale akcije i događaji koji uključuju miša ili tipkovnicu (događaji Click i DblClick, događaji Focus te događaj Scroll) nisu pokriveni u ovom poglavlju. Za više informacija o događajima Click i DblClick, pogledajte odlomke “Pokretanje akcija klikom na gumb” i “Razumijevanje fokusa” u 3. poglavlju “Forme, kontrole i izbornici”, te odlomke “Događaj Click” i “Događaj DblClick” u priručniku *Microsoft Visual Basic 6.0 Language Reference* biblioteke *Microsoft Visual Basic 6.0 Reference Library*. Također pogledajte “Događaj Scroll”, u stalnoj pomoći.

Sadržaj

- Odgovaranje na događaje miša
- Otkrivanje tipki miša
- Otkrivanje stanja SHIFT, CTRL i ALT
- Povlačenje i ispuštanje
- Povlačenje i ispuštanje tipa OLE
- Prilagođavanje pokazivača miša
- Odgovaranje na događaje tipkovnice
- Prekidanje pozadinske obrade

Primjer aplikacije: Mouse.vbp

Većina primjera programskog koda u ovom poglavlju uzeta je iz primjera aplikacije Mouse.vbp. Tu aplikaciju možete pronaći u direktoriju Samples.

Odgovaranje na događaje miša

Događaje MouseDown, MouseUp i MouseMove možete upotrijebiti kako bi vašim aplikacijama omogućili odgovaranje na položaj i stanje miša. Sljedeći popis ne uključuje događaje povlačenja, koji su predstavljeni u odlomku "Povlačenje i ispuštanje", kasnije u ovom poglavlju. Ove događaje miša prepoznaje većina kontrola.

događaj	opis
MouseDown	Pojavljuje se kad korisnik pritisne bilo koju tipku miša.
MouseUp	Pojavljuje se kad korisnik otpusti bilo koju tipku miša.
MouseMove	Pojavljuje se svaki put kad je pokazivač miša pomaknut na novi položaj na ekranu.

Forma može prepoznati događaj miša kad je pokazivač iznad dijela forme gdje nema kontrola. Kontrola može prepoznati događaj miša kad je pokazivač iznad kontrole.

Kad korisnik drži tipku miša pritisnutom, objekt nastavlja prepoznavati sve događaje miša sve dok korisnik ne otpusti tipku. To se događa čak i kad se pokazivač pomakne s objekta.

Tri događaja miša koriste sljedeće argumente.

argument	opis
Button	Argument polja bitova u kojem tri najmanje značajna bita dobivaju status tipki miša.
Shift	Argument polja bitova u kojem tri najmanje značajna bita dobivaju status tipki SHIFT, CTRL i ALT.
x, y	Položaj pokazivača miša, koristeći koordinatni sustav objekta koji prima događaj miša.

Argument polja bitova vraća informaciju u pojedinim bitovima, gdje svaki bit pokazuje je li određeni uvjet uključen ili isključen. Upotrebom binarnog obilježavanja, tri bita na lijevoj strani označena su kao *najznačajniji*, a tri desna bita kao *najmanje značajni*. Tehnike za programiranje s ovim argumentima opisane su u odlomcima "Otkrivanje tipki miša" i "Otkrivanje stanja SHIFT, CTRL i ALT" kasnije u ovom poglavlju.

Događaj MouseDown

Događaj MouseDown je najčešće korišten događaj od tri događaja miša. Na primjer, može biti upotrijebljen za ponovno određivanje položaja kontrola na formi tijekom izvođenja aplikacije ili za stvaranje grafičkih efekata. Događaj MouseDown se izaziva kad je pritisnuta tipka miša.

Napomena Događaji miša se koriste za prepoznavanje i odgovaranje na razna stanja miša kao odvojeni događaji i ne bi trebali biti miješani s događajima Click i DblClick. Događaj Click prepoznaje kad je tipka miša pritisnuta i otpuštena, ali samo kao jedinstvenu akciju - klik. Događaji miša se također razlikuju od događaja Click i DblClick po tome što vam omogućuju razlikovanje lijeve, desne i srednje tipke miša te tipki SHIFT, CTRL i ALT.

Korištenje događaja MouseDown s postupkom Move

Događaj MouseDown kombinira se s postupkom Move za pomicanje naredbenog gumba na drugačiji položaj na formi. Novi položaj se ustvrđuje položajem pokazivača miša. Kad korisnik klikne bilo gdje na formu (osim na kontrolu), kontrola se pomiče na položaj pokazivača.

Jedan potprogram, Form_MouseDown, izvodi ovu akciju:

```
Private Sub Form_MouseDown(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  Command1.Move X, Y
End Sub
```

Postupak Move postavlja gornji lijevi kut kontrole naredbenog gumba na položaj pokazivača miša, naznačen argumentima *x* i *y*. Možete promijeniti ovaj potprogram tako da postavlja središte kontrole na položaj pokazivača miša:

```
Private Sub Form_MouseDown(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  Command1.Move (X - Command1.Width / 2), _
  (Y - Command1.Height / 2)
End Sub
```

Korištenje događaja MouseDown s postupkom Line

Primjer aplikacije “Click-A-Line” odgovara na klik mišem tako da nacrta liniju od prethodnog položaja crtanja do novog položaja pokazivača miša. Ova aplikacija koristi događaj MouseDown i postupak Line. Korištenjem sljedeće sintakse, postupak Line će nacrtati liniju od posljednje nacrtane točke do točke (*x2*, *y2*):

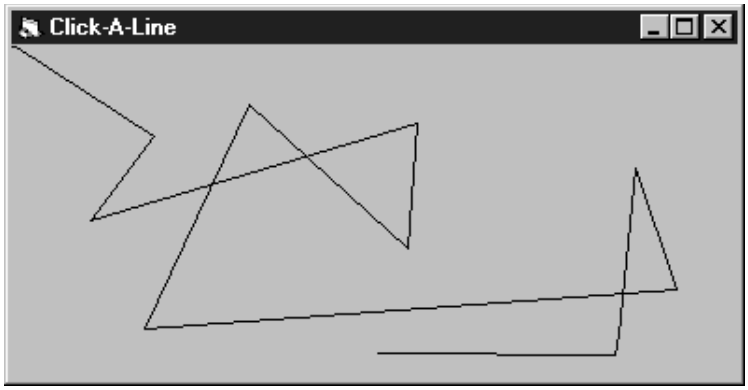
Line -(*x2*, *y2*)

Aplikacija “Click-A-Line” koristi praznu formu s jednim potprogramom, Form_MouseDown:

```
Private Sub Form_MouseDown(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  Line -(X, Y)
End Sub
```

Prva linija ima početak u gornjem lijevom kutu, koji je podrazumijevani početak. Nakon toga, uvijek kad je pritisnuta tipka miša, aplikacija crta ravnu liniju koja se proteže od kraja prethodne linije do trenutnog položaja pokazivača miša. Rezultat je niz povezanih linija, kao što je prikazano na slici 11.1.

Slika 11.1 Povezane linije crtaju se uvijek kad je pozvan događaj



MouseDown

Za više informacija Pogledajte “Događaj MouseDown” u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Događaj MouseMove

Događaj MouseMove pojavljuje se kad se pokazivač miša pomiče po ekranu. Forme i kontrole prepoznaju događaj MouseMove kad je pokazivač miša unutar njihovih okvira.

Korištenje događaja MouseMove s postupkom Line

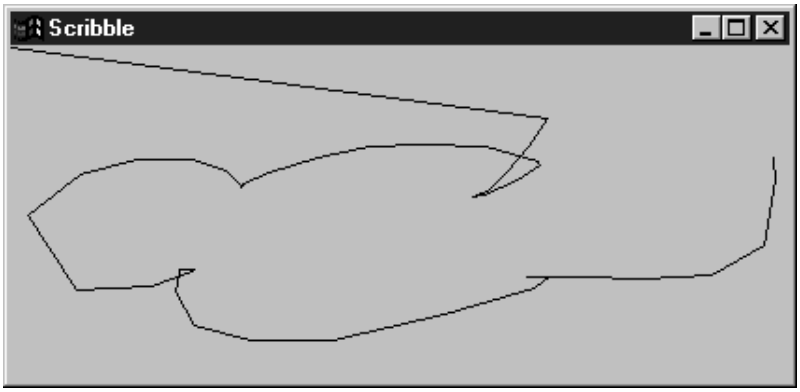
Grafički postupci mogu proizvesti vrlo različite efekte kad se koriste u potprogramu MouseMove umjesto u potprogramu MouseDown. Na primjer, u odlomku “Događaj MouseDown” ranije u ovom poglavlju, postupak Line crtao je povezane dijelove linija. U aplikaciji Scribble opisanoj u nastavku, upotrijebljen je isti postupak za potprogram Form_MouseMove kako bi proizveo neprekidnu zaobljenu liniju umjesto povezanih dijelova.

U aplikaciji Scribble, događaj MouseMove se prepoznaje uvijek kad pokazivač miša promijeni položaj. Sljedeći programski kod crta liniju između trenutnog i prethodnog položaja pokazivača.

```
Private Sub Form_MouseMove(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  Line -(X, Y)
End Sub
```

Slično potprogramu MouseDown, linija stvorena potprogramom MouseMove počinje u gornjem lijevom kutu, kao što je prikazao na slici 11.2.

Slika 11.2 Događaj MouseMove i postupak Line stvaraju jednostavnu aplikaciju za skiciranje



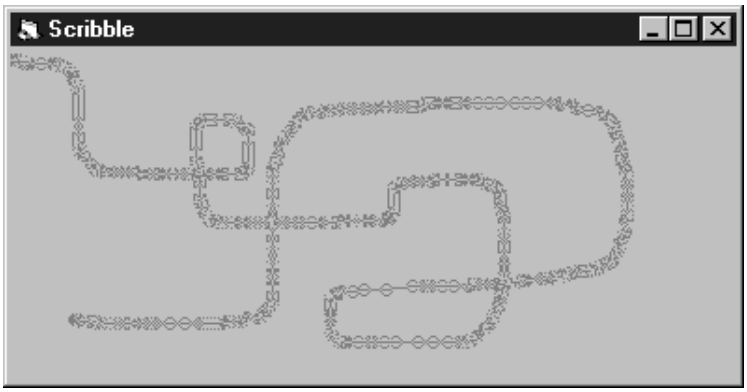
Kako radi događaj MouseMove

Koliko se puta poziva događaj MouseMove dok korisnik pomiče pokazivač miša po ekranu? Ili, recimo to drugim riječima, kad pomaknete pokazivač od vrha do dna ekrana, koliko je položaja bilo upleteno?

Visual Basic ne stvara nužno događaj MouseMove za svaki piksel preko kojeg prijeđe pokazivač miša. Operativno okruženje stvara ograničen broj poruka miša u sekundi. Kako bi vidjeli koliko često su događaji MouseMove stvarno prepoznati, možete proširiti aplikaciju Scribble sljedećim programskim kodom tako da se crta mali krug na svakom položaju gdje je prepoznat događaj MouseMove. Rezultati su prikazani na slici 11.3.

```
Private Sub Form_MouseMove(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  Line -(X, Y)
  Circle (X, Y), 50
End Sub
```

Slika 11.3 Prikaz gdje se pojavljuju događaji MouseMove



Uočite da što brže korisnik pomiče pokazivač, to se rjeđe prepoznaju događaji MouseMove između svake dvije točke. Puno međusobno bliskih krugova pokazuje da je korisnik sporo pomicao miša.

Vaša aplikacija može prepoznati puno događaja MouseMove u brzom slijedu. Zbog toga, potprogram događaja MouseMove ne bi trebao raditi ništa što zahtijeva dulje vrijeme računanja.

Za više informacija Pogledajte "Događaj MouseMove" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Događaj MouseUp

Događaj MouseUp pojavljuje se kad korisnik otpusti tipku miša. Događaj MouseUp je koristan suradnik događajima MouseDown i MouseMove. Sljedeći primjer pokazuje kako sva tri događaja mogu biti upotrijebljena zajedno.

Aplikacija Scribble je korisnija ako omogućuje povlačenje samo dok se tipka miša drži pritisnutom, a crtanje se prekida kad se otpusti tipka. Da biste to napravili, aplikacija bi trebala odgovoriti na tri akcije:

- Korisnik je pritisnuo tipku miša (MouseDown).
- Korisnik pomiče pokazivač miša (MouseMove).
- Korisnik je otpustio tipku miša (MouseUp).

Događaji MouseDown i MouseUp će aplikaciji reći da uključi i isključi crtanje. To određujete stvaranjem varijable na razini forme koja predstavlja stanje crtanja. Upišite sljedeći izraz u odjeljak Declarations modula koda forme:

```
Dim CrtajSad As Boolean
```

Varijabla CrtajSad će predstavljati dvije vrijednosti: vrijednost True će značiti "nacrtaj liniju", a vrijednost False će značiti "ne crtaj liniju".

Budući da se varijable u pravilu pokreću vrijednošću 0 (False), aplikacija počinje isključenim crtanjem. Nakon toga prva linija u potprogramima MouseDown i MouseUp uključuje ili isključuje crtanje određivanjem vrijednosti varijable CrtajSad na razini forme:

```
Private Sub Form_MouseDown(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  CrtajSad = True
  CurrentX = X
  CurrentY = Y
End Sub

Private Sub Form_MouseUp(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  CrtajSad = False
End Sub
```

Potprogram MouseMove crta liniju samo ako varijabla CrtajSad ima vrijednost True. Inače, ne poduzima nikakvu akciju:

```
Private Sub MouseMove(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  If CrtajSad Then Line -(X, Y)
End Sub
```

Svaki put kad korisnik pritisne tipku miša, izvodi se potprogram događaja MouseDown i uključuje crtanje. Nakon toga, dok korisnik drži pritisnutu tipku miša, potprogram događaja MouseMove ponavljano se izvodi kako se pokazivač miša pomiče po ekranu.

Uočite da je u postupku Line ispuštena prva krajnja točka linije, zbog čega Visual Basic počinje crtati na trenutnom položaju pokazivača miša. U pravilu, koordinate crtanja podudaraju se s posljednjom nacrtanom točkom; svojstva forme CurrentX i CurrentY se obnavljaju u potprogramu Form_MouseDown.

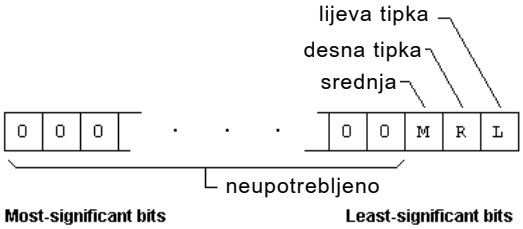
Za više informacija Pogledajte "Događaj MouseUp" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Otkrivanje tipki miša

Svoju aplikaciju možete učiniti još moćnijom ako napišete programski kod koji će različito odgovarati na događaje miša, ovisno o tome koja je tipka miša pritisnuta te jesu li pritisnute tipke SHIFT, CTRL ili ALT. Kako bi pružili takve mogućnosti, upotrijebite argumente *Button* i *Shift* s potprogramima događaja MouseDown, MouseUp i MouseMove. Tehnike za korištenje argumenta *Shift* opisane su u odlomku "Otkrivanje stanja SHIFT, CTRL i ALT" kasnije u ovom poglavlju.

Događaji MouseDown, MouseUp i MouseMove koriste argument *Button* za određivanje koja je tipka ili tipke miša pritisnuta. Argument *Button* je argument polja bitova – vrijednost u kojoj svaki bit predstavlja stanje ili uvjet. Te vrijednosti su predstavljene kao cijeli brojevi. Tri najmanje značajna (najniža) bita predstavljaju lijevu, desnu i srednju tipku miša, kao što je prikazano na slici 11.4.

Slika 11.4 Kako bitovi predstavljaju stanje miša



Podrazumijevana vrijednost svakog bita je 0 (False). Ako ni jedna tipka miša nije pritisnuta, binarna vrijednost ova tri bita je 000. Ako pritisnete lijevu tipku, binarna vrijednost, ili uzorak, mijenja se u 001. Vrijednost bita koji predstavlja lijevu tipku mijenja se s 0 (False) na 1 (True).

Argument *Button* koristi ili decimalnu vrijednost ili konstantu koja predstavlja ove binarne uzorke. Sljedeća tablica ispisuje binarne vrijednosti za bitove, jednake decimalne vrijednosti i konstante Visual Basica:

binarna vrijednost	decimalna vrijednost	konstanta	značenje
001	1	vbLeftButton	Pritisnuta je lijeva tipka.
010	2	vbRightButton	Pritisnuta je desna tipka.
100	4	vbMiddleButton	Pritisnuta je srednja tipka.

Napomena Visual Basic pruža konstante koje predstavljaju binarne vrijednosti argumenata *Button* i *Shift*. Te konstante mogu se koristiti izmjenjivo s njihovim odgovarajućim decimalnim vrijednostima. Međutim, sve vrijednosti nemaju odgovarajuće konstante. Vrijednosti za neke kombinacije argumenata *Button* i/ili *Shift* izvedene su jednostavnim zbrajanjem decimalnih vrijednosti.

Srednja tipka je dodijeljena vrijednosti 4. Istovremeni pritisak lijeve i desne tipke miša proizvodi jednoznamenkastu vrijednost od 3 (1 + 2). Kod miša s tri tipke, istovremeni pritisak sve tri tipke proizvest će decimalnu vrijednost od 7 (4 + 2 + 1). Sljedeća tablica ispisuje preostale vrijednosti tipaka izvedene iz mogućih kombinacija tipki:

binarna vrijednost	decimalna vrijednost	konstanta	značenje
000	0		Niti jedna tipka nije pritisnuta.
011	3	vbLeftButton + vbRightButton	Pritisnute su lijeva i desna tipka.
101	5	vbLeftButton + vbMiddleButton	Pritisnute su lijeva i srednja tipka.
110	6	vbRightButton + vbMiddleButton	Pritisnute su desna i srednja tipka.
111	7	vbRightButton + vbMiddleButton + vbLeftButton	Pritisnute su sve tri tipke.

Korištenje argumenta Button s događajima MouseDown i MouseUp

Argument *Button* upotrebljavate s događajem MouseDown za otkrivanje koja je tipka miša pritisnuta te s događajem MouseUp za otkrivanje koja je tipka miša otpuštena. Budući da za svaki događaj može biti postavljen samo jedan bit, ne možete ispitati jesu li dvije ili više tipaka korištene u isto vrijeme. Drugim riječima, događaji MouseDown i MouseUp prepoznaju samo jedan pritisak na tipku u određenom trenutku.

> **Napomena** Suprotno tome, događaj MouseMove možete upotrijebiti za ispitivanje jesu li dvije ili više tipki miša istovremeno pritisnute. Ovaj događaj možete također upotrijebiti za ispitivanje je li pritisnuta određena tipka miša, neovisno o tome je li u istom trenutku pritisnuta druga tipka ili nije. Za više informacija, pogledajte "Korištenje argumenta Button s događajem MouseMove", kasnije u ovom poglavlju.

Možete odrediti koja tipka uzrokuje događaj MouseDown ili MouseUp jednostavnim kodom. Sljedeći potprogram ispituje je li argument *Button* jednak 1, 2 ili 4:

```
Private Sub Form_MouseDown(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  If Button = 1 Then Print "Pritisnuli ste _
    lijevu tipku miša."
  If Button = 2 Then Print "Pritisnuli ste _
    desnu tipku miša."
  If Button = 4 Then Print "Pritisnuli ste _
    srednju tipku miša."
End Sub
```

Ako korisnik pritisne više od jedne tipke, Visual Basic će takvu akciju protumačiti kao dva ili više odvojenih događaja MouseDown. Visual Basic će postaviti bit za prvu pritisnutu tipku, ispisati poruku za tu tipku, pa napraviti isto za iduću tipku. Slično tome, Visual Basic tumači otpuštanje dvije ili više tipki kao odvojene događaje MouseUp.

Sljedeći potprogram ispisuje poruku kad se otpusti pritisnuta tipka:

```
Private Sub Form_MouseUp(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  If Button = 1 Then Print "Otpustili ste _
    lijevu tipku miša."
  If Button = 2 Then Print " Otpustili ste _
    desnu tipku miša."
  If Button = 4 Then Print " Otpustili ste _
    srednju tipku miša."
End Sub
```

Korištenje argumenta Button s događajem MouseMove

Za događaj MouseMove, argument *Button* pokazuje cjelovito stanje tipki miša – ne samo koja je tipka uzrokovala događaj, kao kod događaja MouseDown i MouseUp. Ova dodatna informacija je pružena zato što mogu biti postavljeni svi, neki ili ni jedan od bitova. To se uspoređuje s samo jednim bitom po događaju u potprogramima MouseDown i MouseUp.

Ispitivanje pojedine tipke

Ako ispitujete događaj MouseMove za jednakost s vrijednošću 001 (decimalno 1), ispitujete je li pritisnuta *samo* lijeva tipka miša dok se miš pomiče. Ako je pritisnuta druga tipka zajedno s lijevom tipkom, sljedeći kod neće ispisati ništa:

```
Private Sub Form_MouseMove(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  If Button = 1 Then Print “Pritiskate _
    samo lijevu tipku miša.”
End Sub
```

Kako bi ispitali koja se tipka drži pritisnutom, upotrijebite operator And. Sljedeći kod ispisuje poruku za svaku pritisnutu tipku, neovisno o tome je li pritisnuta druga tipka miša:

```
Private Sub Form_MouseMove(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  If Button And 1 Then Print “Pritiskate _
    lijevu tipku miša.”
  If Button And 2 Then Print “Pritiskate _
    desnu tipku miša.”
End Sub
```

Istovremeno pritiskanje obje tipke miša ispisat će obje poruke na formu. Događaj MouseMove prepoznaje višestruka stanja tipki miša.

Ispitivanje više tipaka

U većini slučajeva, upotrebljavate događaj MouseMove za izdvajanje koja je tipka ili tipke pritisnuta.

Nadogradnjom na prethodne primjere, možete upotrijebiti izraz If...Then...Else za određivanje jesu li pritisnute lijeva, desna ili obje tipke miša. Sljedeći primjer ispituje tri stanja tipki (pritisnuta lijeva tipka, pritisnuta desna tipka i pritisnute obje tipke) i ispisuje odgovarajuću poruku.

Dodajte sljedeći kod u događaj forme MouseMove:

```
Private Sub Form_MouseMove(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  If Button = 1 Then
    Print “Pritiskate lijevu tipku miša.”
  ElseIf Button = 2 Then
    Print “Pritiskate desnu tipku miša.”
  ElseIf Button = 3 Then
    Print “Pritiskate obje tipke miša.”
  End If
End Sub
```

Mogli bi također upotrijebiti operator And s izrazom Select Case za otkrivanje stanja *Button* i *Shift*. Operator And kombiniran s izrazom Select Case izdvaja moguća stanja tipki miša s tri tipke i zatim ispisuje odgovarajuću poruku.

Stvorite varijablu s imenom IspitivanjeTipke u odjeljku Declarations forme:

```
Dim IspitivanjeTipke As Integer
Dodajte sljede}i kod u doga|aj MouseMove forme:
Private Sub Form_MouseMove(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
IspitivanjeTipke = Button And 7
  Select Case IspitivanjeTipke
    Case 1 ‘ ili vbLeftButton
      Print “Pritiskate lijevu tipku miša.”
    Case 2 ‘ ili vbRightButton
      Print “Pritiskate desnu tipku miša.”
    Case 4 ‘ ili vbMiddleButton
      Print “Pritiskate srednju tipku miša.”
    Case 7
      Print “Pritiskate sve tri tipke miša.”
  End Select
End Sub
```

Korištenje argumenta Button za poboljšavanje grafičkih aplikacija s mišem

Argument *Button* možete upotrijebiti za poboljšavanje aplikacije Scribble opisane u odlomku “Događaj MouseMove” ranije u ovom poglavlju. Kao dodatak crtanju neprekinute linije kad je pritisnuta lijeva tipka miša i prestanku crtanja kad je tipka otpuštena, aplikacija može crtati ravnu liniju od posljednje nacrtane točke do pokazivača miša kad korisnik pritisne desnu tipku miša.

Kad pišete programski kod, često je korisno označiti svaki važan događaj te željeni odgovor. Sljedeća tri važna događaja su događaji miša:

- Form_MouseDown: Ovaj događaj poduzima različite akcije ovisno o stanju tipki miša: ako je pritisnuta lijeva tipka, postavlja varijablu `CrtajSad` na True i obnavlja koordinate crtanja; ako je pritisnuta desna tipka miša, crta liniju.
- Form_MouseUp: Ako je otpuštena lijeva tipka, postavlja varijablu `CrtajSad` na False.
- Form_MouseMove: Ako je varijabla `CrtajSad` postavljena na True, crta liniju.

Varijabla `CrtajSad` je određena u odjeljku Declarations forme:

```
Dim CrtajSad As Boolean
```

Potprogram MouseDown mora izvesti različite akcije, ovisno o tome je li događaj uzrokovala lijeva ili desna tipka miša:

```
Private Sub Form_MouseDown(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  If Button = vbLeftButton Then
    CrtajSad = True
    CurrentX = X
    CurrentY = Y
  ElseIf Button = vbRightButton Then
    Line -(X, Y)
  End If
End Sub
```

Sljedeći potprogram MouseUp isključuje crtanje samo ako je otpuštena lijeva tipka miša:

```
Private Sub Form_MouseUp(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  If Button = vbLeftButton Then CrtajSad = False
End Sub
```

Uočite da unutar potprograma MouseUp bit postavljen na 1 (vbLeftButton) ukazuje da je pripadajuća tipka miša otpuštena pa je crtanje isključeno.

Sljedeći potprogram MouseMove jednak je onom iz verzije aplikacije Scribble kojeg možete pronaći u odlomku “Događaj MouseMove” ranije u ovom poglavlju.

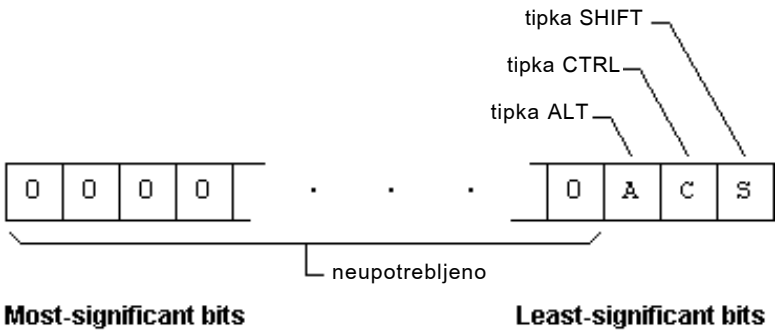
```
Private Sub Form_MouseMove(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  If CrtajSad Then Line -(X, Y)
End Sub
```

Otkrivanje stanja SHIFT, CTRL i ALT

Događaji miša i tipkovnice upotrebljavaju argument Shift za određivanje jesu li pritisnute tipke SHIFT, CTRL i ALT, te ako jesu, u kojoj kombinaciji. Ako je pritisnuta tipka SHIFT, argument Shift je 1; ako je pritisnuta tipka CTRL, argument Shift je 2; ako je pritisnuta tipka ALT, argument Shift je 4. Kako bi ustvrdili kombinacije tih tipki, upotrijebite zbroj njihovih vrijednosti. Na primjer, ako su pritisnute tipke SHIFT i ALT, argument Shift ima vrijednost 5 (1 + 4).

Tri najmanje značajna bita u argumentu Shift odgovaraju stanju tipki SHIFT, CTRL i ALT, kao što je prikazano na slici 11.5.

Slika 11.5 Kako bitovi predstavljaju stanje tipki SHIFT, CTRL i ALT



Svaki ili svi bitovi u argumentu *Shift* mogu biti postavljeni, ovisno o stanju tipki SHIFT, CTRL i ALT. Te vrijednosti i konstante ispisane su u sljedećoj tablici:

binarna vrijednost	decimalna vrijednost	konstanta	značenje
001	1	vbShiftMask	Pritisnuta je tipka SHIFT.
010	2	vbCtrlMask	Pritisnuta je tipka CTRL.
100	4	vbAltMask	Pritisnuta je tipka ALT.
011	3	vbShiftMask + vbCtrlMask	Pritisnute su tipke SHIFT i CTRL.
101	5	vbShiftMask + vbAltMask	Pritisnute su tipke SHIFT i ALT.
110	6	vbCtrlMask + vbAltMask	Pritisnute su tipke CTRL i ALT.
111	7	vbCtrlMask + vbAltMask + vbShiftMask	Pritisnute su tipke SHIFT, CTRL i ALT.

Kao i kod argumenta *Button* događaja miša, možete upotrijebiti izraz If...Then...Else ili operator And u kombinaciji s izrazom Select Case za otkrivanje jesu li pritisnute tipke SHIFT, CTRL i ALT, te ako jesu, u kojoj kombinaciji.

Otvorite nov projekt i dodajte varijablu ShiftTest u odjeljak Declarations forme:

```
Dim ShiftTest As Integer
```

Dodajte sljedeći kod u događaj MouseDown forme:

```
Private Sub Form_MouseDown(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  ShiftTest = Shift And 7
  Select Case ShiftTest
    Case 1        ‘ ili vbShiftMask
      Print “Pritisnuli ste tipku SHIFT.”
    Case 2        ‘ ili vbCtrlMask
      Print “Pritisnuli ste tipku CTRL.”
    Case 4        ‘ ili vbAltMask
      Print “Pritisnuli ste tipku ALT.”
    Case 3
      Print “Pritisnuli ste tipke SHIFT i CTRL.”
    Case 5
      Print “Pritisnuli ste tipke SHIFT i ALT.”
    Case 6
      Print “Pritisnuli ste tipke CTRL i ALT.”
    Case 7
      Print “Pritisnuli ste tipke SHIFT, CTRL i ALT.”
    End Select
End Sub
```

Povlačenje i ispuštanje

Kad oblikujete aplikacije u Visual Basicu, često povlačite kontrole po formi. Osobine povlačenja i ispuštanja u Visual Basicu omogućuju vam pružanje te sposobnosti korisniku tijekom izvođenja aplikacije. Akcija držanja pritisnute tipke miša i pomicanja kontrole naziva se *povlačenje* (*dragging*), a akcija otpuštanja tipke *ispuštanje* (*dropping*).

Napomena Povlačenje kontrole tijekom izvođenja ne mijenja automatski njezin položaj – promjenu položaja morate sami programirati, kao što je opisano u odlomku “Promjena položaja kontrole”, kasnije u ovom poglavlju. Često se povlačenje koristi samo za naznačivanje da treba biti izvedena neka akcija; kontrola zadržava svoj izvoran položaj nakon što korisnik otpusti tipku miša.

Korištenjem sljedećih svojstava, događaja i postupaka povlačenja i ispuštanja, možete odrediti značenje operacije povlačenja te kako će povlačenje biti pokrenuto (ako uopće hoće) za danu kontrolu.

vrsta	stavka	opis
Svojstva	DragMode DragIcon	Omogućuje automatsko ili ručno povlačenje kontrole. Određuje koja će ikona biti prikazana kad se povlači kontrola.
Događaji	DragDrop DragOver	Prepoznaje kad je kontrola ispuštena na objekt. Prepoznaje kad je kontrola prevučena iznad objekta.
Postupak	Drag	Započinje ili prekida ručno povlačenje.

Sve kontrole osim izbornika, mjerača vremena, linija i likova podržavaju svojstva DragMode i DragIcon te postupak Drag. Forme prepoznaju događaje DragDrop i DragOver, ali ne podržavaju svojstva DragMode i DragIcon niti postupak Drag.

Napomena Kontrole mogu biti povlačene samo kad nemaju fokus. Kako bi spriječili kontrolu da dobije fokus, postavite njezino svojstvo TabStop na False.

Omogućavanje automatskog moda povlačenja

Kako bi korisniku omogućili povlačenje kontrole, postavite njezino svojstvo DragMode na 1 – Automatic.

Kad povlačenje postavite na Automatic, povlačenje je uvijek "uključeno". Za veći nadzor nad operacijama povlačenja, upotrijebite postavku 0 – Manual opisanu u odlomku "Nadzor početka i prekida povlačenja" kasnije u ovom poglavlju.

Napomena Dok je u tijeku automatska operacija povlačenja, kontrola koja se povlači ne prepoznaje druge događaje miša.

Mijenjanje ikone povlačenja

Kad povlači kontrolu, Visual Basic koristi siv obris kontrole kao podrazumijevanu ikonu povlačenja. Ovaj obris možete zamijeniti drugim slikama određivanjem svojstva DragIcon. To svojstvo sadržava objekt tipa Picture koji odgovara grafičkoj slici.

Najlakši način postavljanja svojstva DragIcon je korištenje prozora s svojstvima. Odaberite svojstvo DragIcon, te kliknite gumb Properties za odabir datoteke koja sadrži grafičku sliku iz dijaloškog okvira Load Icon.

Svojstvu DragIcon možete dodijeliti ikone iz biblioteke Icon Library uključene s Visual Basicom (ikone se nalaze u direktoriju \Program files\Microsoft Visual Basic\Icons). Možete također stvoriti svoje vlastite ikone povlačenja s nekom grafičkom aplikacijom.

Tijekom izvođenja aplikacije, ikonu povlačenja možete odabrati dodjeljivanjem svojstva DragIcon jedne kontrole isto svojstvu druge kontrole:

```
Set Image1.DragIcon = Image2.DragIcon
```

Svojstvo DragIcon možete također postaviti tijekom izvođenja dodjeljivanjem svojstva Picture jedne kontrole svojstvu DragIcon druge kontrole:

```
Set Image1.DragIcon = Image3.Picture
```

Ili, možete upotrijebiti funkciju LoadPicture:

```
Set Image1.DragIcon = LoadPicture("c:\Program _
files\Microsoft Visual Basic\Icons _
\Computer\Disk04.ico")
```

Za više informacija Za informacije o svojstvu Picture i funkciji LoadPicture, pogledajte 12. poglavlje "Rad s tekstom i grafikom". Također pogledajte odlomke "Svojstvo Picture" i "Funkcija LoadPicture" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Odgovaranje kad korisnik ispusti objekt

Kad korisnik otpusti tipku miša nakon povlačenja kontrole, Visual Basic stvara događaj DragDrop. Na taj događaj možete odgovoriti na puno načina. Zapamtite da se kontrola ne pomiče automatski na novi položaj, ali možete napisati programski kod za postavljanje kontrole na nov položaj (pokazan posljednjim položajem sivog obrisa). Pogledajte "Promjena položaja kontrole", kasnije u ovom poglavlju za više informacija.

Važna su dva izraza kad se raspravlja o operacijama povlačenja i ispuštanja; *izvor* (*source*) i *cilj* (*target*).

izraz	značenje
Izvor (Source)	Kontrola koja je povučena. Ta kontrola može biti bilo koji objekt osim izbornika, mjerača vremena, linije ili lika.
Cilj (Target)	Objekt na koji korisnik ispušta kontrolu. Objekt, koji može biti forma ili kontrola, prepoznaje događaj DragDrop.

Kontrola postaje cilj ako je položaj pokazivača miša unutar njezinih granica kad je otpuštena tipka miša. Forma je cilj ako je pokazivač na praznom dijelu forme.

Događaj DragDrop pruža tri argumenta: *source*, *x* i *y*. Argument *source* je pokazivač na kontrolu koja je ispuštena na cilj.

Budući da je argument *source* određen s As Control, upotrebljavate ga kao i neku kontrolu – možete ukazivati na njegova svojstva i pozivati njegove postupke.

Sljedeći primjer pokazuje kako surađuju izvor i cilj. Izvor je kontrola slike s svojstvom Picture postavljenim da učita probnu datoteku ikone koja predstavlja nekoliko mapa datoteka. Njezino svojstvo DragMode je postavljeno na 1 – Automatic, a njezino svojstvo DragIcon na probnu datoteku ikone povlačenja i ispuštanja. Cilj, također kontrola slike, sadrži sliku otvorenog ormarića.

Dodajte sljedeći potprogram događaju DragDrop druge kontrole slike:

```
Private Sub Image2_DragDrop(Source As Control, _
X As Single, Y As Single)
  Source.Visible = False
  Image2.Picture = LoadPicture("c:\Program _
    Files\Microsoft Visual _
    Basic\Icons\Office\Files03a.ico")
End Sub
```

Povlačenje i ispuštanje kontrole Image1 na kontrolu Image2 uzrokuje nestanak kontrole Image1 i promjenu slike u kontroli Image2 u zatvoreni ormarić. Korištenjem argumenta *source*, svojstvo Visible kontrole Image1 promijenjeno je u False.

> **Napomena** Argument *source* trebate pažljivo koristiti. Iako znate da on uvijek upućuje na kontrolu, ne znate nužno na koji tip kontrole. Na primjer, ako je kontrola okvir s tekstom i pokušate uputiti na Source.Value, rezultat će biti pogreška tijekom izvođenja zato što okviri s tekstom nemaju svojstvo Value.

Možete upotrijebiti Izraz If...Then...Else s ključnom riječi TypeOf za utvrđivanje koji je tip kontrole ispušten.

Za više informacija Pogledajte "If...Then...Else" u priručniku *Microsoft Visual Basic 6.0 Language Reference*, te 9. poglavlje "Programiranje objektima".

Nadzor početka i prekida povlačenja

Visual Basic ima postavku Manual za svojstvo DragMode koja vam daje veći nadzor od postavke Automatic. Postavka Manual omogućuje vam da odredite kad kontrola može i ne može biti povlačena (kad je svojstvo DragMode postavljeno na Automatic, možete uvijek povlačiti kontrolu sve dok se ne promijeni ta postavka).

Na primjer, možete poželjeti omogućiti povlačenje u odgovoru na događaje MouseDown i MouseUp, ili u odgovoru na akciju tipkovnicom ili naredbu izbornika. Postavka Manual vam također omogućuje prepoznavanje događaja MouseDown prije početka povlačenja, tako da možete zapamtiti položaj pokazivača miša.

Kako bi iz programskog koda omogućili povlačenje, ostavite svojstvo DragMode s standardnom postavkom (0 – Manual). Zatim upotrijebite postupak Drag uvijek kad želite početi ili prekinuti povlačenje objekta. Upotrijebite sljedeće konstante Visual Basica za određivanje akcije postupka Drag argumentom *akcija*.

konstanta	vrijednost	značenje
vbCancel	0	Odustajanje od operacije povlačenja.
vbBeginDrag	1	Početak operacije povlačenja.
vbEndDrag	2	Kraj operacije povlačenja.

Sintaksa postupka Drag je sljedeća:

[*objekt.*] **Drag** *akcija*

Ako je argument *akcija* postavljen na vbBeginDrag, postupak Drag započinje povlačenje kontrole. Ako je argument *akcija* postavljen na vbEndDrag, kontrola se ispušta, uzrokujući događaj DragDrop. Ako je argument *akcija* postavljen na vbCancel, odustaje se od povlačenja. Učinak je sličan korištenju vrijednosti vbEndDrag, osim što se ne pojavljuje događaj DragDrop.

Nadogradnjom na primjer dan u odlomku "Odgovaranje kad korisnik ispusti objekt", ranije u ovom poglavlju, kontroli Image1 možete dodati događaj MouseDown koji pokazuje postupak Drag. Postavite svojstvo DragMode kontrole Image1 na 0 – Manual, te dodajte sljedeći potprogram:

```
Private Sub Image1_MouseDown(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  Image1.Drag vbBeginDrag
  Set Image1.DragIcon = LoadPicture("c:\Program _
    files\Microsoft Visual _
    Basic\Icons\DragDrop\Dragfldr.ico")
End Sub
```

Dodavanje potprograma događaja DragOver kontroli Image2 omogućuje vam završavanje povlačenja kad izvor uđe u cilj. Ovaj primjer zatvara ormarić s datotekama kad kontrola Image1 prijeđe preko kontrole Image2.

```
Private Sub Image2_DragOver(Source As Control, _
X As Single, Y As Single, State As Integer)
  Source.Drag vbEndDrag
  Source.Visible = False
  Image2.Picture = LoadPicture("c:\Program _
    files\Microsoft Visual _
    Basic\Icons\Office\Files03a.ico")
End Sub
```

Dodavanje treće kontrole slike formi prikazuje odustajanje od operacije povlačenja. U ovom primjeru svojstvo Picture kontrole Image3 sadrži sliku kante za otpatke. Korištenjem događaja DragOver i argumenta *source*, povlačenje datoteka preko kontrole Image3 poništava operaciju povlačenja.

```
Private Sub Image3_DragOver(Source As Control, _
X As Single, Y As Single, State As Integer)
  Source.Drag vbCancel
End Sub
```

Promjena položaja kontrole

Možete trebati promijeniti položaj izvorne kontrole nakon što korisnik otpusti tipku miša. Kako bi pomaknuli kontrolu na nov položaj miša, upotrijebite postupak Move s bilo kojom kontrolom koja je bila omogućena za pomicanje.

Kontroli možete promijeniti položaj kad je povučena i ispuštena na bilo koje mjesto na formi koje nije zauzeto drugom kontrolom. Kako bi to prikazali, započnite novi projekt Visual Basica, dodajte kontrolu slike na formu i dodijelite joj bilo koju ikonu ili sliku postavljanjem svojstva Picture, te zatim promijenite svojstvo DragMode kontrole slike na 1 – Automatic.

Dodajte sljedeći potprogram događaju DragDrop forme:

```
Private Sub Form_DragDrop(Source As Control, _
X As Single, Y As Single)
  Source.Move X, Y
End Sub
```

Ovaj programski kod možda neće precizno proizvesti učinak koji želite, zato što je gornji lijevi kut kontrole postavljen na položaj pokazivača miša. Sljedeći kod postavlja središte kontrole na položaj pokazivača miša:

```
Private Sub Form_DragDrop(Source As Control, _
X As Single, Y As Single)
  Source.Move (X - Source.Width / 2), _
  (Y - Source.Height / 2)
End Sub
```

Ovaj programski kod najbolje radi kad je svojstvo DragIcon postavljeno na vrijednost drugačiju od standardne (sivi pravokutnik). Kad se koristi sivi pravokutnik, korisnik obično želi da se kontrola precizno pomakne na krajnji položaj sivog pravokutnika. Kako bi to napravili, zapamtite početni položaj pokazivača miša unutar izvorne kontrole. Zatim upotrijebite taj položaj kao pomak kad se kontrola pomakne.

Kako zapamtiti početni položaj pokazivača miša

1. Odredite ručno povlačenje kontrole.
2. Odredite dvije varijable na razini forme, DragX i DragY.
3. Uključite povlačenje kad se pojavi događaj MouseDown.
4. Spremite vrijednosti *x* i *y* u varijable na razini forme kod tog događaja.

Sljedeći primjer pokazuje kako uzrokovati micanje povlačenjem za kontrolu slike s imenom Image1. Svojstvo DragMode kontrole treba biti postavljeno na 0 – Manual tijekom izrade aplikacije. Odjeljak Declarations sadrži varijable na razini forme s imenima DragX i DragY, koje pamte početni položaj pokazivača miša unutar kontrole Image:

```
Dim DragX As Single, DragY As Single
```

Potprogrami MouseDown i MouseUp kontrole uključuju povlačenje, odnosno ispuštaju kontrolu. Kao dodatak, potprogram MouseDown pamti položaj pokazivača miša unutar kontrole u trenutku kad počne povlačenje:

```
Private Sub Image1_MouseDown(Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  Image1.Drag 1
  DragX = X
  DragY = Y
End Sub
```

Potprogram Form_DragDrop zapravo pomiče kontrolu. Za pojednostavljivanje ovog primjera, pretpostavimo da je kontrola Image1 jedina kontrola na formi. Zbog toga cilj može biti jedino sama forma. Potprogram Form_DragDrop ponovno postavlja kontrolu, koristeći vrijednosti varijabli DragX i DragY kao pomake:

```
Private Sub Form_DragDrop(Source As Control, _
X As Single, Y As Single)
  Source.Move (X - DragX), (Y - DragY)
End Sub
```

Uočite da ovaj primjer pretpostavlja da kontrola Image1 i forma koriste iste jedinice u svojim koordinatnim sustavima. Ako ne koriste, tad ćete trebati pretvarati jedne mjerne jedinice u druge.

Za više informacija Za informacije o koordinatnim sustavima, pogledajte 12. poglavlje “Rad s tekstom i grafikom”, te odlomak “Svojstvo ScaleMode” u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Povlačenje i ispuštanje tipa OLE

Jedna od najmoćnijih i najkorisnijih osobina koje možete dodati svojim Visual Basic aplikacijama je sposobnost povlačenja teksta ili grafike iz jedne kontrole u drugu, ili iz kontrole u drugu Windows aplikaciju, i obratno. Povlačenje i ispuštanje tipa OLE vam omogućuje dodavanje te djelotvornosti vašim aplikacijama.

S povlačenjem i ispuštanjem tipa OLE, ne povlačite jednu kontrolu na drugu kontrolu kako bi pozvali neki kod (kao s povlačenjem i ispuštanjem raspravljenim ranije u ovom poglavlju); pomičete *podatke* iz jedne kontrole ili aplikacije u drugu kontrolu ili aplikaciju. Na primjer, korisnik može odabrati i povući niz ćelija u Excelu, te ispustiti taj niz ćelija u kontrolu mreže povezane s podacima u vašoj aplikaciji.

Gotovo sve kontrole Visual Basica podržavaju povlačenje i ispuštanje tipa OLE do određene granice. Neke standardne i ActiveX kontrole (koje su pružene u Professional i Enterprise verzijama Visual Basica) pružaju automatsku podršku povlačenju i ispuštanju tipa OLE, što znači da nije potrebno pisati kod za povlačenje iz ili ispuštanje u kontrolu i tu su uključene:

Mreža podataka (DataGrid)	Okvir za sliku (PictureBox)	Okvir za proširen tekst (Rich Text Box)
Kontrola slike (Image)	Okvir s tekstom (TextBox)	Maskiran okvir za editiranje (Masked Edit Box)

Kako bi za ove kontrole omogućili automatsko OLE povlačenje i ispuštanje, postavite svojstva OLEDragMode i OLEDropMode na Automatic.

Neke kontrole pružaju samo automatsku podršku operaciji OLE povlačenja. Kako bi omogućili automatsko povlačenje iz ovih kontrola, postavite njihovo svojstvo OLEDragMode na Automatic.

Kombinirani okvir (ComboBox)	Popis podataka (DBList)	Okvir s popisom datoteka (FileListBox)
Kombinirani okvir za podatke (DBCombo Box)	Okvir s popisom direktorija (DirectoryListBox)	Okvir s popisom (ListBox)
Pregled stabla (TreeView)	Pregled liste (ListView)	

Neke kontrole podržavaju samo događaje OLE povlačenja i ispuštanja, što znači da ih možete programirati tako da djeluju ili kao izvor ili kao cilj operacija OLE povlačenja i ispuštanja.

Kontrolna kućica (CheckBox)	Kontrola okvira (Frame)	Gumb izbora (OptionButton)
Naredbeni gumb (CommandButton)	Kontrola natpisa (Label)	Okvir s popisom pogona (DriveListBox)
Kontrola podataka (Data)		

Napomena Kako bi odredili podržavaju li druge ActiveX kontrole OLE povlačenje i ispuštanje, učitajte kontrolu u Visual Basic i provjerite postojanje svojstava OLEDragMode i OLEDropMode, ili postojanje postupka OLEDrag (kontrola

koja nema automatsku podršku OLE povlačenju neće imati svojstvo OLEDragMode, ali će imati postupak OLEDrag ako podržava OLE povlačenje kroz programski kod).

Napomena Forme, MDI forme, objekti tipa Document, korisničke kontrole i stranice svojstava sadrže svojstvo OLEDropMode i pružaju podršku samo ručnom povlačenju i ispuštanju.

Korištenjem sljedećih svojstava, događaja i postupaka OLE povlačenja i ispuštanja, možete odrediti kako će dana kontrola odgovoriti na povlačenje i ispuštanje.

vrsta	stavka	opis
Svojstva	OLEDragMode	Omogućuje automatsko ili ručno povlačenje kontrole (ako kontrola podržava ručno, ali ne i automatsko OLE povlačenje, neće imati ovo svojstvo, ali će podržavati postupak OLEDrag i događaje OLE povlačenja i ispuštanja).
	OLEDropMode	Određuje kako će kontrola odgovoriti na ispuštanje.
Događaji	OLEDragDrop	Prepoznaje kad je izvorni objekt ispušten na kontrolu.
	OLEDragOver	Prepoznaje kad je izvorni objekt prevučen iznad kontrole.
	OLEGiveFeedback	Pruža korisničku ikonu povlačenja kao povratnu informaciju korisniku, temeljenu na izvornom objektu.
	OLEStartDrag	Određuje koje oblike podataka i učinke povlačenja (kopiranje, pomicanje ili odbacivanje podataka) podržava izvor kad je započeto povlačenje.
	OLESetData	Pruža podatke kad je ispušten izvorni objekt.
	OLECompleteDrag	Obavještava izvor akcije da je obavljen kad je objekt ispušten na cilj.
Postupak	OLEDrag	Započinje ručno povlačenje.

Automatsko nasuprot ručnog povlačenja i ispuštanja

Korisno je razmišljati o ostvarivanju OLE povlačenja i ispuštanja kao ili *automatskom* ili *ručnom*.

Automatsko povlačenje i ispuštanje znači da, na primjer, možete povući tekst iz jedne kontrole okvira s tekstom u drugu jednostavnim postavljanjem svojstava OLEDragMode i OLEDropMode tih kontrola na Automatic; ne trebate pisati nikakav programski kod za odgovaranje na bilo koji od događaja OLE povlačenja i ispuštanja. Kad povučete niz ćelija iz Excela u dokument Worda, izveli ste operaciju automatskog povlačenja i ispuštanja. Ovisno o tome kako dana kontrola ili aplikacija podržava OLE povlačenje i ispuštanje te koji se tip podataka povlači, automatsko povlačenje i ispuštanje podataka može biti najbolji i najjednostavniji postupak.

Ručno povlačenje i ispuštanje znači da ste odabrali (ili ste bili prisiljeni na to) ručno rukovanje jednim ili više događaja OLE povlačenja i ispuštanja. Ručno ostvarivanje OLE povlačenja i ispuštanja može biti bolji postupak kad želite postići veći nadzor nad

svakim korakom u procesu, kako bi korisniku omogućili prilagođenu vidljivu povratnu informaciju, ili kako bi stvorili svoj vlastiti oblik podataka. Ručno ostvarivanje je jedini izbor kad kontrola ne podržava automatsko povlačenje i ispuštanje.

Također je korisno odrediti opći model operacija OLE povlačenja i ispuštanja. U operacijama povlačenja i ispuštanja, objekt iz kojeg se podaci povlače označuje se kao *izvor* (*source*). Objekt u kojeg se podaci ispuštaju označuje se kao *cilj* (*target*). Visual Basic pruža svojstva, događaje i postupke za nadzor i odgovaranje na akcije koje utječu i na izvor i na cilj. Također je korisno prepoznati da izvor i cilj mogu biti u različitim aplikacijama, u istoj aplikaciji ili čak u istoj kontroli. Ovisno o scenariju, možete trebati napisati programski kod za ili izvor ili cilj, ili za oboje.

Omogućavanje automatskog OLE povlačenja i ispuštanja

Ako vaše kontrole podržavaju automatsko povlačenje i ispuštanje, možete povući podatke iz i/ili ispustiti podatke u kontrolu Visual Basica postavljanjem svojstava OLEDragMode i/ili OLEDropMode te kontrole na Automatic. Na primjer, možete trebati povući tekst iz kontrole okvira s tekstom u dokument Worda za Windowse, ili omogućiti kontroli okvira s tekstom da prihvati podatke povučene iz dokumenta Worda za Windowse.

Kako bi omogućili povlačenje iz kontrole okvira s tekstom, postavite njezino svojstvo OLEDragMode na Automatic. Tijekom izvođenja aplikacije, možete odabrati tekst upisan u kontrolu okvira s tekstom i odvući ga u otvoren dokument Worda za Windowse.

Kad povučete tekst iz kontrole okvira s tekstom u dokument Worda za Windowse, taj tekst se, u pravilu, radije prebacuje, a ne kopira u dokument. Ako držite pritisnutom tipku CTRL dok ispuštate tekst, on će biti radije kopiran, a ne prebačen. To je podrazumijevano ponašanje za sve objekte ili aplikacije koje podržavaju OLE povlačenje i ispuštanje. Kako bi ograničili tu operaciju dopuštajući podacima da se samo prebacuju ili samo kopiraju, trebate promijeniti automatsko ponašanje korištenjem tehnika ručnog povlačenja i ispuštanja. Za više informacija, pogledajte “Korištenje miša i tipkovnice za promjenu učinaka ispuštanja i povratne informacije korisniku”, kasnije u ovom poglavlju.

Kako bi kontroli okvira s tekstom omogućili automatsko prihvaćanje podataka u operaciji OLE povlačenja i ispuštanja, postavite njezino svojstvo OLEDropMode na Automatic. Tijekom izvođenja aplikacije, podaci povučeni iz aplikacije koja podržava OLE u kontrolu okvira s tekstom biti će radije prebačeni, a ne kopirani osim ako tijekom ispuštanja ne držite pritisnutu tipku CTRL, ili ne promijenite standardno ponašanje programskim kodom.

Automatska podrška povlačenju i ispuštanju podataka ima svoja ograničenja; neka od tih ograničenja proizlaze iz djelotvornosti samih kontrola. Na primjer, ako pomaknete tekst iz dokumenta Worda za Windowse u kontrolu okvira s tekstom, sva proširena oblikovanja teksta u dokumentu Worda bit će izgubljena jer kontrola okvira s tekstom ne podržava takvo oblikovanje. Slična ograničenja postoje kod većine kontrola. Drugo ograničenje automatskih operacija je u tome da nemate potpuni nadzor nad tipom podatka koji se povlači i/ili ispušta.

Napomena Kad povlačite podatke, možete uočiti da pokazivač miša naznačuje podržava li objekt preko kojeg se prevlači OLE povlačenje i ispuštanje za tip podatka koji povlačite. Ako objekt podržava OLE povlačenje i ispuštanje za taj tip podatka, prikazuje se pokazivač "ispuštanja". Ako objekt to ne podržava, prikazuje se pokazivač "nema ispuštanja".

Objekt DataObject OLE povlačenja i ispuštanja

Povlačenje i ispuštanje tipa OLE koristi isti model *izvora* i *cilja* kao i jednostavne tehnike povlačenja i ispuštanja upravljane događajima, raspravljene u odlomku "Povlačenje i ispuštanje". U ovom slučaju, međutim, ne povlačite jednu kontrolu na drugu kontrolu kako bi pozvali neki kod, pomičete *podatke* iz jedne kontrole ili aplikacije u drugu kontrolu ili aplikaciju. Na primjer, korisnik odabire i povlači niz ćelija u Excelu (*izvor*) te ispušta niz ćelija u kontrolu mreže povezane s podacima (*cilj*) u vašoj aplikaciji.

U Visual Basicu, nositelj, ili spremište, tih podataka je objekt DataObject – on je sredstvo kojim se podaci pomiču iz izvora u cilj. On to čini pružanjem postupaka potrebnih za spremanje, dohvaćanje i ispitivanje podataka. Sljedeća tablica ispisuje svojstvo i postupke koje upotrebljava objekt DataObject.

vrsta	stavka	opis
svojstvo	Files	Sadržava imena datoteka povučenih u ili iz Windows Explorera.
postupci	Clear	Briše sadržaj objekta DataObject.
	GetData	Dohvaća podatke iz objekta DataObject.
	GetFormat	Utvrđuje je li određen oblik podataka dostupan u objektu DataObject.
	SetData	Postavlja podatke u objekt DataObject, ili pokazuje da je određen oblik dostupan na zahtjev.

Ovi postupci, korišteni s događajima OLE povlačenja i ispuštanja, omogućuju vam upravljanje podacima u objektu DataObject na strani izvora i na strani cilja (ako su obje strane unutar vaše Visual Basic aplikacije). Na primjer, u objekt DataObject možete postaviti podatke na strani izvora korištenjem postupka SetData, te zatim upotrijebiti postupak GetData kako bi prihvatili podatke na strani cilja.

Postupak Clear koristi se za brisanje sadržaja objekta DataObject na strani izvora kad je pokrenut događaj OLEStartDrag. Kad se podaci iz kontrole povlače u automatskoj operaciji povlačenja, njezini oblici podataka su postavljeni u objekt DataObject prije pokretanja događaja OLEStartDrag. Ako ne želite upotrijebiti podrazumijevane oblike, upotrijebite postupak Clear. Ako želite nešto dodati standardnim oblicima, nemojte upotrijebiti postupak Clear.

Svojstvo Files omogućuje vam spremanje imena niza datoteka koje zatim mogu biti povučene i ispuštene u cilj. Pogledajte "Povlačenje datoteka iz Windows Explorera", u stalnoj pomoći, za više informacija o ovom svojstvu.

Možete također odrediti oblik u kojem će podaci biti preneseni. Postupci SetData i GetData koriste sljedeće argumente za postavljanje ili dohvaćanje podataka iz objekta DataObject:

argument	opis
Data	Omogućuje vam određivanje tipa podatka koji je postavljen u objekt DataObject (neobavezan argument ako je određen argument *Format*; inače, ovaj argument je obavezan).
Format	Omogućuje vam određivanje nekoliko različitih oblika koje izvor može podržati, bez potrebe za učitavanjem podataka za svaki oblik (neobavezan argument ako je određen argument *Data* ili ako Visual Basic prepoznaje oblik; inače, obavezan je).

Napomena Kad se podaci ispuste na cilj, a njihov oblik nije određen, Visual Basic je sposoban otkriti je li to bitmapirana slika, metadatoteka, poboljšana metadatoteka ili tekst. Svi ostali oblici moraju biti izričito određeni ili će biti stvorena pogreška.

Argument *format* koristi sljedeće konstante ili vrijednosti za određivanje oblika podatka:

konstanta	vrijednost	značenje
vbCFText	1	tekst
vbCFBitmap	2	bitmapirana slika (.bmp)
vbCFMetafile	3	metadatoteka (.wmf)
vbCFEMetafile	14	poboljšana metadatoteka (.emf)
vbCFDIB	8	bitmapirana slika neovisna o uređaju (.dib ili .bmp)
vbCFPalette	9	paleta boja
vbCFFiles	15	popis datoteka
vbCFRTF	-16639	oblik proširenog teksta (.rtf)

Postupci SetData, GetData i GetFormat koriste argumente *data* i *format* za vraćanje ili tipa podatka u objektu DataObject ili za dohvaćanje samog podatka ako je oblik sukladan s *ciljem*. Na primjer:

```
Private Sub txtIzvor_OLEStartDrag(Data As _
VB.DataObject, AllowedEffects As Long)
  Data.SetData txtIzvor.SelText, cbCFText
End Sub
```

U ovom primjeru, *data* je tekst označen u okviru s tekstom txtIzvor, a *format* je određen kao tekst (vbCFText).

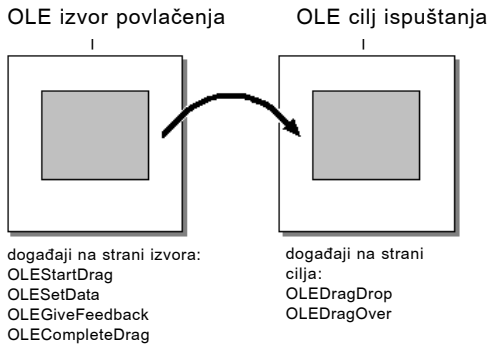
Napomena Trebali bi upotrebljavati oblik podatka vbCFDIB umjesto oblika vbCFBitmap i vbCFPalette, u većini slučajeva. Oblik vbCFDIB sadrži i bitmapirane slike i palete pa je zbog toga poželjniji postupak prenošenja bitmapirane slike. Međutim, možete također odrediti oblike vbCFBitmap i vbCFPalette zbog cjelovitosti. Ako ne želite upotrebljavati oblik vbCFDIB, morate odrediti i oblik vbCFBitmap i oblik vbCFPalette tako da se bitmapirana slika i paleta mogu ispravno smjestiti u objekt DataObject.

Za više informacija Pogledajte "Stvaranje korisničkog oblika podatka", kasnije u ovom poglavlju, za informacije o određivanju vaših vlastitih oblika podataka.

Kako radi OLE povlačenje i ispuštanje

Kad se izvodi operacija OLE povlačenja i ispuštanja, stvaraju se određeni događaji na stranama izvora i cilja. Događaji pridruženi s izvornim objektom se stvaraju uvijek, bez obzira je li operacija povlačenja i ispuštanja automatska ili ručna. Događaji na strani cilja, međutim, stvaraju se samo kod ručne operacije ispuštanja. Sljedeća slika prikazuje koji se događaji pojavljuju uz mogućnost odgovaranja na izvoru povlačenja, te koji se događaji pojavljuju uz mogućnost odgovaranja na cilju ispuštanja.

Slika 11.6 Događaji na stranama izvora i cilja



Na koje ćete događaje trebati odgovoriti ovisi o tome kako ste odabrali ostvariti djelotvornost povlačenja i ispuštanja. Na primjer, možda ste stvorili aplikaciju s okvirom s tekstom za kojeg želite da automatski prihvati povučene podatke iz druge aplikacije. U takvom slučaju, jednostavno postavite svojstvo OLEDropMode kontrole na Automatic. Ako želite omogućiti podacima da također automatski budu povučeni iz kontrole okvira s tekstom, postavite svojstvo OLEDragMode na Automatic.

Ako, međutim, želite promijeniti standardne pokazivače miša ili proširiti djelotvornost stanjima tipki miša i zamjenskim tipkama tipkovnice, trebate ručno odgovoriti na događaje na stranama izvora i cilja. Slično tome, ako želite ispitati podatke prije nego što su ispušteni na kontrolu (kako bi provjerili da su podaci sukladni, na primjer), ili stvoriti kašnjenje kad se podaci učitavaju u objekt DataObject (tako da se na početku ne moraju učitavati višestruki oblici), trebat ćete upotrijebiti ručne operacije OLE povlačenja i ispuštanja.

Budući da možete povlačiti i ispuštati podatke u brojne kontrole Visual Basica i Windows aplikacije – s raznim ograničenjima i zahtjevima – ostvarivanje OLE povlačenja i ispuštanja može se kretati od jednostavnog do vrlo složenog. Najjednostavnije ostvarivanje, svakako, bilo bi povlačenje i ispuštanje između dva automatizirana objekta, bez obzira je li objekt dokument Worda, proračunska tablica Excela ili kontrola u vašoj aplikaciji koja je postavljena na Automatic. Određivanje višestrukih oblika podataka koji bi trebali biti prihvatljivi vašem cilju ispuštanja moglo bi biti složenije.

Pokretanje povlačenja

Što se događa u temeljnoj operaciji OLE povlačenja i ispuštanja unutar vaše Visual Basic aplikacije? Kad korisnik povuče podatke iz izvora OLE povlačenja (kontrole okvira s tekstom, na primjer) odabirom te držanjem pritisnute tipke miša, pokreće se događaj OLEStartDrag i zatim možete spremiti podatke ili jednostavno odrediti oblike koje podržava izvor. Također trebate odrediti dopušta li izvor kopiranje ili premještanje podataka, ili oboje.

Za više informacija Pogledajte "Pokretanje operacije OLE povlačenja", kasnije u ovom poglavlju, za više informacija o postupku OLEDrag, događaju OLEStartDrag, korištenjem postupka SetData za određivanje podržanih oblika podataka, te postavljanju podataka u objekt DataObject.

Povlačenje preko cilja

Dok korisnik povlači preko cilja, pokreće se događaj OLEDragOver cilja, ukazujući da je izvor unutar njegovih granica. Nakon toga određujete što će cilj napraviti ako se podaci ovdje ispuste – hoće li se kopirati, prebaciti ili odbaciti. Po dogovoru, standard je obično prebacivanje, ali može biti i kopiranje.

Kad cilj odredi koji će učinak ispuštanja biti izveden ako se ovdje ispusti izvor, pokreće se događaj OLEGiveFeedback. Događaj OLEGiveFeedback se koristi za pružanje vidljive povratne informacije korisniku o tome koja će akcija biti poduzeta ako se ispusti odabir – znači, pokazivač miša će biti promijenjen kako bi ukazao na akcije kopiranja, prebacivanja, ili akciju "nema ispuštanja".

Dok se izvor pomiče unutar granica cilja – ili ako korisnik pritisne tipke SHIFT, CTRL ili ALT dok tipku miša drži pritisnutom – učinak ispuštanja može biti promijenjen. Na primjer, umjesto dopuštanja kopiranja ili prebacivanja, podaci mogu biti odbačeni.

Ako korisnik izađe iz cilja ili pritisne tipku ESC, na primjer, tada operacija povlačenja može biti otkazana ili promijenjena (pokazivač miša može se promijeniti kako bi pokazao da objekt preko kojeg se trenutno prelazi neće prihvatiti podatke).

Za više informacija Pogledajte "Povlačenje OLE izvora povlačenja preko OLE cilja ispuštanja", kasnije u ovom poglavlju, za više informacija o događajima OLEDragOver i OLEGiveFeedback.

Završavanje povlačenja

Kad korisnik ispusti izvor na cilj, pokreće se događaj OLEDragDrop cilja. Cilj pita izvor za oblik podataka koje sadrži (ili podržava, ako podaci nisu bili smješteni u izvoru kad je povlačenje počelo) te zatim prihvaća ili odbacuje podatke.

Ako su podaci bili spremljeni kad je započelo povlačenje, cilj dohvaća podatke korištenjem postupka GetData. Ako podaci nisu bili spremljeni kad je započelo povlačenje, podaci se dohvaćaju pokretanjem događaja OLESetData izvora te zatim korištenjem postupka SetData.

Kad su podaci prihvaćeni ili odbačeni, pokreće se događaj OLECompleteDrag, a izvor zatim poduzima odgovarajuću akciju: ako su podaci prihvaćeni i određeno je prebacivanje, izvor briše podatke, na primjer.

Za više informacija Pogledajte "Ispuštanje OLE izvora povlačenja na OLE cilj ispuštanja", kasnije u ovom poglavlju, za više informacija o događaju OLEDragDrop, događaju OLECompleteDrag, te korištenju postupaka GetFormat i GetData za dohvaćanje podataka iz objekta DataObject.

Pokretanje operacije OLE povlačenja

Ako želite biti u mogućnosti odrediti koji su oblici podatka ili učinci ispuštanja (kopiranje, prebacivanje ili bez ispuštanja) podržani, ili ako kontrola iz koje želite povući podatke ne podržava automatsko povlačenje, morat ćete svoju operaciju OLE povlačenja izvesti ručno.

Prva faza ručne operacije povlačenja i ispuštanja je pozivanje postupka OLEDrag, postavljanje dopuštenih učinaka ispuštanja, određivanje podržanih oblika podataka, te, neobavezno, postavljanje podataka u objekt DataObject.

Postupak OLEDrag upotrebljavate za ručno pokretanje operacije povlačenja, a događaj OLEStartDrag za određivanje dopuštenih učinaka akcije ispuštanja te podržanih oblika podataka.

Postupak OLEDrag

Općenito, postupak OLEDrag se poziva iz događaja MouseMove objekta kad su podaci odabrani, lijeva tipka miša se drži pritisnutom, i pomiče se miš.

Postupak OLEDrag ne pruža nikakve argumente. Njegova temeljna namjena je pokretanje ručnog povlačenja te zatim dopuštanje događaju OLEStartDrag da odredi uvjete operacije povlačenja (na primjer, određivanje što će se dogoditi kad se podaci odvuku u drugu kontrolu).

Ako izvorna kontrola podržava svojstvo OLEDragMode, morate postaviti to svojstvo na Manual kako bi imali ručni nadzor nad operacijom povlačenja te zatim upotrijebiti postupak OLEDrag na kontroli. Ako kontrola podržava ručno, ali ne i automatsko OLE povlačenje, neće imati svojstvo OLEDragMode, ali će podržavati postupak OLEDrag i događaje OLE povlačenja i ispuštanja.

Napomena Postupak OLEDrag će također djelovati ako je svojstvo OLEDragMode izvorne kontrole postavljeno na Automatic.

Određivanje učinka ispuštanja i oblika podataka

U ručnoj operaciji OLE povlačenja, kad korisnik započne povlačiti izvor i pozove se postupak OLEDrag, pojavljuje se događaj OLEStartDrag kontrole. Upotrijebite taj događaj za određivanje učinaka ispuštanja i oblika podataka koje podržava izvor.

Događaj OLEStartDrag koristi dva argumenta za određivanje podržanih oblika podataka te hoće li podaci biti kopirani ili premješteni kad se ispuste (učinci ispuštanja).

Napomena Ako u događaju OLEStartDrag nisu određeni učinci ispuštanja ni oblici podataka, ručno povlačenje neće biti pokrenuto.

Argument AllowedEffects

Argument *allowedeffects* određuje koje učinke ispuštanja podržava izvor povlačenja. Na primjer:

```
Private Sub txtIzvor_OLEStartDrag(Data As _
VB.DataObject, AllowedEffects As Long)
  AllowedEffects = vbDropEffectMove Or _
  vbDropEffectCopy
End Sub
```

Cilj zatim može upitati izvor povlačenja za tu informaciju te prikladno odgovoriti.

Argument *allowedeffects* koristi sljedeće vrijednosti za određivanje učinaka ispuštanja:

konstanta	vrijednost	opis
vbDropEffectNone	0	Cilj ispuštanja ne može prihvatiti podatke.
vbDropEffectCopy	1	Rezultat ispuštanja je kopiranje. Izvorni podaci ostaju nedirnuti u izvoru povlačenja.
vbDropEffectMove	2	Izvor podataka uklanja podatke.

Argument Format

Određivanjem argumenta *format* događaja OLEStartDrag određujete koje oblike podataka objekt podržava. Kako bi to napravili, upotrijebite postupak SetData. Na primjer, u primjeru koji koristi kontrolu okvira za prošireni tekst kao izvor i kontrolu okvira s tekstom kao cilj, možete odrediti sljedeće podržane oblike:

```
Private Sub rtbIzvor_OLEStartDrag(Data As _
VB.DataObject, AllowedEffects As Long)
  AllowedEffects = vbDropEffectMove Or _

  vbDropEffectCopy
  Data.SetData , vbCFText
  Data.SetData , vbCFRTF
End Sub
```

Cilj može upitati izvor da ustvrdi koji su oblici podataka podržani te zatim prikladno odgovori – na primjer, ako cilj ne podržava oblik ispuštenih podataka, odbacit će ispuštene podatke. U ovom slučaju, jedini oblici podataka koji je podržava izvor su oblik teksta i oblik proširenog teksta.

Za više informacija Pogledajte "Objekt DataObject OLE povlačenja i ispuštanja", ranije u ovom poglavlju, za više informacija o vrijednostima oblika za postupak SetData.

Postavljanje podataka u objekt DataObject

U većini slučajeva, posebno ako izvor podržava više od jednog oblika, ili ako troši puno vremena za stvaranje podataka, možete postaviti podatke u objekt DataObject samo kad to zatraži cilj. Međutim, možete postaviti podatke u objekt DataObject kad počnete operaciju povlačenja tako da upotrijebite postupak SetData u događaju OLEStartDrag. Na primjer:

```
Private Sub txtIzvor_OLEStartDrag(Data As _
VB.DataObject, AllowedEffects As Long)
  Data.Clear
  Data.SetData txtIzvor.SelText, vbCFText
End Sub
```

Ovaj primjer čisti podrazumijevane oblike podataka iz objekta DataObject korištenjem postupka Clear, određuje oblik podatka (text) odabranih podataka, te postavlja podatke u objekt DataObject postupkom SetData.

Povlačenje OLE izvora povlačenja preko OLE cilja ispuštanja

S ručnim ciljem, možete ustanoviti i odgovoriti na položaj izvornih podataka unutar cilja i odgovoriti na stanje tipki miša te tipki SHIFT, CTRL i ALT. Ako su i izvor i cilj ručni, možete prilagoditi standardno vidljivo ponašanje pokazivača miša.

kako bi učinili ovo	upotrijebite
Utvrđivanje i odgovaranje na položaj izvornog objekta.	Argument *state* događaja OLEDragOver.
Odgovaranje na stanje tipki miša.	Argument *button* događaja OLEDragDrop i OLEDragOver.
Odgovaranje na stanje tipki SHIFT, CTRL i ALT.	Argumenti *shift* događaja OLEDragDrop i OLEDragOver.
Promjena standardnog vidljivog ponašanja pokazivača miša.	Argument *effect* događaja OLEDragOver i argument *effect* događaja OLEGiveFeedback.

Za više informacija Za više informacija o promjeni pokazivača miša, pogledajte "Povlačenje OLE izvora povlačenja preko OLE cilja ispuštanja", prije. Za više informacija o korištenju argumenata *button* i *shift*, pogledajte "Korištenje miša i tipkovnice za promjenu učinaka ispuštanja i povratne informacije korisniku", kasnije u ovom poglavlju.

Argument State događaja OLEDragOver

Ovisno o njegovom položaju, argument *effect* može biti promijenjen kako bi pokazivao trenutno prihvatljiv učinak ispuštanja.

Argument *state* događaja OLEDragOver omogućuje vam odgovaranje na kretanje izvornih podataka kad uđu u, prijeđu preko ili napuste ciljnu kontrolu. Na primjer, kad izvorni podaci uđu u ciljnu kontrolu, argument *state* se postavlja na vrijednost vbEnter.

Kad se izvor povlačenja pomakne unutar granica cilja ispuštanja, argument *state* se postavlja na vrijednost vbOver. Ovisno o položaju (argumenti *x* i *y*) pokazivača miša, možete trebati promijeniti učinak povlačenja. Uočite da se događaj OLEDragOver stvara nekoliko puta u sekundi, čak i ne pomičete pokazivač miša.

Argument *state* događaja OLEDragOver određuje kad podaci uđu u, prijeđu preko i napuste ciljnu kontrolu korištenjem sljedećih konstanti:

konstanta	vrijednost	značenje
vbEnter	0	Podaci su povučeni unutar opsega cilja.
vbLeave	1	Podaci su povučeni izvan opsega cilja.
vbOver	2	Podaci su i dalje unutar opsega cilja, ali je pomaknut miš, promijenjena je tipka miša ili tipkovnice, ili je protekao određeni sistemski određen iznos vremena.

Pružanje korisničke vidljive povratne informacije korisniku

Ako želite promijeniti standardno vidljivo ponašanje pokazivača miša u operaciji OLE povlačenja i ispuštanja, možete upravljati događajem OLEDragOver na strani cilja i događajem OLEGiveFeedback na strani izvora.

Povlačenje i ispuštanje tipa OLE pruža automatsku vidljivu povratnu informaciju tijekom operacije povlačenja i ispuštanja. Na primjer, kad počnete povlačenje, pokazivač miša se mijenja kako bi ukazao da je pokrenuto povlačenje. Kad prijeđete preko objekata koji ne podržavaju OLE ispuštanje, pokazivač miša se mijenja u pokazivač "nema ispuštanja".

Promjena pokazivača, miša, tako da naznačuje kako će kontrola odgovoriti ako podaci budu ispušteni na nju uključuje dva koraka: otkrivanje tipa podataka u objektu DataObject upotrebom postupka GetFormat, te zatim određivanje argumenta *effect* događaja OLEDragOver za obavješćivanje izvora koji su učinci ispuštanja dopušteni za tu kontrolu.

Događaj OLEDragOver

Kad je svojstvo OLEDropMode ciljne kontrole postavljeno na Manual, događaj OLEDragOver se pokreće uvijek kad povlačeni podaci prijeđu preko kontrole.

Argument *effect* događaja OLEDragOver se koristi za određivanje vrste akcije koja će biti poduzeta ako je objekt ispušten. Kad je ova vrijednost određena, pokreće se događaj OLEGiveFeedback izvora. Događaj OLEGiveFeedback sadrži svoj vlastiti argument *effect*, koji se koristi za pružanje vidljive povratne informacije korisniku o tipu akcije koja će biti poduzeta kad se odabir povuče – znači, pokazivač miša se mijenja kako bi naznačio kopiranje, prebacivanje ili učinak "nema ispuštanja".

Argument *effect* događaja OLEDragOver koristi sljedeće konstante za naznačivanje akcije ispuštanja:

konstanta	vrijednost	opis
vbDropEffectNone	0	Cilj ispuštanja ne može prihvatiti podatke.
vbDropEffectCopy	1	Rezultat ispuštanja je kopiranje. Originalni podaci su nedirnuti izvorom povlačenja.
vbDropEffectMove	2	Izvor povlačenja prebacuje podatke.

Napomena Argument *effect* događaja OLEDragOver i OLEGiveFeedback pokazuje iste učinke ispuštanja (kopiranje, prebacivanje ili "nema ispuštanja") kao i argument *allowedeffects* događaja OLEStartDrag. Oni se razlikuju jedino po tome što događaj OLEStartDrag određuje koji su učinci dopušteni, a događaji OLEDragOver i OLEGiveFeedback koriste argument *effect* za ukazivanje izvoru koji će od tih učinaka biti izvedeni.

Sljedeći primjer programskog koda pita objekt DataObject za sukladan oblik podataka za ciljnu kontrolu. Ako su podaci sukladni, argument *effect* obavješćuje izvor da će biti izvedeno prebacivanje ako se podaci ispuste. Ako podaci nisu sukladni, izvor će biti obaviješten i bit će prikazan pokazivač miša tipa "nema ispuštanja".

```
Private Sub txtCilj_OLEDragOver(Data As VB.DataObject, _
Effect As Long, Button As Integer, Shift As Integer, _
X As Single, Y As Single, State As Integer)
  If Data.GetFormat(vbCFText) Then
    Effect = vbDropEffectMove And Effect
  Else
    Effect = vbDropEffectNone
  End If
End Sub
```

Kad se podaci izvora prevuku iznad cilja, i pokrene se događaj OLEDragOver, izvor kazuje cilju koje učinke dopušta (kopiranje, prebacivanje ili "nema ispuštanja"). Nakon toga morate odabrati pojedini učinak koji će se dogoditi ako se podaci ispuste. Argument *effect* događaja OLEDragOver obavješćuje izvor koju akciju ispuštanja podržava, a izvor zatim obavješćuje korisnika korištenjem događaja OLEGiveFeedback za promjenu pokazivača miša.

Događaj OLEGiveFeedback

Kako bi promijenili standardno ponašanje pokazivača miša temeljeno na argumentu *effect* događaja OLEDragOver, trebate ručno odrediti nove vrijednosti pokazivača miša korištenjem događaja OLEGiveFeedback. Događaj OLEGiveFeedback izvora pokreće se automatski kad je postavljen argument *effect* događaja OLEDragOver.

Događaj OLEGiveFeedback sadrži dva argumenta (*effect* i *defaultcursors*) koji vam omogućuju promjenu pokazivača miša u operaciji OLE povlačenja i ispuštanja.

Argument *effect*, slično ostalim događajima OLE povlačenja i ispuštanja, određuje hoće li podaci biti kopirani, prebačeni ili odbačeni. Svrha ovog argumenta u događaju OLEGiveFeedback, međutim, je da vam omogući pružanje korisničke vidljive povratne informacije korisniku promjenom pokazivača miša za ukazivanje na te akcije.

konstanta	vrijednost	opis
vbDropEffectNone	0	Cilj ispuštanja ne može prihvatiti podatke.
vbDropEffectCopy	1	Rezultat ispuštanja je kopiranje. Originalni podaci su nedirnuti izvorom povlačenja.
vbDropEffectMove	2	Izvor povlačenja prebacuje podatke.
vbDropEffectScroll	&H80000000&	Pomicanje će početi ili se trenutno događa u cilju. Vrijednost se koristi zajedno s drugim vrijednostima.

Napomena Vrijednost vbDropEffectScroll može biti upotrijebljena u nekim aplikacijama ili kontrolama za naznačivanje da korisnik uzrokuje pomicanje pokretanjem pokazivača miša blizu ruba prozora aplikacije. Pomicanje je automatski podržano u nekim, ali ne u svim standardnim kontrolama Visual Basica. Možda ćete trebati programirati učinak pomicanja ako povlačite podatke u aplikaciju koja sadrži trake za pomicanje – Word za Windowse, na primjer.

Argument *defaultcursors* određuje hoće li biti korišten standardni OLE pokazivač. Postavljanje ovog argumenta na False omogućuje vam određivanje vaših vlastitih pokazivača korištenjem svojstva Screen.MousePointer objekta Screen.

U većini slučajeva, određivanje korisničkih pokazivača miša je nepotrebno zato što OLE rukuje standardnim ponašanjem miša. Ako odlučite odrediti korisničke pokazivače miša korištenjem događaja OLEGiveFeedback, trebat ćete računati na svaki mogući učinak, uključujući pomicanje. Također je dobra ideja programirati za učinke koji kasnije mogu biti dodani, stvaranjem mogućnosti vraćanja nadzora nad pokazivačem miša postupku OLE ako se otkrije nepoznat učinak.

Sljedeći primjer programskog koda postavlja argumente *effect* i *defaultcursors* te određuje korisničke pokazivače (datoteke .ico ili .cur) za učinke kopiranja, prebacivanja ili pomicanja postavljanjem svojstava MousePointer i MouseIcon objekta Screen. Ovaj primjer također vraća nadzor nad pokazivačem miša postupku OLE ako se otkrije nepoznat učinak.

```
Private Sub txtIzvor_OLEGiveFeedback(Effect As Long, _
DefaultCursors As Boolean)
  DefaultCursors = False
  If Effect = vbDropEffectNone Then
    Screen.MousePointer = vbNoDrop
  ElseIf Effect = vbDropEffectCopy Then
    Screen.MousePointer = vbCustom
    Screen.MouseIcon = LoadPicture("c:\copy.ico")
  ElseIf Effect = (vbDropEffectCopy Or _
  vbDropEffectScroll) Then
    Screen.MousePointer = vbCustom
    Screen.MouseIcon = LoadPicture("c:\copyscrl.ico")
  ElseIf Effect = vbDropEffectMove Then
    Screen.MousePointer = vbCustom
    Screen.MouseIcon = LoadPicture("c:\move.ico")
  ElseIf Effect = (vbDropEffectMove Or _
  vbDropEffectScroll) Then
    Screen.MousePointer = vbCustom
    Screen.MouseIcon = LoadPicture("c:\movescrl.ico")
  Else
    ' Ako je dodan novi oblik kojeg ne razumijemo,
    ' dopuštanje postupku OLE da rukuje s
    ' ispravnim standardnim pokazivačima.
    DefaultCursors = True
  End If
End Sub
```

Napomena Trebali bi uvijek obnoviti pokazivače miša u događaju OLECompleteDrag ako odredite korisničke pokazivače miša u događaju OLEGiveFeedback. Za više informacija o obavješćivanju izvora kad su podaci ispušteni, pogledajte "Ispuštanje OLE izvora povlačenja na OLE cilj ispuštanja", u nastavku.

Za više informacija Pogledajte "Prilagođavanje pokazivača miša", kasnije u ovom poglavlju, za informacije o postavljanju svojstava MousePointer i MouseIcon.

Ispuštanje OLE izvora povlačenja na OLE cilj ispuštanja

Ako vaš cilj podržava ručne operacije OLE povlačenja i ispuštanja, možete nadzirati što će se događa kad se pokazivač pomakne unutar cilja i možete odrediti koju vrstu podataka će cilj prihvatiti. Kad korisnik ispusti izvorni objekt na ciljnu kontrolu, koristi se događaj OLEDragDrop za upit objektu DataObject za sukladne oblike podataka, te zatim za dohvaćanje podataka.

Događaj OLEDragDrop također obavješćuje izvor akcije ispuštanja, omogućujući mu da obriše izvorne podatke ako je bilo određeno prebacivanje, na primjer.

Dohvaćanje podataka

Događaj OLEDragDrop pokreće se kad korisnik ispusti izvor na cilj. Ako su podaci postavljeni u objektu DataObject kad započne operacija povlačenja, mogu biti dohvaćeni kad se pokrene događaj OLEDragDrop, korištenjem postupka GetData. Međutim, ako su bili određeni samo podržani oblici izvora kad započne operacija povlačenja, tada će postupak GetData automatski pokrenuti događaj OLESetData u izvoru kako bi smjestio podatke u, te zatim dohvatio podatke iz, objekta DataObject.

Sljedeći primjer dohvaća podatke koji su postavljeni u objekt DataObject kad je pokrenuta operacija povlačenja. Operacija povlačenja mogla je biti pokrenuta ručno (korištenjem postupka OLEDrag na izvor) ili automatski (postavljanjem svojstva OLEDragMode izvora na Automatic). Povlačeni podaci dohvaćaju se postupkom GetData objekta DataObject. Postupak GetData pribavlja vam konstante koje predstavljaju tipove podataka koje podržava objekt DataObject. U ovom slučaju, dohvaćamo podatke kao tekst.

```
Private Sub txtCilj_OLEDragDrop(Data As _
VB.DataObject, Effect As Long, Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  txtCilj.Text = Data.GetData(vbCFText)
End Sub
```

Za više informacija Za potpunu listu oblika konstanti postupka GetData, pogledajte “Objekt DataObject OLE povlačenja i ispuštanja”, ranije u ovom poglavlju.

Upiti objektu DataObject

Možda ćete trebati upitati objekt DataObject za tipove podataka koji su ispušteni na cilj. Upotrijebite postupak GetFormat u izrazu If...Then kako bi odredili tipove podataka koje može prihvatiti ciljna kontrola. Ako su podaci unutar objekta DataObject sukladni, akcija ispuštanja će biti dovršena.

```
Private Sub txtCilj_OLEDragDrop(Data As _
VB.DataObject, Effect As Long, Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  If Data.GetFormat(vbCFText) Then
    txtCilj.Text = Data.GetData(vbCFText)
  End If
End Sub
```

Postavljanje podataka u objekt DataObject

Kad cilj upotrebljava postupak GetData za dohvaćanje podataka iz izvora, događaj OLESetData se pokreće samo ako podaci nisu postavljeni u izvor kad je pokrenuta operacija povlačenja.

U većini slučajeva, posebno ako izvor podržava više od jednog oblika, ili ako troši puno vremena za stvaranje podataka, možete postaviti podatke u objekt DataObject samo kad to zatraži cilj. Događaj OLESetData dopušta izvoru da odgovori na samo jedan zahtjev za postojeći oblik podataka.

Na primjer, ako su podržani oblici podataka određeni korištenjem događaja OLEStartDrag kad je pokrenuta operacija povlačenja, ali podaci nisu postavljeni u objekt DataObject, koristi se događaj OLESetData za postavljanje specifičnog oblika podataka u objekt DataObject:

```
Private Sub txtIzvor_OLESetData(Data As _
VB.DataObject, DataFormat As Integer)
  If DataFormat = vbCFText Then
    Data.SetData txtIzvor.SelText, vbCFText
  End If
End Sub
```

Obavješćivanje izvora kad su podaci ispušteni

Argument *effect* događaja OLEDragDrop određuje kako će podaci biti utjelovljeni u cilj kad se ispuste podaci. Kad je ovaj argument postavljen, na izvoru se pokreće događaj OLECompleteDrag s svojim argumentom *effect* postavljenim na ovu vrijednost. Izvor zatim može poduzeti odgovarajuću akciju: ako je određeno prebacivanje, izvor briše podatke, na primjer.

Argument *effect* događaja OLEDragDrop koristi iste konstante kao i argument *effect* događaja OLEDragOver za ukazivanje na akciju ispuštanja. Sljedeća tablica ispisuje te konstante:

konstanta	vrijednost	opis
vbDropEffectNone	0	Cilj ispuštanja ne može prihvatiti podatke.
vbDropEffectCopy	1	Rezultat ispuštanja je kopiranje. Originalni podaci su nedirnuti izvorom povlačenja.
vbDropEffectMove	2	Izvor povlačenja prebacuje podatke.

Sljedeći primjer postavlja argument *effect* za naznačivanje akcije ispuštanja.

```
Private Sub txtCilj_OLEDragDrop(Data As _
VB.DataObject, Effect As Long, Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  If Data.GetFormat(vbCFText) Then
    txtCilj.Text = DataGetData(vbCFText)
  End If
  Effect = vbDropEffectMove
End Sub
```

Na strani izvora, događaj OLECompleteDrag se pokreće kad je izvor ispušten na cilj, ili kad je otkazana operacija OLE povlačenja i ispuštanja. Događaj OLECompleteDrag je posljednji događaj u operaciji povlačenja i ispuštanja.

Događaj OLECompleteDrag sadrži samo jedan argument (*effect*), koji se koristi za obavješćivanje izvora akcije koja je poduzeta kad su podaci ispušteni na cilj.

Argument *effect* vraća iste vrijednosti koje su upotrijebljene za argument *effect* ostalih događaja OLE povlačenja i ispuštanja: vrijednosti vbDropEffectNone, vbDropEffectCopy i vbDropEffectMove.

Postavljanjem ovog argumenta nakon što je cilj prepoznao povlačenje i izvor je ispušten na cilj, na primjer, izvor će obrisati originalne podatke u kontroli. Trebate također upotrijebiti događaj OLECompleteDrag za obnavljanje pokazivača miša ako ste u događaju OLEGiveFeedback odredili korisnički pokazivač miša. Na primjer:

```
Private Sub txtIzvor_OLECompleteDrag(Effect As Long)
  If Effect = vbDropEffectMove Then
    txtIzvor.SelText = ""
  End If
  Screen.MousePointer = vbDefault
End Sub
```

Korištenje miša i tipkovnice za promjenu učinaka ispuštanja i povratne informacije korisniku

Možete poboljšati događaje OLEDragDrop i OLEDragOver korištenjem argumenata *button* i *shift* za odgovaranje na stanje tipki miša te tipki SHIFT, CTRL i ALT. Na primjer, kad povlačite podatke u kontrolu, možete omogućiti korisniku da izvede operaciju kopiranja pritiskanjem tipke CTRL, ili operaciju prebacivanja pritiskanjem tipke SHIFT.

U sljedećem primjeru, argument *shift* događaja OLEDragDrop koristi se za otkrivanje je li pritisnuta tipka CTRL kad se ispuštaju podaci. Ako je, izvodi se kopiranje. Ako nije, izvodi se prebacivanje.

```
Private Sub txtCilj_OLEDragDrop(Data As _
VB.DataObject, Effect As Long, Button As Integer, _
Shift As Integer, X As Single, Y As Single)
  If Shift And vbCtrlMask Then
```

```
    txtCilj.Text = Data.GetData(vbCFText)
    Effect = vbDropEffectCopy
  Else
    txtCilj.Text = Data.GetData(vbCFText)
    Effect = vbDropEffectMove
  End If
End Sub
```

Argument *button* može se upotrijebiti za izdvajanje i odgovaranje na razna stanja tipki miša. Na primjer, možete poželjeti dopustiti korisniku da prebaci podatke istovremenim pritiskom lijeve i desne tipke miša.

Za naznačivanje korisniku koja će akcija biti poduzeta kad se izvorni objekt prevuče preko cilja uz pritisnutu tipku miša ili tipke SHIFT, CTRL i ALT, možete postaviti argumente *shift* i *button* događaja OLEDragOver. Na primjer, kako bi obavijestili korisnika koja će akcija biti poduzeta kad je pritisnuta tipka CTRL tijekom operacije povlačenja, možete dodati sljedeći programski kod u događaj OLEDragOver:

```
Private Sub txtCilj_OLEDragOver(Data As VB.DataObject, _
Effect As Long, Button As Integer, Shift As Integer, _
X As Single, Y As Single, State As Integer)
  If Shift and vbCtrlMask Then
    Effect = vbDropEffectCopy
  Else
    Effect = vbDropEffectMove
  End If
End Sub
```

Za više informacija Pogledajte odlomke “Otkrivanje tipki miša” i “Otkrivanje stanja SHIFT, CTRL i ALT”, ranije u ovom poglavlju za više informacija o odgovaranju na stanja miša i tipkovnice.

Stvaranje korisničkog oblika podatka

Ako su vam oblici podataka pruženi s Visual Basicom nedovoljni za neke određene namjene, možete stvoriti korisnički oblik podatka za upotrebu u operaciji OLE povlačenja i ispuštanja. Na primjer, korisnički oblik podatka je koristan ako vaša aplikacija definira jedinstven oblik podatka kojeg trebate povući između dva primjera vaše aplikacije, ili unutar same aplikacije.

Kako bi stvorili korisnički oblik podatka, trebate pozvati funkciju RegisterClipboardFormat Windows API-ja. Na primjer:

```
Private Declare Function RegisterClipboardFormat Lib _
“user32.dll” Alias “RegisterClipboardFormatA” _
(ByVal lpszFormat$ As Integer)
Dim MojOblik As Integer
```

Kad to jednom odredite, možete upotrebljavati svoj korisnički oblik kao i sve ostale oblike podataka objekta DataObject. Na primjer:

```
Dim a() As Byte
a = Data.GetData(MojOblik)
```

Kako bi upotrijebili ovu djelotvornost, trebate postaviti podatke u, i dohvatiti podatke iz objekta DataObject kao matricu tipa Byte. Nakon toga možete dodijeliti svoj korisnički oblik podatka varijabli stringa jer se ona automatski pretvara.

Oprez Dohvaćanje vašeg korisničkog oblika podatka postupkom GetData može proizvesti nepredvidljive rezultate.

Budući da Visual Basic ne razumije vaš korisnički oblik podatka (jer ste ga definirali), nema način za otkrivanje veličine podatka. Visual Basic može ustvrditi veličinu memorije za matricu tipa Byte bjer je nju dodijelio Windows, ali operativni sustav obično dodjeljuje više memorije nego što je potrebno.

Zbog toga, kad dohvaćate korisnički oblik podatka, dobit ćete matricu tipa Byte koja sadrži najmanje, a vjerojatno i više od toga, broj bajtova koje je izvor stvarno postavio u objekt DataObject. Nakon toga trebate ispravno protumačiti vaš korisnički oblik podatka kad je dohvaćen iz objekta DataObject. Na primjer, u jednostavnom stringu, trebate potražiti karakter NULL te odrezati string na tu duljinu.

Povlačenje datoteka iz Windows Explorera

Možete upotrijebiti OLE povlačenje i ispuštanje za povlačenje datoteka iz Windows Explorera u odgovarajuću kontrolu Visual Basica, ili obratno. Na primjer, možete odabrati raspon tekstualnih datoteka u Windows Exploreru te ih sve otvoriti u jednoj kontroli okvira s tekstom, povlačenjem i ispuštanjem odabranog na kontrolu.

Kako bi to prikazali, sljedeći postupak koristi kontrolu okvira s tekstom te događaje OLEDragOver i OLEDragDrop za otvaranje raspona tekstualnih datoteka korištenjem svojstva Files te oblika podatka vbCFFiles objekta DataObject.

Kako povući tekstualne datoteke u kontrolu okvira s tekstom iz Windows Explorera

1. Započnite novi projekt u Visual Basicu.
2. Dodajte kontrolu okvira s tekstom na formu. Postavite njezino svojstvo OLEDropMode na Manual. Postavite njezino svojstvo **MultiLine** na **True** i obrišite vrijednost svojstva **Text**.
3. Dodajte funkciju za odabir i označavanje raspona datoteka. Na primjer:

```
Sub DropFile(ByVal txt As TextBox, ByVal strFN$)
  Dim iFile As Integer
  iFile = FreeFile

  Open strFN For Input Access Read Lock Read Write As #iFile
  Dim Str$, strLine$
  While Not EOF(iFile) And Len(Str) <= 32000
```

```
      Line Input #iFile, strLine$
      If Str <> "" Then Str = Str & vbCrLf
      Str = Str & strLine
   Wend
   Close #iFile

   txt.Selstart = Len(txt)
   txt.SelLength = 0
   txt.SelText = Str
End Sub
```

4. Dodajte sljedeći potprogram događaju OLEDragOver. Postupak GetFormat se koristi za ispitivanje sukladnosti oblika podatka (vbCFFiles).

```
Private Sub Text1_OleDragOver(Data As VB.DataObject, _
Effect As Long, Button As Integer, Shift As Integer, _
X As Single, Y As Single, State As Integer)
   If Data.GetFormat(vbCFFiles) Then
      ' Ako su podaci u ispravnom obliku, obavijesti
      ' izvor o akciji koja će biti poduzeta.
      Effect = vbDropEffectCopy And Effect
      Exit Sub
   End If
   ' Ako podaci nisu željenog oblika, nema ispuštanja.
   Effect = vbDropEffectNone

End Sub
```

5. Na kraju, dodajte sljedeći potprogram događaju OLEDragDrop.

```
Private Sub Text1_OLEDragDrop(Data As _
VB.DataObject, Effect As Long, Button As Integer, _
Shift As Integer, X As Single, Y As Single)
   If Data.GetFormat(vbCFFiles) Then
      Dim vFN

      For Each vFN In Data.Files
          DropFile Text1, vFN
      Next vFN
   End If
End Sub
```

6. Pokrenite aplikaciju, otvorite Windows Explorer, označite nekoliko tekstualnih datoteka, i povucite ih u kontrolu okvira s tekstom. Svaka od tekstualnih datoteka će biti otvorena u okviru s tekstom.

Prilagođavanje pokazivača miša

Svojstva MousePointer i MouseIcon možete upotrijebiti za prikaz korisničke ikone, kursora ili bilo kojeg iz niza unaprijed određenih pokazivača miša. Promjena pokazivača miša daje vam način obavješćivanja korisnika da se obrađuju dugotrajni pozadinski zadaci, da može promijeniti veličinu kontroli ili prozoru, ili da dana kontrola ne podržava povlačenje i ispuštanje, na primjer. Upotrebom korisničkih ikona ili pokazivača miša, možete izraziti neograničen opseg vidljivih informacija o stanju i djelotvornosti vaše aplikacije.

Sa svojstvom MousePointer možete odabrati bilo koji od šesnaest unaprijed određenih pokazivača. Ovi pokazivači predstavljaju razne sistemske događaje i procedure. Sljedeća tablica opisuje neke od tih pokazivača te njihove moguće načine upotrebe u vašoj aplikaciji.

pokazivač miša	konstanta	opis
	vbHourGlass	Upozorava korisnika na promjene u stanju aplikacije. Na primjer, prikaz pješčanog sata kazuje korisniku da čeka.
	vbSizePointer	Obavještava korisnika o promjenama djelovanja. Na primjer, pokazivač s dvije strelice kazuje korisniku da može promijeniti veličinu prozora.
	vbNoDrop	Opominje korisnika da akcija ne može biti izvedena. Na primjer, pokazivač "nema ispuštanja" kazuje korisniku da ne može ispustiti datoteku na taj položaj.

Svaki izbor pokazivača je predstavljen postavkom cjelobrojne vrijednosti. Standardna postavka je 0 – Default i obično se prikazuje kao standardni pokazivač Windowsa u obliku strelice. Međutim, ovu postavku nadzire operativni sustav i može ju promijeniti ako korisnik promijeni sistemske postavke miša. Kako bi u svojoj aplikaciji nadzirali pokazivač miša, postavite svojstvo MousePointer na prikladnu vrijednost.

Potpuni popis pokazivača miša dostupna je odabirom svojstva MousePointer kontrole ili forme te prolazom kroz spuštajuću listu postavki, ili korištenjem pretraživača objekata te traženjem stavke MousePointerConstants.

Kad postavite svojstvo MousePointer za kontrolu, pokazivač se pojavljuje kad je miš iznad odgovarajuće kontrole. Kad postavite svojstvo MousePointer za formu, odabrani pokazivač se pojavljuje kad je miš iznad praznih područja forme te kad je iznad kontrola koje imaju svojstvo MousePointer postavljeno na 0 – Default.

Tijekom izvođenja aplikacije možete postaviti vrijednost pokazivača miša korištenjem cjelobrojnih vrijednosti ili konstantama Visual Basica za pokazivač miša. Na primjer:

```
Form1.MousePointer = 11                ' ili vbHourglass
```

Za više informacija Za potpunu listu konstanti pokazivača miša, pogledajte "Konstante MousePointer" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Ikone i kursori

Pokazivač miša možete postaviti tako da prikazuje korisničke ikone ili kursore. Upotreba korisničkih ikona ili kursora omogućuje vam daljnju promjenu izgleda ili djelotvornosti vaše aplikacije. Ikone su jednostavno datoteke s nastavkom .ico, kao one dostavljene s Visual Basicom. Kursori su datoteke s nastavkom .cur i, slično ikonama, u suštini su bitmapirane slike. Međutim, kursori su stvoreni posebno kako bi pokazali korisniku kako će se odvijati akcije pokrenute mišem – oni mogu predstavljati stanje miša i trenutan položaj unosa podataka.

Kursori također sadrže informaciju *vruće točke* (*hot spot*). Vruća točka je piksel koji prati položaj kursora – *x* i *y* koordinate. U pravilu, vruća točka je postavljena u središtu kursora. Ikone, kad se učitaju u Visual Basic kroz svojstvo MouseIcon, pretvaraju se u oblik kursora i vruća točka se postavlja u središnji piksel. Ikone i kursori razlikuju se po tome što položaj vruće točke u datoteci tipa .cur može biti promijenjen, dok se u datoteci tipa .ico ne može mijenjati. Datoteke kursora mogu biti editirane u aplikaciji Image Editor, koja je dostupna u dodatku Windows SDK.

Kako bi upotrijebili korisničku ikonu ili kursor, odredite svojstva MousePointer i MouseIcon.

Kako upotrijebiti datoteku tipa .ico kao pokazivač miša

1. Odaberite formu ili kontrolu i postavite svojstvo MousePointer na 99 – Custom.
2. Učitajte datoteku tipa .ico u svojstvo MouseIcon. Na primjer, za formu:

```
Form1.MouseIcon = LoadPicture("c:\Program _
Files\Microsoft Visual _
Basic\Icons\Computer\Disk04.ico")
```

Oba svojstva moraju biti odgovarajuće podešena kako bi se ikona pojavila kao pokazivač miša. Ako ikona nije učitana u svojstvo MouseIcon kad je svojstvo MousePointer postavljeno na 99 – Custom, upotrijebit će se podrazumijevan pokazivač miša. Slično tome, ako svojstvo MousePointer nije postavljeno na 99 – Custom, zanemaruje se postavka svojstva MouseIcon.

Napomena Visual Basic ne podržava datoteke animiranih kursora (.ani).

Za više informacija Pogledajte "Svojstvo MouseIcon" i "Svojstvo MousePointer" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Odgovaranje na događaje tipkovnice

Događaji tipkovnice, zajedno s događajima miša, su temeljni elementi suradnje korisnika s vašom aplikacijom. Klikovi mišem i pritisci na tipke pokreću događaje i pružaju način unosa podataka te osnovne oblike kretanja kroz prozore i izbornike.

Iako operativni sustav pruža neograničenu podlogu za sve te akcije, ponekad ih je korisno ili potrebno promijeniti ili poboljšati. Događaji KeyPress, KeyUp i KeyDown omogućuju vam izradu tih promjena i poboljšanja.

Programiranje vaše aplikacije tako da odgovara na događaje tipkovnice označava se kao pisanje *rukovatelja tipkovnicom* (*keyboard handler*). Rukovatelj tipkovnicom može raditi na dvije razine: na razini kontrole i na razini forme. Rukovatelj na razini kontrole (*niska razina*) omogućuje vam programiranje određene kontrole. Na primjer, možete trebati pretvoriti sav tekst upisan u kontrolu okvira s tekstom u velika slova. Rukovatelj *na razini forme* omogućuje formi da prva reagira na događaje tipkovnice. Nakon toga fokus može biti prebačen na kontrolu ili kontrole na formi, a događaji mogu biti ponovljeni ili pokrenuti.

Sa takvim događajima tipkovnice možete napisati programski kod koji će rukovati većinom tipki na standardnim tipkovnicama. Za informacije o postupanju s međunarodnim skupovima karaktera i tipkovnicama, pogledajte 16 poglavlje “Međunarodna izdanja”.

Pisanje rukovatelja tipkovnicom niske razine

Visual Basic pruža tri događaja koje prepoznaju forme te sve kontrole koje prihvaćaju unos podataka tipkovnicom. Ovi događaji su opisani u sljedećoj tablici.

događaj tipkovnice	pojavljuje se
KeyPress	Kad je pritisnuta tipka koja odgovara ASCII karakteru.
KeyDown	Kad je pritisnuta bilo koja tipka na tipkovnici.
KeyUp	Kad je otpuštena bilo koja tipka na tipkovnici.

Samo objekt koji ima fokus može primiti događaj tipkovnice. Za događaje tipkovnice, forma ima fokus samo ako je aktivna i nijedna kontrola na formi nema fokus. To se događa samo na praznim formama i formama na kojima su sve kontrole onemogućene. Međutim, ako svojstvo KeyPreview forme postavite na True, forma će prihvaćati sve događaje tipkovnice za svaku kontrolu na formi prije nego što kontrola prepozna te događaje. To je iznimno korisno kad želite izvesti istu akciju uvijek kad je pritisnuta ista tipka, neovisno o tome koja kontrola ima fokus u tom trenutku.

Događaji KeyDown i KeyUp pružaju najnižu razinu odgovora na tipkovnicu. Upotrijebite te događaje za otkrivanje stanja koja događaj KeyPress ne može otkriti, na primjer:

- Posebne kombinacije tipaka SHIFT, CTRL i ALT.
- Kursorske tipke. Zapamtite da neke kontrole (naredbeni gumbi, gumbi izbora i kontrolne kućice) ne prihvaćaju događaje kursorskih tipki: umjesto toga, kursorske tipke uzrokuju pomicanje na drugu kontrolu.
- tipke PAGEUP i PAGEDOWN.
- Razlikovanje numeričke tipkovnice od brojeva s tipki glavnog dijela tipkovnice.
- Odgovaranje na tipke kad budu otpuštene na jednak način kao i kad su pritisnute (događaj KeyPress odgovara samo na pritisak tipke).
- Funkcijske tipke koje nisu dodane naredbama izbornika.

Događaji tipkovnice nisu međusobno isključivi. Kad korisnik pritisne tipku, stvaraju se oba događaja, KeyDown i KeyPress, iza kojih slijedi događaj KeyUp kad korisnik otpusti tipku. Kad korisnik pritisne jednu od tipki koje događaj KeyPress ne otkriva, pojavljuje se samo događaj KeyDown, iza kojeg slijedi događaj KeyUp.

Prije korištenja događaja KeyUp i KeyDown, provjerite je li dovoljan događaj KeyPress. Ovaj događaj otkriva tipke koje odgovaraju svim standardnim ASCII karakterima: slovima, brojkama te znakovima interpunkcije na standardnoj tipkovnici, kao i tipkama ENTER, TAB i BACKSPACE. Općenito je lakše napisati kod za događaj KeyPress.

Također bi trebali uzeti u obzir korištenje prečica kombinacijom tipki i pristupnih tipki, koje su opisane u odlomku "Osnove izbornika" u 3. poglavlju "Forme, kontrole i izbornici". Prečice kombinacijom tipki moraju biti dodijeljene naredbama izbornika, ali mogu uključivati funkcijske tipke (uključujući neke kombinacije funkcijskih tipki s zamjenskim tipkama). Prečice kombinacijom tipki možete dodijeliti bez pisanja dodatnog koda.

> Napomena Skup karaktera tipa Windows ANSI (American National Standards Institute) podudara se s 256 karaktera koji uključuju standardnu latinsku abecedu, oznake izdavaštva (kao simbol autorskog prava, dugu crticu i oznaku nastavka izraza), kao i puno zamjenskih i slova s naglascima. Ovi karakteri su predstavljeni jedinstvenom brojčanom vrijednošću od 1 bajta (0 – 255). Skup ASCII (American Standard Code for Information Interchange) je u biti podskup (0 – 127) skupa ANSII karaktera i predstavlja standardna slova, brojke i znakove interpunkcije na standardnoj tipkovnici. Ova dva skupa karaktera se često označuju kao međusobno zamjenjivi.

Događaj KeyPress

Događaj KeyPress pojavljuje se kad je pritisnuta bilo koja tipka koja odgovara ASCII karakteru. Skup ASCII karaktera predstavlja ne samo slova, brojke i znakove interpunkcije na standardnoj tipkovnici nego također i većinu kontrolnih tipki. Događaj KeyPress, međutim, od kontrolnih tipki prepoznaje samo tipke ENTER, TAB i BACKSPACE. Ostale funkcijske, kursorske i tipke za editiranje mogu biti otkrivene događajima KeyDown i KeyUp.

Upotrijebite događaj KeyPress uvijek kad želite obraditi standardne ASCII karaktere. Na primjer, ako želite prisiliti sve karaktere u okviru s tekstom da budu velika slova, možete upotrijebiti ovaj događaj za promjenu vrste slova dok se upisuju:

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
  KeyAscii = Asc(UCase(Chr(KeyAscii)))
End Sub
```

Argument *keyascii* vraća cjelobrojnu vrijednost koja odgovara kodu ASCII karaktera. Gornji potprogram koristi funkciju Chr za pretvaranje ASCII koda karaktera u odgovarajući karakter, funkciju UCase za pretvaranje karaktera u veliko slovo, a funkciju Asc za pretvaranje rezultata natrag u kod karaktera.

Upotrebom istih ASCII kodova karaktera, možete ispitati je li pritisnuta tipka koju prepoznaje događaj KeyPress. Na primjer, sljedeći potprogram događaja koristi događaj KeyPress za otkrivanje je li korisnik pritisnuo tipku BACKSPACE:

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
  If KeyAscii = 8 Then MsgBox "Pritisnuli _
  ste tipku BACKSPACE."
End Sub
```

Možete također upotrijebiti konstante Visual Basica za kodove tipki umjesto kodova karaktera. Tipka BACKSPACE u prethodnom primjeru ima ASCII vrijednost od 8. Vrijednost konstante za tipku BACKSPACE je vbKeyBack.

Za više informacija Za potpunu listu kodova karaktera, pogledajte odlomke "Skup karaktera (0 – 127)" i "Skup karaktera (128 – 255)" u priručniku *Microsoft Visual Basic 6.0 Language Reference*. Potpuni popis konstanti kodova tipki s pripadajućim ASCII vrijednostima dostupna je u odlomku "Konstante kodova tipki" u istom priručniku ili korištenjem pretraživača objekata te traženjem pojma KeyCodeConstants.

Događaj KeyPress možete također upotrijebiti za promjenu standardnog ponašanja određenih tipki. Na primjer, pritisak tipke ENTER kad ne postoji gumb tipa Default na formi uzrokuje zvučno upozorenje. Taj pisak možete zaobići presretanjem tipke ENTER (kod karaktera 13) u događaju KeyPress.

```
Private Sub Text1_KeyPress(KeyAscii As Integer)
  If KeyAscii = 13 Then KeyAscii = 0
End Sub
```

Događaji KeyDown i KeyUp

Događaji KeyUp i KeyDown prijavljuju točno fizičko stanje same tipkovnice: tipka je pritisnuta (događaj KeyDown) i tipka je otpuštena (KeyUp). Suprotno tome, događaj KeyPress ne prijavljuje stanje tipkovnice izravno – on ne prepoznaje pritisnuto ili otpušteno stanje tipke, on jednostavno pribavlja karakter koji ta tipka predstavlja.

Drugi primjer pomaže u prikazivanju razlike. Kad korisnik upiše veliko slovo "A", događaj KeyDown dobiva ASCII kod za "A". Događaj KeyDown dobiva isti kod i kad korisnik upiše malo slovo "a". Kako bi otkrili je li pritisnuti karakter veliko ili malo slovo, ovi događaji koriste argument *shift*. Suprotno tome, događaj KeyPress obrađuje oblike velikog i malog slova kao dva odvojena ASCII karaktera.

Događaji KeyDown i KeyUp vraćaju informaciju o upisanom karakteru pružanjem sljedeća dva argumenta.

argument	opis
Keycode	Ukazuje na stvarnu pritisnutu tipku. U gornjem slučaju, "A" i "a" se vraćaju kao ista tipka. Ova slova imaju istu vrijednost argumenta *keycode*. Zapamtite da se "1" na glavnim tipkama i "1" na numeričkoj tipkovnici vraćaju kao različite tipke, iako stvaraju isti karakter.
Shift	Ukazuje na stanje tipki SHIFT, CTRL i ALT. Samo istraživanjem ovog argumenta možete otkriti je li upisano veliko ili malo slovo.

Argument Keycode

Argument *keycode* prepoznaje tipku po ASCII vrijednosti ili konstanti koda tipke. Kodovi tipki za tipke s slovima su isti kao ASCII kodovi karaktera velikih slova. Tako je argument *keycode* i za "A" i za "a" jednak vrijednosti koju vraća izraz Asc("A"). Sljedeći primjer koristi događaj KeyDown za otkrivanje je li pritisnuta tipka "A":

```
Private Sub Text1_KeyDown(KeyCode As Integer, _
Shift As Integer)
  If KeyCode = vbKeyA Then MsgBox "Pritisnuli _
  ste tipku A."
End Sub
```

Pritiskanje SHIFT + "A" ili "A" bez tipke SHIFT prikazuje okvir s porukom – znači, argument je istinit u oba slučaja. Kako bi otkrili je li pritisnut oblik velikog ili malog slova trebate upotrijebiti argument *shift*. Pogledajte odlomak "Argument Shift" kasnije u ovom poglavlju.

Kodovi tipki za brojeve i znakove interpunkcije isti su kao i ASCII kodovi broja na tipki. Tako je argument *keycode* i za "1" i za "!" jednak vrijednosti koju vraća izraz Asc("1"). Ponovno, kako bi ustanovili je li upisan karakter "!" morate upotrijebiti argument *shift*.

Događaji KeyDown i KeyUp mogu prepoznati većinu kontrolnih tipki na standardnoj tipkovnici. To uključuje funkcijske tipke (F1 – F16), tipke za editiranje (HOME, PAGE UP, DELETE i tako dalje), kursorske tipke (STRELICA DESNO, LIJEVO, GORE i DOLJE), te numeričku tipkovnicu. Ove tipke mogu biti ispitane korištenjem konstante koda tipke ili odgovarajuće ASCII vrijednosti. Na primjer:

```
Private Sub Text1_KeyDown(KeyCode As Integer, _
Shift As Integer)
  If KeyCode = vbKeyHome Then MsgBox "Pritisnuli _
  ste tipku HOME."
End Sub
```

Za više informacija Za potpunu listu kodova karaktera, pogledajte odlomke "Skup karaktera (0 – 127)" i "Skup karaktera (128 – 255)" u priručniku *Microsoft Visual Basic 6.0 Language Reference*. Potpuni popis konstanti kodova tipki s pripadajućim ASCII vrijednostima dostupna je korištenjem pretraživača objekata te traženjem pojma KeyCodeConstants.

Argument Shift

Događaji tipkovnice koriste argument *shift* na isti način kao i događaji miša – kao cjelobrojne vrijednosti i vrijednosti konstanti koje predstavljaju tipke SHIFT, CTRL i ALT. Argument *shift* možete upotrijebiti s događajima KeyDown i KeyUp kako bi razlikovali karaktere velikih i malih slova, ili za ispitivanje raznih stanja tipkovnice.

Nadogradnjom na prethodni primjer, možete upotrijebiti argument shift za otkrivanje je li pritisnut oblik velikog slova.

```
Private Sub Text1_KeyDown(KeyCode As Integer, _
Shift As Integer)
  If KeyCode = vbKeyA And Shift = 1 _
  Then MsgBox “Pritisnuli ste veliko slovo A.”
End Sub
```

Slično događajima miša, događaji KeyUp i KeyDown mogu otkriti tipke SHIFT, CTRL i ALT posebno ili kao kombinacije. Sljedeći primjer ispituje određena stanja ovih tipki.

Otvorite novi projekt i dodajte varijablu ShiftKey u odjeljak Declarations forme:

```
Dim ShiftKey As Integer
```

Dodajte formi kontrolu okvira s tekstom te ovaj potprogram u događaj KeyDown:

```
Private Sub Text1_KeyDown(KeyCode As Integer, _
Shift As Integer)
  ShiftKey = Shift And 7
  Select Case ShiftKey
    Case 1               ‘ ili vbShiftMask
      Print “Pritisnuli ste tipku SHIFT.”
    Case 2               ‘ ili vbCtrlMask
      Print “Pritisnuli ste tipku CTRL.”
    Case 4               ‘ ili vbAltMask
      Print “Pritisnuli ste tipku ALT.”
    Case 3
      Print “Pritisnuli ste SHIFT i CTRL.”
    Case 5
      Print “Pritisnuli ste SHIFT i ALT.”
    Case 6
      Print “Pritisnuli ste CTRL i ALT.”
    Case 7
      Print “Pritisnuli ste SHIFT, CTRL i ALT.”
  End Select
End Sub
```

Sve dok kontrola okvira s tekstom ima fokus, svaka tipka ili kombinacija tipki, kad se pritisne, ispisuje odgovarajuću poruku na formu.

Za više informacija Pogledajte “Otkrivanje stanja SHIFT, CTRL i ALT” ranije u ovom poglavlju.

Pisanje rukovatelja tipkovnicom na razini forme

Svaki događaj KeyDown i KeyUp pridružen je određenom objektu. Kako bi napisali rukovatelja tipkovnicom koji se primjenjuje za sve objekte na formi, postavite svojstvo KeyPreview forme na True. Kad je svojstvo KeyPreview forme postavljeno na True, forma prepoznaje događaje KeyPress, KeyUp i KeyDown za sve kontrole na formi prije nego što same kontrole prepoznaju te događaje. To čini vrlo lakim pružanje zajedničkog odgovora na određen pritisak tipke.

Svojstvo KeyPreview forme možete postaviti na True u prozoru s svojstvima ili kroz programski kod u potprogramu Form_Load:

```
Private Sub Form_Load
  Form1.KeyPreview = True
End Sub
```

Možete ispitati razna stanja tipaka na formi određivanjem varijable ShiftKey te korištenjem izraza Select Case. Sljedeći potprogram će ispisati poruku na formu neovisno o tome koja kontrola ima fokus.

Otvorite novi projekt i dodajte varijablu ShiftKey u odjeljak Declarations forme:

```
Dim ShiftKey As Integer
```

Dodajte formi kontrolu okvira s tekstom i kontrolu naredbenog gumba. Dodajte sljedeći potprogram događaju KeyDown forme:

```
Private Sub Form1_KeyDown(KeyCode As Integer, _
Shift As Integer)
  ShiftKey = Shift And 7
  Select Case ShiftKey
    Case 1        ‘ ili vbShiftMask
      Print “Pritisnuli ste tipku SHIFT.”
    Case 2        ‘ ili vbCtrlMask
      Print “Pritisnuli ste tipku CTRL.”
    Case 4        ‘ ili vbAltMask
      Print “Pritisnuli ste tipku ALT.”
  End Select
End Sub
```

Ako ste odredili prečicu kombinacijom tipki za kontrolu izbornika, događaj Click za tu kontrolu izbornika pojavljuje se automatski kad korisnik pritisne tu kombinaciju tipki, i ne pojavljuje se događaj tipkovnice.

Slično tome, ako je na formi naredbeni gumb s svojstvom Default postavljenim na True, tipka ENTER će uzrokovati pojavljivanje događaja Click za taj naredbeni gumb umjesto događaja tipkovnice. Ako postoji naredbeni gumb s svojstvom Cancel postavljenim na True, tipka ESC će uzrokovati pojavljivanje događaja Click za taj naredbeni gumb umjesto događaja tipkovnice.

Na primjer, ako naredbenom gumbu dodate potprogram događaja Click te zatim postavite svojstvo Default ili svojstvo Cancel na True, pritisak tipki ENTER ili ESC će zaobići događaj KeyDown. Ovaj potprogram zatvara aplikaciju:

```
Private Sub Command1_Click()
  End
End Sub
```

Uočite da tipka TAB pomiče fokus s kontrole na kontrolu i ne izaziva događaj tipkovnice osim ako je svaka kontrola na formi onemogućena ili ima svojstvo TabStop postavljeno na False.

Kad je svojstvo KeyPreview forme postavljeno na True, forma prepoznaje događaje tipkovnice prije kontrola, ali se događaji i dalje pojavljuju za kontrole. Kako bi to spriječili, možete postaviti na 0 argumente *keyascii* ili *keycode* u potprogramima forme za događaje tipkovnice. Na primjer, ako na formi ne postoji podrazumijevani gumb, možete upotrijebiti tipku ENTER za pomicanje fokusa s kontrole na kontrolu:

```
Private Sub Form_KeyPress(KeyAscii As Integer)
Dim NexTabIndex As Integer, i As Integer
  If KeyAscii = 13 Then
    If Screen.ActiveControl.TabIndex = Count - 1 Then
      NextTabIndex = 0
    Else
      NextTabIndex = Screen.ActiveControl.TabIndex + 1
    End If
    For i = 0 To Count - 1
      If Me.Controls(i).TabIndex = NextTabIndex Then
        Me.Controls(i).SetFocus
        Exit For
      End If
    Next i
    KeyAscii = 0
  End If
End Sub
```

Budući da ovaj programski kod postavlja argument *keyascii* na 0 kad je on 13, kontrole nikad neće prepoznati kad je pritisnuta tipka ENTER, i njihovi potprogrami događaja tipkovnice nikad neće biti pozvani.

Prekidanje pozadinske obrade

Vaša aplikacija može iskoristiti dugu pozadinsku obradu kako bi ostvarila određene zadatke. Ako je to slučaj, korisno je pružiti korisniku način da se prebaci na drugu aplikaciju, ili da prekine ili otkaže pozadinski zadatak. Radno okruženje Windowsa daje korisnicima prvu mogućnost: prebacivanje na drugu aplikaciju korištenjem kombinacije tipki ALT + TAB, na primjer. Ostale mogućnosti možete pružiti pisanjem programskog koda koji će odgovarati kad korisnik klikne gumb za otkazivanje ili pritisne tipku ESC.

U razmatranju kako to ostvariti u vašoj aplikaciji, važno je razumjeti kako operativni sustav rukuje zadacima iz raznih aplikacija. Windows je prvenstveno višezadaćni operativni sustav, što znači da se vrijeme nezaposlenosti procesora učinkovito dijeli među pozadinskim zadacima. Te pozadinske zadatke može proizvesti aplikacija s kojom radi korisnik, druga aplikacija, ili možda neki događaji kojima upravlja sustav. Međutim, prednost se uvijek daje aplikaciji s kojom radi korisnik. Na taj način uvijek su osigurani trenutni odgovori miša i tipkovnice.

Pozadinska obrada može se svrstati u dvije kategorije: stalnu i povremenu. Primjer stalnog zadatka bilo bi kopiranje datoteke s poslužitelja. Povremeno ažuriranje vrijednosti bilo bi primjer povremenog zadatka. Korisnik može prekinuti ili poništiti oba tipa zadatka. Međutim, budući da je pozadinska obrada obično složen predmet, važno je na prvom mjestu uzeti u obzir kako se ti zadaci pokreću. Idući odlomak “Omogućavanje korisniku da prekine zadatke”, opisuje ta razmatranja i tehnike.

Omogućavanje korisniku da prekine zadatke

Tijekom dugih pozadinskih zadataka, vaša aplikacija ne može odgovoriti na ulazne podatke korisnika. Zbog toga, trebali bi korisniku omogućiti način za prekidanje i otkazivanje pozadinske obrade pisanjem koda za događaje miša ili tipkovnice. Na primjer, kad se izvodi dugi pozadinski zadatak, možete prikazati dijaloški okvir koji sadrži gumb Cancel kojeg korisnik može pokrenuti pritiskom na tipku ENTER (ako je fokus na gumbu Cancel) ili ga može kliknuti mišem.

> Napomena Korisniku također možete dati vidljiv znak kad se obrađuje dug zadatak. Na primjer, možete korisniku pokazati kako napreduje zadatak (upotrebom kontrole Label ili kontrole Gauge, na primjer), ili promjenom pokazivača miša u pješčani sat.

Postoji nekoliko tehnika, ali ne istog načina, za pisanje koda koji će rukovati pozadinskom obradom. Jedan način za omogućavanje korisniku da prekine zadatke je prikaz gumba Cancel i omogućavanje izvođenja njegovog događaja Click. Možete to napraviti postavljanjem koda za vaš pozadinski zadatak u događaj Timer, korištenjem sljedećih smjernica.

- Upotrebljavajte statičke varijable za informaciju koja se mora održati između pojava potprograma događaja Timer.
- Kad događaj Timer dobije nadzor, dopustite mu da se izvodi neznatno duže od vremena kojeg ste odredili za svojstvo Interval. Tako će vaš pozadinski zadatak sigurno iskoristiti svaki komadić procesorskog vremena kojeg mu sustav može dati. Sljedeći događaj Timer će jednostavno čekati u redu poruka sve dok se ne izvede prethodni.
- Upotrebljavajte lijepe velike vrijednosti – pet do deset sekundi – za svojstvo Interval mjerača vremena, jer se na taj način postiže djelotvornija obrada. Ravnopravna višezadaćnost sprječava blokiranje ostalih aplikacija, a korisnici obično podnose neznatno kašnjenje pri otkazivanju dugog zadatka.
- Upotrebljavajte svojstvo Enabled mjerača vremena kao zastavicu za sprečavanje pokretanja pozadinskih zadataka kad se već izvode.

Za više informacija Pogledajte “Korištenje kontrole mjerača vremena” u 7. poglavlju “Korištenje standardnih kontrola Visual Basica”.

Korištenje funkcije DoEvents

Iako su događaji Timer najbolji alat za pozadinsku obradu, posebno za vrlo duge zadatke, funkcija DoEvents pruža prikladan način omogućavanja otkazivanja zadatka. Na primjer, sljedeći programski kod prikazuje gumb “Process” koji se mijenja u gumb “Cancel” kad je kliknut. Ponovni klik prekida zadatak koji taj gumb izvodi.

```
‘ Originalni natpis ovog gumba je “Process”.
Private Sub Command1_Click()
  ‘ Statičke varijable su djeljive za
  ‘ sve primjere potprograma.
  Static blnProcessing As Boolean
  Dim lngCt As Long
  Dim intYieldCt As Integer
  Dim dblDummy As Double
  ‘ Kad je gumb kliknut, ispitivanje
  ‘ je li obrada već u tijeku.
  If blnProcessing Then
    ‘ Ako je obrada u tijeku, poništi ju.
    blnProcessing = False
  Else
    Command1.Caption = “Cancel”
    blnProcessing = True
    lngCt = 0
    ‘ Izvođenje milijun množenja s tekućim
    ‘ zarezom. Nakon svake tisuće,
    ‘ provjera poništavanja.
    Do While blnProcessing And (lngCt < 1000000)
```

```
      For intYieldCt = 1 to 1000
        lngCt = lngCt + 1
        dblDummy = lngCt * 3.14159
      Next intYieldCt
      ‘ Izraz DoEvents dopušta pojavljivanje
      ‘ drugim događajima, uključujući
      ‘ pritisak na ovaj gumb drugi put.
      DoEvents
    Loop
    blnProcessing = False
    Command1.Caption = “Process”
    MsgBox lngCt & “ množenja je izvedeno”
  End If
End Sub
```

Funkcija DoEvents prebacuje nadzor jezgri operativnog okruženja. Nadzor se vraća vašoj aplikaciji odmah čim su sve ostale aplikacije u okruženju imale priliku odgovoriti na neriješene događaje. Ovim načinom trenutna aplikacija neće izgubiti fokus, ali će biti omogućena obrada pozadinskih događaja.

Rezultati ovog ustupanja možda neće uvijek biti ono što ste očekivali. Na primjer, sljedeći programski kod događaja Click čeka deset sekundi nakon klika gumba kako bi prikazao poruku. Ako je gumb kliknut dok već čeka, klikovi će biti završeni u obrnutom redoslijedu.

```
Private Sub Command2_Click()
  Static intKlik As Integer
  Dim intBrojKlika As Integer
  Dim dblVrijemeKraja As Double
    ‘ Svaki put kad je gumb kliknut,
    ‘ daje mu se jedinstven broj.
  intKlik = intKlik + 1
  intBrojKlika = intKlik
    ‘ Deset sekundi čekanja.
  dblVrijemeKraja = Timer + 100
  Do While dblVrijemeKraja > Timer
    ‘ Ne radi ništa osim dopuštanja
    ‘ drugim aplikacijama da obrade
    ‘ svoje događaje.
    DoEvents
  Loop
  MsgBox “Klik” & intBrojKlika & “ je završen.”
End Sub
```

Možda ćete trebati spriječiti potprogram događaja koji predaje nadzor s funkcijom DoEvents od ponovnog pozivanja prije vraćanja funkcije DoEvents. Inače, potprogram bi mogao završiti u beskrajnom pozivanju, sve dok se ne potroše sistemski izvori. Takav događaj možete spriječiti privremenim onesposobljavanjem kontrole ili postavljanjem statične varijable “zastavice”, kao u prethodnom primjeru.

Izbjegavanje funkcije DoEvents kod upotrebe općih podataka

Moglo bi biti savršeno sigurno za funkciju da ponovno bude pozvana kad je ustupila nadzor funkcijom DoEvents. Na primjer, ovaj potprogram ispituje proste brojeve i koristi funkciju DoEvents za povremeno omogućavanje ostalih aplikacija da obrade događaje:

```
Function PrimeStatus(TestVal As Long) As Integer
  Dim Lim As Integer
  PrimeStatus = True
  Lim = Sqr(TestVal)
  For I = 2 To Lim
    If TestVal Mod I = 0 Then
      PrimeStatus = False
      Exit For
    End If
    If I Mod 200 = 0 Then DoEvents
  Next I
End Function
```

Ovaj kod poziva izraz DoEvents jednom u svakih 200 ponavljanja. To omogućuje potprogramu PrimeStatus da nastavi izračunavanja koliko je potrebno sve dok ostatak okoline odgovara na događaje.

Razmotrite što se događa tijekom poziva funkcije DoEvents. Izvođenje koda aplikacije je Privremeno odgođeno dok ostale forme i aplikacije obrađuju događaje. Jedan od tih događaja mogao bi biti klik gumba koji ponovno pokreće potprogram PrimeStatus.

To uzrokuje ponovno pokretanje potprograma PrimeStatus, ali budući da svako pojavljivanje funkcije ima prostor na slogu za svoje parametre i lokalne varijable, nema sukoba. Svakako, ako se PrimeStatus pozove previše puta, mogla bi se pojaviti pogreška Premalo prostora na slogu (Out of Stack Space).

Situacija bi bila bitno drugačija da PrimeStatus koristi ili mijenja varijable na razini modula ili opće podatke. U tom slučaju, izvođenje idućeg primjera potprograma PrimeStatus prije nego što se vrati funkcija DoEvents moglo bi kao rezultat imati drugačije vrijednosti u podacima na razini modula ili u općim podacima od onih prije poziva funkcije DoEvents. Rezultati funkcije PrimeStatus bili bi nakon toga nepredvidljivi.

Za više informacija Pogledajte odlomke “Funkcija DoEvents” i “Postupak Refresh” u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

12. POGLAVLJE

Rad s tekstom i grafikom

Visual Basic uključuje usavršene tekstualne i grafičke sposobnosti za korištenje u vašim aplikacijama. Ako o tekstu razmislite kao o vidljivom elementu, vidjet ćete da veličina, oblik i boja mogu biti iskorišteni za poboljšanje predstavljene informacije. Isto kao što novine koriste naslove, stupce i grafičke oznake za rastavljanje teksta u manje dijelove, svojstva teksta mogu vam pomoći u poboljšavanju važnih pojmova i zanimljivih detalja.

Visual Basic također pruža grafičke sposobnosti koje vam omogućuju veliku fleksibilnost u oblikovanju, uključujući dodavanje animacije prikazivanjem niza slika.

Ovo poglavlje opisuje načine postavljanja i rukovanja tekstom i grafikom. Uključeni su detalji o oblikovanju, pismima, paletama boja te ispisu. Spajanjem ovih sposobnosti s ispravnim pojmovima oblikovanja, možete optimizirati privlačnost i jednostavnost korištenja svoje aplikacije.

Sadržaj

- Rad s pismima
- Prikazivanje teksta na formama i okvirima za sliku
- Oblikovanje brojeva, datuma i vremena
- Rad s odabranim tekstom
- Prenošenje teksta i grafike s objektom Clipboard
- Razumijevanje koordinatnog sustava
- Korištenje grafičkih kontrola
- Korištenje grafičkih postupaka
- Rad s bojama
- Korištenje objekta tipa Picture
- Ispis

Primjeri aplikacija: Blanker.vbp, Palettes.vbp

Neki od primjera koda u ovom poglavlju uzeti su iz primjera Blanker (Blanker.vbp) i Palettes (Palettes.vbp). Ove aplikacije ćete pronaći u direktoriju Samples.

Rad s pismima

Tekst se prikazuje korištenjem *pisma* (*font*) – skupa karaktera istog oblika, dostupnih u određenoj veličini, stilu i težini.

Operativni sustavi Windows 95/98 i Windows NT pružaju vama i vašim korisnicima potpun skup standardnih pisama. Pisma tipa TrueType su promjenjive veličine, što znači da mogu ponovno stvoriti karakter bilo koje veličine. Kad odaberete TrueType pismo, ono se iscrtava u odabranoj veličini točaka i prikazuje kao bitmapirana slika na ekranu.

Kod tiskanja, odabrano TrueType pismo ili pisma se iscrtavaju u odgovarajućoj veličini i zatim šalju pisaču. Zbog toga, nema potrebe za odvajanjem pisama za ekran i za pisač. Međutim, pisma pisača zamijeniti će TrueType pisma, ako je na raspolaganju jednako pismo, što povećava brzinu ispisa.

Odabir pisama za vašu aplikaciju

Zapamtite da korisnik vaše aplikacije možda neće imati pisma koja ste upotrijebili za stvaranje aplikacije. Ako odaberete TrueType pismo kojeg nema korisnik, Windowsi odabiru najsličnije pismo na sustavu korisnika. Ovisno o oblikovanju vaše aplikacije, to može prouzročiti probleme korisniku. Na primjer, pismo koje odaberu Windowsi može povećati tekst tako da će tekstovi natpisa izlaziti iz svojih okvira.

Jedan način izbjegavanja problema s pismima je distribucija potrebnih pisama zajedno s vašom aplikacijom. Vjerojatno ćete trebati dobiti dopuštenje vlasnika autorskog prava pisma kako bi ga mogli distribuirati sa svojom aplikacijom.

Svoju aplikaciju možete također programirati tako da među dostupnim pismima na operativnom sustavu provjeri postojanje pisama koje koristite. Ako se vaše pismo ne nalazi na operativnom sustavu korisnika, možete programirati aplikaciju tako da sama odabere drugačije pismo s liste.

Drugi način zaobilaženja problema s pismima je korištenje pisama koje korisnici najvjerojatnije imaju na svojim sustavima. Ako upotrebljavate pisma iz specifične verzije Windowsa, trebat ćete odrediti tu verziju kao sistemski zahtjev za vašu aplikaciju.

Provjera dostupnih pisama

Vaša aplikacija može lako provjeriti jesu li potrebna pisma dostupna na sustavu i pisaču korisnika. Svojstvo Fonts odgovara objektima Printer i Screen. Matrica koju vraća svojstvo Fonts je popis svih pisama dostupnih pisaču ili ekranu. Možete proći kroz matricu ovog svojstva, te zatim potražiti odgovarajuće stringove imena. Ovaj primjer programskog koda određuje ima li sustav pismo pisača koje odgovara pismu odabrane forme:

```
Private Sub Form_Click()
Dim I As Integer, Zastavica As Boolean
  For I = 0 To Printer.FontCount - 1
    Zastavica = StrComp(Font.Name, Printer.Fonts(I), 1)
    If Zastavica = True Then
          Debug.Print "Postoji odgovarajuće pismo."
          Exit For
    End If
  Next I
End Sub
```

Za više informacija Za informacije o pismima u sustavima istočne Azije, pogledajte "Značenja pisma, prikaza i tiskanja u DBCS okruženju" u 16. poglavlju "Međunarodna izdanja". Za informacije o postavljanju svojstava pisma, pogledajte "Određivanje karakteristika pisma", kasnije u ovom poglavlju.

Stvaranje svojih vlastitih tipova pisma

Ako postavite pokazivač na dodatak Standard OLE Types korištenjem dijaloškog okvira References, možete upotrijebiti klasu StdFont za stvaranje svojih vlastitih tipova pisma. Ako pogledate pretraživač objekata, uočit ćete da tamo postoje klase StdFont i Font. Klasa Font izvedena je iz baze klase StdFont i podržavaju je sve kontrole.

Možete upotrijebiti sljedeću sintaksu:

```
Dim MojePismo As Font
```

Međutim, ne možete upotrijebiti ovo:

```
Dim Moje Pismo As New Font
```

Umjesto toga, kako bi stvorili svoj vlastiti tip pisma ili slike, upotrijebite programski kod sličan sljedećem:

```
Dim MojePismo As New StdFont
With MojePismo
  .Bold = True
  .Name = "Arial"
End With
Set Text1.Font = MojePismo
```

Određivanje karakteristika pisma

Forme, kontrole koje prikazuju tekst (kao tekst ili natpise), te objekti tipa Printer podržavaju svojstvo Font, koje određuje vidljive karakteristike teksta, uključujući:

- Ime pisma (oblik)
- Veličinu pisma (u točkama)
- Posebne karakteristike (podebljano, nakošeno, podvučeno ili prekriženo)

Za detalje o objektu tipa Printer, pogledajte “Ispis iz aplikacije” kasnije u ovom poglavlju.

Određivanje svojstava pisma

Tijekom izrade aplikacije možete odrediti svako od svojstava pisma dvostrukim klikom svojstva Font u prozoru sa svojstvima te postavljanjem svojstava u dijaloškom okviru Fonts.

Tijekom izvođenja aplikacije, karakteristike pisma određujete postavljanjem svojstava objekta Font za svaku formu i kontrolu. Sljedeća tablica opisuje svojstva za objekt Font.

svojstvo	tip	opis
Name	String	Određuje ime pisma, kao Arial ili Courier.
Size	Single	Određuje veličinu pisma u točkama (72 točke po inču kod tiskanja, jedan inč je 2,54 centimetra).
Bold	Boolean	Ako je True, pismo je podebljano.
Italic	Boolean	Ako je True, pismo je nakošeno.
StrikeThrough	Boolean	Ako je True, Visual Basic crta liniju preko teksta.
Underline	Boolean	Ako je True, pismo je podvučeno.
Weight	Integer	Vraća ili postavlja težinu pisma. Iznad određene težine, svojstvo Bold se prisiljava na vrijednost True.

Na primjer, sljedeći izrazi postavljaju razna svojstva pisma za kontrolu natpisa s imenom lblGodinaUDatum:

```
With lblGodinaUDatum.Font
   .Name = “Arial”         ‘ Promjena pisma u Arial.
   .Bold = True            ‘ Podebljavanje pisma.
End With
```

Redoslijed po kojem postavljate svojstva pisma je nebitan, jer sva pisma ne podržavaju sve varijacije pisama. Prvo postavite svojstvo Name. Nakon toga možete postaviti bilo koje od svojstava s vrijednošću tipa Boolean, kao Bold i Italic, na vrijednost True ili False.

Skup svojstava pisma možete također spremiti u objekt Fonts. Možete odrediti objekt Font kao što određujete i svaki drugi objekt, korištenjem klase StdFont:

```
Dim MojePismo As New StdFont
With MojePismo
   .Name = "Arial"
   .Size = 10
   .Bold = True
End With
```

Napomena Prije nego što stvorite objekt Font, morate upotrijebiti dijaloški okvir References (dostupan iz izbornika Project) za stvaranje pokazivača na dodatak Standard OLE Types.

Nakon toga se možete lako prebacivati s jednog skupa svojstava pisma na drugi, postavljanjem objekta Font forme ili kontrole na novi objekt:

```
Set lblGodinaUDatum.Font = MojePismo
```

Za više informacija Pogledajte "Objekt Font" u priručniku *Microsoft Visual Basic 6.0 Language Reference* biblioteke *Microsoft Visual Basic 6.0 Language Reference Library*.

Rad s malim pismima

Neka pisma ne podržavaju veličine manje od 8 točaka. Kad postavite svojstvo Size za jedno od takvih pisama na veličinu manju od 8 točaka, svojstvo Name ili svojstvo Size će se automatski promijeniti u drugačije pismo ili drugačiju veličinu. Kako bi izbjegli nepredvidljive rezultate, svaki put kad postavite svojstvo Size na veličinu pisma manju od 8 točaka, ponovno ispitajte vrijednosti svojstva Name i svojstva Size nakon takvog postavljanja.

Primjena svojstava pisma za specifične objekte

Učinak postavljanja svojstava pisma razlikuje se ovisno o tehnici upotrijebljenoj za prikaz teksta. Ako je tekst određen sa svojstvom (kao svojstva Text ili Caption), tad promjena svojstva pisma utječe na cijeli tekst u toj kontroli. Kontrole natpisa, okviri s tekstom, kontrole okvira, naredbeni gumbi, kontrolne kućice i sve kontrole datotečnog sustava koriste svojstvo za određivanje teksta.

Ako aplikacija prikazuje tekst postupkom Print, tad promjena svojstva pisma utječe na sve primjene tog postupka nakon promjene svojstva. Tekst ispisan prije promjene svojstva neće biti promijenjen. Samo forme, okviri za sliku, te objekti Debug i Printer podržavaju postupak Print.

Budući da promjene u svojstvima pisma utječu na sav tekst u okvirima s tekstom i kontrolama natpisa, u tim kontrolama ne možete miješati pisma. Ako trebate miješati pisma (na primjer, kako bi podebljali neke riječi, ali ostavili ostale u normalnom pismu), stvorite okvir za sliku i upotrijebite postupak Print za prikaz teksta. Odlomak "Prikazivanje teksta na formama i okvirima za sliku", kasnije u ovom poglavlju, objašnjava kako upotrebljavati postupak Print.

Svojstvo FontTransparent

Forme i okviri za sliku imaju dodatno svojstvo pisma, FontTransparent. Kad je svojstvo FontTransparent postavljeno na True, pozadina se prikazuje putem teksta prikazanom na formi ili okviru za sliku. Slika 12.1 pokazuje učinke svojstva FontTransparent.

Slika 12.1 Učinci svojstva FontTransparent

svojstvo FontTransparent postavljeno na True

svojstvo FontTransparent postavljeno na False

Prikazivanje teksta na formama i okvirima za sliku

Kako bi prikazali tekst na formi ili okviru za sliku, upotrijebite postupak Print ispred kojeg se nalazi ime forme ili okvira za sliku. Kako bi poslali izlazni tekst na pisač, upotrijebite postupak Print u objektu Printer.

Korištenje postupka Print

Sintaksa postupka Print je:

[*objekt.*] **Print** [*listaizlaza*] [{ ; | , }]

Argument *objekt* je neobavezan; ako je izostavljen, postupak Print primjenjuje se za trenutnu formu.

Na primjer, sljedeći izrazi ispisuju poruke na:

- Formi imena MojaForma:

```
MojaForma.Print "Ovo je forma."
```

- Okviru za sliku imena picMalaPoruka:

```
picMalaPoruka.Print "Ovo je okvir za sliku."
```

- Trenutnoj formi:

```
Print "Ovo je trenutna forma."
```

- objektu Printer:

```
Printer.Print "Ovaj tekst ide na pisač."
```

Argument *listaizlaza* je tekst koji se pojavljuje na formi ili okviru za sliku. Više stavki u argumentu *listaizlaza* moraju biti razdvojene zarezom ili točka-zarezom ili s oboje,

kao što je objašnjeno u odlomku “Prikaz različitih stavki u istoj liniji”, kasnije u ovom poglavlju.

Odrezani tekst

Ako je forma ili okvir za sliku premalen za prikaz cijelog teksta, tekst će biti odrezan. Gdje će forma ili okvir za sliku odrezati tekst ovisi o koordinatama položaja na kojem ste počeli ispisivati tekst. Ne možete se pomicati kroz formu ili okvir za sliku.

Slojevitost

Kad ispisujete tekst na formu, tekst se pojavljuje u sloju koji se nalazi *ispod* svih kontrola postavljenih na formu. Ispis na formu će tako obično najbolje raditi na formama posebno stvorenim za sadržavanje teksta. Za više informacija o tome kako se tekst i grafika pojavljuju u slojevima na formi, pogledajte “Slojevitost grafike sa svojstvima AutoRedraw i ClipControls”, kasnije u ovom poglavlju.

Prikaz različitih stavki u istoj liniji

Stavke koje prikazujete ili ispisujete mogu uključivati vrijednosti svojstava, konstante i varijable (tekstualne ili brojčane). Postupak Print, raspravljen u odlomku “Prikazivanje teksta na formama i okvirima za sliku” ispisuje vrijednosti brojčanih stavki. Kod ispisa brojeva pozitivnih vrijednosti ispisuje se znak razmaka prije i poslije broja. Brojevi negativnih vrijednosti prikazuju znak minusa umjesto vodećeg razmaka.

Upotrijebite točku-zarez (;) ili zarez (,) za odvajanje jedne stavke od druge. Ako koristite točku-zarez, Visual Basic ispisuje jednu stavku iza druge, bez ubačenih razmaka. Ako koristite zarez, Visual Basic nastavlja ispis sljedeće stavke na mjestu idućeg tabulatora stupca.

Na primjer, sljedeći izraz ispisuje na trenutnu formu:

```
Print “Vrijednost X-a je “; X; “a vrijednost Y-a je”; Y
```

Ako varijabla X sadrži vrijednost 2, a varijabla Y vrijednost 7, izraz će dati ovakav izlaz:

```
Vrijednost X-a je 2, a vrijednost Y-a je 7
```

U pravilu, svaki postupak Print ispisuje tekst i pomiče se na iduću liniju. Ako nema stavki, postupak Print jednostavno preskače liniju. Niz izraza Print (u sljedećem primjeru, za okvir za sliku s imenom picBrojLinije) automatski koristi odvojene linije:

```
picBrojLinije.Print “Ovo je linija 1.”
picBrojLinije.Print “Ovo je linija 2.”
```

Međutim, postavljanjem točke-zareza (ili zareza) na kraj prvog izraza, uzrokovat ćete pojavljivanje izlaza idućeg izraza Print u istoj liniji:

```
picBrojLinije.Print “Ovo sve se pojavljuje “;
picBrojLinije.Print “u istoj liniji.”
```

Prikaz izlaza naredbe Print na određenom položaju

Postavljanje izlaza naredbe Print možete nadzirati određivanjem koordinata crtanja, korištenjem jedne ili obje sljedeće tehnike:

- Korištenjem postupka Cls (clear, očisti) za brisanje forme ili okvira za sliku te obnavljanje koordinata na početak (0, 0).
- Postavljanjem koordinata crtanja svojstvima CurrentX i CurrentY.

Postupak Cls

Sav tekst i grafika na objektu koji su stvoreni postupkom Print i grafičkim postupcima mogu biti obrisani postupkom Cls. Postupak Cls također obnavlja koordinate crtanja na početak (0, 0), što je u pravilu gornji lijevi kut. Na primjer, ovi izrazi brišu:

- Okvir za sliku imena Picture1:

```
Picture1.Cls
```

- Trenutnu formu:

```
Cls
```

Određivanje koordinata crtanja

Koordinate crtanja formi i okvira za sliku možete izravno odrediti svojstvima CurrentX i CurrentY. Na primjer, ovi izrazi obnavljaju koordinate crtanja na gornji lijevi kut za kontrolu Picture1 te za trenutnu formu:

- Okvir za sliku imena Picture1:

```
Picture1.CurrentX = 0
Picture1.CurrentY = 0
```

- Trenutna forma:

```
CurrentX = 0
CurrentY = 0
```

Svaki novi tekst kojeg ispisujete pojavljuje se iznad svakog teksta i grafike koje se već nalaze na tom položaju. Kako bi obrisali samo dio teksta, nacrtajte okvir postupkom Line i popunite ga bojom pozadine. Imajte na umu da koordinate crtanja određene svojstvima CurrentX i CurrentY obično mijenjaju položaj kad koristite grafičke postupke.

U pravilu, forme i okviri za sliku koriste koordinatni sustav gdje svaka jedinica predstavlja twip (1440 twipova jednako je inču, a otprilike 567 twipova jednako je centimetru). Možda ćete trebati promijeniti svojstvo Scale forme, okvira za sliku ili objekta

Printer iz twipova u točke (points), zato što se visina teksta mjeri u točkama. Korištenje iste jedinice mjere za tekst i za objekt u koji ćete ispisivati tekst olakšava izračunavanje položaja teksta.

Za više informacija Za više informacija o twipovima i koordinatama crtanja, pogledajte "Razumijevanje koordinatnog sustava", kasnije u ovom poglavlju.

Postupci TextHeight i TextWidth

Prije korištenja postupka Print, možete upotrijebiti postupke TextHeight i TextWidth za utvrđivanje gdje postaviti svojstva CurrentX i CurrentY. Postupak TextHeight vraća visinu linije teksta, uzimajući u obzir veličinu i stil pisma u objektu. Sintaksa je:

[*objekt.*] **TextHeight** (*string*)

Ako argument *string* sadrži ugrađene karaktere oznake kraja reda (Chr(13)), tada tekst sadrži više linija, a postupak TextHeight vraća visinu broja linija u tekstu koji se nalazi u ovom argumentu. Ako ne postoje ugrađeni karakteri oznake kraja reda, postupak TextHeight uvijek vraća visinu jedne linije teksta.

Jedan način korištenja postupka TextHeight je postavljanje svojstva CurrentY na određenu liniju. Na primjer, sljedeći izrazi postavljaju koordinate crtanja na početak pete linije:

```
CurrentY = TextHeight("primjer") * 4
CurrentX = 0
```

Pretpostavljajući da ne postoje oznake kraja reda u uzorku teksta, trebate koristiti ovu sintaksu za postavljanje svojstva CurrentY na *n-tu* liniju:

CurrentY = [*objekt.*] **TextHeight** (*string*) * (n – 1)

Ako je izostavljen argument *objekt*, postupak se primjenjuje za trenutnu formu. Argument *objekt* može biti forma, okvir za sliku ili objekt Printer.

Postupak TextWidth vraća širinu stringa, uzimajući u obzir veličinu i stil pisma objekta. Ovaj postupak je koristan zato što većina pisama ima karaktere razmjerne širine. Postupak TextWidth pomaže vam u otkrivanju je li širina stringa veća od širine forme, okvira za sliku ili objekta Printer.

Na primjer, sljedeći izrazi koriste postupke TextWidth i TextHeight za centriranje teksta u okvir postavljanjem svojstava CurrentX i CurrentY. Ime okvira u ovom primjeru je Jelovnik.

```
CurrentX = (BoxWidth - TextWidth("Jelovnik")) / 2
CurrentY = (BoxHeight - TextHeight("Jelovnik")) / 2
```

Za više informacija Pogledajte odlomke "Postupak TextHeight" i "Postupak TextWidth" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Oblikovanje brojeva, datuma i vremena

Visual Basic pruža veliku fleksibilnost u prikazivanju oblika brojeva, kao i oblika datuma i vremena. Možete jednostavno prikazati međunarodne oblike za brojeve, datume i vremena.

Funkcija Format pretvara brojčanu vrijednost u tekstualni string i daje vam nadzor nad izgledom stringa. Na primjer, možete odrediti broj decimalnih mjesta, nule na početku ili kraju, te oblike valute. Sintaksa je:

Format(*izraz* [, *oblik* [, *prvidantjedna* [, *prvidangodine*]]])

Argument *izraz* određuje broj za pretvaranje, a argument *oblik* je string sastavljen od simbola koji pokazuju kako oblikovati broj. Najčešće korišteni simboli su ispisani u sljedećoj tablici.

simbol	opis
0	Oznaka mjesta znamenke; ispisuje nulu na početku ili kraju na tom položaju, ako je potrebno.
#	Oznaka mjesta znamenke; nula na početku ili kraju se nikad ne ispisuje.
.	Decimalna oznaka mjesta.
,	Razdvojnik tisuća.
- + $ ( ) razmak	Slovčani karakter; karakteri se prikazuju upravo kako su upisani u string za oblikovanje.

Argument *prvidantjedna* je konstanta koja određuje prvi dan tjedna; argument *prvidangodine* je konstanta koja određuje prvi dan godine. Oba argumenta su neobavezna. Za više informacija o ovim konstantama, pogledajte odlomak “Funkcija Format” u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Imenovani oblici

Visual Basic pruža nekoliko standardnih oblika za korištenje s funkcijom Format. Umjesto označavanja simbola u argumentu *oblik*, možete odrediti te oblike imenom u argumentu *oblik* funkcije Format. Uvijek zatvorite ime oblika u dvostruke navodnike (“”).

Sljedeća tablica ispisuje imena oblika koje možete upotrebljavati.

imenovani oblik	opis
General Number	Prikazuje brojeve bez razdvojnika tisuća.
Currency	Prikazuje brojeve s razdvojnikom tisuća, ako je potrebno; prikazuje dvije znamenke desno od decimalnog razdvojnika. Izlaz se temelji na postavkama sustava korisnika.
Fixed	Prikazuje najmanje jednu znamenku lijevo i dvije znamenke desno od decimalnog razdvojnika.

imenovani oblik	opis
Standard	Prikazuje broj s razdvojnikom tisuća, najmanje jednu znamenku lijevo i dvije znamenke desno od decimalnog razdvojnika.
Percent	Množi vrijednost sa 100 sa znakom postotka na kraju.
Scientific	Koristi standardnu znanstvenu notaciju.
General Date	Prikazuje datum i vrijeme ako argument *izraz* sadržava oboje. Ako je argument *izraz* samo datum ili vrijeme, nedostajuća informacija se ne prikazuje. Prikaz datuma je određen postavkama sustava korisnika.
Long Date	Koristi oblik Long Date određen postavkama sustava korisnika.
Medium Date	Koristi oblik dd-mmm-gg (na primjer, 03-Apr-93). Prikaz datuma je određen postavkama sustava korisnika.
Short Date	Koristi oblik Short Date određen postavkama sustava korisnika.
Long Time	Prikazuje vrijeme korištenjem oblika određenog postavkama sustava korisnika; uključuje sate, minute i sekunde.
Medium Time	Prikazuje sate, minute i oznaku “AM” ili “PM” korištenjem oblika “hh:mm AM/PM”.
Short Time	Prikazuje sate i minute korištenjem oblika hh:mm.
Yes/No	Svaka brojčana vrijednost različita od nule (obično -1) je Yes. Nula je No.
True/False	Svaka brojčana vrijednost različita od nule (obično -1) je True. Nula je False.
On/Off	Svaka brojčana vrijednost različita od nule (obično -1) je On. Nula je Off.

Funkcija Format podržava puno drugih posebnih znakova, kao što su oznaka mjesta postotka i potencije.

Za više informacija Pogledajte odlomak “Funkcija Format” u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Oblici brojeva

Sljedeća pretvaranja brojeva pretpostavljaju da je u kontrolnom panelu Windowsa država postavljena na “English (United States)”.

sintaksa oblika	rezultat
Format(8315.4, “00000.00”)	08315.40
Format(8315.4, “#####.##”)	8315.4
Format(8315.4, “##,##0.00”)	8,315.40
Format(315.4, “$##0.00”)	$315.40

U ovim primjerima, simbol za decimalni razdvojnik je točka (.), a simbol za razdvojnik tisuća je zarez (,). Međutim, karakter razdvojnika koji se stvarno prikazuje ovisi o državi određenoj u kontrolnom panelu Windowsa.

Ispis oblikovanih datuma i vremena

Kako bi ispisali oblikovane datume i vremena, upotrijebite funkciju Format sa simbolima koji predstavljaju datum i vrijeme. Ovi primjeri koriste funkcije Now i Format za prepoznavanje i oblikovanje trenutnog datuma i vremena. Sljedeći primjeri pretpostavljaju da je dijaloški okvir Regional Settings kontrolnog panela Windowsa postavljen na "English (United States)".

sintaksa oblika	rezultat
Format(Now, "*m/d/yy*")	1/27/93
Format(Now, "*dddd, mmmm dd, yyyy*")	Wednesday, January 27, 1993
Format(Now, "*d-mmm*")	27-Jan
Format(Now, "*mmmm-yy*")	January-93
Format(Now, "*hh:mm* AM/PM")	07:18 AM
Format(Now, "*h:mm:ss* a/p")	7:18:00 a
Format(Now, "*d-mmmm h:mm*")	3-January 7:18

Korištenjem funkcije Now s oblikom "*ddddd*" i "*ttttt*", možete ispisati trenutan datum i vrijeme u obliku koji odgovara odabiru u dijaloškom okviru Regional Settings kontrolnog panela Windowsa.

država	sintaksa oblika	rezultat
Švedska	Format(Now, "*ddddd ttttt*")	1992-12-31 18:22:38
Velika Britanija	Format(Now, "*ddddd ttttt*")	31/12/92 18:22:38
Kanada (francuski)	Format(Now, "*ddddd ttttt*")	92-12-31 18:22:38
SAD	Format(Now, "*ddddd ttttt*")	12/31/92 6:22:38 PM

Za više informacija Za više informacija o razmišljanjima o međunarodnim postavkama kod upotrebe funkcije Format, pogledajte "Funkcije svjesne poprišta" u 16. poglavlju "Međunarodna izdanja". Za više informacija o datumima temeljenim na mjestu sustava, pogledajte "Pisanje međunarodnog koda u Visual Basicu" u 16. poglavlju "Međunarodna izdanja".

Rad s odabranim tekstom

Okviri s tekstom i kombinirani okviri imaju niz svojstava za odabrani tekst koja su posebno korisna u radu s odlagalištem (Clipboard). Ta svojstva, koja se odnose na blok odabranog (osvijetljenog) teksta unutar kontrole, omogućuju vam da stvorite funkcije

rezanja i lijepljenja za korisnika. Sljedeća svojstva mogu biti mijenjana tijekom izvođenja aplikacije.

svojstvo	opis
SelStart	Cijeli broj tipa Long koji određuje početni položaj odabranog bloka teksta. Ako nema odabranog teksta, ovo svojstvo određuje položaj mjesta ubacivanja. Postavka od 0 pokazuje položaj točno ispred prvog karaktera u okviru s tekstom ili kombiniranom okviru. Postavka jednaka duljini teksta u okviru s tekstom ili kombiniranom okviru ukazuje na položaj točno iza posljednjeg karaktera u kontroli.
SelLength	Cijeli broj tipa Long koji određuje broj odabranih karaktera.
SelText	String koji sadrži odabrane karaktere (ili prazan string, ako nema odabranih karaktera).

Postavljanjem svojstava SelStart i SelLength možete nadzirati koji je tekst odabran. Na primjer, ovi izrazi odabiru sav tekst u okviru s tekstom:

```
Text1.SetFocus
‘ Početak odabira prije prvog karaktera.
Text1.SelStart = 0
‘ Odabir do kraja teksta.
Text1.SelLength = Len(Text1.Text)
```

Ako svojstvu SelText dodijelite novi string, taj string zamjenjuje odabrani tekst, a mjesto ubacivanja postavlja se točno nakon kraja novoubačenog teksta. Na primjer, sljedeći izraz zamjenjuje odabrani tekst stringom “Upravo sam ubačen!”:

```
Text1.SelText = “Upravo sam ubačen!”
```

Ako nema odabranog teksta, string se jednostavno uljepljuje u okvir s tekstom na mjesto ubacivanja.

Za više informacija Pogledajte odlomke “Svojstvo SelStart” “Svojstvo SelLength” i “Svojstvo SelText” u priručniku *Microsoft Visual Basic 6.0 Controls Reference* biblioteke *Microsoft Visual Basic 6.0 Reference Library*.

Prenošenje teksta i grafike objektom Clipboard

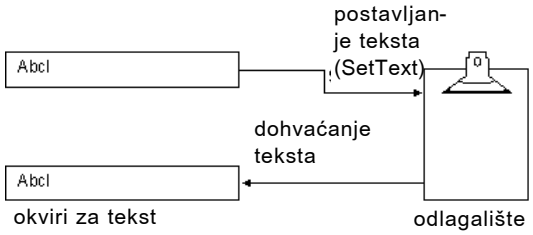
Objekt Clipboard (odlagalište) nema svojstava ni događaja, ali ima nekoliko postupaka koji vam omogućuju prenošenje podataka u i iz odlagališta okoline. Postupci odlagališta dijele se u tri kategorije. Postupci GetText i SetText koriste se za prenošenje teksta. Postupci GetData i SetData prenose grafiku. Postupci GetFormat i Clear rade i s tekstualnim i s grafičkim oblicima.

Za više informacija Za informacije u prenošenju podataka unutar vaše aplikacije ili između aplikacija, pogledajte “Povlačenje i ispuštanje tipa OLE” u 11. poglavlju “Odgovaranje na događaje miša i tipkovnice”.

Rezanje, kopiranje i lijepljenje teksta s odlagalištem

Dva od najkorisnijih postupaka odlagališta su postupci SetText i GetText. Ova dva postupka prenose podatke tipa string u i iz odlagališta, kao što je prikazano na slici 12.2.

Slika 12.2 Micanje podataka u i iz odlagališta postupcima SetText i



GetText

Postupak SetText kopira tekst u odlagalište, zamjenjujući svaki tekst koji je tu prije bio spremljen. Postupak SetText koristite kao izraz. Sintaksa je:

Clipboard.SetText *podaci* [, *oblik*]

Postupak GetText vraća tekst spremljen u odlagalištu. Koristite ga kao funkciju:

odredište = **Clipboard.GetText()**

Kombiniranjem postupaka SetText i GetText sa svojstvima odabiranja predstavljenim u odlomku “Rad s odabranim tekstom”, možete lako napisati naredbe za kopiranje, rezanje i lijepljenje za okvir s tekstom. Sljedeći potprogrami događaja ostvaruju te naredbe za kontrole s imenima mnuKopiraj, mnuIzreži i mnuUlijepi:

```
Private Sub mnuKopiraj_Click()
  Clipboard.Clear
  Clipboard.SetText Text1.SelText
End Sub

Private Sub mnuIzreži_Click()
  Clipboard.Clear
  Clipboard.SetText Text1.SelText
  Text1.SelText = “”
End Sub

Private Sub mnuUlijepi_Click()
  Text1.SelText = Clipboard.GetText()
End Sub
```

Napomena Primjer najbolje radi ako postoje kontrole izbornika, jer možete koristiti izbornike dok kontrola Text1 ima fokus.

Uočite da potprogrami Kopiraj i Izreži najprije prazne odlagalište postupkom Clear. Odlagalište se ne čisti automatski zato što možda trebate u odlagalište smjestiti podatke u nekoliko raznih oblika, kao što je opisano u odlomku "Rad s više oblika u odlagalištu" kasnije u ovom poglavlju. Oba potprograma, Kopiraj i Izreži, zatim kopiraju odabrani tekst iz okvira s tekstom Text1 u odlagalište sljedećim izrazom:

```
Clipboard.SetText Text1.SelText
```

U naredbi Ulijepi, postupak GetText vraća string teksta koji je trenutno u odlagalištu. Izraz dodjeljivanja zatim kopira taj string u odabrani dio okvira s tekstom (Text1.SelText). Ako trenutno nema odabranog teksta, Visual Basic postavlja taj tekst na mjesto ubacivanja u okviru s tekstom:

```
Text1.SelText = Clipboard.GetText()
```

Ovaj kod pretpostavlja da se sav tekst prebacuje u i iz okvira s tekstom Text1, ali korisnik može kopirati, izrezivati i uljepljivati između okvira Text1 i kontrola na drugim formama.

Budući da cijelo okruženje koristi odlagalište, korisnik također može prenositi tekst između okvira Text1 i bilo koje aplikacije korištenjem odlagališta.

Rad sa svojstvom ActiveControl

Ako želite da naredbe Kopiraj, Izreži i Ulijepi rade sa svakim okvirom s tekstom koji ima fokus, upotrijebite svojstvo ActiveControl objekta Screen. Sljedeći programski kod pruža pokazivač na svaku kontrolu koja ima fokus:

```
Screen.ActiveControl
```

Ova dio koda možete upotrijebiti kao i svaki drugi pokazivač na kontrolu. Ako znate da je kontrola okvir s tekstom, možete ukazati na bilo koje od svojstava podržanih od okvira s tekstom, uključujući svojstva Text, SelText i SelLength. Sljedeći kod pretpostavlja da je okvir s tekstom aktivna kontrola, i koristi svojstvo SelText:

```
Private Sub mnuKopiraj_Click()
  Clipboard.Clear
  Clipboard.SetText Screen.ActiveControl.SelText
End Sub

Private Sub mnuIzreži_Click()
  Clipboard.Clear
  Clipboard.SetText Screen.ActiveControl.SelText
  Screen.ActiveControl.SelText = ""
End Sub

Private Sub mnuUlijepi_Click()
  Screen.ActiveControl.SelText = Clipboard.GetText()
End Sub
```

Rad s više oblika u odlagalištu

U odlagalište zapravo možete istovremeno postaviti više dijelova podataka, sve dok je svaki dio drugačijeg oblika. To je korisno zato što ne znate koja će aplikacija uljepljivati podatke, pa pribavljanje podataka u nekoliko raznih oblika proširuje mogućnost da ćete ih pružiti u obliku koji može upotrijebiti druga aplikacija. Ostali postupci odlagališta – GetData, SetData i GetFormat – omogućuju vam da postupate s oblicima podataka različitim od teksta dobavljanjem broja koji određuje oblik. Ovi oblici su opisani u sljedećoj tablici, zajedno s pripadajućim brojem.

konstanta	opis
vbCFLink	Veza za razmjenu dinamičkih podataka.
vbCFText	Tekst. Svi primjeri ranije u ovom poglavlju upotrebljavaju ovaj oblik.
vbCFBitmap	Bitmapirana slika.
vbCFMetafile	Metadatoteka.
vbCFDIB	Bitmapirana slika neovisna o uređaju.
vbCFPalette	Paleta boja.

Posljednja četiri oblika možete upotrijebiti kad izrezujete i uljepljujete podatke iz kontrola okvira za sliku. Sljedeći kod pruža općenite naredbe izrezivanja, kopiranja i lijepljenja koje rade sa svakom od standardnih kontrola.

```
Private Sub mnuKopiraj_Click()
  Clipboard.Clear
  If TypeOf Screen.ActiveControl Is TextBox Then
    Clipboard.SetText Screen.ActiveControl.SelText
  ElseIf TypeOf Screen.ActiveControl Is ComboBox Then
    Clipboard.SetText Screen.ActiveControl.Text
  ElseIf TypeOf Screen.ActiveControl Is PictureBox Then
    Clipboard.SetData Screen.ActiveControl.Picture
  ElseIf TypeOf Screen.ActiveControl Is ListBox Then
    Clipboard.SetText Screen.ActiveControl.Text
  Else
    ‘ Ni jedna akcija nema smisla za ostale kontrole.
  End If
End Sub

  Private Sub mnuIzreži_Click()
    ‘ Najprije treba napraviti isto kao kod kopiranja.
    mnuKopiraj_Click
    ‘ Sada se briše sadržaj aktivne kontrole.
    If TypeOf Screen.ActiveControl Is TextBox Then
       Screen.ActiveControl.SelText = “”
  ElseIf TypeOf Screen.ActiveControl Is ComboBox Then
```

```
    Screen.ActiveControl.Text = “”
  ElseIf TypeOf Screen.ActiveControl Is PictureBox Then
    Screen.ActiveControl.Picture = LoadPicture()
  ElseIf TypeOf Screen.ActiveControl Is ListBox Then
    Screen.ActiveControl.RemoveItem Screen.ActiveControl.ListIndex
  Else
    ‘ Ni jedna akcija nema smisla za ostale kontrole.
  End If
End Sub

Private Sub mnuUlijepi_Click()
  If TypeOf Screen.ActiveControl Is TextBox Then
    Screen.ActiveControl.SelText = Clipboard.GetText()
  ElseIf TypeOf Screen.ActiveControl Is ComboBox Then
    Screen.ActiveControl.Text = Clipboard.GetText()
  ElseIf TypeOf Screen.ActiveControl Is PictureBox Then
    Screen.ActiveControl.Picture = Clipboard.GetData()
  ElseIf TypeOf Screen.ActiveControl Is ListBox Then
    Screen.ActiveControl.AddItem Clipboard.GetText()
  Else
    ‘ Ni jedna akcija nema smisla za ostale kontrole.
  End If
End Sub
```

Provjera oblika podataka u odlagalištu

Postupak GetFormat možete upotrijebiti za utvrđivanje jesu li podaci u odlagalištu određenog oblika. Na primjer, možete onemogućiti naredbu Ulijepi ovisno u tome jesu li podaci u odlagalištu sukladni s trenutno aktivnom kontrolom.

```
Private Sub mnuEditiraj_Click()
‘ događaj Click za izbornik Editiraj.
  mnuIzreži.Enabled = True
  mnuKopiraj.Enabled = True
  mnuUlijepi.Enabled = False
  If TypeOf Screen.ActiveControl = TextBox Then
    If Clipboard.GetFormat(vbCFText) Then _
    mnuUlijepi.Enabled = True
  ElseIf TypeOf Screen.ActiveControl Is ComboBox Then
    If Clipboard.GetFormat(vbCFText) Then _
    mnuUlijepi.Enabled = True
  ElseIf TypeOf Screen.ActiveControl Is ListBox Then
    If Clipboard.GetFormat(vbCFText) Then _
    mnuUlijepi.Enabled = True
  ElseIf TypeOf Screen.ActiveControl Is PictureBox Then
    If Clipboard.GetFormat(vbCFBitmap) Then mnuUlijepi.Enabled = True
```

```
    Else
      ‘ Ne može se izrezivati ili kopirati iz
      ‘ ostalih tipova kontrola.
      mnuIzreži.Enabled = False
      mnuKopiraj.Enabled = False
    End If
End Sub
```

Napomena Možete također provjeriti ostale oblike podataka s konstantama vbCFPalette, vbCFDIB i vbCFMetafile. Ako želite zamijeniti paletu slike korištenjem operacija odlagališta, bolje je iz odlagališta zatražiti konstantu vbCFBitmap nego konstantu vbCFDIB. Pogledajte “Rad s 256 boja”, kasnije u ovom poglavlju, za više informacija o radu sa paletom boja.

Za više informacija Pogledajte “Objekt Clipboard” u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Razumijevanje koordinatnog sustava

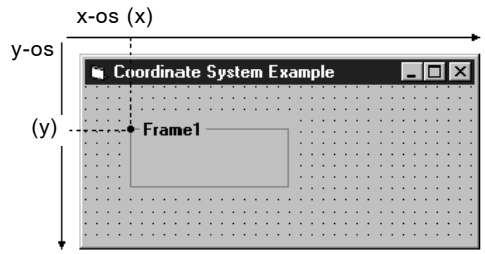
Svaka grafička operacija opisana u ovom poglavlju (uključujući promjenu veličine, pomicanje i crtanje) koristi koordinatni sustav područja crtanja ili spremnika. Iako možete upotrijebiti koordinatni sustav za postizanje grafičkih efekata, također je važno znati kako koristiti koordinatni sustav za određivanje položaja formi i kontrola u vašoj aplikaciji.

Koordinatni sustav je dvodimenzionalna mreža koja određuje položaje na ekranu, u formi ili drugom spremniku (kao što su okvir za sliku ili objekt Printer). Položaj na toj mreži određujete korištenjem koordinata u obliku:

(*x, y*)

Vrijednost x-a je položaj točke duž x-osi, s podrazumijevanim položajem od 0 na lijevom kraju. Vrijednost y-a je položaj točke duž y-osi, s podrazumijevanim položajem od 0 na gornjem kraju. Ovaj koordinatni sustav je prikazan na slici 12.3.

Slika 12.3 Koordinatni sustav forme



Za koordinatni sustav Visual Basica primjenjuju se sljedeća pravila:

- Kad pomaknete kontrolu ili joj promijenite veličinu, koristite koordinatni sustav spremnika kontrole. Ako nacrtate objekt izravno na formi, forma je spremnik. Ako nacrtate kontrolu unutar kontrole okvira ili okvira za sliku, kontrola okvira ili okvir za sliku je spremnik.
- Svi grafički postupci i postupci tipa Print koriste koordinatni sustav spremnika. Na primjer, izrazi koji crtaju unutar okvira za sliku koriste koordinatni sustav te kontrole.
- Izrazi koji mijenjaju veličinu ili pomiču formu uvijek izražavaju položaj i veličinu forme u twipovima.

  Kad stvorite kod za promjenu veličine ili pomicanje forme, uvijek najprije trebate provjeriti svojstva Height i Width objekta Screen kako bi bili sigurni da će se forma uklopiti u ekran.
- Gornji lijevi kut ekrana je uvijek (0, 0). Podrazumijevan koordinatni sustav svakog spremnika počinje s koordinatama (0, 0) u gornjem lijevom kutu spremnika.

Jedinice mjere koje se koriste za određivanje položaja duž ovih osi zbirno se nazivaju *mjerilo* (*scale*). U Visual Basicu, svaka os u koordinatnom sustavu ima svoje vlastito mjerilo.

Možete promijeniti smjer osi, početnu točku i mjerilo koordinatnog sustava, ali za sada upotrebljavajte standardni sustav. Odlomak “Promjena koordinatnog sustava objekta”, kasnije u ovom poglavlju, raspravlja kako napraviti takve promjene.

Objašnjenje twipova

U pravilu, sva pomicanja, promjene veličine, te izrazi za grafičko crtanje Visual Basica koriste jedinicu od jednog twipa. *Twip* je jedna dvadesetina (1/20) točke ispisa (1440 twipova jednako je jednom inču, a 567 twipova jednako je jednom centimetru). Ovakve mjere određuju veličinu objekta kod ispisa. Stvarni fizički razmaci na ekranu različiti su ovisno o veličini monitora.

Promjena koordinatnog sustava objekta

Koordinatni sustav za određen objekt (formu ili kontrolu) postavljate korištenjem svojstava mjerila i postupkom Scale. Koordinatni sustav možete koristiti na jedan od tri različita načina:

- Upotrijebite podrazumijevano mjerilo.
- Odaberite jedno od više standardnih mjerila.
- Stvorite korisničko mjerilo.

Promjena mjerila koordinatnog sustava može pojednostaviti položaj i promjenu veličine grafike na formi. Na primjer, aplikacija koja stvara stupčaste grafikone u okviru za sliku može promijeniti koordinatni sustav kako bi podijelila kontrolu u četiri stupca,

gdje svaki predstavlja stupac u grafikonu. Sljedeći odlomci objašnjavaju kako postaviti podrazumijevana, standardna i korisnička mjerila za promjenu koordinatnog sustava.

Korištenje podrazumijevanog mjerila

Svaka forma i okvir za sliku ima nekoliko svojstava mjerila (ScaleLeft, ScaleTop, ScaleWidth, ScaleHeight i ScaleMode) i jedan postupak (Scale), koje možete upotrijebiti za određivanje koordinatnog sustava. Podrazumijevano mjerilo za objekte u Visual Basicu postavlja koordinate (0, 0) u gornji lijevi kut objekta. Podrazumijevano mjerilo koristi twipove.

Ako se želite vratiti na podrazumijevano mjerilo, upotrijebite postupak Scale bez argumenata.

Odabir standardnog mjerila

Umjesto izravnog određivanja jedinica, možete ih odrediti u granicama standardnog mjerila, postavljanjem svojstva ScaleMode na jedno od postavki prikazano u sljedećoj tablici.

postavka svojstva ScaleMode	opis
0	Korisnički određeno. Ako izravno postavite vrijednosti svojstava ScaleWidth, ScaleHeight, ScaleTop ili ScaleLeft, svojstvo ScaleMode se automatski postavlja na 0.
1	Twipovi. Ovo je podrazumijevano mjerilo. U jednom inču je 1440 twipova.
2	Točke. U jednom inču su 72 točke.
3	Pikseli. Piksel je najmanja jedinica razlučivosti monitora ili pisača. Broj piksela po inču ovisi o razlučivosti uređaja.
4	Karakteri. Kod ispisa, karakter je visok šestinu (1/6) inča i širok dvanaestinu (1/12) inča.
5	Inči.
6	Milimetri.
7	Centimetri.

Svi modovi u tablici, osim modova 0 i 3, vrijede za duljine kod ispisa. Na primjer, stavka koja je dvije jedinice duga kad je svojstvo ScaleMode postavljeno na 7, duga je dva centimetra kad se ispiše.

```
‘ Postavljanje mjerila na inče za ovu formu.
ScaleMode = 5
‘ Postavljanje mjerila na piksele za picPicture1.
picPicture1.scaleMode = 3
```

Postavljanje vrijednosti za svojstvo ScaleMode bit će uzrok Visual Basicu da ponovno odredi svojstva ScaleWidth i ScaleHeight tako da budu u skladu s novim mjerilom.

Svojstva ScaleTop i ScaleLeft se zatim postavljaju na 0. Izravno postavljanje vrijednosti svojstava ScaleWidth, ScaleHeight, ScaleTop ili ScaleLeft automatski postavlja svojstvo ScaleMode na 0.

Stvaranje korisničkog mjerila

Možete upotrijebiti svojstva ScaleLeft, ScaleTop, ScaleWidth i ScaleHeight kako bi stvorili korisničko mjerilo. Za razliku od postupka Scale, ova svojstva mogu biti upotrijebljena za određivanje mjerila ili za dobivanje informacije o trenutnom mjerilu koordinatnom sustavu.

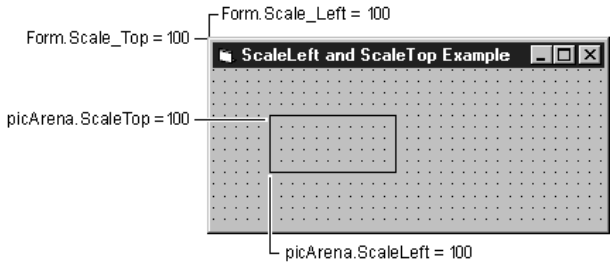
Korištenje svojstava ScaleLeft i ScaleTop

Svojstva ScaleLeft i ScaleTop dodjeljuju brojčane vrijednosti gornjem lijevom kutu objekta. Na primjer, sljedeći izrazi postavljaju vrijednost gornjeg lijevog kuta trenutne forme i gornjeg lijevog kuta okvira za sliku sa imenom picArena.

```
ScaleLeft = 100
ScaleTop = 100
picArena.ScaleLeft = 100
picArena.ScaleTop = 100
```

Ove vrijednosti mjerila prikazane su na slici 12.4.

Slika 12.4 Svojstva ScaleLeft i ScaleTop za formu i kontrolu



Ovi izrazi određuju gornji lijevi kut kao (100, 100). Iako ovi izrazi ne mijenjaju izravno veličinu ili položaj tih objekata, mijenjaju učinak kasnijih izraza. Na primjer, sljedeća naredba koja postavi svojstvo Top kontrole na 100 postavit će tu kontrolu na sam vrh njezinog spremnika.

Korištenje svojstava ScaleWidth i ScaleHeight

Svojstva ScaleWidth i ScaleHeight određuju jedinice u granicama trenutne širine i visine područja crtanja. Na primjer:

```
ScaleWidth = 1000
ScaleHeight = 500
```

Ovi izrazi određuju vodoravnu jedinicu kao jednu tisućinku (1/1000) trenutne unutarnje širine forme te okomitu jedinicu kao jednu petstotinku (1/500) trenutne unutarnje visine forme. Ako forma kasnije promijeni veličinu, jedinice ostaju iste.

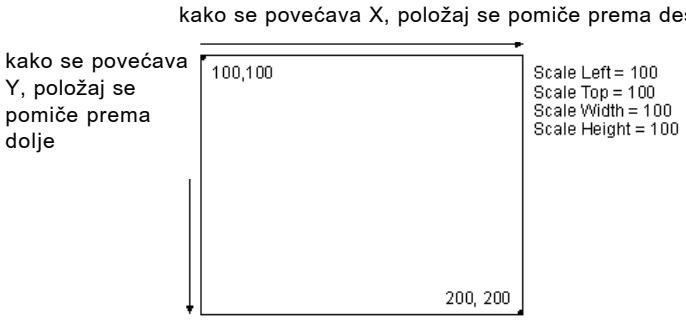
Napomena Svojstva ScaleWidth i ScaleHeight određuju jedinice unutar granica unutarnjih dimenzija objekta; te dimenzije ne uključuju debljinu okvira ni visinu izbornika ili naslova. Stoga, svojstva ScaleWidth i ScaleHeight uvijek pokazuju iznos prostora dostupnog *unutar* objekta. Razlika između unutarnjih i vanjskih dimenzija (određenih svojstvima Width i Height) posebno je važna kod formi koje mogu imati debeli rub. Jedinice se također mogu razlikovati; svojstva Width i Height se uvijek iskazuju u granicama koordinatnog sustava *spremnika*; sama svojstva ScaleWidth i ScaleHeight određuju koordinatni sustav objekta.

Postavljanje svojstava za promjenu koordinatnog sustava

Vrijednosti sva četiri svojstva mjerila mogu uključivati razlomke te također mogu biti negativni brojevi. Negativne postavke svojstava ScaleWidth i ScaleHeight mijenjaju orijentaciju koordinatnog sustava.

Mjerilo prikazano na slici 12.5 ima svojstva ScaleLeft, ScaleTop, ScaleWidth i ScaleHeight postavljena na 100.

Slika 12.5 Mjerilo se izvodi od (100, 100) do (200, 200)



Korištenje postupka Scale za promjenu koordinatnog sustava

Djelotvorniji način promjene koordinatnog sustava, osim postavljanja pojedinih svojstava, je korištenje postupka Scale. Korisničko mjerilo određujete korištenjem sljedeće sintakse:

[objekt.]Scale (x1, y1) – (x2, y2)

Vrijednosti x1 i y1 određuju postavke svojstava ScaleLeft i ScaleTop. Razlika između dvije x-koordinate i dvije y-koordinate određuje postavke svojstava ScaleWidth i ScaleHeight, respektivno. Na primjer, pretpostavimo da ste postavili koordinatni sustav za formu određivanjem krajnjih točaka (100, 100) i (200, 200):

```
Scale (100, 100) - (200, 200)
```

Ovaj izraz određuje formu kao 100 jedinica široku i 100 jedinica visoku. S ovako postavljenim mjerilom, sljedeći izraz pomiče kontrolu lika jednu petinu puta preko forme:

```
shpPokretač.Left = shpPokretač.Left + 20
```

Određivanje vrijednosti x1 veće od x2 (x1 > x2) ili vrijednosti y1 veće od y2 (y1 > y2) ima isti učinak kao postavljanje svojstava ScaleWidth ili ScaleHeight na negativne vrijednosti.

Pretvaranje mjerila

Upotrijebite postupke ScaleX i ScaleY za pretvaranje iz jednog moda mjerila u drugi mod mjerila. Ovi postupci imaju sljedeću sintaksu:

[*objekt.*]**ScaleX** (*vrijednost* [, *izMjerila* [, *uMjerilo*]]

[*objekt.*]**ScaleY** (*vrijednost* [, *izMjerila* [, *uMjerilo*]]

Odredišni *objekt* je forma, okvir za sliku ili objekt tipa Printer. *Vrijednost* je izražena u koordinatnom sustavu određenom modom mjerila *izMjerila*. Vraćena vrijednost je izražena u modu mjerila određenom argumentom *uMjerilo*, ili modom mjerila *objekta* ako je argument *uMjerilo* izostavljen. Ako je izostavljen argument *izMjerila*, mod mjerila za *vrijednost* je HIMETRIČAN.

HIMETRIJA je mod mjerila koji određuje fizičke veličine. Na primjer, broj HIMETRIČNIH jedinica u liniji od 10 centimetara je 10 000. Rezultirajuća linija nacrtana na ekranu je duga deset centimetara, neovisno o veličini područja slikovnog prikaza. Za informacije o HIMETRIČNOM modu mjerila i fizičkim veličinama, pogledajte dodatak Microsoft Windows SDK.

Sljedeći izraz razvlači sadržaj kontrole okvira za sliku MojaSlika na dvostruku širinu. MojaSlika.Picture.Width vraća širinu slike sadržane u kontroli okvira za sliku, što je HIMETRIČKA vrijednost koja treba biti pretvorena u mod mjerila forme Form1.

```
Form1.PaintPicture MojaSlika.Picture, X, Y, _
Form1.Scale(MojaSlika.Picture.Width) * 2
```

Sljedeći primjer pokazuje dva jednako vrijedna načina za određivanje svojstva Width forme na širinu od *np* piksela.

```
‘ Svojstvo ScaleMode forme je postavljeno na piksele.
ScaleMode = vbPixels
‘ Način 1:
‘ Privremeno postavljanje svojstva ScaleMode forme na twipove.
ScaleMode = vbTwips
‘ ScaleX() vraća vrijednost u twipovima.
Width = Width - ScaleWidth + ScaleX(np, vbPixels)
‘ Vraćanje svojstva ScaleMode forme na piksele.
ScaleMode = vbPixels
‘ Način 2:
‘ Pretvaranje iz piksela u twipove bez
‘ mijenjanja svojstva ScaleMode forme.
Width = Width + ScaleX(np - ScaleWidth, vbPixels, vbTwips)
```

Za više informacija Pogledajte "Postupci ScaleX, ScaleY" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Korištenje grafičkih kontrola

Visual Basic pruža tri kontrole oblikovane za stvaranje grafičkih učinaka u aplikaciji:

- Kontrola slike (Image)
- Kontrola linije (Line)
- Kontrola lika (Shape)

Prednosti grafičkih kontrola

Kontrole slike, linije i lika vrlo su korisne za stvaranje slika tijekom izrade aplikacije. Jedna prednost grafičkih kontrola je što one zahtijevaju manje sistemskih izvora od ostalih kontrola Visual Basica, što poboljšava izvođenje vaše aplikacije Visual Basica.

Druga prednost grafičkih kontrola je da možete stvarati slike s manje programskog koda nego s grafičkim postupcima. Na primjer, kako bi na formu postavili krug, možete upotrijebiti ili postupak Circle ili kontrolu lika. Postupak Circle traži od vas da stvorite krug s programskim kodom tijekom izvođenja, dok kontrolu lika možete jednostavno nacrtati na formi i postaviti odgovarajuća svojstva tijekom izrade aplikacije.

Ograničenja grafičkih kontrola

Dok su grafičke kontrole oblikovane kako bi poboljšale izvođenje s malim zahtjevima prema aplikaciji, one ostvaruju taj cilj ograničavanjem ostalih osobina uobičajenih za kontrole u Visual Basicu. Grafičke kontrole:

- Ne mogu se pojaviti iznad ostalih kontrola, osim ako su unutar spremnika koji se može pojaviti iznad ostalih kontrola (kao okvir za sliku).
- Ne mogu dobiti fokus tijekom izvođenja.
- Ne mogu služiti kao spremnici za druge kontrole.
- Nemaju svojstvo hWnd.

Za više informacija Za informacije o grafičkim postupcima, pogledajte "Korištenje grafičkih postupaka" kasnije u ovom poglavlju. Za informacije o grafičkim kontrolama, pogledajte odlomke "Korištenje kontrole slike" "Korištenje kontrole linije" i "Korištenje kontrole lika" u 7. poglavlju "Korištenje standardnih kontrola Visual Basica". Za informacije o učincima grafike na izvođenje vaše aplikacije, posebno pogledajte "Redukcija u grafici" u 15. poglavlju "Oblikovanje u korist izvođenja i sukladnosti".

Dodavanje slika vašoj aplikaciji

Slike se mogu prikazivati na tri mjesta u aplikaciji Visual Basica:

- na formi
- u okviru za sliku
- u kontroli slike

Slike mogu doći iz aplikacija za crtanje, kao one koje dolaze s raznim verzijama Microsoft Windowsa, drugih grafičkih aplikacija, ili biblioteka sličica (clip-art). Visual Basic pruža veliku zbirku ikona koje možete upotrijebiti kao slike u aplikacijama. Visual Basic vam omogućuje dodavanje .jpg i .gif datoteka, kao i .bmp, .dib, .ico, .cur, .wmf i .emf datoteka vašim aplikacijama. Za više informacija o grafičkim oblicima koje podržava Visual Basic, pogledajte odlomke "Korištenje kontrole slike" i "Korištenje kontrole okvira za sliku" u 7. poglavlju " Korištenje standardnih kontrola Visual Basica".

Upotrebljavate razne tehnike za dodavanje slike formi, okviru za sliku ili kontroli slike ovisno o tome želite li dodati sliku tijekom izrade ili tijekom izvođenja aplikacije.

Dodavanje slike tijekom izrade aplikacije

Postoje dva načina dodavanja slike tijekom izrade aplikacije:

- Učitajte sliku na formu, ili u okvir za sliku ili kontrolu slike iz datoteke slike:

  U prozoru sa svojstvima, odaberite svojstvo Picture s liste svojstava i kliknite gumb Properties. Visual Basic će prikazati dijaloški okvir, iz kojeg možete odabrati datoteku slike.

  Ako postavite svojstvo Picture za formu, slika koju odaberete se prikazuje na formi, iza svih kontrola koje ste postavili na nju. Slično tome, ako postavite svojstvo Picture za okvir za sliku, slika se prikazuje u okviru, iza svih kontrola koje ste postavili u njega.

- Ulijepite sliku na formu ili u okvir za sliku ili kontrolu slike.

  Kopirajte sliku iz druge aplikacije (kao Microsoft Paint) u odlagalište. Vratite se u Visual Basic, odaberite formu, okvir za sliku ili kontrolu slike, te u izborniku Edit, odaberite Paste.

Kad ste jednom postavili svojstvo Picture za formu, okvir za sliku ili kontrolu slike – ili učitavanjem ili uljepljivanjem slike – u okviru s postavkom ovog svojstva pojavljuje se riječ "(Bitmap)" "(Icon)" ili "(Metafile)". Kako bi promijenili postavku, učitajte ili ulijepite drugu sliku. Kako bi ponovno postavili svojstvo Picture na "(None)", dvokliknite na riječ prikazanu u okviru s postavkom i pritisnite tipku DEL.

Dodavanje slike tijekom izvođenja aplikacije

Postoje četiri načina za dodavanje slike tijekom izvođenja aplikacije:

- Upotrijebite funkciju LoadPicture za određivanje imena datoteke te dodjelu slike svojstvu Picture.

  Sljedeći izraz učitava datoteku Auti.bmp u okvir za sliku imena picPrikaz (ime kontrole možete odrediti postavljanjem njezinog svojstva Name):

  ```
  picPrikaz.Picture = LoadPicture("C:\Slike\Auti.bmp")
  ```

  Novu sliku možete učitati na formu, u okvir za sliku ili kontrolu slike kad god to želite. Učitavanje nove slike potpuno zamjenjuje postojeću sliku, iako izvorne datoteke slika nikad nisu pogođene.

- Upotrijebite funkciju LoadResPicture za dodjeljivanje slike iz datoteke .res projekta u svojstvo Picture.

  Sljedeći izraz učitava bitmapiranu sliku izvornog ID-a 10, iz datoteke izvora u okvir za sliku imena picIzvor:

  ```
  Set picIzvor.Picture = LoadResPicture(10, vbResBitmap)
  ```

- Kopirajte sliku iz jednog objekta u drugi.

  Kad je jednom slika učitana ili ulijepljena na formu ili u okvir za sliku ili kontrolu slike, možete ju dodijeliti drugim formama, okvirima za sliku ili kontrolama slike tijekom izvođenja aplikacije. Na primjer, ovaj izraz kopira sliku iz okvira za sliku imena picPrikaz u kontrolu slike imena imgPrikaz:

  ```
  Set imgPrikaz.Picture = picPrikaz.Picture
  ```

- Kopirajte sliku iz objekta Clipboard (odlagališta).

Za više informacija Za više informacija o kopiranju slike iz odlagališta, pogledajte "Rad s više oblika u odlagalištu", ranije u ovom poglavlju.

Za informacije o datotekama izvora, pogledajte "Rad s datotekama izvora" u 8. poglavlju "Više o programiranju".

Napomena Ako učitate ili ulijepite slike iz datoteka tijekom izrade aplikacije, slike se snimaju i učitavaju s formom, a aplikacija kopira slike iz jednog objekta u drugi. Zatim, ako stvorite .exe datoteku, ne trebate davati svojim korisnicima kopije datoteka slika; sama .exe datoteka sadrži slike. Također, razmislite o isporuci .res datoteke te korištenju funkcije LoadResPicture. Datoteka tipa .res ugrađuje se u .exe datoteku, a bitmapirane slike snimljene su standardnom obliku kojeg može čitati svaki editor izvora. Ako učitate slike tijekom izvođenja aplikacije funkcijom LoadPicture, morate isporučiti datoteke slika svojim korisnicima zajedno sa svojom aplikacijom.

Uklanjanje slika tijekom izvođenja aplikacije

Funkciju LoadPicture možete također upotrijebiti za uklanjanje slike tijekom izvođenja aplikacije a da je ne zamijenite drugom slikom. Sljedeći izraz uklanja sliku iz kontrole slike imena imgPrikaz:

```
Set imgPrikaz.Picture = LoadPicture("")
```

Pomicanje slika i mijenjanje njihove veličine

Ako je forma, okvir za sliku ili kontrola slike pomaknuta (tijekom izrade ili tijekom izvođenja aplikacije), njihova slika se automatski pomiče s njima. Ako forma, okvir za sliku ili kontrola slike promijene veličinu tako da su premale za prikazivanje slike, slike se odrezuju s desne i donje strane. Slika se također odrezuje ako ju učitate ili kopirate na formu ili u okvir za sliku ili kontrolu slike koje su premale da bi prikazale cijelu sliku.

Svojstvo AutoSize

Ako želite da se okvir za sliku automatski proširi kako bi se prilagodio novoj slici, postavite svojstvo AutoSize okvira za sliku na True. Nakon toga, kad se tijekom izvođenja učita ili kopira slika u okvir za sliku, Visual Basic automatski proširuje kontrolu prema dolje i prema desno dovoljno da može prikazati cijelu sliku. Ako je slika koju učitate veća od rubova forme, pojavit će se odrezana jer se veličina forme neće promijeniti.

Svojstvo AutoSize također možete upotrijebiti za automatsko smanjivanje okvira za sliku kako bi odrazio veličinu nove slike.

Napomena Kontrole slike nemaju svojstvo AutoSize, ali automatski mijenjaju svoju veličinu kako bi odgovarala slici učitanoj u njih. Forme nemaju svojstvo AutoSize, i ne povećavaju se automatski kako bi prikazale cijelu sliku.

Svojstvo Stretch kontrola slike

Ako želite da se slika u kontroli slike automatski proširi kako bi popunila određenu veličinu, upotrijebite svojstvo Stretch. Kad je svojstvo Stretch postavljeno na False, kontrola slike automatski prilagođuje svoju veličinu kako bi odgovarala slici učitanoj u nju. Kako bi promijenili veličinu slike tako da odgovara kontroli slike, postavite svojstvo Stretch kontrole slike na True.

Odabir umjetnosti za kontrolu slike

Gdje ćete nabaviti datoteke slika? Ako želite ikone, možete upotrijebiti biblioteku ikona uključenu s Visual Basicom. Datoteke ikona možete pronaći unutar poddirektorija glavnog direktorija Visual Basica (\Vb\Graphics\Icons). Možete stvoriti .bmp datoteke s aplikacijom Microsoft Paint, ili možete kupiti zbirku sličica koja uključuje datoteke bitmapiranih slika ili ikona, ili metadatoteke. Također možete stvoriti datoteku izvora (.res) koja sadrži slike.

Za više informacija Pogledajte "Rad s datotekama izvora" u 8. poglavlju "Više o programiranju" za više informacija o stvaranju datoteke izvora.

Uvod u grafička svojstva formi i kontrola

Forme i razne kontrole imaju grafička svojstva. Sljedeća tablica ispisuje ta svojstva.

kategorija	svojstva
obrada prikazivanja	AutoRedraw, ClipControls
trenutan položaj crtanja	CurrentX, CurrentY
tehnike crtanja	DrawMode, DrawStyle, DrawWidth, BorderStyle, BorderWidth
tehnike popunjavanja	FillColor, FillStyle
boje	BackColor, ForeColor, BorderColor, FillColor

Forme i okviri za sliku imaju dodatna svojstva:

- Svojstva mjerila, kao što je opisano u odlomku “Promjena koordinatnog sustava objekta”, ranije u ovom poglavlju.
- Svojstva pisma, kao što je opisano u odlomku “Određivanje karakteristika pisma”, ranije u ovom poglavlju.

Postoje dva svojstva formi i okvira za sliku koja ćete vjerojatno odmah poželjeti upotrijebiti: BackColor i ForeColor. Svojstvo BackColor boja pozadinu područja crtanja. Ako je svojstvo BackColor postavljeno na svijetloplavu boju, tada će cijelo područje biti svijetloplavo nakon što ga očistite. Svojstvo ForeColor (prednji plan) određuje boju teksta i grafike koji se iscrtavaju na objektu, iako vam neki grafički postupci daju mogućnost korištenja drugačije boje. Za više informacija o bojama, pogledajte “Rad s bojama”, kasnije u ovom poglavlju.

Stvaranje trajne grafike svojstvom AutoRedraw

Svaka forma i okvir za sliku imaju svojstvo AutoRedraw. Ovo svojstvo je tipa Boolean i, kad je postavljeno na True, uzrokuje snimanje grafičkog izlaza u memoriju. Svojstvo AutoRedraw možete upotrijebiti za stvaranje trajne grafike.

Trajne slike

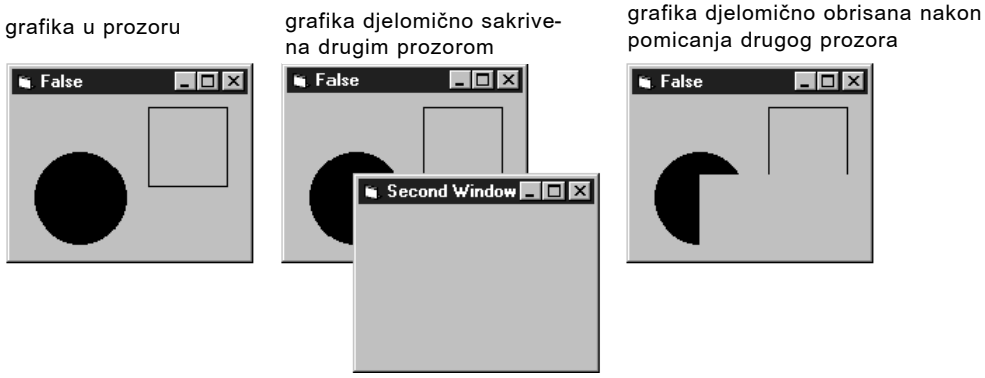
Operativni sustav Microsoft Windows upravlja ekranskom slikom kako bi stvorio privid preklapajućih prozora. Kad se jedan prozor pomakne iznad drugog, privremeno ga skrivajući, te se zatim ponovno pomakne na stranu, prozor i njegov sadržaj trebaju biti ponovno prikazani. Windowsi vode računa o ponovnom prikazivanju prozora i kontrola. Vaša Visual Basic aplikacija ipak mora rukovati ponovnim prikazivanjem slika na formi ili u okviru za sliku.

Ako stvorite sliku na formi korištenjem grafičkih postupaka, obično želite da se ponovno pojavi točno kako ste ju postavili (*trajna slika*). Za stvaranje stalne slike možete upotrijebiti svojstvo AutoRedraw.

Svojstvo AutoRedraw i forme

Podrazumijevana postavka svojstva AutoRedraw je False. Kad je svojstvo AutoRedraw postavljeno na False, sve slike stvorene grafičkim postupcima koje se pojavljuju na formi su izgubljene ako ih drugi prozor privremeno prekrije. Također, slike koje se pružaju izvan rubova forme su izgubljene ako povećate formu. Učinci postavljanja svojstva AutoRedraw na False su prikazani na slici 12.6.

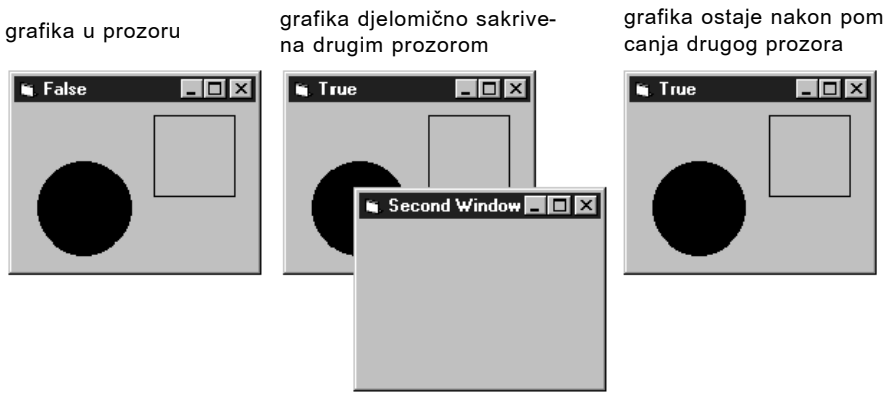
Slika 12.6 Učinci postavljanja svojstva AutoRedraw na False



Kad je svojstvo AutoRedraw forme postavljeno na True, Visual Basic primjenjuje grafičke postupke na "slikarskom platnu" u memoriji. Aplikacija kopira sadržaj te memorije slikarskog platna za ponovno prikazivanje slike privremeno sakrivene drugim prozorom. U većini slučajeva, veličina tog slikarskog platna za forme je veličina ekrana. Ako je svojstvo MaxButton forme postavljeno na False, a rub forme nije promjenjive veličine, veličina slikarskog platna jednaka je veličini forme.

Ovo slikarsko platno također omogućuje aplikaciji da snimi slike koje se pružaju izvan rubova forme kad je forma promjenjive veličine. Učinci postavljanja svojstva AutoRedraw na True su prikazani na slici 12.7.

Slika 12.7 Učinci postavljanja svojstva AutoRedraw na True



Svojstvo AutoRedraw i okviri za sliku

Kad je svojstvo AutoRedraw okvira za sliku postavljeno na True, Visual Basic u memoriju snima samo vidljiv sadržaj okvira za sliku. Razlog tome je što je memorija slikarskog platna koja se koristi za snimanje sadržaja okvira za sliku jednake veličine kao i okvir za sliku. Slike koje se pružaju izvan okvira za sliku su odsječene i više se ne pojavljuju, čak i ako se promijeni veličina okvira za sliku.

Korištenje nepostojane grafike

Možete ostaviti svojstvo AutoRedraw na False za formu i sve njezine okvire za sliku kako bi sačuvali memoriju. Međutim, tada slike nisu automatski trajne: morate upravljati ponovnim iscrtavanjem svake slike u kodu prema potrebi.

Možete uključiti kod u događaju Paint forme ili okvira za sliku koji ponovno iscrtava sve linije, krugove te točke prema potrebi. Ovakav pristup obično radi najbolje kad imate ograničen broj slika koje jednostavno možete rekonstruirati.

Potprogram događaja Paint poziva se uvijek kad dio forme ili okvira za sliku treba biti ponovno iscrtan – na primjer, kad se makne prozor koji je pokrivao objekt, ili kad promjena veličine uzrokuje povratak slike u vidno područje. Ako je svojstvo AutoRedraw postavljeno na True, potprogram Paint objekta se nikad ne poziva osim ako ga vaša aplikacija izričito ne pozove. Vidljiv sadržaj objekta spremljen je u memoriji slikarskog platna, pa događaj Paint nije potreban.

Imajte na umu da odluka korištenja nepostojane grafike može utjecati na način iscrtavanja grafike na formi ili spremniku. Sljedeći odlomak “Rezanje područja svojstvom ClipControls”, te odlomak “Slojevitost grafike sa svojstvima AutoRedraw i ClipControls”, kasnije u ovom poglavlju, raspravljaju ostale čimbenike koji mogu odlučiti trebate li ili ne koristiti nepostojanu grafiku.

Promjena svojstva AutoRedraw tijekom izvođenja

Postavku svojstva AutoRedraw možete promijeniti tijekom izvođenja aplikacije. Ako je svojstvo AutoRedraw postavljeno na False, slike i izlaz postupka Print bit će ispisani samo na ekranu, a ne i u memoriji. Ako očistite objekt postupkom Cls, sav izlaz zapisan nakon što je svojstvo AutoRedraw postavljeno na True neće biti obrisan. Taj izlaz se zadržava u memoriji, te morate ponovno postaviti svojstvo AutoRedraw na False te upotrijebiti postupak Cls kako bi ga obrisali.

Za više informacija Kako bi naučili o utjecajima svojstva AutoRedraw na izvođenje, pogledajte “Optimiziranje brzine prikaza” u 15. poglavlju “Oblikovanje u korist izvođenja i sukladnosti”.

Područja rezanja svojstvom ClipControls

Svaka forma, okvir za sliku i kontrola okvira ima svojstvo ClipControls. Ovo svojstvo je tipa Boolean i, kad je postavljeno na True, uzrokuje određivanje područja rezanja u spremniku kad se u njemu crta oko svih kontrola osim:

- Kontrole lika
- Kontrole linije
- Kontrole slike
- Kontrole natpisa
- Svih ActiveX grafičkih kontrola

Postavljanjem svojstva ClipControls na False, možete poboljšati brzinu kojom se forma iscrtava na ekranu. Poboljšanje brzine je najveće na formama s puno kontrola koje se ne preklapaju, kao dijaloški okviri.

Područja rezanja

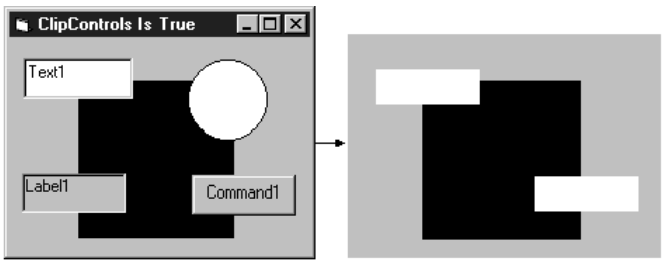
Rezanje je proces određivanja područja na formi ili spremniku koje će biti iscrtano kad se prikaže forma ili spremnik. Obris koji se koristi za određivanje područja na formi ili spremniku koja će biti iscrtana ili "izrezana", određuje *područje rezanja* za tu formu ili spremnik. Područja rezanja su korisna kad aplikacije temeljene na Windowsima trebaju spremiti jedan dio prikaza te istovremeno ponovno iscrtati ostatak.

Rezanje formi i spremnika

Podrazumijevana postavka svojstva ClipControls je True. Kad se svojstvo ClipControls postavljeno na True, Windowsi određuju područje rezanja za pozadinu forme ili spremnika prije događaja Paint. To područje rezanja okružuje sve negrafičke kontrole. Kod korištenja svojstva ClipControls, kontrole natpisa djeluju kao grafičke kontrole.

Tijekom događaja Paint, Windowsi ponovno iscrtavaju samo pozadinu unutar područja rezanja, zaobilazeći negrafičke kontrole. Slika 12.8 prikazuje formu s četiri kontrole, okvirom nacrtanim postupkom Line, te područjem rezanja stvorenim za tu formu postavljanjem svojstva ClipControls na True. Uočite da područje rezanja nije odrezalo područje oko kontrola natpisa i linije na formi. Okvir nacrtan na pozadini postupkom Line iscrtava se samo u području rezanja.

Slika 12.8 Područje rezanja stvoreno kad je ClipControls postavljeno na True



forma sa dvije negrafičke kontrole, kontrolom lika, kontrolom natpisa, i okvirom stvorenim postupkom Line

područja rezanja prikazana svijetlo sivom; kontrola lika i kontrola natpisa nizu odrezane. Okvir se iscrtava samo u području rezanja

Kad je svojstvo ClipControls postavljeno na False, Windowsi ne određuju područje rezanja za pozadinu forme ili spremnika prije događaja Paint. Također, izlaz grafičkih postupaka unutar događaja Paint pojavljuje se samo na dijelovima forme ili spremnika koji trebaju biti ponovno iscrtani. Budući da izračunavanje i upravljanje područjem rezanja troši vrijeme, postavljanje svojstva ClipControls na False može uzrokovati brže prikazivanje formi s puno nepreklapajućih kontrola (kao složeni dijaloški okviri).

Napomena Zaobilazite ugnježđavanje kontrola sa svojstvom ClipControls postavljenim na True unutar kontrola kojima je svojstvo ClipControls postavljeno na False. Takav način može kao rezultat imati neispravno iscrtavanje ugniježđenih kontrola. Kako bi to ispravili, postavite svojstvo ClipControls na True i za spremnike i za kontrole.

Za više informacija Pogledajte "Optimiziranje brzine prikaza" u 15. poglavlju "Oblikovanje u korist izvođenja i sukladnosti".

Slojevitost grafike sa svojstvima AutoRedraw i ClipControls

Različite kombinacije svojstava AutoRedraw i ClipControls imaju različite učinke na način iscrtavanja grafičkih kontrola i grafičkih postupaka na ekranu.

Dok stvarate grafiku, imajte na umu da se grafičke kontrole i natpisi, negrafičke kontrole, te grafički postupci pojavljuju na različitim slojevima u spremniku. Ponašanje tih slojeva ovisi o tri čimbenika:

- Postavci svojstva AutoRedraw.
- Postavci svojstva ClipControls.
- Pojavljuju li se grafički postupci unutar ili izvan događaja Paint.

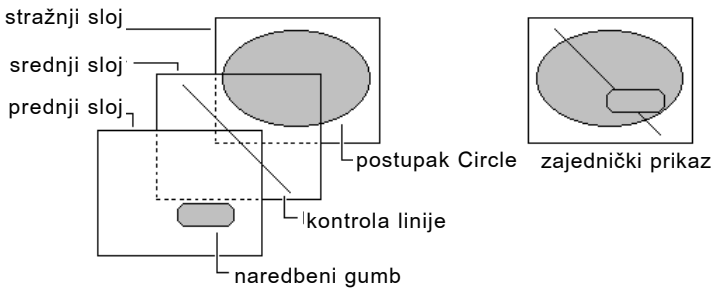
Normalna slojevitost

Obično, slojevi forme ili drugih spremnika su, od vrha prema dolje, sljedeći:

sloj	sadrži
Prednji	Negrafičke kontrole kao naredbeni gumbi, kontrolne kućice i kontrole datoteka.
Srednji	Grafičke kontrole i kontrole natpisa.
Stražnji	Prostor za crtanje na formi ili spremniku. Ovdje se pojavljuju rezultati grafičkih postupaka.

Sve na jednom sloju prekriva sve na sloju iza, pa će se grafika koju stvorite grafičkim kontrolama pojavljivati iza ostalih kontrola na formi, a sva grafika koju stvorite grafičkim postupcima pojavljivat će se iza svih grafičkih i negrafičkih kontrola. Normalan raspored slojeva prikazan je na slici 12.9.

Slika 12.9 Normalna slojevitost grafike na formi



Učinci slojevitosti

Možete proizvesti normalnu slojevitost korištenjem nekog od nekoliko pristupa. Kombiniranje postavki svojstava AutoRedraw i ClipControls te postavljanje grafičkih postupaka unutar ili izvan događaja Paint utječe na slojevitost i izvođenje aplikacije.

Sljedeća tablica ispisuje učinke stvorene različitim kombinacijama svojstava AutoRedraw i ClipControls te postavljanjem grafičkih postupaka.

AutoRedraw	ClipControls	grafički postupci unutar/izvan događaja Paint	Ponašanje slojevitosti
True	True (standardno)	Događaj Paint zanemaren	Normalna slojevitost.
True	False	Događaj Paint zanemaren	Normalna slojevitost. Forme s puno kontrola koje se ne preklapaju mogu se brže iscrtavati zato što se ne izračunava ili stvara područje rezanja.
False (standardno)	True (standardno)	Unutar	Normalna slojevitost.
False	True	Izvan	Negrafičke kontrole na vrhu. Grafički postupci i grafičke kontrole pojavljuju se pomiješano u srednjem i stražnjem sloju. Nije preporučeno.
False	False	Unutar	Normalna slojevitost, utječe samo na piksele koji su prethodno pokriveni ili se pojavljuju kod promjene veličine forme.
False	False	Izvan	Grafički postupci i sve kontrole pojavljuju se izmiješano u sva tri sloja. Nije preporučeno.

Učinci svojstva AutoRedraw

Postavljanje svojstva AutoRedraw na True uvijek proizvodi normalnu slojevitost. Iako je korištenje svojstva AutoRedraw najlakši način raslojavanja grafike, aplikacije s velikim formama mogu trpjeti od usporenog izvođenja zbog memorijskih zahtjeva ovog svojstva.

Učinci svojstva ClipControls

Kad je svojstvo AutoRedraw postavljeno na True, postavke svojstva ClipControls nemaju učinka na slojevitost grafike na formi ili u spremniku. Međutim, svojstvo ClipControls može utjecati na brzinu prikaza forme. Kad je ovo svojstvo postavljeno na False, aplikacija ne stvara područje rezanja. Nepostojanje izračunavanja ili iscrtavanja za zaobilaženje rupa u području rezanja može uzrokovati brže prikazivanje forme.

Također, kad su oba svojstva, AutoRedraw i ClipControls, postavljena na False, aplikacija ponovno iscrtava samo piksele na formi ili spremniku koji su izloženi:

- Pokrivanju forme ili spremnika drugim prozorom te zatim pomicanjem tog prozora na stranu.
- Promjeni veličine forme ili spremnika.

Učinci događaja Paint

Kad je svojstvo AutoRedraw postavljeno na False, najbolje mjesto upotrebe grafičkih postupaka je iz događaja Paint forme ili spremnika. Ograničavanje grafičkih postupaka na događaj Paint uzrokuje iscrtavanje tih postupaka po predvidljivom redu.

Upotreba grafičkih postupaka izvan događaja Paint kad je svojstvo AutoRedraw postavljeno na False može proizvesti nestalnu grafiku. Svaki put kad se izlaz grafičkog postupka pojavi na formi ili spremniku, može pokriti neku kontrolu ili grafički postupak koji se tu već nalazi (ako je svojstvo ClipControls postavljeno na False). Kad aplikacija koristi više od nekoliko grafičkih postupaka za stvaranje vizualnih efekata, upravljanje izlaznim rezultatima može biti iznimno teško osim ako svi postupci nisu postavljeni u događaj Paint.

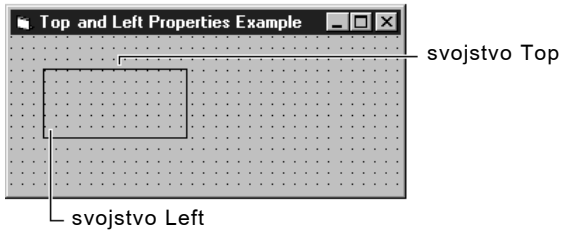
Dinamičko pomicanje kontrola

S Visual Basicom, jedan od najlakših efekata je postizanje pomicanja kontrole tijekom izvođenja aplikacije. Možete izravno promijeniti svojstva koja određuju položaj kontrole ili možete upotrijebiti postupak Move.

Korištenje svojstava Left i Top

Svojstvo Left je udaljenost između gornjeg lijevog kuta kontrole i lijeve strane forme. Svojstvo Top je udaljenost između gornjeg lijevog kuta kontrole i vrha forme. Slika 12.10 pokazuje svojstva Left i Top kontrole.

Slika 12.10 Svojstva Left i Top



Kontrolu možete pomaknuti promjenom postavki svojstava Left i Top izrazima sličnim ovim:

```
txtPolje1.Left = txtPolje1.Left + 200
txtPolje1.Top = txtPolje1.Top - 300
```

Pomicanje kontrole linije

Kao što je prije spomenuto, kontrole linije nemaju svojstva Left i Top. Umjesto toga, upotrebljavate posebna svojstva kako bi nadzirali položaj kontrola linije na formi. Sljedeća tablica ispisuje ta svojstva te kako ona određuju položaj kontrole linije.

svojstvo	opis
X1	X-koordinata početka linije. Dana je u jedinicama trenutnog mjerila. Početak linije je kraj stvoren kad počnete crtati.
Y1	Y-koordinata početka linije.
X2	X-koordinata kraja linije. Kraj linije je kraj stvoren kad prestanete crtati.
Y2	Y-koordinata kraja linije.

Primjer JumpyLine aplikacije Blanker nasumce mijenja položaj kontrole linije na formi DemoForm korištenjem ovih izraza:

```
‘ Postavljanje nasumičnog X položaja za kraj 1. linije.
linLineCtl.X1 = Int(DemoForm.Width * Rnd)
‘ Postavljanje nasumičnog Y položaja za kraj 1. linije.
linLineCtl.Y1 = Int(DemoForm.Height * Rnd)
‘ Postavljanje nasumičnog X položaja za kraj 2. linije.
linLineCtl.X2 = Int(DemoForm.Width * Rnd)
‘ Postavljanje nasumičnog Y položaja za kraj 2. linije.
linLineCtl.Y2 = Int(DemoForm.Height * Rnd)
‘ Brisanje zalutalih piksela iz pokretne linije.
Cls
‘ Stanka u prikazivanju prije idućeg pomaka.
Delay
```

Korištenje postupka Move

Promjena svojstava Left i Top ili X i Y proizvodi trzav efekt dok se kontrola najprije pomiče vodoravno pa zatim okomito. Postupak Move proizvodi glatkije dijagonalno pomicanje.

Sintaksa postupka Move je:

[*objekt.*]**Move** *left* [, *top*[, *width*[, *height*]]]

Objekt je forma ili kontrola koja će biti pomaknuta. Ako je *objekt* izostavljen, pomiče se trenutna forma. Argumenti *left* i *top* su nove postavke za svojstva Left i Top *objekta*, dok su argumenti *width* i *height* nove postavke za njegova svojstva Width i Height. Obavezan je samo argument *left*, ali, kako bi odredili ostale argumente, morate uključiti sve argumente koji se u listi argumenata pojavljuju prije argumenta kojeg želite odrediti.

Apsolutno pomicanje

Apsolutno pomicanje se pojavljuje kad pomaknete objekt na određene koordinate u njegovom spremniku. Sljedeći izraz koristi apsolutno pomicanje kako bi pomaknuo kontrolu imena txtPolje1 na koordinate (100, 200):

```
txtPolje1.Move 100, 200
```

Relativno pomicanje

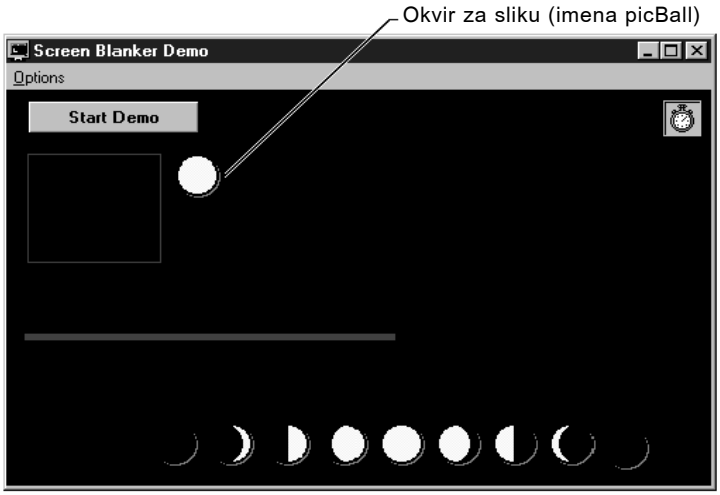
Relativno pomicanje se pojavljuje kad pomaknete objekt određivanjem udaljenosti za koju se treba pomaknuti od svog trenutnog položaja. Sljedeći izraz koristi relativno pomicanje za pomak kontrole txtPolje1 na položaj 100 twipova prema dolje i desno od njegovog trenutnog položaja:

```
txtPolje1.Move txtPolje1.Left + 100, txtPolje1.Top + 100
```

Ovaj dio pokazuje pomicanje kontrole u primjeru aplikacije Blanker. Primjer Rebound pomiče okvir za sliku dijagonalno po formi, pa okvir za sliku izgleda kao da se “odbija” od rubova forme. Ovaj primjer koristi okvir za sliku umjesto kontrole slike zato što treptanje kontrole slike tijekom pomicanja traži njezino ponovno iscrtavanje.

Slika 12.11 prikazuje glavnu formu aplikacije Blanker (formu DemoForm) te okvir za sliku koji se koristi u ovom primjeru.

Slika 12.11 Okvir za sliku (imena picBall) u aplikaciji Blanker



Ime okvira za sliku je picBall. Ova kontrola počinje se micati po formi nakon što odaberete naredbu Rebound iz izbornika Options te kliknete gumb Start Demo. Potprogram događaja tog naredbenog gumba zatim poziva potprogram CtlMoveDemo.

Potprogram CtlMoveDemo nasumce odabire jedan početni smjer iz ove četiri mogućnosti:

- Lijevo i gore
- Desno i gore
- Lijevo i dolje
- Desno i dolje

Okvir za sliku picBall pomiče se duž odabranog smjera sve dok kontrola ne dosegne jedan od četiri ruba forme. Nakon toga okvir za sliku mijenja smjer udaljavajući se od ruba koji je dosegnuo; varijabla Motion nadzire smjer. Na primjer, kad se okvir za sliku pomiče lijevo i gore, taj dio potprograma mijenja vrijednost varijable Motion i zapovijeda programskom kodu da pomakne okvir picBall u drugom smjeru.

Sljedeći izrazi dolaze iz potprograma CtlMoveDemo aplikacije Blanker:

```
Select Case Motion
Case 1
   ‘ Ako je kretanje prema lijevo i gore,
   ‘ pomicanje kontrole za 20 twipova.
   picBall.Move picBall.Left - 20, picBall.Top - 20
   ‘ Ako kontrola dotakne lijevi rub,
   ‘ promjena pomicanja prema desno i gore.
   If picBall.Left <= 0 Then
      Motion = 2
```

```
' Ako kontrola dotakne gornji rub,
' promjena pomicanja prema lijevo i dolje.
ElseIf picBall.Top <= 0 Then
  Motion = 4
End If
```

Uočite da linija koda koja pomiče kontrolu picBall oduzima 20 twipova od trenutnih vrijednosti njezinih svojstava Left i Top kako bi uspostavila nov položaj kontrole. Tako je osigurano da se kontrola uvijek pomiče relativno prema njezinom trenutnom položaju.

Brzina i glatkoća pomicanja kontrole ovise o broju twipova (ili drugih jedinica) korištenih u postupku Move. Povećavanje broja twipova povećava brzinu, ali smanjuje glatkoću pomicanja. Smanjivanje broja twipova smanjuje brzinu, ali poboljšava glatkoću pomicanja kontrole.

Za više informacija Za dodatne informacije o postupku Move, pogledajte "Postupak Move" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Dinamička promjena veličine kontrola

U aplikaciji Visual Basica, tijekom izvođenja aplikacije možete promijeniti veličinu i oblik okvira za sliku, kontrole slike ili forme, isto kako im možete promijeniti položaj.

Sljedeća svojstva utječu na veličinu.

svojstvo	primjena	opis
Align	okviri za sliku i kontrole podataka	Ako je postavljeno za poravnavanje okvira za sliku uz vrh (1) ili dno (2) forme, širina okvira za sliku uvijek je jednaka širini unutrašnjosti forme. Ako je postavljeno za poravnavanje okvira za sliku uz lijevu (3) ili desnu (4) stranu forme, visina okvira za sliku je visina unutrašnjosti forme.
Height	sve forme i sve kontrole osim mjerača vremena, izbornika i linija	Visina objekta izražena u modu mjerila forme (twipovi u pravilu).
Width	sve forme i sve kontrole osim mjerača vremena, izbornika i linija	Širina objekta izražena u modu mjerila forme (twipovi u pravilu).
AutoSize	kontrole natpisa i okviri za sliku	Ako je True, Visual Basic uvijek prilagođuje dimenzije okvira za sliku veličini sadržaja.
Stretch	kontrole slike	Ako je True, bitmapirana slika ili metadatoteka se rasteže kako bi odgovarala veličini kontrole slike. Ako je False, veličina kontrole slike mijenja se kako bi odgovarala veličini bitmapirane slike ili metadatoteke koju sadržava.

U ovom primjeru, naredbeni gumb imena cmdRasti postaje sve veći svaki put kad ga korisnik klikne:

```
Private Sub cmdRasti_Click()
  cmdRasti.Height = cmdRasti.Height + 300
  cmdRasti.Width = cmdRasti.Width + 300
End Sub
```

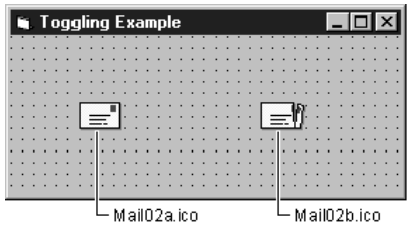
Stvaranje jednostavne animacije

Možete stvoriti jednostavnu animaciju promjenom slika tijekom izvođenja aplikacije. Najlakši način da to napravite je prebacivanje između dvije slike. Možete također upotrijebiti niz slika za stvaranje animacije s više slika. Također, dinamičkim pomicanjem slike, možete stvoriti razrađenije efekte.

Prebacivanje između dvije slike

Neke ikone mogu biti upotrijebljene u parovima. Na primjer, u poddirektoriju \Vb\Graphics\Icons postoje dvije odgovarajuće ikone omotnice, jedna s neotvorenom omotnicom te jedna s poderanom omotnicom, kao što je prikazano na slici 12.12. Promjenom, ili *prebacivanjem*, između njih dvije, možete stvoriti animaciju koja vašem korisniku pokazuje stanje pošte.

Slika 12.12 Ikone pošte



Sljedeći izraz mijenja svojstvo Picture kontrole slike s imenom imgStanjePošte kako bi prebacio njezinu sliku s neotvorene omotnice na otvorenu omotnicu.

```
imgStanjePošte.Picture = imgOtvorenaPošta.Picture
```

Kruženje kroz više slika

Možete također kružiti kroz više slika kako bi napravili duže animacije. Ova tehnika je u osnovi ista kao i prebacivanje između dvije slike, ali od aplikacije zahtijeva odabir bitmapirane slike koja će djelovati kao trenutna slika. Jedan način nadziranja pojedinih slika u animaciji je korištenje matrice kontrola.

Za više informacija Pogledajte “Stvaranje matrica objekata” u 9. poglavlju “Programiranje objektima” za više informacija o matricama kontrola.

Primjer aplikacije Blanker uključuje animaciju koja prikazuje rotirajući mjesec. Primjer Spinning Moon koristi matricu od devet kontrola slike za stvaranje animacije. Da bi vidjeli kako slike u matrici kontrola međusobno djeluju tijekom izvođenja aplikacije, odaberite Spinning Moon iz izbornika Options, te kliknete gumb Start Demo, koji poziva potprogram ImageDemo.

Korištenje grafičkih postupaka

Kao dodatak grafičkim kontrolama, Visual Basic pruža nekoliko postupaka za stvaranje grafike. Grafički postupci, rezimirani u sljedećoj tablici, primjenjuju se za forme i okvire za sliku.

postupak	opis
Cls	Čisti svu grafiku i izlazne rezultate naredbe Print.
PSet	Postavlja boju pojedinog piksela.
Point	Vraća vrijednost boje određene točke.
Line	Crta liniju, pravokutnik ili popunjeni okvir.
Circle	Crta krug, elipsu ili luk.
PaintPicture	Crta grafiku na određenim položajima.

Napomena Postupak Print se također može smatrati grafičkim postupkom, jer se njegovi izlazni rezultati zapisuju u objekt i snimaju u memoriju slike (ako je uključeno svojstvo AutoRedraw) baš kao i postupci PSet, Line i Circle. Za više informacija o postupku Print, pogledajte "Prikazivanje teksta na formama i okvirima za sliku", ranije u ovom poglavlju.

Prednosti grafičkih postupaka

Grafički postupci dobro rade u okolnostima gdje korištenje grafičkih kontrola zahtijeva previše posla. Na primjer, stvaranje linija mreže na grafikonu zahtijevalo bi matricu kontrola linije, ali i samo malu količinu koda korištenjem postupka Line. Praćenje položaja kontrola linije u matrici dok forma mijenja veličinu je više posla od jednostavnog ponovnog iscrtavanja linija postupkom Line.

Kad želite da se na formi kratko pojavi vizualni efekt, kao bljesak boje kad prikažete dijalog About, možete napisati nekoliko linija programskog koda za takav privremeni efekt umjesto korištenja druge kontrole.

Grafički postupci nude neke vizualne efekte koji nisu dostupni u grafičkim kontrolama. Na primjer, možete stvoriti lukove ili obojati pojedine piksele samo korištenjem grafičkih postupaka. Slike koje stvorite ovim grafičkim postupcima pojavljuju se na formi u svom vlastitom sloju. Taj sloj je ispod svih ostalih kontrola na formi, pa korištenje grafičkih postupaka može prikladno djelovati kad želite stvoriti grafiku koja se pojavljuje ispod svega ostalog u aplikaciji.

Za više informacija Pogledajte "Slojevitost grafike sa svojstvima AutoRedraw i ClipControls", ranije u ovom poglavlju.

Ograničenja grafičkih postupaka

Stvaranje slika grafičkim postupcima ima svoje mjesto u kodu, što znači da trebate pokrenuti aplikaciju kako bi vidjeli učinak grafičkog postupka. Grafički postupci zbog toga nisu prikladni kao grafičke kontrole, kad treba stvoriti jednostavne dizajnerske elemente sučelja. Promjena izgleda grafičkih kontrola tijekom izrade aplikacije je lakša od mijenjanja i ispitivanja programskog koda za grafički postupak.

Za više informacija Za informacije o stvaranju grafičkih aplikacija s događajima miša te postupcima Line ili Move, pogledajte odlomke "Događaj MouseDown" "Događaj MouseMove" i "Korištenje argumenta Button za poboljšavanje grafičkih aplikacija s mišem " u 11. poglavlju "Odgovaranje na događaje miša i tipkovnice".

Osnove crtanja grafičkim postupcima

Svaki grafički postupak crta izlazni rezultat na formu, u okvir za sliku ili u objekt tipa Printer. Kako bi naznačili gdje želite crtati, ispred grafičkog postupka napišite ime forme ili kontrole okvira za sliku. Ako izostavite objekt, Visual Basic pretpostavlja da želite crtati na formi kojoj je pridodan taj programski kod. Na primjer, sljedeći izrazi crtaju točku na:

- Formi imena MojaForma

```
MojaForma.Pset (500, 500)
```

- Okviru za sliku, imena picSlika1

```
picSlika1.Pset (500, 500)
```

- Trenutnoj formi

```
PSet (500, 500)
```

Svako područje crtanja ima svoj vlastiti koordinatni sustav koji određuje koje će se jedinice primjenjivati za koordinate. Kao dodatak, svako područje crtanja ima svoj vlastiti kompletan skup grafičkih svojstava.

Za više informacija Pogledajte "Ispis iz aplikacije", kasnije u ovom poglavlju, za više informacija o objektu tipa Printer. Pogledajte "Razumijevanje koordinatnog sustava", ranije u ovom poglavlju, za više informacija o koordinatama.

Čišćenje područja crtanja

Svaki put kad želite očistiti područje crtanja i početi ispočetka, upotrijebite postupak Cls. Određeno područje crtanja će biti ponovno iscrtano u boji pozadine (BackColor):

[*objekt.*]**Cls**

Upotreba postupka Cls bez određenog objekta čisti formu kojoj je pridodan takav programski kod.

Crtanje točaka

Nadziranje pojedinog piksela je jednostavna grafička operacija. Postupak PSet postavlja boju piksela na određenoj točki:

[*objekt.*]**PSet** (*x, y*)[, *boja*]

Argumenti *x* i *y* su jednostruke preciznosti, pa mogu prihvatiti unos cijelog broja ili razlomka. Unos može biti svaki brojčani izraz, uključujući varijable.

Ako ne uključite argument *boja*, postupak PSet postavlja piksel na boju crtanja (ForeColor). Na primjer, sljedeći izrazi postavljaju razne točke na trenutnoj formi (formi kojoj je pridodan taj programski kod), MojaForma, te okviru za sliku picSlika1:

```
PSet (300, 100)
PSet (10.75, 50.33)
MojaForma.PSet (230, 1000)
picSlika1.PSet (1.5, 3.2)
```

Dodavanje argumenta *boja* daje vam veći nadzor:

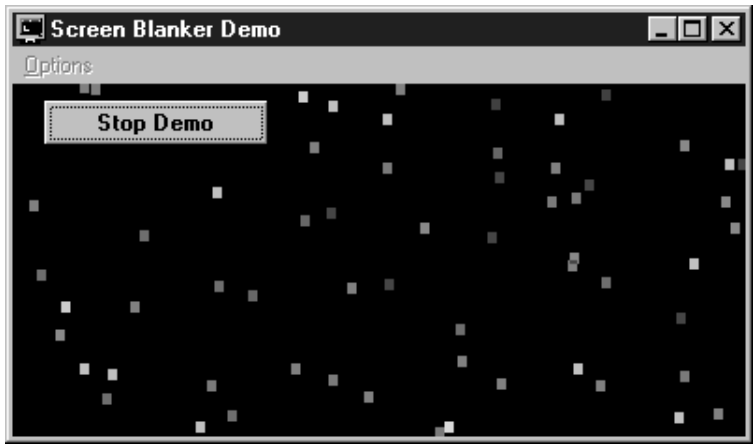
```
‘ Postavljanje točke 50, 75 u tamnoplavu.
PSet (50, 75), RGB(0, 0, 255)
```

Aplikacija Blanker crta točke u nasumce odabranim bojama kako bi stvorila primjer Confetti. potprogram PSetDemo stvara konfete:

```
Sub PSetDemo()
   ‘ Postavljanje varijable Red na nasumičnu vrijednost.
   R = 255 * Rnd
   ‘ Postavljanje varijable Green na nasumičnu vrijednost.
   G = 255 * Rnd
   ‘ Postavljanje varijable Blue na nasumičnu vrijednost.
   B = 255 * Rnd
   ‘ Postavljanje vodoravnog položaja.
   XPos = Rnd * ScaleWidth
   ‘ Postavljanje okomitog položaja.
   YPos = Rnd * ScaleHeight
   ‘ Crtanje točke s nasumičnom bojom.
   PSet (XPos, YPos), RGB(R, G, B)
End Sub
```

Prikaz rezultirajućih konfeta prikazan je na slici 12.13.

Slika 12.13 Prikaz konfeta u aplikaciji Blanker



Kako bi "obrisali" točku, postavite je na boju pozadine:

```
PSet (50, 75), BackColor
```

Kao što je opisano u idućem odlomku "Crtanje linija i likova", ispred (*x, y*) koordinata može napisati ključnu riječ Step, što čini točku relativnom prema posljednjem nacrtanom položaju.

Postupak Point je blisko povezan s postupkom PSet, ali vraća vrijednost boje na određenom položaju:

```
PointColor = Point (500, 500)
```

Za više informacija Za više informacija, pogledajte odlomke "Postupak PSet" i "Postupak Point" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Crtanje linija i likova

Iako čišćenje područja crtanja te crtanje pojedinih točaka može biti korisno, najzanimljiviji grafički postupci crtaju cijele linije i likove.

Crtanje linija

Kako bi nacrtali liniju između dvije koordinate, upotrijebite jednostavan oblik postupka Line, koji ima ovu sintaksu:

[*objekt.*]**Line** [(*x1, y1*)]–(*x2, y2*)[, *boja*]

Objekt je neobavezan; ako je izostavljen, postupak crta na formi kojoj je pridodan programski kod (trenutnoj formi). Prvi par koordinata je također neobavezan. Kao kod svih vrijednosti koordinata, argumenti *x* i *y* mogu biti cijeli brojevi ili razlomci. Na primjer, ovaj izraz crta standardnu liniju na formi:

```
Line (500, 500)-(2000, 2000)
```

Visual Basic crta liniju koja uključuje prvu krajnju točku, ali ne posljednju krajnju točku. Takvo ponašanje je korisno kod crtanja zatvorenog oblika od točke do točke. Kako bi nacrtali posljednju točku, upotrijebite ovu sintaksu:

PSet [**Step**] (0, 0)[, *boja*]

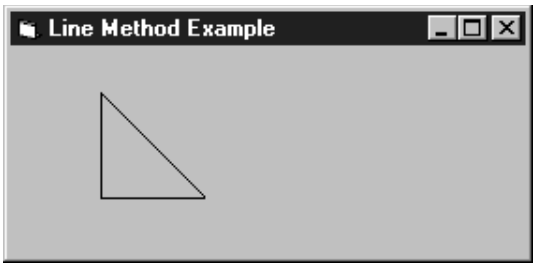
Prvi par koordinata (*x1, y1*) je neobavezan. Ako izostavite te koordinate, Visual Basic koristi trenutan x, y položaj objekta (koordinate crtanja) kao krajnju točku. Trenutan položaj može biti određen svojstvima CurrentX i CurrentY, ali inače je jednak posljednjoj točki nacrtanoj grafičkim ili Print postupkom. Ako niste prethodno koristili neki grafički ili Print postupak ili niste određivali svojstva CurrentX i CurrentY, podrazumijevani položaj je gornji lijevi kut objekta.

Na primjer, sljedeći izrazi crtaju trokut povezivanjem tri točke.

```
‘ Postavljanje x-koordinate početne točke.
CurrentX = 1500
‘ Postavljanje y-koordinate početne točke.
CurrentY = 500
‘ Crtanje linije prema dolje i desno od početne točke.
Line -(3000, 2000)
‘ Crtanje linije prema lijevo od trenutne točke.
Line -(1500, 2000)
‘ Crtanje linije gore i desno prema početnoj točki.
Line -(1500, 500)
```

Rezultat je prikazan na slici 12.14.

Slika 12.14 Trokut nacrtan postupkom Line



Aplikacija Blanker upotrebljava postupak Line za stvaranje zanimljivih uzoraka. Kako bi to vidjeli, odaberite Crossfire iz izbornika Options te kliknite gumb Start Demo.

Ključna riječ Step

Postupci PSet, Line i Circle određuju jednu ili više točaka korištenjem ove sintakse:

(*x, y*)

Ispred svake od tih točaka možete napisati ključnu riječ Step, određujući da je položaj točke relativan prema posljednjoj nacrtanoj točki. Visual Basic dodaje vrijednosti x i y vrijednostima posljednje nacrtane točke. Na primjer, ovaj izraz:

```
Line (100, 200)-(150, 250)
```

je jednak ovom:

```
Line (100, 200)-Step(50, 50)
```

U većini situacija, ključna riječ Step pošteđuje vas neprekidnog praćenja posljednje nacrtane točke. Često će vam biti zanimljiviji relativni položaji dvije točke nego njihovi apsolutni položaji.

Korištenje argumenta boje

Kako bi promijenili boju linije, upotrijebite neobavezan argument *boja* s grafičkim postupcima. Na primjer, ovaj izraz crta tamno plavu liniju:

```
Line (500, 500)-(2000, 2000), RGB(0, 0, 255)
```

Ako izostavite argument *boja*, boju crtanja određuje svojstvo ForeColor objekta gdje se crta linija.

Crtanje okvira

Upotrebom postupka Line možete nacrtati popunjene okvire. Sljedeći primjer crta okvir s gornjim lijevim kutom na (500, 500) i mjeri 1000 twipova na svakoj strani:

```
Line (500, 500)-Step(1000, 0)
Line -Step(0, 1000)
Line -Step(-1000, 0)
Line -Step(0, -1000)
```

Međutim, Visual Basic pruža puno jednostavniji način crtanja okvira. Kad upotrijebite opciju B s postupkom Line, Visual Basic crta pravokutnik, tretirajući određene točke kao suprotne kutove pravokutnika. Stoga, možete zamijeniti četiri linije prethodnog primjera sljedećom linijom:

```
Line (500, 500)-Step(1000, 1000), B
```

Uočite da su prije opcije B potrebna dva zareza, kako bi ukazali da je preskočen argument boje. Sintaksa postupka Line je pokrivena u odlomku "Crtanje linija i likova", ranije u ovom poglavlju.

Svojstva FillStyle i FillColor

Sve dok ne promijenite postavku svojstva FillStyle, okvir se pojavljuje prazan (okvir se ne popunjava s podrazumijevanom postavkom svojstva FillStyle, koja je 1 – Transparent). Svojstvo FillStyle možete promijeniti na bilo koju od postavki ispisanih u sljedećoj tablici.

postavka	opis
0	Popunjen. Ispunjava okvir bojom određenom za svojstvo FillColor.
1	Proziran (standardno). Grafički objekt se pojavljuje prazan, bez obzira na to koja se boja koristi.
2	Vodoravne linije.
3	Okomite linije.
4	Dijagonalne linije prema gore.
5	Dijagonalne linije prema dolje.
6	Ukrižane linije.
7	Dijagonalno ukrižane linije.

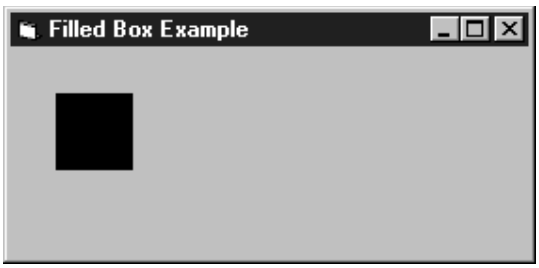
Prema tome, postavljanje svojstva FillStyle na 0 popunit će okvir bojom postavljenom za svojstvo FillColor.

Drugi način popunjavanja okvira je određivanje opcije **F** nakon opcije **B** (zapamtite da **F** ne može biti upotrijebljen bez **B**). Kad upotrijebite opciju **F**, postupak Line zanemaruje svojstva FillColor i FillStyle. Okvir je uvijek popunjen kad upotrijebite opciju **F**. Sljedeći izraz popunjava okvir punim uzorkom, koristeći svojstvo ForeColor:

```
Line (500, 500)-Step(1000, 1000), BF
```

Rezultat je prikazan na slici 12.15.

Slika 12.15 Okvir popunjen punim uzorkom



Crtanje krugova

Postupak Circle crta razne kružne i eliptične (ovalne) likove. Kao dodatak, postupak Circle crta lukove (dijelove kružnice) i odsječke kruga. Možete proizvesti puno vrsta savijenih linija korištenjem varijacija postupka Circle.

Kako bi nacrtali krug, Visual Basic treba položaj središta kruga te dužinu njegova polumjera. Sintaksa za savršen krug je:

[*objekt.*]**Circle** [**Step**](*x, y*), *polumjer*[, *boja*]

Uglate zagrade pokazuju da su *objekt* i ključna riječ Step neobavezni. Ako ne odredite objekt, pretpostavlja se trenutna forma. Argumenti x i y su koordinate središta, a *polumjer* je polumjer kruga. Na primjer, ovaj izraz crta krug sa središtem na (1200, 1000) te polumjerom od 750:

```
Circle (1200, 1000), 750
```

Točan učinak ovog izraza ovisi o veličini i koordinatnom sustavu forme. Budući da je veličina forme nepoznata, ne znate hoće li krug biti vidljiv. Korištenje svojstava područja crtanja postavlja središte kruga u središte forme:

```
Circle ((ScaleWidth + ScaleLeft) / 2, (ScaleHeight + _
ScaleTop) / 2)), ScaleWidth / 4
```

Za sad, sve što trebate znati o svojstvima ScaleWidth i ScaleHeight je da ona pomažu kod postavljanja slike u središte forme.

Za više informacija Odlomak “Promjena koordinatnog sustava objekta”, ranije u ovom poglavlju, raspravlja detaljno svojstva ScaleWidth i ScaleHeight.

> **Napomena** Polumjer kruga je uvijek određen u granicama vodoravnih jedinica. Ako vaš koordinatni sustav koristi iste vodoravne i okomite jedinice (što u pravilu čini), može zanemariti ovu činjenicu. Međutim, ako upotrebljavate korisničko mjerilo, vodoravne i okomite jedinice mogu odgovarati različitim udaljenostima. U prethodnim primjerima, polumjer je određen u vodoravnim jedinicama, a stvarna visina kruga je zajamčeno jednaka njegovoj stvarnoj širini.

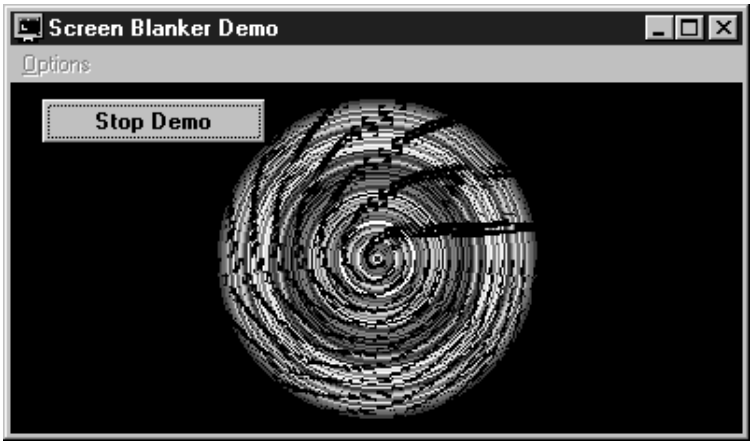
Aplikacija Blanker stvara krugove kao dio primjera Rainbow Rug. Taj primjer crta nizove krugova linijama od crtica koji se nalaze oko središta forme. Vremenom će krugovi nalikovati istkanom kružnom prostiraču. Potprogram CircleDemo stvara krugove u primjeru Rainbow Rug sa sljedećim izrazima:

```
Sub CircleDemo()
  Dim Radius
‘ Postavljanje varijable Red na nasumičnu vrijednost.
  R = 255 * Rnd
‘ Postavljanje varijable Green na nasumičnu vrijednost.
  G = 255 * Rnd
```

```
‘ Postavljanje varijable Blue na nasumičnu vrijednost.
  B = 255 * Rnd
‘ Postavljanje x-koordinate u sredinu forme.
  XPos = ScaleWidth / 2
‘ Postavljanje y-koordinate u sredinu forme.
  YPos = ScaleHeight / 2
‘ Postavljanje polumjera između 0 i 50% visine forme.
  Radius = (YPos * 0.9) + 1) * Rnd
‘ Crtanje kruga upotrebom nasumične boje.
  Circle (XPos, YPos), Radius, RGB(R, G, B)
End Sub
```

Rezultati primjera Rainbow Rug prikazani su na slici 12.16.

Slika 12.16 Primjer Rainbow Rug u aplikaciji Blanker



Crtanje lukova

Kako bi nacrtali lukove postupkom Circle, za određivanje *početka* i *kraja* luka trebate pružiti argumente kuta u radijanima. Sintaksa za crtanje luka je:

[*objekt.*]**Circle** [**Step**](*x, y*), *polumjer*, [*boja*], *početak*, *kraj*[, *omjer*]

Ako su argumenti *početak* ili *kraj* negativni, Visual Basic crta liniju koja povezuje središte kruga s negativnom krajnjom točkom. Na primjer, sljedeći potprogram crta tortu s izrezanim odsječkom.

```
Private Sub Form_Click()
  Const PI = 3.14159265
  Circle (3500, 1500), 1000, -PI / 2, - PI / 3
End Sub
```

Napomena Formula za pretvaranje iz stupnjeva u radijane je množenje stupnjeva sa Pi/180.

Crtanje elipsi

Omjer slike kruga nadzire hoće li se krug pojaviti savršeno zaokružen (krug) ili rastegnut (elipsa). Potpuna sintaksa postupka Circle je:

[*objekt.*]**Circle** [**Step**](*x, y*), *polumjer*, [*boja*], *početak*, *kraj*[, *omjer*]

Argumenti *početak* i *kraj* su neobavezni, ali potrebni su zarezi ako želite preskočiti argumente. Na primjer, ako uključite argumente *polumjer* i *omjer*, ali ne i argumente *boja*, *početak* i *kraj*, morate dodati četiri zareza za redom kako bi naznačili da preskačete tri argumenta:

```
Circle (1000, 1000), 500, , 2
```

Argument *omjer* određuje omjer okomite prema vodoravnoj dimenziji. Ovdje je *omjer* pozitivan broj s plivajućim zarezom. To znači da možete odrediti cjelobrojne ili razlomačke vrijednosti, ali ne i negativne vrijednosti. Velike vrijednosti *omjera* proizvode elipse razvučene po okomitoj osi, dok male vrijednosti *omjera* proizvode elipse razvučene po vodoravnoj osi. Budući da elipsa ima dva polumjera – jedan vodoravan x-polumjer i jedan okomit y-polumjer – Visual Basic pridružuje jedini argument *polumjer* naredbe Circle dužoj osi. Ako je *omjer* manji od jedan, *polumjer* je x-polumjer; ako je *omjer* veći ili jednak jedan, *polumjer* je y-polumjer.

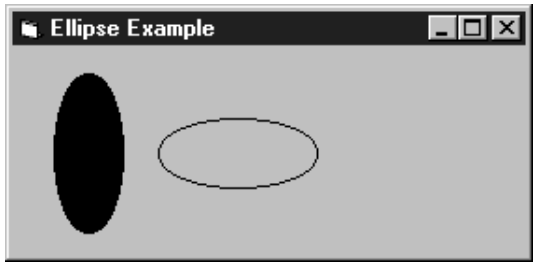
> **Napomena** Argument *omjer* uvijek određuje omjer između okomite i vodoravne dimenzije u granicama stvarne fizičke udaljenosti. Kako bi osigurao da se to događa (čak i kad upotrebljavate korisničko mjerilo), polumjer je određen u granicama vodoravnih jedinica.

Sljedeći potprogram pokazuje kako različite vrijednosti *omjera* određuju hoće li naredba Circle upotrijebiti argument *polumjer* kao x-polumjer ili y-polumjer elipse:

```
Private Sub Form_Click()
‘ Crtanje popunjene elipse.
  FillStyle = 0
  Circle (600, 1000), 800, , 3
‘ Crtanje prazne elipse.
  FillStyle = 1
  Circle (1800, 1000), 800, , 1 / 3
End Sub
```

Izlazni rezultat je prikazan na slici 12.17.

Slika 12.17 Elipse nacrtane postupkom Circle



Za više informacija Za više informacija o crtanju krugova i lukova, pogledajte "Crtanje krugova" ranije u ovom poglavlju.

Crtanje grafike na određenim položajima

Možete nacrtati grafiku na određenim položajima na formi, u okviru za sliku te prema objektu tipa Printer, upotrebom postupka PaintPicture. Sintaksa postupka PaintPicture je:

[*objekt.*]**PaintPicture** *slika*, *odrX*, *odrY*[, *odrŠirina*[, *odrVisina* _

[, *izvX*[, *izvY*[, *izvŠirina*[, *izvVisina* [, O*p*]]]]]]]

Odredišni *objekt* je forma, okvir za sliku ili objekt Printer gdje će *slika* biti prikazana. Ako je *objekt* izostavljen, pretpostavljena je trenutna forma. Argument *slika* mora biti objekt tipa Picture, kao iz svojstva Picture forme ili kontrole.

Argumenti *odrX* i *odrY* su vodoravan i okomit položaj na kojima će slika biti prikazana u modu mjerila *objekta*. Argumenti *odrŠirina* i *odrVisina* su neobavezni i određuju širinu i visinu s kojima će slika biti prikazana na odredišnom *objektu*.

Argumenti *izvX* i *izvY* su neobavezni i određuju x-koordinatu i y-koordinatu gornjeg lijevog kuta područja izrezivanja unutar *slike*.

Neobavezan argument *Op* određuje operaciju rastera (kao AND ili XOR) koja će biti izvedena na slici dok se iscrtava na odredišnom *objektu*.

Postupak PaintPicture može se upotrijebiti umjesto funkcije BitBlt Windows API-ja za izvođenje široke raznolikosti operacija bitovima dok se pravokutni blok grafike pomiče s jednog položaja na bilo koji drugi položaj.

Na primjer, postupak PaintPicture možete koristiti za stvaranje višestrukih kopija iste bitmapirane slike, te ih posložiti na formi. Upotreba ovog postupka je brža od pomicanja kontrola slike po formi. Sljedeći kod slaže 100 kopija kontrole slike i preokreće svaku sliku vodoravno davanjem negativne vrijednosti za argument *odrŠirina*.

```
For i = 0 To 10
  For j = 0 To 10
    Form1.PaintPicture picF.Picture, j * _
    picF.Width, i * picF.Height, _
    picF.Width, -picF.Height
Next j, i
```

Za više informacija Pogledajte "Postupak PaintPicture" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Određivanje širine linije

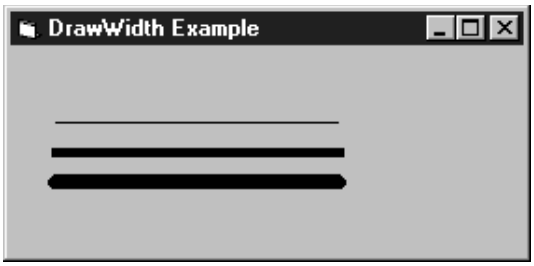
Svojstvo DrawWidth određuje širinu linije izlaznih rezultata grafičkih postupaka.
Svojstvo BorderWidth određuje debljinu obrisa kontrole linije i kontrola lika.

Sljedeći potprogram crta linije nekoliko različitih širina.

```
Private Sub Form_Click()
  DrawWidth = 1
  Line (100, 1000)-(3000, 1000)
  DrawWidth = 5
  Line (100, 1500)-(3000, 1500)
  DrawWidth = 8
  Line (100, 2000)-(3000, 2000)
End Sub
```

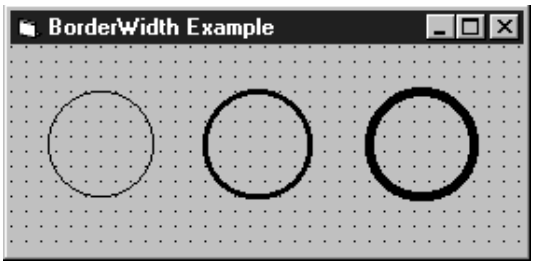
Rezultati su prikazani na slici 12.18.

Slika 12.18 Učinci mijenjanja svojstva DrawWidth



Slika 12.19 prikazuje tri kontrole lika s različitim vrijednostima svojstva BorderWidth.

Slika 12.19 Učinci mijenjanja svojstva BorderWidth



Određivanje punih ili isprekidanih linija

Svojstvo DrawStyle određuje hoće li linije stvorene grafičkim postupcima biti pune ili isprekidane. Svojstvo BorderStyle kontrole lika nudi istu funkciju kao svojstvo DrawStyle, ali se primjenjuje za razne objekte.

Napomena Svojstvo BorderStyle kontrole lika i kontrole linije služi drugačijim namjenama i koristi drugačije postavke od svojstva BorderStyle u drugim objektima, u kontrolama i u formama. Za kontrole lika i linije, svojstvo BorderStyle radi slično svojstvu DrawStyle kao što je opisano u ovom odlomku. Za forme i druge kontrole, svojstvo BorderStyle određuje hoće li kontrola ili forma imati rub te ako hoće, je li rub nepomičan ili promjenjive veličine.

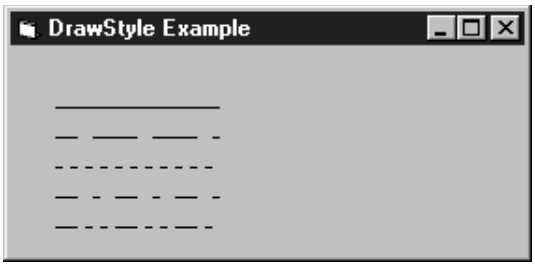
Stilovi popunjavanja i popunjavanja iznutra

Stil popunjavanja iznutra (svojstvo DrawStyle ili BorderStyle = 6) je gotovo jednak stilu popunjavanja. Oba stvaraju punu liniju. Razlika između ove dvije postavke postaje očita kad upotrijebite široku liniju za crtanje okvira ili kontrole lika. U takvim slučajevima, stil popunjavanja crta liniju tako da je pola linije unutar, a pola linije izvan okvira ili lika. Stil popunjavanja iznutra crta liniju tako da je cijela linija unutar okvira ili lika. Pogledajte odlomak “Crtanje okvira”, ranije u ovom poglavlju, da vidite kako se crta okvir.

Sljedeći potprogram prikazuje sve podržane postavke svojstva DrawStyle stvaranjem petlje u kojoj postavke idu od 0 do 6, jedna po jedna vrijednost. Rezultati su prikazani na slici 12.20.

```
Private Sub Form_Click()
  Dim I As Integer, Y As Long
  For I = 0 To 6
    DrawStyle = I
    Y = (200 * I) + 1000
    Line (200, Y)-(2400, Y)
  Next I
End Sub
```

Slika 12.20 Učinci mijenjanja svojstva DrawStyle



Za više informacija Pogledajte odlomke “Svojstvo DrawStyle” i “Svojstvo BorderStyle” u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Nadzor prikaza korištenjem svojstva DrawMode

Svojstvo DrawMode određuje što se događa kad nacrtate jednu sliku iznad druge. Iako promjena svojstva DrawMode obično ima neki učinak (posebno sa sustavima boje), često nije potrebno upotrebljavati ovo svojstvo kad crtate na praznoj ili čisto bijeloj pozadini, ili na pozadini s bojama koje se bitno ne razlikuju.

Svojstvo DrawMode možete postaviti na vrijednosti od 1 do 16. Uobičajene postavke ispisane su u sljedećoj tablici.

postavka	opis
4 – Not Copy Pen	Crta obratno od uzorka linije, neovisno o tome što je već nacrtano.
7 – Xor Pen	Prikazuje razliku između uzorka linije i postojećeg prikaza, kao što je objašnjeno kasnije u ovom odlomku. Crtanje objekta dvaput s ovim modom obnavlja pozadinu točno kakva je i bila.
11 – Nop	Nema operacije. Zapravo, isključuje crtanje.
13 – Copy Pen	Standardna postavka. Prihvaća uzorak linije, neovisno o tome što je već nacrtano.

Za više informacija Pogledajte "Svojstvo DrawMode" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Postavka Xor Pen

Postavka 7 – Xor Pen, svojstva DrawMode, korisna je za animaciju. Crtanje linije dvaput obnavlja postojeći prikaz točno kako je izgledao prije nego što je linija nacrtana. To omogućuje stvaranje objekta koji se "pomiče iznad" pozadine bez da ju kvari, budući da možete obnoviti pozadinu dok pomičete objekt. Većina modova nema jamstvo čuvanja stare pozadine.

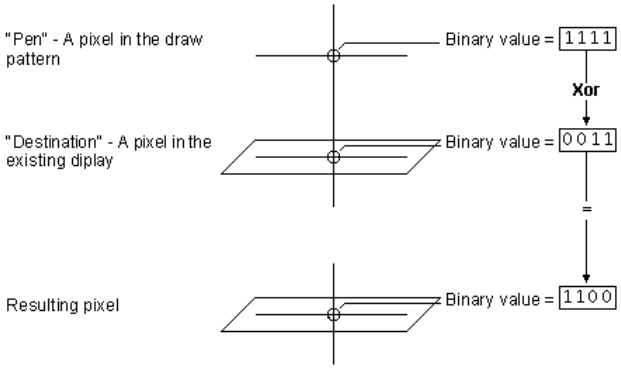
Na primjer, sljedeći programski kod pomiče krug svaki put kad je kliknut miš. Bez obzira koji je uzorak ispod kruga, obnavlja se.

```
Private Sub Form_Click()
  ForeColor = 255: DrawMode = 7
  Circle (CurrentX, CurrentY), 1000
  CurrentX = CurrentX + 220
  CurrentY = CurrentY + 220
  Circle (CurrentX, CurrentY), 1000
End Sub
```

Mod crtanja Xor Pen (i većina ostalih postavki svojstva DrawMode) radi uspoređujući svaki pojedini piksel u uzorku crtanja (nazvan "Olovka" (Pen)) s pripadajućim pikselom u postojećem području (nazvan "Odredište"). Na jednobojnim sustavima, piksel se uključuje ili isključuje, a Visual Basic izvodi jednostavnu logičku operaciju: uključuje piksel ako je uključen piksel olovke ili odredišta, ali ne ako su oba uključena.

Na sustavima u boji, svakom pikselu je dodijeljena vrijednost boje. Za postavke svojstva DrawMode kao što je Xor Pen, Visual Basic uspoređuje svaki odgovarajući par piksela u olovci i odredištu i izvodi binarno (bitovno) uspoređivanje. Rezultat određuje vrijednost boje rezultirajućeg piksela, kao što je prikazano na slici 12.21.

Slika 12.21 Korištenje postavke Xor Pen za postavljanje binarne vrijednosti piksela u liniji



Stvaranje grafike kad se učita forma

Kad stvarate grafiku koja se pojavljuje na formi kad se forma učita, razmislite o postavljanju grafičkih postupaka u događaj Form_Paint. Tako nacrtana grafika bit će automatski ponovno iscrtana kod svakog događaja crtanja. Ako postavite grafiku u događaj Form_Load, postavite svojstvo AutoRedraw forme na True. U tom slučaju, događaj Form_Load trebao bi prikazati formu, te zatim nacrtati grafiku. Zapamtite, forme nisu vidljive tijekom događaja Form_Load. Budući da Visual Basic ne obrađuje grafičke postupke na formi koja nije vidljiva, grafički postupci u događaju Form_Load se zanemaruju osim ako svojstvo AutoRedraw nije postavljeno na True.

Rad s bojama

Visual Basic upotrebljava dosljedan sustav za sva svojstva boje i grafičke postupke. Boja je predstavljena cijelim brojem tipa Long, i ta vrijednost ima isto značenje u svim sklopovima koji određuju boju.

Određivanje boja tijekom izvođenja aplikacije

Postoje četiri načina određivanja vrijednosti boje tijekom izvođenja aplikacije:

- Korištenje funkcije RGB.
- Korištenje funkcije QBColor za odabir jedne od 16 boja aplikacije Microsoft QuickBasic.
- Korištenje jedne od ugrađenih konstanti ispisanih u pretraživaču objekata.
- Izravan unos vrijednosti boje.

Ovaj odlomak raspravlja kako upotrebljavati funkcije RGB i QBColor kao jednostavne načine određivanja boje. Pogledajte "Korištenje svojstava boje" kasnije u ovom poglavlju za informacije o korištenju konstanti za određivanje boje ili o izravnom unosu vrijednosti boje.

Korištenje funkcije RGB

Funkciju RGB možete upotrijebiti za određivanje bilo koje boje.

Kako upotrijebiti funkciju RGB za određivanje boje

1. Dodijelite svakoj od tri osnovne boje (crvena, zelena i plava) broj između 0 i 255, gdje 0 označuje najmanji intenzitet, a 255 najveći.
2. Postavite ta tri broja kao ulazne podatke funkciji RGB, koristeći redoslijed crvena-zelena-plava.
3. Dodijelite rezultat svojstvu boje ili argumentu boje.

Svaka vidljiva boja može biti proizvedena kombiniranjem jedne ili više od tri osnovne boje. Na primjer:

```
‘ Postavljanje pozadine na zelenu.
Form1.BackColor = RGB(0, 120, 0)
‘ Postavljanje pozadine na žutu.
Form2.BackColor = RGB(255, 255, 0)
‘ Postavljanje točke u tamnoplavu.
PSet (100, 100), RGB(0, 0, 64)
```

Za više informacija Za informacije o funkciji RGB, pogledajte "Funkcija RGB" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Korištenje svojstava boje

Puno kontrola u Visual Basicu ima svojstva koja određuju boje upotrijebljene za prikaz kontrole. Imajte na umu da se neka od tih svojstava također primjenjuju za kontrole koje nisu grafičke. Sljedeća tablica ispisuje svojstva boje.

svojstvo	opis
BackColor	Određuje boju pozadine forme ili kontrole koja se koristi za crtanje. Ako promijenite vrijednost svojstva BackColor nakon korištenja grafičkih postupaka za crtanje, nova boja pozadine briše postojeću grafiku.
ForeColor	Određuje boju kojom će grafički postupci stvarati tekst ili grafiku u formi ili kontroli. Promjena vrijednosti svojstva ForeColor ne djeluje na tekst ili grafiku koja je već stvorena.
BorderColor	Određuje boju ruba kontrole lika.
FillColor	Određuje boju kojom se popunjavaju krugovi stvoreni postupkom Circle i okviri stvoreni postupkom Line.

Za više informacija Za detaljne informacije o ovim svojstvima boja, pogledajte odlomke “Svojstva BackColor, ForeColor” “Svojstvo BorderColor” i “Svojstvo FillColor” u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Određivanje boja

Svojstva boje mogu upotrijebiti svaki od nekoliko postupaka za određivanje vrijednosti boje. Funkcija RGB opisana u odlomku “Rad s bojama”, ranije u ovom poglavlju, je jedan način određivanja boja. Ovaj dio raspravlja dva daljnja načina određivanja boja:

- Korištenje definiranih konstanti
- Korištenje izravnih postavki boje

Korištenje definiranih konstanti

Ne trebate razumjeti kako se stvaraju vrijednosti boja ako upotrijebite ugrađene konstante ispisane u pretraživaču objekata. Kao dodatak, ugrađene konstante ne trebaju biti prethodno određene. Na primjer, možete upotrijebiti konstantu vbRed uvijek kad želite odrediti crveno kao argument boje ili postavku svojstva boje:

```
BackColor = vbRed
```

Korištenje izravnih postavki boje

Korištenje funkcije RGB ili ugrađenih konstanti za određivanje boje su posredni postupci. Oni su posredni jer ih Visual Basic tumači u jednostruki pristup koji koristi za predstavljanje boje. Ako shvatite kako su boje predstavljene u Visual Basicu, možete dodijeliti brojeve svojstvima boja i argumentima koji izravno određuju boju. U većini slučajeva, puno je lakše unijeti te brojeve u heksadecimalnom sustavu.

Valjan opseg normalne RGB boje je 0 do 16 777 215 (&HFFFFFF&). Svaka postavka boje (svojstvo ili argument) je cjeli broj predstavljen s 4 bajta. Viši bajt broja u tom opsegu jednak je 0. Niža 3 bajta, od najmanje značajnog do najznačajnijeg bajta, određuju količinu crvene, zelene i plave, respektivno. Crvena, zelena i plava komponenta su predstavljene brojem između 0 i 255 (&HFF) svaka.

Nadalje, možete odrediti boju kao heksadecimalni broj korištenjem ove sintakse:

```
&HPPZZCC&
```

PP označuje količinu plave, ZZ količinu zelene, a CC količinu crvene komponente. Svaki od ova tri dijela je dvoznamenkasti heksadecimalni broj od 00 do FF. Srednja vrijednost je 80. Stoga, sljedeći broj određuje sivu boju, što je srednja količina sve tri boje:

```
&H808080&
```

Postavljanje najznačajnijeg bita na 1 mijenja značenje vrijednosti boje. Ona više ne predstavlja RGB boju, nego opću boju okruženja određenu kroz kontrolni panel Windowsa. Vrijednosti koje odgovaraju tim općim bojama sustava su u opsegu od &H80000000 do &80000015.

Napomena Iako možete odrediti preko 16 milijuna različitih boja, nisu svi sustavi sposobni prikazati ih točno. Za više informacija o tome kako Windowsi predstavljaju boje, pogledajte "Rad s 256 boja", kasnije u ovom poglavlju.

Korištenje sistemskih boja

Kad određujete boje kontrola ili formi u svojoj aplikaciji, koristite boje koje su određene od operativnog sustava umjesto određenih vrijednosti boja. Ako odredite sistemske boje, kad korisnik vaše aplikacije promijeni vrijednosti sistemskih boja na svojem računalu, vaša aplikacija će automatski odražavati korisnički određene vrijednosti boja.

Svaka sistemska boja ima definiranu konstantu i izravnu postavku boje. Viši bajt postavki boja za sistemske boje razlikuje se onih za normalne RGB boje. Za RGB boje, viši bajt jednak je 0 dok je kod sistemskih boja viši bajt jednak 8. Ostatak broja upućuje na određenu sistemsku boju. Na primjer, heksadecimalni broj koji se koristi za predstavljanje boje naslova aktivnog prozora je &80000002&.

Kad tijekom izrade aplikacije odredite svojstva boje u prozoru sa svojstvima, odabir kartice System omogućuje vam odabir sistemskih postavki, koje su automatski pretvorene u heksadecimalnu vrijednost. Definirane konstante za sistemske brojeve možete također pronaći u pretraživaču objekata.

Rad s 256 boja

Visual Basic podržava 256 boja na sustavima s videokarticama i pogoniteljima prikaza koji rukuju sa 256 ili više boja. Sposobnost istovremenog prikaza 256 boja je osobito dragocjena u multimedijskim aplikacijama ili aplikacijama koje trebaju prikazati slike kvalitete bliske fotografskoj.

Možete prikazati 256-bojne slike i odrediti do 256 boja za grafičke postupke u:

- formama
- okvirima za sliku
- kontrolama slike (samo prikaz slike)

Napomena Podrška za 256 boja ne primjenjuje se za Windows metadatoteke. Visual Basic prikazuje metadatoteke korištenjem standardne palete od 16 VGA boja.

Palete boja

Palete boja pružaju bazu za podršku 256 boja u aplikacijama Visual Basica. U raspravljanju o paletama, važno je razumjeti odnos između različitih tipova palete. *Hardverska paleta* sadrži 256 unosa određujući stvarne RGB vrijednosti koje će biti prikazane na ekranu. *Hardverska kopirana paleta* (*halftone palette*) sadrži unaprijed određeni niz od 256 RGB vrijednosti na raspolaganju samim Windowsima. *Logička paleta* je skup do 256 RGB vrijednosti sadržanih unutar bitmapirane ili slike drugačijeg tipa.

Windowsi mogu crtati koristeći 256 boja u hardverskoj paleti. Dvadeset od tih 256 boja, nazivane *statičke boje*, rezervirane su od sustava i ne mogu biti promijenjene od aplikacije. Statičke boje uključuju 16 boja u standardnoj VGA paleti (iste kao i boje određene funkcijom QBColor Visual Basica), uz četiri dodatne nijanse sive. Sistemska kopirana paleta uvijek sadrži te statičke boje.

Prozor prednjeg plana (prozor s fokusom) određuje 236 nestatičkih boja u hardverskoj paleti. Svaki put kad se promijeni hardverska paleta, svi prozori u pozadini ponovno se iscrtavaju korištenjem tih boja. Ako boje u logičkoj paleti pozadinskog prozora potpuno ne odgovaraju bojama koje su trenutno u hardverskoj paleti, Windowsi će dodijeliti najbliži par.

Prikaz slika s 256 boja

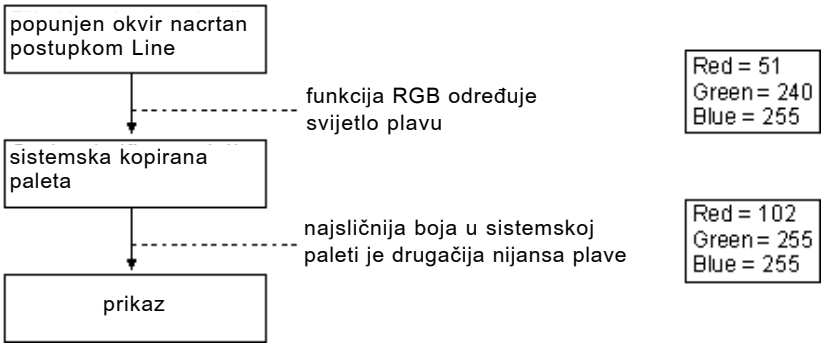
Forme, okviri za sliku i kontrole slike automatski prikazuju slike u 256 boja ako hardver i softver prikaza korisnika mogu podržati toliko boja na ekranu. Ako sustav korisnika podržava manje boja od slike, Visual Basic će preslikati sve boje u najsličnije od dostupnih.

Na prikazima sa stvarnim bojama (true-color, 16 milijuna boja), Visual Basic uvijek upotrebljava točnu boju. Na jednobojnim prikazima ili prikazima sa 16 boja, Visual Basic će diterirati boje pozadine te boje postavljene svojstvom FillColor. *Diteriranje* je proces koji se koristi za oponašanje boja koje nisu dostupne iz videokartica i pogonitelja prikaza.

Crtanje s paletama boja

Kod video pogonitelja s 256 boja, možete koristiti do 256 boja s grafičkim postupcima. U pravilu, 256 boja dostupnih u Visual Basicu su boje u kopiranoj paleti sustava. Iako možete odrediti točnu boju upotrebom funkcije RGB, stvarno prikazana boja bit će najsličnija boja iz kopirane palete, kao što je prikazano na slici 12.22.

Slika 12.22 Usklađivanje boja od određene boje do prikaza



Iako je podrazumijevana paleta Visual Basica sistemska kopirana paleta, prikaz boja možete također nadzirati svojstvima PalleteMode i Pallete formi, korisničkih kontrola i korisničkih dokumenata. U tom slučaju, usklađivanje boja je isto, osim što će te boje biti usklađene s najsličnijom bojom u hardverskoj paleti.

Upravljanje višestrukim paletama boja

Kad radite s paletama boja, imajte na umu da većina prikaza može prikazati samo 256 boja istovremeno na ekranu.

Ovo ograničenje postaje važno kad u svojoj aplikaciji upotrebljavate više od jedne palete. Na primjer, na jednoj formi, možete prikazati bitmapiranu sliku od 256 boja u kontroli slike dok prikazujete drugu sliku u okviru za sliku. Ako logičke palete ove dvije slike ne sadržavaju potpuno istih 256 boja, Windowsi moraju odlučiti koja logička paleta prva postavlja svoje boje u hardversku paletu. Zapamtite: hardverska paleta određuje što se zapravo pojavljuje na ekranu.

Slična situacija događa se kad vaša Visual Basic aplikacija ima dvije ili više formi s različitim logičkim paletama. Kad neka od tih formi primi fokus, njezina logička paleta nadzire hardversku paletu. To često može rezultirati slabijim prikazom od optimalnog na sustavima s 256 boja. Kao programer Visual Basica, možete nadzirati hardversku paletu upotrebom svojstva PaletteMode.

Svojstvo PaletteMode

Kad oblikujete aplikacije koje se mogu izvoditi na sustavima s 256 boja, možete nadzirati način kojim Windowsi odabiru kontrole prikaza određivanjem svojstva PaletteMode forme, korisničke kontrole ili korisničkog dokumenta (korisničke kontrole i korisnički dokumenti dostupni su samo u verzijama Professional i Enterprise). Sve kontrole sadržane na formi, korisničkoj kontroli ili korisničkom dokumentu bit će prikazane temeljeno na svojstvu PaletteMode. Sljedeća tablica prikazuje raspoložive postavke svojstva PaletteMode:

mod	konstanta	primjena za
Halftone	vbPaletteModeHalftone	forme, korisničke kontrole, korisnički dokumenti
UseZOrder	vbPaletteModeUseZOrder	forme, korisničke kontrole, korisnički dokumenti
Custom	vbPaletteModeCustom	forme, korisničke kontrole, korisnički dokumenti
Container	vbPaletteModeContainer	korisničke kontrole
None	vbPaletteModeNone	korisničke kontrole
Object	vbPaletteModeObject	activeX kreatori koji sadrže paletu

Svojstvo PaletteMode primjenjuje se samo za prikaze s 256 boja. Na prikazima s high-color bojama i true-color bojama, odabirom boje rukuje upravljački program slike korištenjem palete s 32 000 ili 16 milijuna boja, respektivno. Čak i ako programirate na sustavu koji ima prikaz s high-color ili true-color bojama, možete trebati odrediti svojstvo PaletteMode, jer većina vaših korisnika možda koristi prikaze s 256 boja.

Svojstvo PaletteMode može biti postavljeno tijekom izrade aplikacije u prozoru sa svojstvima, ili promijenjeno tijekom izvođenja kroz kod. Primjer aplikacije Palettes prikazuje učinke prikazivanja slika sa različitim paletama korištenjem nekoliko različitih postavki svojstva PaletteMode.

Napomena Za prethodne verzije Visual Basica, svojstvo PaletteMode podudara se sa svojstvom UseZOrder.

Mod Halftone svojstva PaletteMode

Podrazumijevani mod za forme i korisničke dokumente je mod Halftone. U ovom modu, sve kontrole, slike sadržane na formi ili grafički postupci koriste sistemsku kopiranu paletu.

Mod Halftone je dobar izbor u većini slučajeva jer pruža kompromis između slika na vašoj formi, te boja upotrijebljenih u drugim formama ili slikama. Ovaj mod, međutim, može rezultirati smanjenjem kvalitete nekih slika. Na primjer, slika s paletom od 256 nijansi sive može izgubiti detalje ili prikazati neočekivane tragove drugih boja.

Mod UseZOrder svojstva PaletteMode

Z-redoslijed (Z-order) je relativno raspoređivanje koje određuje kako se kontrole međusobno preklapaju na formi. Kad je svojstvo PaletteMode forme koja ima fokus postavljeno na UseZOrder, paleta najgornje kontrole uvijek ima prednost. To znači da svaki put kad nova kontrola postane najgornja (na primjer, kad učitate novu sliku u okvir za sliku), hardverska paleta bit će ponovno preslikana. To će često uzrokovati popratni efekt poznat kao bljesak palete: prikaz na trenutak bljesne dok se prikazuju nove boje, i u trenutnoj formi i u svim ostalim vidljivim formama ili aplikacijama.

Iako postavka UseZOrder pruža najtočnije tumačenje boja, dolazi do gubljenja brzine. Dodatno, ovaj postupak kao posljedicu može imati pojavljivanje diterirane boje pozadine forme ili kontrola koje nemaju slike. Postavljanje svojstva PaletteMode na postavku UseZOrder je najbolji izbor kad je točan prikaz najgornje slike važniji od smetnji bljeskanja palete, ili kad trebate održavati sukladnost prema nazad s ranijim verzijama Visual Basica.

Mod Custom svojstva PaletteMode

Ako trebate precizniji nadzor nad stvarnim prikazom boja, možete upotrijebiti sliku s 256 boja za određivanje korisničke palete. Kako bi to napravili, dodijelite sliku s 256 boja (datoteku tipa .bmp, .cur, .ico, .dib ili .gif) svojstvu Palette forme i postavite svojstvo PaletteMode na postavku Custom. Bitmapirana slika ne mora biti vrlo velika: čak i samo jedan piksel može odrediti do 256 boja za formu ili okvir za sliku. To je zato što logička paleta bitmapirane slike može sadržavati do 256 boja, neovisno o tom pojavljuju li se te boje u slici.

Kao i kod podrazumijevane postavke, boje koje odredite upotrebom funkcije RGB moraju također postojati u slici. Ako se boja ne slaže, bit će preslikana najsličnijom bojom iz logičke palete slike koja je dodijeljena svojstvu Palette.

Kako bi svojstvo PaletteMode postavili na Custom tijekom izvođenja, dodajte sljedeći kod u događaj Form_Load (uz pretpostavku da je slika koja sadržava vašu odabranu paletu dodijeljena kontroli slike imena Slika1):

```
‘ Dodjela palete iz kontrole Slika1 formi.
Form1.Palette = Slika1.Picture
‘ Korištenje moda Custom
Form1.PaletteMode = vbPaletteModeCustom
```

Alternativno, možete upotrijebiti objekt tipa Picture za postizanje istog učinka bez dodatne kontrole slike:

```
Dim objPic As Picture
Set objPic = LoadPicture(App.Path & “\Pastela.bmp”)
‘ Dodjela palete objekta slike formi.
Form1.Palette = objPic
‘ Korištenje moda Custom
Form1.PaletteMode = vbPaletteModeCustom
```

Postavka Custom svojstva PaletteMode je vaš najbolji izbor ako želite održavati jedinstvenu paletu kroz svoju aplikaciju.

Napomena Korištenje postavke Custom svojstva PaletteMode može također poboljšati izvođenje vaše aplikacije u slučajevima kad ne koristite nikakve slike sa 256 boja. Ako postavite svojstvo PaletteMode forme na Custom i ostavite svojstvo Palette prazno, vaša forma će se učitavati brže jer se ne pojavljuje slaganje paleta.

Za više informacija Kako bi naučili više o objektu tipa Picture, pogledajte "Korištenje objekta tipa Picture", kasnije u ovom poglavlju.

Ostali modovi paleta

Dvije dodatne postavke svojstva PaletteMode dostupne su kad stvarate korisničke kontrole: postavke Container i None. Mod Container preslikava paletu korisničke kontrole i svih sadržanih kontrola u okolne boje spremnika (forme ili korisničkog dokumenta) tijekom izvođenja aplikacije. Ako spremnik ne pribavljuje okolnu paletu, bit će pozvan mod Halftone. Budući da možda nećete unaprijed znati gdje će vaša korisnička kontrola biti razvijena, taj mod može spriječiti vašu kontrolu od sukobljavanja s ostalim postupcima rukovanja paletom.

Mod None radi baš ono što možete očekivati: potpuno uklanja rukovanje paletom. Kod stvaranja korisničke kontrole koja ne prikazuje slike ili grafiku, postavljanje svojstva PaletteMode na None poboljšava izvođenje uklanjanjem dodatne nadgradnje potrebne za obradu poruka palete.

Korištenje relativnih boja palete

Kad oblikujete prikaze s 256 boja, neke boje mogu se pojaviti diterirane. To može učiniti tekst i ostale grafičke elemente teškima za čitanje. Određivanjem relativne boje palete, Visual Basic će prikazati najbližu nediteriranu približnu vrijednost određene boje na prikazima s 256 boja dok će i dalje prikazivati točnu boju na većim dubinama boja.

Kako bi prisilili Visual Basic da upotrijebi najbližu temeljnu, prije nego diteriranu, boju za dano svojstvo, postavite broj 2 u najvažniji bajt svojstva boje. Na primjer, kako bi prisilili pozadinu forme da bude obojana temeljnom svijetlonarančastom bojom možete upotrijebiti sljedeći kod:

```
Private Function PaletaRGB(RGB As Long) As Long
  PaletaRGB = &H02000000 Or RGB
End Function
```

Ako postavite ovo tijekom izvođenja aplikacije:

```
Form1.BackColor = &H00C0E0FF&     ' diterirana svijetlonarančasta
```

I dodate sljedeće u događaj Form_Click:

```
Private Sub Form1_Click()
  Form1.BackColor = PaletaRGB(Form1.BackColor)
End Sub
```

Tijekom izvođenja, kad se klikne forma, boja pozadine će se promijeniti u temeljnu, prije nego u diteriranu, nijansu. Ona sad koristi najbližu boju iz kopirane palete. Ovaj učinak možda neće biti vidljiv na sustavima koji rade s dubinom boje veće od 256 boja.

Korištenje objekta tipa Picture

Objekt tipa Picture je u nekim pogledima sličan objektu tipa Printer – ne možete ga vidjeti, ali je unatoč tome koristan. O objektu tipa Picture možete razmišljati kao o nevidljivom okviru za sliku kojeg možete upotrebljavati kao privremeno područje za slike. Na primjer, sljedeći kod učitava bitmapiranu sliku u objekt tipa Picture te ju koristi za postavljanje svojstva Picture kontrole okvira za sliku:

```
Private Sub Command1_Click()
  Dim objPic As Picture
  Set objPic = LoadPicture("Leptir.bmp")
  Set Picture1.Picture = objPic
End Sub
```

Objekt tipa Picture podržava bitmapirane slike, slike tipa GIF, slike tipa JPEG, metadatoteke i ikone.

Korištenje matrica objekata tipa Picture

Možete također upotrijebiti matricu objekata tipa Picture za čuvanje niza grafika u memoriji bez korištenja forme koja sadržava više okvira za sliku ili kontrola slike. To je prikladno za stvaranje animacijskih sljedova ili drugih aplikacija gdje se traže brze izmjene slika. Odredite matricu na razini modula:

```
Dim objPics(1) As Picture
```

Dodajte sljedeći kod u događaj Form_Load:

```
' Učitavanje bitmapa u matricu objekata Picture.
Set objPics(0) = LoadPicture("Leptir1.bmp")
Set objPics(1) = LoadPicture("Leptir2.bmp")
```

U događaju Timer zatim možete kružiti kroz slike:

```
Static intBrojač As Integer
If intBrojač = 0 Then
  intBrojač = 1
Else
  intBrojač = 0
End If
' Korištenje postupka PaintPicture
' za prikaz bitmapa na formi.
PaintPicture objPics(intBrojač), 0, 0
```

Dodavanjem petlje za povećavanje x i y koordinata, lako možete napraviti da bitmapirane slike leptira "lete" po formi.

Korištenje objekta tipa Picture umjesto Windows API-ja

Postoji puno stvari koje možete učiniti s bitmapiranim slikama, ikonama ili metadatotekama u Windows API-ju, ali objekt tipa Picture već čini većinu toga za vas. To znači da je bolje koristiti objekt tipa Picture umjesto Windows API gdje je to moguće. Objekt tipa Picture vam također omogućuje korištenje datoteka tipa .jpg i .gif, što nije moguće s Windows API-jem.

Ne postoji izravna veza između svojstava Picture.Handle i PictureBox.hDC. Svojstvo hDC okvira za sliku je hvataljka koju pruža operativni sustav za sklop uređaja kontrole okvira za sliku. Svojstvo Handle objekta tipa Picture je zapravo hvataljka GDI objekta koji je sadržan u objektu tipa Picture.

Postoje dva potpuno različita načina za crtanje grafike u prozoru. Možete upotrijebiti funkcije BitBlt ili StretchBlt na svojstvu hDC objekta, ili možete upotrijebiti postupak PaintPicture na objektu tipa Picture ili svojstvu Picture. Ako imate kontrolu slike, možete koristiti samo postupak PaintPicture jer kontrole slike nemaju svojstvo hDC.

Za više informacija Za više informacija o Windows API-ju, pogledajte 4. dio "Pristup DLL-ovima i Windows API-ju" u vodiču *Microsoft Visual Basic 6.0 Component Tools Guide* biblioteke *Microsoft Visual Basic 6.0 Reference Library*.

Ispis

Ispis je jedan od najsloženijih zadataka koje izvode aplikacije temeljene na Windowsima. Dobri rezultati ovise o zajedničkom radu svih dijelova procesa. Slabi rezultati mogu izaći iz problema u vašoj aplikaciji, razlikama u pogoniteljima pisača, ili ograničenim sposobnostima pisača. Iako je dobra ideja ispitati svoju aplikaciju sa uobičajeno korištenim pisačima te pogoniteljima pisača, ne možete ispitati sve moguće kombinacije koje mogu imati korisnici.

Ispis iz vaše aplikacije sadržava ova tri dijela:

- Programski kod u vašoj aplikaciji koji počinje proces tiskanja.
- Pogonitelji pisača instalirani i na vaš sustav i na sustav korisnika vaše aplikacije.
- Sposobnosti pisača dostupnih korisnicima vaše aplikacije.

Programski kod u vašoj aplikaciji određuje tip i kvalitetu izlaznih rezultata tiskanja dostupnih iz vaše aplikacije. Na kvalitetu ispisa također mogu utjecati pogonitelji pisača i pisači korisnika. Ovaj dio bavi se omogućavanjem ispisa iz aplikacije Visual Basica. Za informacije o ispisu iz razvojnog okruženja Visual Basica, pogledajte "Ispis informacija u prozoru za trenutan upis naredbi" u 13. poglavlju "Traženje i obrada pogrešaka".

Ispis iz aplikacije

Visual Basic pruža tri tehnike za ispis teksta i grafike.

- Možete proizvesti izlazne rezultate koje želite na formi te zatim ispisati formu korištenjem postupka PrintForm.
- Možete poslati tekst i grafiku pisaču određivanjem standardnog pisača iz članova zbirke Printers.
- Možete poslati tekst i grafiku objektu tipa Printer te ih zatim tiskati korištenjem postupaka NewPage i EndDoc.

Sljedeći odlomci istražuju prednosti i mane ova tri pristupa.

Korištenje postupka PrintForm

Postupak PrintForm šalje sliku određene forme pisaču. Kako bi ispisali informacije iz vaše aplikacije postupkom PrintForm, najprije te informacije morate prikazati na formi te zatim ispisati tu formu postupkom PrintForm. Sintaksa je sljedeća:

[*forma.*]**PrintForm**

Ako izostavite ime forme, Visual Basic ispisuje trenutnu formu. Postupak PrintForm ispisuje cijelu formu, čak i ako dio forme nije vidljiv na ekranu. Međutim, ako forma sadrži grafiku, grafika se ispisuje samo ako je svojstvo AutoRedraw forme postavljeno na True. Kad je ispis dovršeno, postupak PrintForm poziva postupak EndDoc za čišćenje pisača.

Na primjer, možete poslati tekst pisaču tako da ga ispišete na formi i zatim pozovete postupak PrintForm sljedećim izrazima:

```
Print "Ovdje je neki tekst."
PrintForm
```

Postupak PrintForm je daleko najlakši način tiskanja iz vaše aplikacije. Budući da bi mogao slati informacije pisaču u razlučivosti ekrana korisnika (tipično 96 točaka po inču), rezultati mogu biti razočaravajući na pisačima s puno većom razlučivosti (tipično 300 točaka po inču za laserske pisače). Rezultati se mogu razlikovati, ovisno o objektima na vašoj formi.

Za više informacija Pogledajte "Postupak PrintForm" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Korištenje zbirke Printers

Zbirka Printers je objekt koji sadrži sve pisače dostupne u operativnom sustavu. Popis pisača isti je kao i onaj dostupan u dijaloškom okviru Print Setup ili kontrolnom panelu Windowsa. Svaki pisač u zbirci ima jedinstven indeks za identifikaciju. Počevši od 0, na svaki pisač u zbirci može se upućivati njegovim brojem.

Neovisno o tome koji postupak tiskanja upotrebljavate, svi izlazni rezultati za ispis iz aplikacije Visual Basica usmjereni su objektu tipa Printer, kojeg početno predstavlja podrazumijevani pisač određen u kontrolnom panelu Windowsa. Unatoč tome, kao podrazumijevani pisač možete odrediti bilo koji pisač u zbirci Printers.

Kako bi odabrali pisač iz zbirke, upotrijebite sljedeću sintaksu:

Set Printer = **Printers**(*n*)

Sljedeći izrazi ispisuju imena uređaja svih pisača na operativnom sustavu u prozor za trenutan upis naredbi:

```
Private Sub Command1_Click()
Dim x As Printer
  For Each x In Printers
    Debug.Print x.DeviceName
  Next
End Sub
```

Napomena Ne možete stvoriti nove primjere objekta Printer u kodu, i ne možete izravno dodati ili maknuti pisače iz zbirke Printers. Kako bi dodali ili maknuli pisače na svom sustavu, upotrijebite kontrolni panel Windowsa.

Korištenje objekta Printer

Objekt tipa Printer je prostor za crtanje neovisan o uređaju koji podržava postupke Print, PSet, Line, PaintPicture i Circle za stvaranje teksta i grafike. Ove postupke možete koristiti na objektu Printer baš kao što to možete na formi ili okviru za sliku. Objekt Printer također ima sva svojstva pisma opisana ranije u ovom poglavlju. Kad završite postavljanje informacija u objekt Printer, upotrijebite postupak EndDoc za slanje izlaznih rezultata pisaču. Kad se aplikacije zatvaraju, automatski upotrebljavaju postupak EndDoc za slanje svih informacija koje su na čekanju u objektu Printer.

Objekt Printer pruža najbolju kvalitetu ispisa za niz raznih pisača jer Windowsi prevode tekst i grafiku iz prostora za crtanje neovisnog o uređaju objekta Printer tako da najbolje odgovaraju razlučivosti i sposobnostima pisača. Možete također ispisati dokumente s više stranica korištenjem postupka NewPage objekta Printer.

Glavna mana korištenja objekta Printer je količina koda potrebna za dobivanje najboljih rezultata. Ispis bitmapiranih slika iz objekta Printer također može dulje potrajati i zbog toga usporiti izvođenje aplikacije.

Ispis objektom tipa Printer

Postoji nekoliko načina postavljanja teksta i grafike u objekt Printer. Kako bi ispisivali s objektom Printer, učinite nešto od sljedećeg:

- Dodijelite određen član zbirke Printers objektu Printer ako želite ispisati na pisač koji nije podrazumijevani pisač.
- Postavite tekst i grafiku u objekt Printer.
- Ispišite sadržaj objekta Printer postupkom NewPage ili EndDoc.

Svojstva objekta tipa Printer

Svojstva objekta Printer početno se poklapaju sa onima podrazumijevanog pisača postavljenim u kontrolnom panelu Windowsa. Tijekom izvođenja aplikacije, možete postaviti bilo koje od svojstava objekta Printer, koja uključuju: PaperSize, Height, Width, Orientation, ColorMode, Duplex, TrackDefault, Zoom, DriverName, DeviceName, Port, Copies, PaperBin i PrintQuality. Za više detalja i sintaksi za ove postupke, pogledajte priručnik *Microsoft Visual Basic 6.0 Language Reference*.

Ako je svojstvo TrackDefault postavljeno na True i promijenite podrazumijevan pisač u kontrolnom panelu Windowsa, vrijednosti svojstava objekta Printer će odražavati svojstva novog podrazumijevanog pisača.

Neka svojstva ne možete promijeniti na sredini stranice jednom kad je svojstvo postavljeno. Promjene tih svojstava će utjecati samo na sljedeće stranice. Sljedeći izrazi pokazuju kako možete ispisati svaku stranicu korištenjem različite kvalitete ispisa:

```
For brojstranice = 1 To 4
  Printer.PrintQuality = -1 * brojstranice
  Printer.Print “Kvaliteta ove stranice je”; brojstranice
  Printer.NewPage
Next
```

Vrijednosti kvalitete ispisa kreću se u opsegu od -4 do -1, ili mogu biti pozitivan cijeli broj koji odgovara razlučivosti tiskanja u točkama po inču (dots per inch, DPI). Na primjer, sljedeći kod će postaviti razlučivost pisača na 300 DPI:

```
  Printer.PrintQuality = 300
```

Za više informacija Za informacije o svojstvima objekta Printer, pogledajte odgovarajuće svojstvo u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

> **Napomena** Učinci vrijednosti svojstava objekta Printer ovise o pogoniteljima pribavljenim od proizvođača pisača. Neke postavke svojstava možda neće imati učinka, ili će nekoliko različitih postavki svojstava imati isti učinak. Postavke izvan prihvatljivog raspona mogu, ali ne moraju uzrokovati grešku. Za više informacija o određenim pogoniteljima, pogledajte dokumentaciju proizvođača.

Svojstva mjerila

Objekt tipa Printer ima ova svojstva mjerila:

- ScaleMode
- ScaleLeft i ScaleTop
- ScaleWidth i ScaleHeight
- Zoom

Svojstva ScaleLeft i ScaleTop određuju x i y koordinate, gornjeg lijevog kuta ispisane stranice. Promjenom vrijednosti svojstava ScaleLeft i ScaleTop, možete stvoriti lijeve i gornje margine na ispisanoj stranici. Na primjer, možete upotrijebiti svojstva ScaleLeft i ScaleTop za centriranje tiskane forme (PFrm) na stranici korištenjem ovih izraza:

```
Printer.ScaleLeft = -((Printer.Width - PFrm.Width) / 2)
Printer.ScaleTop = -((Printer.Height - PFrm.Height) / 2)
```

Većina pisača podržava svojstvo Zoom. Ovo svojstvo određuje postotak veličine u kojoj će izlazni rezultat biti ispisan. Podrazumijevana vrijednost svojstva Zoom je 100, što naznačuje da će izlazni rezultat biti tiskan sa 100 posto svoje veličine (u stvarnoj veličini). Svojstvo Zoom možete upotrijebiti za stvaranje stranice koju tiskate manjom ili većom od stvarne veličine papira. Na primjer, postavljanjem svojstva Zoom na 50 vaša tiskana stranica će biti velika pola širine i pola duljine stranice papira. Sljedeća sintaksa postavlja svojstvo Zoom na polovicu veličine podrazumijevanog objekta Printer:

```
Printer.Zoom = 50
```

Smještanje teksta i grafike

Možete postaviti svojstva CurrentX i CurrentY za objekt Printer, baš kao što to možete za forme ili okvire za sliku. Kod objekta Printer, ova svojstva određuju gdje će biti smješten izlazni rezultat na trenutnoj stranici. Sljedeći izrazi postavljaju koordinate crtanja u gornji lijevi kut trenutne stranice:

```
Printer.CurrentX = 0
Printer.CurrentY = 0
```

Možete također upotrijebiti postupke TextHeight i TextWidth za smještanje teksta u objekt Printer. Za više informacija o korištenju ovih postupaka teksta, pogledajte “ Prikaz izlaza naredbe Print na određenom položaju”, ranije u ovom poglavlju.

Ispis formi objektom Printer

Možda će vam trebati da vaša aplikacija ispisuje jednu ili više formi zajedno s informacijama na tim formama, posebno ako oblikovanje forme odgovara ispisanom dokumentu kao račun ili karta za zapis radnog vremena. Kao najlakši način da to napravite, upotrijebite postupak PrintForm. Za najbolju kvalitetu na laserskom pisaču upotrijebite postupak Print i grafičke postupke s objektom Print. Imajte na umu da korištenje objekta Print traži više planiranja, zato što morate osvježiti formu u objektu Printer prije ispisa.

Osvježavanje forme u objektu Printer može također zahtijevati osvježavanje:

- Obrisa forme, uključujući naslovnu traku i trake s izbornicima.
- Kontrola i njihovih sadržaja, uključujući tekst i grafiku.
- Izlaznih rezultata grafičkih postupaka pridruženih izravno na formu, uključujući postupak Print.

Opseg u kojem morate osvježiti te elemente u objektu Printer ovisi o vašoj aplikaciji te koliko forme trebate tiskati.

Osvježavanje teksta i grafike na formi

Kad stvarate tekst i grafiku na formi korištenjem postupaka Print, Line, Circle, PaintPicture ili PSet, možete također trebati pojavljivanje kopije tih izlaznih rezultata u objektu Printer. Najlakši način ostvarivanja toga je pisanje potprograma neovisnog o uređaju koji će osvježiti tekst i grafiku.

Na primjer, sljedeći potprogram koristi postupak PaintPicture za ispis forme ili svojstva Picture objekta u bilo koji objekt izlaznog rezultata, kao pisač ili druga forma:

```
Sub IspisSvugdje(Izv As Object, Odr As Object)
  Odr.PaintPicture Izv.Picture, Odr.Width / 2, Odr.Height / 2
  If Odr Is Printer Then
    Printer.EndDoc
  End If
End Sub
```

Zatim možete pozvati ovaj potprogram i proslijediti mu objekte izvora i odredišta:

```
IspisSvugdje MojaForma, Printer
IspisSvugdje MojaForma, TvojaForma
```

Za više informacija Za više informacija, pogledajte odlomke “Postupak Print” “Postupak Line” “Postupak Circle” “Postupak PSet” ili “Postupak PaintPicture” u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Ispis kontrola na formi

Objekt Printer može prihvatiti izlazne rezultate postupka Print i grafičkih postupaka (kao postupak Line ili PSet). Ipak, ne možete postaviti kontrole izravno na objekt Printer. Ako vaša aplikacija treba tiskati kontrole, morate napisati potprograme koje će u objektu Printer ponovno iscrtati svaki tip kontrole koji koristite, ili morate upotrijebiti postupak PrintForm.

Ispis sadržaja objekta Printer

Jednom kad ste postavili tekst i grafiku u objekt Printer, upotrijebite postupak EndDoc za ispis sadržaja. Postupak EndDoc upućuje stranicu dalje i šalje sve izlazne rezultate na čekanju *spooleru*. *Spooler* presreće posao ispisa na njegovom putu do pisača i šalje ga disku ili memoriji, gdje se posao ispisa drži sve dok pisač ne bude spreman za njega. Na primjer:

```
Printer.Print “Ovo je prva od dvije linije teksta.”
Printer.Print “Ovo je druga od dvije linije teksta.”
Printer.EndDoc
```

> **Napomena** Visual Basic automatski poziva postupak EndDoc ako vaša aplikacija završi a da ga izričito ne pozove.

Stvaranje dokumenata s više stranica

Kad ispisujete duže dokumente, možete odrediti u kodu gdje želite početak nove stranice korištenjem postupka NewPage. Na primjer:

```
Printer.Print “Ovo je 1. stranica.”
Printer.NewPage
Printer.Print “Ovo je 2. stranica.”
Printer.EndDoc
```

Opoziv ispisa

Možete završiti trenutan posao ispisa upotrebom postupka KillDoc. Na primjer, korisniku možete dijaloškim okvirom postaviti upit da odluči želi li dokument ispisati ili završiti ispis:

```
Sub TiskajIliNe()
  Printer.Print “Ovo je prva linija za _
  pokazivanje postupka KillDoc”
  Printer.Print “Ovo je druga linija za _
  pokazivanje postupka KillDoc”
  Printer.Print “Ovo je treća linija za _
  pokazivanje postupka KillDoc”
```

```
  If vbNo = MsgBox("Ispisati ovaj lijep dokument?", _
  vbYesNo) Then
    Printer.KillDoc
  Else
    Printer.EndDoc
  End If
End Sub
```

Ako upravitelj ispisom, Printer Manager, operativnog sustava rukovodi poslom ispisa, postupak KillDoc briše cijeli posao koji ste poslali pisaču. Međutim, ako upravitelj ispisom ne nadzire posao ispisa, prva stranica je možda već poslana pisaču, i postupak KillDoc na nju neće imati utjecaja. Količina podataka poslanih pisaču neznatno se mijenja ovisno o pogoniteljima pisača.

Napomena Ne možete upotrijebiti postupak KillDoc za završavanje posla ispisa koji je pokrenut postupkom PrintForm.

Hvatanje pogrešaka pisača

Tijekom ispisa mogu se pojaviti uhvatljive pogreške tijeka izvođenja. Sljedeća tablica ispisuje neke primjere koji migu biti prijavljeni:

broj pogreške	poruka pogreške
396	**Property cannot be set within, a page (Svojstvo ne može biti postavljeno unutar stranice).** Ova pogreška se pojavljuje kad je isto svojstvo različito postavljeno na istoj stranici.
482	**Printer Error (Pogreška pisača).** Visual Basic prijavljuje ovu pogrešku uvijek kad pogonitelj pisača vrati kod pogreške.
483	**Printer driver does not support the property (Pogonitelj pisača ne podržava svojstvo).** Ova pogreška se pojavljuje kod pokušaja upotrebe svojstva koje trenutan pogonitelj pisača ne podržava.
484	**Printer driver unavailable (Nedostupan pogonitelj pisača).** Ova pogreška se pojavljuje kad u datoteci WIN.INI nedostaju informacije o pisaču ili su nedostatne.

Napomena Pogreške pisača ne moraju se uvijek pojaviti trenutno. Ako izraz prouzroči pogrešku pisača, pogreška možda neće biti oživljena do izvođenja sljedeće linije koja poziva taj pisač.

Za više informacija Za detaljnu raspravu o greškama tijekom izvođenja, pogledajte 13. poglavlje "Traženje i obrada pogrešaka".

13. POGLAVLJE

Traženje i obrada pogrešaka

Bez obzira na to koliko je vješt programski kod, pogreške se mogu pojaviti (i vjerojatno hoće). Idealno, potprogrami Visual Basica uopće ne trebaju kod za rukovanje pogreškama. Na žalost, ponekad su datoteke pogreškom obrisane, na diskovnim pogonima nastaje manjak prostora, ili se pogoni u mreži neočekivano odspajaju. Takve mogućnosti mogu u vašem kodu uzrokovati pogreške tijekom izvođenja. Kako bi rukovali tim pogreškama, vašim potprogramima trebate dodati kod za rukovanje pogreškama.

Ponekad se pogreške također pojavljuju unutar vašeg koda; ovaj tip pogreške se obično označava kao *bug*. Manje pogreške – na primjer, pokazivač koji se ne ponaša prema očekivanju – mogu biti razočaravajuće ili neprikladne. Ozbiljnije pogreške mogu uzrokovati prestanak reagiranja aplikacije na naredbe, što će vjerojatno zahtijevati od korisnika da ponovno pokrene aplikaciju, gubeći sav posao koji nije bio snimljen.

Proces traženja i popravljanja pogrešaka u vašoj aplikaciji poznat je kao *otkrivanje pogrešaka* (*debugging*). Visual Basic pruža nekoliko raznih alata za pomoć pri analiziranju načina rada vaše aplikacije. Ti alati za otkrivanje pogrešaka su posebno korisni u traženju izvora pogrešaka, ali možete također upotrijebiti te alate za pokuse sa promjenama u vašoj aplikaciji ili za učenje kako rade druge aplikacije.

Ovo poglavlje pokazuje kako koristiti alate za otkrivanje pogrešaka koji su uključeni u Visual Basic i objašnjava kako rukovati *pogreškama tijekom izvođenja* – pogreškama koje se pojavljuju dok se izvodi vaš kod i rezultat su pokušaja završavanja neispravnih operacija.

Sadržaj

- Kako rukovati pogreškama
- Oblikovanje rukovatelja pogreškama
- Hijerarhija rukovanja pogreškama
- Ispitivanje rukovanja pogreškama stvaranjem pogrešaka
- Ugrađeno rukovanje pogreškama
- Centralizirano rukovanje pogreškama
- Isključivanje rukovanja pogreškama
- Rukovanje pogreškama sa ActiveX sastavnim dijelovima

- Pristupi otkrivanju pogrešaka
- Izbjegavanje pogrešaka
- Vrijeme izrade, vrijeme izvođenja i mod prekida
- Korištenje prozora za otkrivanje pogrešaka
- Korištenje moda prekida
- Izvođenje odabranih dijelova vaše aplikacije
- Nadziranje stoga poziva
- Ispitivanje podataka i potprograma prozorom za neposredan upis naredbi
- Razmatranje posebnog otkrivanja pogrešaka
- Savjeti za otkrivanje pogrešaka

Primjer aplikacije: Errors.vbp

Većina primjera koda u ovom poglavlju uzeta je iz primjera aplikacije Errors.vbp. Ovaj primjer aplikacije možete pronaći u direktoriju Samples.

Kako rukovati pogreškama

Idealno, potprogrami Visual Basica uopće ne trebaju kod za rukovanje pogreškama. Stvarnost pokazuje da problemi hardvera ili neočekivane akcije korisnika mogu uzrokovati pogreške tijekom izvođenja koje zaustavljaju vaš kod, i obično korisnik ne može napraviti ništa kako bi aplikacija nastavila s izvođenjem. Ostale pogreške možda neće prekinuti kod, ali mogu uzrokovati nepredvidljivo djelovanje koda.

Na primjer, sljedeći potprogram vraća True ako određena datoteka postoji i False ako ne postoji, ali ne sadrži kod za rukovanje greškom:

```
Function DatotekaPostoji(imedatoteke) As Boolean
  DatoPostoji = (Dir(imedatoteke) <> “”)
End Function
```

Funkcija Dir vraća prvu datoteku koja odgovara navedenom imenu datoteke (danom sa ili bez zamjenskih karaktera, imena pogona, ili staze); ona vraća prazan string ako podudarna datoteka nije pronađena.

Ovaj kod izgleda kao da pokriva oba moguća rezultata poziva funkcije Dir. Međutim, ako oznaka pogona određena u argumentu nije valjani pogon, pojavit će se pogreška “Nedostupan pogon” (Device unavailable). Ako je navedeni pogon meki disk, ova funkcija će raditi ispravno samo ako je disketa u disketnom pogonu. Ako nije, Visual Basic predstavlja pogrešku “Disketa nije spremna” (Disk not ready) i zaustavlja izvođenje vašeg koda.

Kako bi izbjegli ovu situaciju, možete upotrijebiti osobine rukovanja pogreškama u Visual Basicu za presretanje pogrešaka i poduzimanje ispravne akcije (presretanje pogreške je također znano kao *hvatanje* (*trapping*) pogreške). Kad se pogreška pojavi, Visual Basic postavlja razna svojstva objekta pogreške, Err, kao što je broj pogreške, opis, i tako dalje. Možete upotrijebiti objekt Err i njegova svojstva u rutini rukovanja greškom tako da vaša aplikacija inteligentno odgovori na situaciju pogreške.

Na primjer, problemi sa uređajima, kao što je neispravan pogon ili prazan disketni pogon, mogu biti obrađeni sljedećim programskim kodom:

```
Function DatotekaPostoji(imedatoteke) As Boolean
  Dim Por As String
  ‘ Uključivanje hvatanja pogreške tako da rukovatelj
  ‘ greškom odgovori ako se otkrije neka pogreška.
  On Error Goto ProvjeraPogreške
    DatoPostoji = (Dir(imedatoteke) <> “”)
    ‘ Izbjegavanje izvođenja rukovatelja pogreškom
    ‘ ako se ne pojavi pogreška.
    Exit Function

ProvjeraPogreške:          ‘ Grananje ovamo ako se pojavi pogreška.
  ‘ Određivanje konstanti za predstavljanje
  ‘ ugrađenih kodova pogreške Visual Basica.
  Const mnGreDisketaNijeSpremna) = 71, _
  mnGreNedostupanPogon = 68
  ‘ vbExclamation, vbOK, vbCancel, vbCritical
  ‘ i vbOKCancel su konstante određene
  ‘ u tipskoj biblioteci VBA.
  If (Err.Number = mnGreDisketaNijeSpremna) Then
    Por = “Stavite disketu u disketni pogon “
    Por = Por & “i zatvorite vrata pogona ako postoje.”
    ‘ Prikaz okvira s porukom sa ikonom upozorenja
    ‘ i s gumbima OK i Cancel.
    If MsgBox(Por, vbExclamation & vbOKCancel) = vbOK Then
          Resume
    Else
          Resume Next
    End If
  ElseIf Err.Number = mnGreNedostupanPogon Then
    Por = “Ovaj pogon ili staza ne postoje: “
    Por = Por & imedatoteke
    MsgBox Por, vbExclamation
    Resume Next
  Else
    Por = “Neočekivana pogreška #” & Str(Err.Number)
    Por = Por & “se pojavila: “ & Err.Description
    ‘ Prikaz okvira s porukom s ikonom znaka Stop
    ‘ i gumbom OK.
```

```
      MsgBox Por, vbCrticial
      Stop
    End If
    Resume
End Function
```

U ovom kodu, svojstvo Number objekta Err sadrži broj pridružen s greškom tijeka izvođenja koja se pojavila; svojstvo Description sadrži kratak opis pogreške.

Kad Visual Basic stvori pogrešku "Disketa nije spremna", taj dio koda predstavlja poruku kazujući korisniku da odabere jedan od dva gumba – OK ili Cancel. Ako korisnik odabere OK, naredba Resume vraća kontrolu naredbi u kojoj se pogreška pojavila i pokušava ponovno izvesti tu naredbu. To uspijeva ako je korisnik ispravio problem; inače, aplikacija se vraća na rukovatelja pogreškama.

Ako korisnik odabere gumb Cancel, izraz Resume Next vraća kontrolu naredbi iza one u kojoj se pojavila pogreška (u ovom slučaju, naredbi Exit Function).

Ako se pojavi pogreška "Nedostupan pogon", ovaj kod predstavlja poruku koja opisuje problem. Izraz Resume Next zatim potiče nastavak izvođenja funkcije u naredbi koja je iza izraza u kojem se pojavila pogreška.

Ako se pojavi neočekivana pogreška, prikazuje se kratak opis pogreške i programski kod se zaustavlja na naredbi Stop.

Aplikacija koju stvorite može ispraviti pogrešku ili zatražiti korisnika da promijeni uvjete koji su uzrokovali pogrešku. Kako bi to napravili, upotrijebite tehnike kao što su one pokazane u prethodnom primjeru. Idući odlomak detaljno raspravlja te tehnike.

Za više informacija Pogledajte "Smjernice za rukovanje složenim pogreškama" u odlomku "Hijerarhija rukovanja pogreškama", kasnije u ovom poglavlju, za objašnjenje kako koristiti naredbu Stop.

Oblikovanje rukovatelja pogreškama

Rukovatelj pogreškama (*error handler*) je rutina za hvatanje i odgovaranje na pogreške u vašoj aplikaciji. Trebat ćete dodati rukovatelje pogreškama svakom potprogramu gdje očekujete mogućnost pogreške (trebate pretpostaviti da svaka naredba Basica može proizvesti pogrešku osim ako izričito znate da neće). Postupak oblikovanja rukovatelja pogreškama sadržava tri koraka:

1. Postavite, ili *omogućite*, hvatanje pogreške kazujući aplikaciji gdje da se grana (koju rutinu rukovanja greškom treba izvesti) kad se pojavi pogreška.

   Izraz On Error omogućuje hvatanje i usmjeruje aplikaciju prema naznačenoj oznaci početka rutine rukovanja greškom.

   U primjeru aplikacije Errors.vbp, funkcija FileExists sadrži rutinu rukovanja greškom imena CheckError.

2. Napišite rutinu rukovanja pogreškom koja odgovara na sve pogreške koje možete predvidjeti. Ako se kontrola zaista grana u hvatanje na nekoj točki, za hvatanje se kaže da je *aktivno*.

   Rutina `ProvjeraPogreške` (CheckError) rukuje pogreškom koristeći izraz If...Then...Else koji odgovara na vrijednost u svojstvu Number objekta Err, što je brojčani kod koji odgovara grešci Visual Basica. U prošlom primjeru, ako se stvori pogreška "Disketa nije spremna", poruka traži od korisnika da ubaci disketu i zatvori vrata pogona. Drugačija poruka se prikazuje ako se pojavi pogreška "Nedostupan pogon". Ako se stvori bilo koja druga pogreška, prikazuje se odgovarajući opis i aplikacija se zaustavlja.

3. Izađite iz rutine rukovanja pogreškom.

   U slučaju pogreške "Disketa nije spremna", naredba Resume grana kod natrag u izraz gdje se pogreška pojavila. Visual Basic zatim pokušava ponovno izvesti taj izraz. Ako situacija nije promijenjena, pojavljuje se još jedna pogreška i izvođenje se grana natrag u rutinu rukovanja pogreškom.

   U slučaju pogreške "Nedostupan pogon", izraz Resume Next grana kod natrag na izraz iza onog u kojem se pojavila pogreška.

Detalji o izvođenju ovih koraka pruženi su u ostatku ove teme. Pozovite se na primjer funkcije DatotekaPostoji dok čitate te korake.

Postavljanje hvatanja pogreške

Hvatanje pogreške je omogućeno kad Visual Basic izvede izraz On Error, koji određuje rukovatelja pogreškom. Takva zamka za pogreške ostaje omogućena sve dok je aktivan potprogram koji ju sadrži – znači, sve dok se za taj potprogram ne izvedu naredbe Exit Sub, Exit Function, Exit Property, End Sub, End Function ili End Property. Iako istovremeno može biti omogućena samo jedna zamka za pogreške u svakom danom potprogramu, možete stvoriti nekoliko alternativnih zamki za pogreške i omogućiti različite zamke u različito vrijeme. Također možete onemogućiti zamku za pogreške korištenjem posebnog stanja izraza On Error – On Error GoTo 0.

Kako bi postavili zamku za pogreške koja skače na rutinu rukovanja pogreškama, upotrijebite izraz On Error GoTo *linija*, gdje *linija* naznačuje oznaku koja identificira kod rukovanja pogreškom. U primjeru funkcije DatotekaPostoji, oznaka je ProvjeraGreške (iako je dvotočka dio oznake, ne koristi se u izrazu On Error GoTo *linija*).

Za više informacija Za više informacija o onemogućavanju rukovanja pogreškama, pogledajte "Isključivanje rukovanja pogreškama", kasnije u ovom poglavlju.

Pisanje rutine rukovanja pogreškom

Prvi korak pisanja rutine za rukovanje pogreškom je dodavanje oznake linije koja obilježava početak rutine rukovanja pogreškom. Oznaka linije treba imati opisno ime i završavati dvotočkom. Uobičajeno pravilo je postavljanje rutine rukovanja pogreškom na kraj potprograma s naredbom Exit Sub, Exit Function ili Exit Property neposredno prije oznake linije. To potprogramu omogućuje izbjegavanje izvođenja koda za rukovanje pogreškom ako se pogreška ne pojavi.

Glavni dio rutine za rukovanje pogreškom sadrži programski kod koji zapravo rukuje pogreškom, obično u obliku izraza Case ili If...Then...Else. Trebate odrediti koje će pogreške vjerojatno pojaviti i pružiti smjer akcije za svaku, na primjer, tražeći od korisnika da ubaci disketu u slučaju pogreške "Disketa nije spremna". Uvijek mora biti pružena mogućnost rukovanja svim neočekivanim pogreškama korištenjem uvjeta Else ili Case Else – u slučaju primjera funkcije DatotekaPostoji, ta mogućnost upozorava korisnika i završava rad aplikacije.

Svojstvo Number objekta Err sadrži brojčani kod koji predstavlja posljednju pogrešku tijekom izvođenja. Korištenjem objekta Err u kombinaciji s izrazom Select Case ili If...Then...Else, možete poduzeti određene akcije za svaku pogrešku koja se pojavi.

> **Napomena** String sadržan u svojstvu Description objekta Err objašnjava pogrešku pridruženu trenutnom broju pogreške. Točno izražavanje opisa može se razlikovati među raznim verzijama Microsoft Visual Basica. Zbog toga, radije upotrijebite svojstvo Err.Number nego svojstvo Err.Description, za prepoznavanje specifične pogreške koja se pojavila.

Izlaz iz rutine rukovanja pogreškom

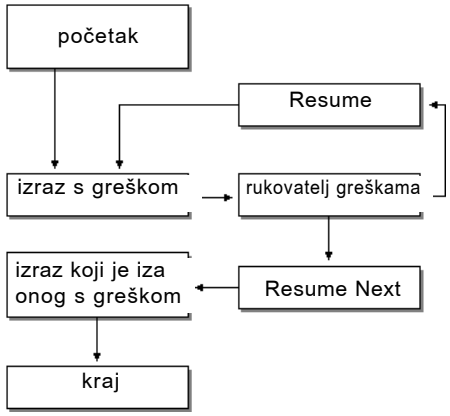
Primjer funkcije DatotekaPostoji koristi naredbu Resume unutar rukovatelja pogreškom za ponovno izvođenje izraza koji je izvorno prouzročio pogrešku, te koristi izraz Resume Next za vraćanje izvođenja na izraz iza onog u kojem se pogreška pojavila. Postoje drugi načini izlaza iz rutine rukovanja pogreškom. Ovisno o okolnostima, možete to napraviti koristeći bilo koji od izraza prikazanih u sljedećoj tablici.

izraz	opis
Resume [0]	Izvođenje aplikacije nastavlja se izrazom koji je prouzročio pogrešku ili s posljednjim izvedenim pozivom izvan potprograma koji sadrži rutinu rukovanja pogreškom. Upotrijebite ovaj izraz za ponavljanje operacije nakon ispravljanja stanja koje je uzrokovalo pogrešku.
Resume Next	Nastavljanje izvođenja aplikacije s izrazom odmah iza onog koji je uzrokovao pogrešku. Ako se pogreška pojavila izvan potprograma koji sadrži rukovatelja pogreškom, izvođenje se nastavlja s izrazom odmah iza poziva potprograma u kojem se pojavila pogreška, ako pozivan potprogram nema omogućenog rukovatelja pogreškom.
Resume *linija*	Nastavlja izvođenje aplikacije na oznaci određenoj argumentom *linija*, koji je oznaka linije (ili broj linije različit od nule) koja mora biti u istom potprogramu gdje je rukovatelj pogreškom.
Err.Raise Number:=*broj*	Pokreće pogrešku tijekom izvođenja. Kad se izvede ova naredba unutar rutine rukovanja pogreškom, Visual Basic u listi poziva traži drugu rutinu rukovanja pogreškom (*lista poziva* je niz potprograma pozvanih da dođu na trenutnu točku izvođenja. Pogledajte odlomak "Hijerarhija rukovanja pogreškama", kasnije u ovom poglavlju).

Razlika između izraza Resume i Resume Next

Razlika između izraza Resume i Resume Next prikazana je na slici 13.1.

Slika 13.1 Tijek aplikacije s izrazima Resume i Resume Next



Općenito, upotrebljavat ćete izraz Resume uvijek kad rukovatelj pogreškom može ispraviti pogrešku, a izraz Resume Next kad rukovatelj pogreškom ne može ispraviti pogrešku. Možete napisati kod rukovatelja pogreškom tako da postojanje pogreške tijekom izvođenja nikad ne bude otkriveno korisniku ili da nema prikazivanja poruka o grešci i omogućavanja korisniku da unese ispravke.

Na primjer, potprogram Function u sljedećem primjeru programskog koda koristi rukovanje pogreškom za izvođenje "sigurnog" dijeljenja njegovih argumenata bez otkrivanja pogrešaka koje se mogu pojaviti. pogreške koje se mogu pojaviti kod izvođenja dijeljenja su:

pogreška	uzrok
"Dijeljenje s nulom" (Division by zero)	Brojnik je različit od nule, ali je nazivnik jednak nuli.
"Prekoračenje" (Overflow)	I brojnik i nazivnik su jednaki nuli (tijekom dijeljenja s pomičnim zarezom).
"Neispravan poziv potprograma" (Illegal procedure call)	Brojnik ili nazivnik nisu numeričke vrijednosti (ili se ne mogu smatrati numeričkom vrijednosti).

U sva tri slučaja, sljedeći potprogram tipa Function hvata te pogreške i vraća vrijednost Null:

```
Function Dijeljenje(brojnik, nazivnik) As Variant
  Dim Por As String
  Const mnPogDijeSaNulom = 11, mnPogPrekora = 6
  Const mnPogLošPoziv = 5
  On Error Goto MatRukovatelj
       Dijeli = brojnik / nazivnik
```

```
    Exit Function
MatRukovatelj:
  If Err.Number = mnPogDijeSNulom Or _
  Err.Number = mnPogPrekora Or _
  Err.Number = mnPogLošPoziv Then
    Dijeli = Null          ‘ Ako je pogreška bila Dijeljenje s nulom,
                           ‘ Prekoračenje ili Neispravan poziv
                           ‘ potprograma, vraćanje vrijednosti Null.
  Else
    ‘ Prikaz poruke neočekivane pogreške.
    Por = “Neočekivana pogreška “ & Err.Number
    Por = Por & “: “ & Err.Description
    MsgBox Por, vbExclamation
  End If                   ‘ U svim slučajevima, izraz Resume Next
                           ‘ nastavlja izvođenje na
  Resume Next              ‘ izrazu Exit Function.
End Function
```

Nastavljanje izvođenja na određenoj liniji

Izraz Resume Next može također biti upotrijebljen tamo gdje se pogreška pojavljuje unutar petlje, i želite ponovno pokrenuti operaciju. Ili, možete upotrijebiti izraz Resume *linija*, koji vraća kontrolu na određenu oznaku linije.

Sljedeći primjer prikazuje upotrebu izraza Resume *linija*. Ova funkcija, kao varijacija primjera funkcije DatotekaPostoji prikazane ranije, omogućuje korisniku da unese označavanje vrste datoteke koju će funkcija vratiti ako datoteka postoji.

```
Function ProvjeraDatoteke As String
  Const mnPogLošeImeDato = 52, _
  mnPogOtvoVrataDiska = 71
  Const mnPogNedostupanPogon = 68, _
  mnPogPogrešnoImeDato = 64
  Dim strUpit As String, strPor As String, _
  strOdreDato As String
  strUpit = “Unesite označavanje datoteke za provjeru:”
Po~etakOvdje:
  strOdreDato = “*.*”      ‘ Početak s
                           ‘ podrazumijevanom postavkom.
  strPor = strPor & vbCrLf & strUpit
  ‘ Neka korisnik promijeni postavku.
  strOdreDato = InputBox(strPor, “Traženje datoteke”, _
  strOdreDato, 100, 100)
  ‘ Izlaz ako korisnik obriše podrazumijevano.
  If strOdreDato = “” Then Exit Function
  In Error GoTo Rukovatelj
    ProvjeraDato = Dir(strOdreDato)
```

```
    Exit Function
Rukovatelj:
  Select Case Err.Number          ‘ Analiza koda pogreške
                                  ‘ i ispis poruke.
    Case mnPogPogrešnoImeDato, mnPogLošeImeDato
      strPor = “Vaše označavanje datoteke je “
      strPor = strPor & “neispravno; probajte drugo.”
    Case mnPogOtvoVrataDiska
      strPor = “Zatvorite vrata pogona i “
      strPor = strPor & “pokušajte ponovno.”
    Case mnPogNedostupanPogon
      strPor = “Pogon koji ste odredili nije “
      strPor = strPor & “pronađen. Pokušajte ponovno.”
    Case Else
      Dim intBrojPog As Integer
      intBrojPog = Err.Number
      Err.Clear                        ‘ Čišćenje objekta Err.
      Err.Raise Number:=intBrojGre     ‘ Obnavljanje pogreške.
  End Select
  Resume PočetakOvdje             ‘ Ovo skače natrag na oznaku
                                  ‘ PočetakOvdje tako da korisnik
                                  ‘ može probati drugo ime datoteke.
End Function
```

Ako je pronađena datoteka koja se podudara s zadanim određivanjem, funkcija vraća ime datoteke. Ako nije pronađena podudarajuća datoteka, funkcija vraća prazan string. Ako se pojavi jedna od neočekivanih pogrešaka, poruka se dodjeljuje varijabli strPor i izvođenje skače natrag na oznaku PočetakOvdje. Na taj način korisniku se daje još jedna prilika da unese valjanu stazu i određivanje datoteke.

Ako je pogreška neočekivana, dio Case Else obnavlja pogrešku tako da idući rukovatelj pogreškama u listi poziva može uhvatiti pogrešku. To je potrebno jer da pogreška nije obnovljena, programski kod bi nastavio izvođenje na liniji Resume PočetakOvdje. Obnavljanjem pogreške vi zapravo uzrokujete ponovno pojavljivanje pogreške; nova pogreška će biti uhvaćena na idućoj razini sloga poziva.

Za više informacija Za više detalja, pogledajte “Hijerarhija rukovanja pogreškama”, kasnije u ovom poglavlju.

Napomena Iako je korištenje *linije* Resume ispravan način pisanja koda, umnažanje skokova na oznake linija može stvoriti programski kod teškim za razumijevanje i traženje pogrešaka.

Hijerarhija rukovanja pogreškama

Omogućen rukovatelj pogreškama je onaj koji je aktiviran izvođenjem izraza On Error i još nije isključen – ili izrazom On Error GoTo 0 ili izlaskom iz potprograma u kojem je omogućen. *Aktivni* rukovatelj pogreškama je onaj u kojem se trenutno obavlja izvođenje. Da bi bio aktivan, rukovatelj pogreškama prvo mora biti omogućen, ali svi omogućeni rukovatelji pogreškama nisu aktivni. Na primjer, nakon naredbe Resume, rukovatelj više nije aktivan, ali je i dalje omogućen.

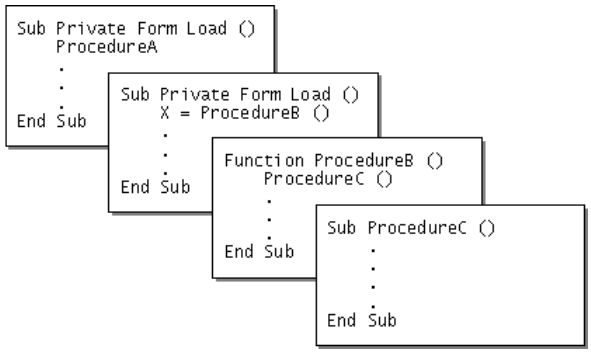
Kad se pojavi pogreška unutar potprograma koji nema omogućenu rutinu rukovanja pogreškama, ili unutar aktivne rutine rukovanja pogreškama, Visual Basic u listi poziva traži drugu omogućenu rutinu rukovanja pogreškama. Lista poziva je slijed poziva koji vode potprogramima koji se trenutno izvode; prikazuje se u dijaloškom okviru Call Stack. Dijaloški okvir Call Stack možete prikazati samo u modu prekida (kad zaustavite izvođenje vaše aplikacije), odabirom stavke Call Stack iz izbornika View ili pritiskom CTRL + L.

Pretraživanje liste poziva

Pretpostavimo pojavljivanje sljedeće liste poziva, kakva je prikazana na slici 13.2:

1. Potprogram događaja poziva potprogram Procedure A.
2. Potprogram Procedure, a poziva potprogram Procedure B.
3. Potprogram Procedure B poziva potprogram Procedure C.

Slika 13.2 Slijed poziva

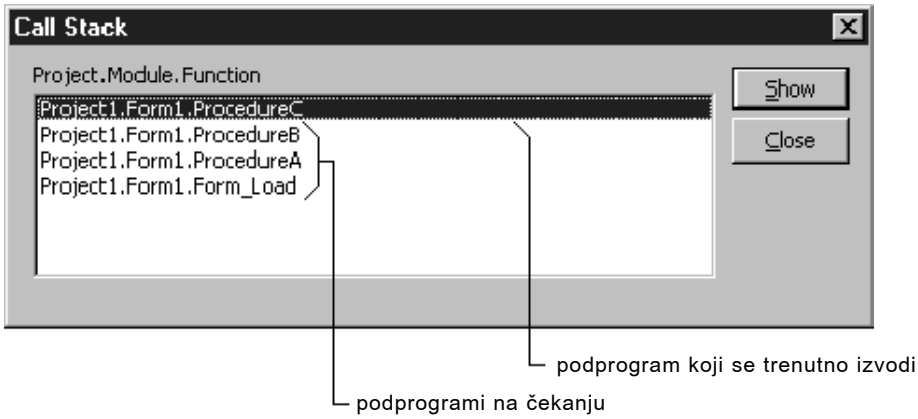


Dok se izvodi potprogram Procedure C, ostali potprogrami su na čekanju, kao što je prikazano u listi poziva u dijaloškom okviru Call Stack.

Za više informacija Za više informacija, pogledajte "Nadziranje stoga poziva", kasnije u ovom poglavlju.

Slika 13.3 prikazuje listu poziva prikazanu u dijaloškom okviru Call Stack.

Slika 13.3 Lista poziva kad su potprogrami na čekanju



Ako se pojavi pogreška u potprogramu Procedure C i taj potprogram nema omogućenog rukovatelja pogreškama, Visual Basic traži unazad kroz potprograme na čekanju u listi poziva – prvo potprogram Procedure B, zatim potprogram Procedure A, te početni potprogram događaja (ali ne dalje) – i izvodi prvog omogućenog rukovatelja pogreškama kojeg pronađe. Ako nigdje u listi poziva ne otkrije omogućenog rukovatelja pogreškama, predstavlja podrazumijevanu poruku neočekivane pogreške i zaustavlja izvođenje.

Ako Visual Basic pronađe rutinu omogućenog rukovatelja pogreškama, izvođenje se nastavlja u toj rutini kao da je pogreška otkrivena u istom potprogramu koji sadržava rukovatelja pogreškama. Ako se u rutini rukovatelja pogreškama izvedu izrazi Resume ili Resume Next, izvođenje se nastavlja kako je pokazano u sljedećoj tablici.

izraz	rezultat
Resume	Ponovno se izvodi poziv potprograma koji je Visual Basic upravo pretražio. U prethodnom primjeru liste poziva, ako potprogram Procedure, a ima omogućenog rukovatelja pogreškom koji uključuje naredbu Resume, Visual Basic ponovno izvodi poziv potprograma Procedure B.
Resume Next	Izvođenje se vraća na naredbu iza zadnje naredbe izvedene u tom potprogramu. To je naredba koja se nalazi iza poziva potprograma kojeg je Visual Basic upravo pretražio. U prethodnom primjeru liste poziva, ako potprogram Procedure, a ima omogućenog rukovatelja pogreškom koji uključuje naredbu Resume Next, izvođenje se vraća na naredbu iza poziva potprograma Procedure B.

Uočite da naredba izvedena u potprogramu *gdje se pronađena rutina rukovanja pogreškom*, nije nužno u potprogramu gdje se pogreška pojavila. Ako to ne uzmete u obzir, vaš programski kod može se izvoditi na načine koje niste željeli. Kako bi kod učinili lakšim za traženje pogrešaka, možete jednostavno otići u mod prekida uvijek kad se pojavi pogreška, kao što je objašnjeno u odlomku "Isključivanje rukovanja pogreškama", kasnije u ovom poglavlju.

Ako opseg pogrešaka rukovatelja pogreškama ne uključuje pogrešku koja se stvarno pojavila, može se pojaviti neočekivana pogreška unutar potprograma s omogućenim rukovateljem pogreškom. U takvom slučaju, potprogram se može izvoditi beskonačno, posebno ako ru'kovatelj pogreškom izvodi naredbu Resume. Kako bi spriječili takve situacije, upotrijebite postupak Raise objekta Err unutar izraza Case Else u rukovatelju. To zapravo stvara pogrešku unutar rukovatelja pogreškom, prisiljavajući Visual Basic da u listi poziva potraži rukovatelja koji se može nositi s pogreškom.

U primjeru potprograma VerifyFile, iz aplikacije Errors.vbp, broj koji je izvorno sadržan u svojstvu Err.Number dodijeljen je varijabli, intErrNum, koja je zatim proslijeđena kao argument postupku Raise objekta Err u izrazu Case Else, tako stvarajući greš'ku. Kad se takva pogreška pojavi unutar aktivnog rukovatelja pogreškom, počinje potraga prema nazad kroz listu poziva.

Dodjela pogrešaka različitim rukovateljima

Učinak traženja prema nazad kroz listu poziva je teško predvidjeti, jer ovisi o tome jesu li izvedene naredbe Resume ili Resume Next u rukovatelju koji uspješno obrađuje pogrešku. Naredba Resume vraća nadzor posljednjem izvedenom pozivu izvan potprograma koji sadrži rukovatelja pogreškom. Izraz Resume Next vraća nadzor svakoj naredbi koja se nalazi neposredno iza posljednjeg izvedenog poziva izvan potprograma koji sadrži rukovatelja pogreškom.

Na primjer, u listi poziva prikazanoj na slici 13.3, ako potprogram Procedure, a ima omogućenog rukovatelja pogreškom, a potprogrami Procedure B i Procedure C nemaju, pogreška koja se pojavi u potprogramu Procedure C biti će obrađena rukovateljem pogreškom potprograma Procedure A. Ako taj rukovatelj pogreškom koristi naredbu Resume, na izlazu, aplikacija nastavlja s pozivom potprograma Procedure B. Međutim, ako rukovatelj pogreškom potprograma Procedure, a koristi izraz Resume Next, na izlazu, aplikacija će nastaviti s naredbom u potprogramu Procedure, a koja se nalazi iza poziva potprograma Procedure B. U oba slučaja rukovatelj pogreškom ne vraća se izravno na potprogram ili izraz gdje se pogreška izvorno pojavila.

Smjernice za rukovanje složenim pogreškama

Kad pišete velike aplikacije Visual Basica koje koriste višestruke module, programski kod rukovanja pogreškama može postati sasvim složen. Imajte na umu ove smjernice:

- Dok ispravljate pogreške u vašem kodu, upotrijebljavajte postupak Raise objekta Err za obnavljanje pogreške u svim rukovateljima pogreškama za slučajeve kad u rukovatelju ne postoji kod za postupanje s određenom pogreškom. To vašoj aplikaciji omogućuje da pokuša ispraviti pogrešku u drugim rutinama rukovanja pogreškama duž liste poziva. Na taj način Visual Basic će sigurno prikazati poruku pogreške ako se pojavi pogreška koju vaš kod ne može obraditi. Kad ispitujete vaš kod, ova tehnika vam pomaže pri otkrivanju pogrešaka koje niste prikladno obradili. Međutim, u samostalnoj izvršnoj datoteci, trebate biti oprezni: ako izvedete postupak Raise, a pogrešku ne uhvati niti jedan drugi postupak, vaša aplikacija će trenutno prestati raditi, bez pojavljivanja bilo kojih događaja QueryUnload ili Unload.
- Upotrijebite postupak Clear ako izričito želite očistiti objekt Err nakon rukovanja pogreškom. To je nužno kod upotrebe ugrađenog rukovatelja pogreškom s izrazom On Error Resume Next. Visual Basic automatski poziva postupak Clear uvijek kad izvodi svaki oblik izraza Resume, Exit Sub, Exit Function, Exit Property ili svaki izraz On Error.
- Ako ne želite da drugi potprogram u listi poziva uhvati pogrešku, upotrijebite naredbu Stop za prisiljavanje završavanja rada vaše aplikacije. Korištenje naredbe Stop dopušta vam pregledavanje sklopa pogreške dok pročišćavate svoj kod u razvojnom okruženju.

  Oprez Obavezno uklonite sve naredbe Stop prije stvaranja izvršne datoteke. Ako samostalna aplikacija Visual Basica (.exe) otkrije naredbu Stop, obrađuje ju kao naredbu End i trenutno završava izvođenje, bez pojavljivanja bilo kojih događaja QueryUnload ili Unload.
- Napišite potprogram za rukovanje pogreškama otporan na propuste kojeg mogu pozivati svi vaši rukovatelji pogreškama kao posljednje utočište za pogreške koje ne mogu obraditi. Taj potprogram otporan na pogreške može izvesti redovan završetak vaše aplikacije izbacivanjem formi i spremanjem podataka.

Za više informacija Pogledajte odlomke “Ispitivanje rukovanja pogreškama stvaranjem pogrešaka”, “Ugrađeno rukovanje pogreškama” i “Vrijeme izrade, vrijeme izvođenja i mod prekida”, kasnije u ovom poglavlju.

Ispitivanje rukovanja pogreškama stvaranjem pogrešaka

Oponašanje pogrešaka je korisno kad ispitujete svoje aplikacije, ili kad želite postupati s određenim stanjem kao da je jednako grešci tijekom izvođenja u Visual Basicu. Na primjer, možda pišete modul koji upotrijebljava objekt određen u vanjskoj aplikaciji, i želite da ostatak vaše aplikacije obradi pogreške koje je vratio objekt kao stvarne pogreške Visual Basica.

Kako bi ispitali postojanje svih mogućih pogrešaka, trebat ćete stvoriti neke od pogrešaka u vašem kodu. Pogrešku u vašem kodu možete stvoriti postupkom Raise:

*objekt***.Raise** *listaargumenata*

Argument *objekt* je obično objekt Err, opće određen objekt pogreške Visual Basica. Argument *listaargumenata* je lista imenovanih argumenata koji mogu biti proslijeđeni s postupkom. Postupak VerifyFile u primjeru aplikacije Errors.vbp koristi sljedeći kod za obnavljanje trenutne pogreške u rukovatelju pogreškom:

```
Err.Raise Number:=intErrNum
```

U tom slučaju, varijabla intErrNum je varijabla koja sadrži broj pogreške koji je pokrenuo rukovatelja pogreškom. Kad programski kod dosegne naredbu Resume, poziva se postupak Clear objekta Err. Potrebno je obnoviti pogrešku kako bi ju proslijedili natrag prethodnom potprogramu na slogu poziva.

Možete također oponašati svaku pogrešku tijeka izvođenja u Visual Basicu pribavljanjem koda pogreške za tu pogrešku:

```
Err.Raise Number:=71      ‘ Oponašanje pogreške “Disketa nije spremna”.
```

Određivanje vaših vlastitih pogrešaka

Ponekad trebate odrediti pogreške kao dodatak onim koje je odredio Visual Basic. Na primjer, aplikacija koja se oslanja na povezivanje modemom može stvoriti pogrešku kad nestane signala. Ako želite stvoriti i hvatati svoje vlastite pogreške, možete dodati brojeve svojih pogrešaka konstanti vbObjectError.

Konstanta vbObjectError rezervira brojeve u opsegu od svog vlastitog pomaka do svog pomaka + 512. Upotreba broja većeg od tog će osigurati da se brojevi vaših pogrešaka ne sukobljavaju s sljedećim verzijama Visual Basica ili drugih Microsoft Basic proizvoda. ActiveX kontrole mogu također određivati svoje vlastite brojeve pogrešaka. Kako bi izbjegli sukobe s njima, potražite upute u dokumentaciji za kontrole koje upotrijebljavate u vašoj aplikaciji.

Kako bi odredili svoje vlastite brojeve pogrešaka, dodajte konstante u odjeljak Declarations vašeg modula:

```
‘ Konstante pogrešaka
Const gNemaSignala = 1 + vbObjectError + 512
Const gNemaOdziva = 2 + vbObjectError + 512
```

Nakon toga možete koristiti postupak Raise kao i s svim ugrađenim pogreškama. U ovakvom slučaju, svojstvo opisa objekta Err vratit će standardni opis – “Pogreška određena aplikacijom ili objektom “ (Application-defined or object defined error). Kako bi pružili svoj vlastiti opis pogreške, trebate ga dodati kao parametar postupku Raise.

Za više informacija Kako bi naučili više o stvaranju vaše vlastite pogreške, pogledajte “Postupak Raise”, u priručniku *Microsoft Visual Basic 6.0 Language Reference* biblioteke *Microsoft Visual Basic 6.0 Reference Library*.

Ugrađeno rukovanje pogreškama

Možda ste naviknuli na programiranje u programskom jeziku koji ne izaziva prigovore – drugim riječima, ne prekida izvođenje vaše koda stvaranjem prigovora kad se pojavi pogreška, nego umjesto toga bilježi pogreške kako bi ih mogli provjeriti kasnije. Programski jezik C radi na taj način, i ponekad vam može biti prikladno slijediti tu praksu u vašem programskom kodu Visual Basica.

Kad provjeravate pogreške odmah nakon svake linije koja može uzrokovati pogrešku, izvodite *ugrađeno rukovanje pogreškama*. Ovaj odlomak objašnjava različite pristupe ugrađenom rukovanju pogreškama, uključujući:

- Pisanje funkcija i izraza koji vraćaju brojeve pogreške kad se pojavi pogreška.
- Izazivanje pogreške Visual Basica u potprogramu obrada pogreške ugrađenim rukovateljem pogreškom u pozivnom potprogramu.
- Pisanje funkcije za vraćanje podatka tipa Variant, te korištenje tog podatka za ukazivanje na pozivni potprogram u kojem se pojavila pogreška.

Vraćanje brojeva pogreške

Postoji niz načina za vraćanje brojeva pogreške. Najjednostavniji način je stvaranje funkcija i izraza koje vraćaju broj pogreške, umjesto vrijednosti, ako se pojavi pogreška. Sljedeći primjer pokazuje kako možete upotrijebiti takav pristup u primjeru funkcije DatotekaPostoji, koji ukazuje postoji li ili ne određena datoteka.

```
Function DatotekaPostoji(p As String) As Long
  If Dir(p) <> “” Then
    DatoPostoji = conUspjeh             ‘ Vraćanje konstante
                                        ‘ koja ukazuje na
  Else                                  ‘ postojanje datoteke.
    DatoPostoji = conNeuspjeh           ‘ Vraćanje konstante
                                        ‘ neuspjeha.
  End If
End Function
Dim Rezultat As Long
Rezultat = DatotekaPostoji(“C:\Testdato.txt”)
If Rezultat = conNeuspjeh Then
  .
  .    ‘ Obrada pogreške.
  .
Else
  .
  .    ‘ Nastavak aplikacije.
  .
End If
```

Ključ ugrađenog rukovanja pogreškama je ispitivanje pogreške odmah nakon svakog izraza ili poziva funkcije. Na taj način, možete oblikovati rukovatelja koji točno predviđa vrstu pogreške koja bi se mogla pojaviti te ju odgovarajuće rješava. Takav pristup ne zahtjeva pojavu stvarne pogreške tijekom izvođenja. To postaje korisno kod rada s API-jem i ostalim DLL potprogramima koji ne podižu prigovore Visual Basica. Umjesto toga, ti potprogrami naznačuju stanje pogreške, ili u povratnoj vrijednosti, ili u jednom od argumenata proslijeđenih potprogramu; provjerite dokumentaciju za potprograme koje upotrijebljavate da bi odredili kako ti potprogrami naznačuju stanje pogreške.

Rukovanje pogreškama u pozivnom potprogramu

Drugi način naznačivanja stanja pogreške je izazivanje pogreške Visual Basica u samom potprogramu, te obrada pogreške u ugrađenom rukovatelju pogreškom unutar pozivnog potprograma. Sljedeći primjer pokazuje isti potprogram DatotekaPostoji, koji izaziva broj pogreške ako nije uspješan. Prije poziva te funkcije, izraz On Error Resume Next postavlja vrijednosti svojstava objekta Err kad se pojavi pogreška, ali bez pokušavanja izvođenja rutine za rukovanje pogreškom.

Iza izraza On Error Resume Next slijedi kod rukovanja pogreškom. Taj programski kod može provjeriti svojstva objekta Err kako bi ustanovio je li se pojavila pogreška. Ako svojstvo Err.Number ne sadrži nulu, pogreška se pojavila, i kod rukovanja pogreškom može poduzeti prikladnu akciju temeljenu na vrijednostima svojstava objekta Err.

```
Function DatotekaPostoji(p As String)
  If Dir(p) <> “” Then
    Err.Raise conUspjeh                ‘ Vraćanje konstante
                                       ‘ koja ukazuje na
  Else                                 ‘ postojanje datoteke.
    Err.Raise conNeuspjeh              ‘ Vraćanje konstante
                                       ‘ neuspjeha.
  End If
End Function
Dim Rezultat As Long
On Error Resume Next
Rezultat = DatotekaPostoji(“C:\Testdato.txt”)
If Err.Number = conNeuspjeh Then
  .
  .        ‘ Obrada pogreške.
  .
Else
  .
  .        ‘ Nastavak aplikacije.
  .
End If
```

Sljedeći primjer koristi oba načina, povratnu vrijednost i jedan od proslijeđenih argumenata za naznačivanje je li stanje pogreške rezultiralo iz poziva funkcije ili nije:

```
Function Snaga(X As Long, P As Integer, _
ByRef Rezultat As Integer) As Long
  On Error Goto ObradaPogreške
  Rezultat = X^P
  Exit Function
ObradaPogreške:
  RezSnaga = conNeuspjeh
End Function
‘ Poziv funkcije Snaga.
Dim lngPovratnaVrijednost As Long, lngMožda Pogreška As Long
lngMožda Pogreška = Snaga(10, 2, lngPovratnaVrijednost)
If lngMožda Pogreška Then
  .
  .        ‘ Obrada pogreške.
  .
Else
  .
  .        ‘ Nastavak aplikacije.
  .
End If
```

Ako je funkcija napisana jednostavno za vraćanje ili vrijednosti rezultata ili koda pogreške, rezultirajuća vrijednost možda će biti u opsegu kodova pogrešaka, pa ih vaš pozivni potprogram neće biti u stanju razlikovati. Korištenjem oba načina, povratne vrijednosti i jednog od proslijeđenih argumenata, vaša aplikacija može ustanoviti da poziv funkcije nije uspio, i poduzeti prikladnu akciju.

Korištenje podataka tipa Variant

Još jedan način vraćanja ugrađene informacije o grešci je iskorištavanje tipa podatka Variant u Visual Basicu te nekih povezanih funkcija. Podatak tipa Variant ima oznaku koja naznačuje tip podatka koji je sadržan u varijabli, i može biti označen kao kod pogreške Visual Basica. Možete napisati funkciju za vraćanje tipa Variant, te upotrijebiti tu oznaku za naznačivanje pozivnom potprogramu da se pojavila pogreška.

Sljedeći primjer pokazuje kako funkcija Snaga može biti napisana za vraćanje podatka tipa Variant.

```
Function Snaga(X As Long, P As Integer) As Variant
  On Error Goto ObradaPogreške
  RezSnaga = X^P
  Exit Function
ObradaPogreške:
  RezSnaga = CVErr(Err.Number)    ‘ Pretvaranje koda pogreške
                                  ‘ u oznaku tipa Variant.
End Function
```

```
‘ Poziv funkcije Snaga.
Dim varPovratnaVrijednost = Snaga(10, 2)
If IsError (varPovratnaVrijednost) Then
   .
   .        ‘ Obrada pogreške.
   .
Else
   .
   .        ‘ Nastavak aplikacije.
   .
End If
```

Centralizirano rukovanje pogreškama

Kad vašoj aplikaciji dodate kod za rukovanje pogreškama, brzo ćete otkriti da stalno iznova obrađujete iste pogreške. Pažljivim planiranjem, možete smanjiti veličinu programskog koda pisanjem nekoliko potprograma koje može pozivati vaš kod rukovanja pogreškama kako bi obradio uobičajene situacije pogreške.

Sljedeći potprogram funkcije PogreškeDatoteke prikazuje poruku prikladnu grešci koja se pojavila i, gdje je moguće, omogućuje korisniku da odabere gumb kako bi odredio koju će daljnju akciju aplikacija poduzeti. Nakon toga vraća kod potprogramu koji je pozvao ovu funkciju. Vrijednost koda ukazuje koju će akciju aplikacija poduzeti. Uočite da korisnički određene konstante kao mnGreNedostupanPogon moraju negdje biti određene (ili globalno, ili na razini modula u modulu koji sadržava ovaj potprogram, ili unutar samog potprograma). Konstanta vbExclamation je određena u objektnoj biblioteci Visual Basica (VB), i zbog toga ne treba biti određena.

```
Function PogreškeDatoteke() As Integer
  Dim intTipPoruke As Integer, strPor As String
  Dim intOdgovor As Integer
  ‘ Povratna vrijednost          Značenje
  ‘ 0                            Resume
  ‘ 1                            Resume Next
  ‘ 2                            Neispravljiva pogreška
  ‘ 3                            Neprepoznata pogreška
  intTipPoruke = vbExclamation
  Select Case Err.Number
    Case mnPogNedostupanPogon           ‘ pogreška 68.
          strPor = “Ovaj uređaj izgleda nedostupan.”
          intTipPoruke = vbExclamation + 4
    Case mnPogDisketaNijeSpremna        ‘ pogreška 71.
          strPor = “Ubacite disketu u pogon “
          strPor = StrPor & “i zatvorite vrata pogona.”
          intTipPoruke = vbExclamation + 4
    Case mnPogIOUre|aja                 ‘ pogreška 57.
          strPor = “Unutarnja pogreška pogona.”
          intTipPoruke = vbExclamation + 4
```

```
    Case mnPogPunDisk                   ‘ pogreška 61.
          strPor = “Disk je pun. Nastaviti?”
          intTipPoruke = vbExclamation + 3
    ‘ pogre{ke 64 i 52.
    Case mnPogLošeImeDato, mnPogLošeImeIliBrojDato
          strPor = “To ime datoteke je neispravno.”
          intTipPoruke = vbExclamation + 4
    Case mnPogNemaStaze                 ‘ pogreška 76.
          strPor = “Ta staza ne postoji.”
          intTipPoruke = vbExclamation + 4
    Case mnPogLošTipDatoteke            ‘ pogreška 54.
          strPor = “Ne mogu otvoriti vašu datoteku “
          strPor = strPor & “takvim tipom pristupa.”
          intTipPoruke = vbExclamation + 4
    Case mnPogDatoVe}Otvorena           ‘ pogreška 55.
          strPor = “Ova datoteka je već otvorena.”
          intTipPoruke = vbExclamation + 4
    Case mnPogUlazIzaKraja              ‘ pogreška 62.
          strPor = “Ova datoteka ima nestandardnu “
          strPor = strPor & “oznaku kraja datoteke, “
          strPor = strPor & “ili je pokušano čitati “
          strPor = strPor & “iza oznake kraja datoteke.”
          intTipPoruke = vbExclamation + 4
    Case Else
          PogreškeDato = 3
          Exit Function
End Select
intOdgovor = MsgBox(strPor, intTipPoruke, “pogreška diska”)
Select Case intOdgovor
  Case 1, 4              ‘ Gumbi OK, Retry.
    PogreškeDato = 0
  Case 5                 ‘ Gumb Ignore.
    PogreškeDato = 1
  Case 2, 3              ‘ Gumbi Cancel, End.
    PogreškeDato = 2
  Case Else
    PogreškeDato = 3
  End Select
End Function
```

Ovaj potprogram obrađuje uobičajene pogreške vezane uz datoteke i pogone. Ako pogreška nije povezana s ulazom/izlazom, vraća se vrijednost 3. Potprogram koji poziva ovu funkciju treba zatim sam obraditi pogrešku, obnoviti pogrešku postupkom Raise, ili pozvati drugi potprogram koji bi obradio pogrešku.

Napomena Dok pišete veće aplikacije, vidjet ćete da upotrijebljavate iste konstante u raznim potprogramima unutar raznih formi i modula. Stvaranje tih konstanti javnim i njihovo određivanje u jednom standardnom modulu može bolje organizirati vaš kod i poštedjeti vas ponavljanog pisanja istih određivanja.

Rukovanje pogreškama možete pojednostaviti pozivom potprograma PogreškeDatoteke uvijek kad imate potprogram koji čita ili piše na disk. Na primjer, vjerojatno ste koristili aplikacije koje vas upozoravaju ako pokušate snimiti datoteku koja već postoji na disku. Obratno, kad pokušate otvoriti datoteku koja ne postoji, većina aplikacija će vas upozoriti da datoteka ne postoji i pitati vas želite li ju stvoriti. U oba primjera, mogu se pojaviti pogreške kad aplikacija proslijedi ime datoteke operativnom sustavu.

Sljedeća rutina provjere koristi vrijednost vraćenu od funkcije PogreškeDatoteke za utvrđivanje koju akciju poduzeti kod događaja pogreške povezane s diskom.

```
Function PotvrdaDatoteke(DatoIme As String, _
Operacija As Integer) As Integer
‘ Parametri:
‘ DatoIme: datoteka koja će biti provjerena i potvrđena.
‘ Operacija: kod za sekvencijalni mod pristupa datoteci
‘ (Output, Input i tako dalje).
‘ Uočite da potprogram radi za binarni i nasumični pristup
‘ jer su poruke uvjetovane stanjem Operacija
‘ različitim od određenih sekvencijalnih modova.
‘ Povratne vrijednosti:
‘ 1        Potvrda operacije neće izazvati problem.
‘ 0        Korisnik odlučuje da ne ide dalje s operacijom.
  Const conSnimiDatoteku = 1, conUčitajDatoteku = 2
  Const conZamijeniDatoteku = 1, conČitajDatoteku = 2
  Const conDodajDatoteci = 3, conNasumičnaDatoteka = 4
  Const conBinarnaDatoteka = 5
  Dim intPotvrda As Integer
  Dim intAkcija As Integer
  Dim intBrojPogreške As Integer, varPor As Variant
  On Error GoTo PotvrdaPogreškeDato      ‘ Uključuje
                                         ‘ hvatanje pogreške.
  DatoIme = Dir(DatoIme)                 ‘ Pogledaj postoji li datoteka.
  On Error GoTo 0                        ‘ Isključuje hvatanje pogreške.
  ‘ Ako korisnik snima tekst u datoteku
  ‘ koja već postoji...
  If DatoIme <> “” And Operacija = conZamijeniDatoteku Then
    varPor = “Datoteka “ & DatoIme
    varPor = varPor & “već postoji na “ & vbCrLf
    varPor = varPor & “disku. Snimanje sadržaja “
    varPor = varPor & vbCrLf
    varPor = varPor & “okvira s tekstom u datoteku “
    varPor = varPor & “ će uništiti trenutan “
    varPor = varPor & “sadržaj datoteke, “ & vbCrLf
    varPor = varPor & “zamjenjujući ga tekstom”
```

```
  varPor = varPor & “iz okvira s tekstom.”
  varPor = varPor & vbCrLf & vbCrLf
  varPor = varPor & “Odaberite OK za zamjenu datoteke, “
  varPor = varPor & “Cancel za zaustavljanje.”
  intPotvrda = MsgBox(varPor, 65, “Poruka o datoteci”)
‘ Ako korisnik želi učitati tekst iz
‘ datoteke koja ne postoji.
ElseIf DatoIme = “” And Operacija = conČitajDatoteku Then
  varPor = “Datoteka “ & DatoIme
  varPor = varPor & “ne postoji.” & vbCrLf
  varPor = varPor & “Želite li ju stvoriti “
  varPor = varPor & “i zatim editirati?” & vbCrLf
  varPor = varPor & vbCrLf & “Odaberite OK za “
  varPor = varPor & “stvaranje datoteke, “
  varPor = varPor & “Cancel za zaustavljanje.”
  intPotvrda = MsgBox(varPor, 65, “Poruka o datoteci”)
‘ Ako DatoIme ne postoji, prisiljavanje potprograma
‘ da vrati 0 postavljanjem intPotvrda na 2.
ElseIf DatoIme = “” Then
  If Operacija = conNasumičnaDatoteka Or _
  Operacija = conBinarnaDatoteka Then
        intPotvrda = 2
  End If
‘ Ako datoteka postoji i operacija nije uspješna,
‘ intPotvrda = 0 i potprogram vraća 1.
End If
‘ Ako okvir nije prikazan, inPotvrda = 0;
‘ Ako korisnik odabere OK, u svakom slučaju,
‘ intPotvrda = 1 i PotvrdaDatoteke treba
‘ vratiti 1 kako bi potvrdila da je namjeravana
‘ operacija u redu. Ako intPotvrda > 1, PotvrdaDatoteke
‘ treba vratiti 0, jer korisnik ne želi
‘ nastaviti dalje operaciju...
If intPotvrda > 1 Then
  PotvrdaDato = 0
Else
  PotvrdaDato = 1
  If intPotvrda = 1 Then
        ‘ Korisnik želi stvoriti datoteku.
        If Operacija = conUčitajDatoteku Then
              ‘ Dodjela conZamijeniDatoteku tako da pozivatelj
              ‘ može razumjeti akciju koja će biti poduzeta.
              Operacija = conZamijeniDatoteku
        End If
```

```
      ‘ Vraćanje koda potvrđujući akciju zamjene
      ‘ postojeće datoteke ili stvaranja nove.
      End If
    End If
Exit Function
PotvrdaPogreškeDato:
intAkcija = PogreškeDato
  Select Case intAkcija
    Case 0
      Resume
    Case 1
      Resume Next
    Case 2
      Exit Function
    Case Else
      intBrojPogreške = Err.Number
      Err.Raise Number:=intBrojPogreške
      Err.Clear
  End Select
End Function
```

Potprogram PotvrdaDatoteke prima specifikaciju za datoteku čije će postojanje biti potvrđeno, s informacijama o modu pristupa koji će biti upotrijebljen kad bude poduzet stvarni pokušaj otvaranja datoteke. Ako treba biti snimljena sekvencionalna datoteka (conZamijeniDatoteku), pronađena je datoteka koja već ima to ime (i zbog toga će biti presnimljena), od korisnika se traži da potvrdi prihvatljivost presnimavanja datoteke.

Ako treba biti otvorena sekvencijalna datoteka (conČitajDatoteku), a datoteka nije pronađena, od korisnika se traži da potvrdi stvaranje nove datoteke. Ako je datoteka otvorena za nasumičan ili binarni pristup, njezino postojanje je potvrđeno (povratna vrijednost 1) ili opovrgnuto (povratna vrijednost 0). Ako se pojavi pogreška pri pozivu funkcije Dir, poziva se potprogram PogreškeDatoteke da analizira pogrešku i od korisnika zatraži razuman smjer akcije.

Isključivanje rukovanja pogreškama

Ako je zamka za pogreške omogućena u potprogramu, ona se automatski onemogućuje kad potprogram završi izvođenje. Međutim, možete trebati onemogućiti zamku za pogreške u potprogramu dok se kod u tom potprogramu i dalje izvodi. Kako bi isključili omogućenu zamku za pogreške, upotrijebite izraz On Error GoTo 0. Kad jednom Visual Basic izvrši taj izraz, pogreške se otkrivaju, ali ne hvataju unutar potprograma. Izraz On Error GoTo 0 možete upotrijebiti za isključivanje rukovanja pogreškama bilo gdje u potprogramu – čak i unutar same rutine rukovanja pogreškama.

Na primjer, pokušajte prolaz korak po korak, korištenjem osobine Step Into, kroz potprogram sličan ovom:

```
Sub PrimjerPogreške()
  On Error GoTo Rukovatelj       ‘ Rukovanje pogreškom
                                 ‘ je omogućeno.
    ‘ pogreške trebaju biti uhvaćene i ispravljene ovdje.
    ‘ Funkcija Kill koristi se za brisanje datoteke.
    Kill “Staradatoteka.xyz”
  On Error GoTo 0                ‘ Rukovanje pogreškom
                                 ‘ se isključuje ovdje.
    Kill “Staradatoteka.xyz”
  On Error GoTo Rukovatelj       ‘ Ponovno se omogućuje
                                 ‘ hvatanje pogreške.
    Kill “Staradatoteka.xyz”
  Exit Sub
Rukovatelj:               ‘ Ovdje ide rutina rukovanja pogreškama.
  MsgBox “pogreška uhvaćena.”
  Resume Next
End Sub
```

Za više informacija Kako bi naučili koristiti osobinu Step Into, pogledajte “Izvođenje odabranih dijelova vaše aplikacije”, kasnije u ovom poglavlju.

Otkrivanje pogrešaka s rukovateljima pogrešaka

Kad tražite pogreške u vašem kodu, može vas zbuniti analiziranje njegovog ponašanja kad stvara pogreške koje hvata rukovatelj pogreškama. Mogli bi dodati izlazni komentar linije On Error u svakom modulu projekta, ali to je također nezgrapno.

Umjesto toga, dok tražite pogreške, možete isključiti rukovatelje pogreškama tako da uđete u mod prekida svaki put kad se pojavi pogreška.

Kako onemogućiti rukovatelje pogreškama kod traženja pogrešaka

1. U pomoćnom izborniku **kodnog prozora** (dostupan je klikom desnom tipkom miša na kodni prozor), odaberite stavku **Toggle**.
2. Odaberite opciju **Break on All Errors**.

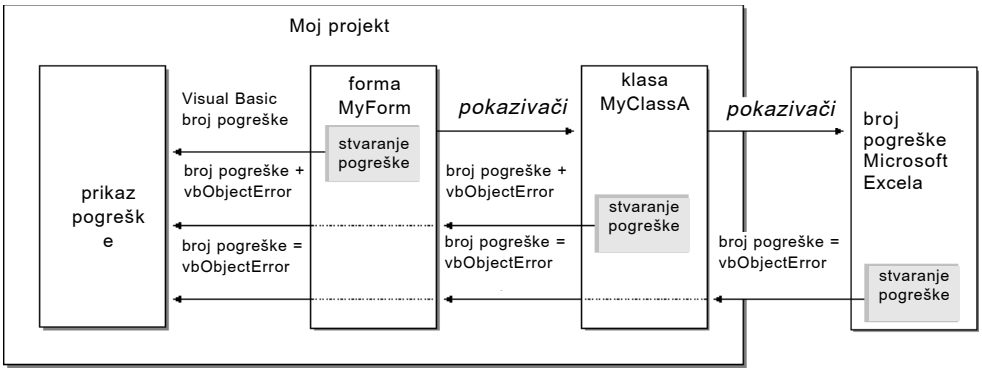
S ovom opcijom odabranom, kad se bilo gdje u projektu pojavi pogreška, ući ćete u mod prekida i kodni prozor će prikazati dio programskog koda u kojem se pojavila pogreška.

Ako ta opcija nije potvrđena, pogreška može, ali i ne mora izazvati prikazivanje poruke pogreške, ovisno o tome gdje se pojavila pogreška. Na primjer, možda je pogreška izazvana vanjskim objektom na kojeg upućuje vaša aplikacija. Ako se u takvom slučaju prikaže poruka, može biti besmislena, ovisno o tome gdje je izvor pogreške.

Rukovanje pogreškama ActiveX sastavnim dijelovima

U aplikacijama koje koriste jedan ili više objekata, postaje teže ustanoviti gdje se pojavljuje pogreška, osobito ako se pojavljuje u objektu druge aplikacije. Na primjer, slika 13.4 prikazuje aplikaciju koja se sastoji od modula forme, koji upućuje na modul klase, koji skreće ukazivanje na objekt Worksheet Microsoft Excela.

Slika 13.4 Obnavljanje pogrešaka između formi, klasa i ActiveX sastavnih dijelova



Ako objekt Worksheet ne obradi određenu pogrešku koja se pojavi u njemu, nego ju umjesto toga obnovi, Visual Basic prosljeđuje pogrešku objektu koji je na njega upućivao, klasi MyClassA. Kad se pogreška pojavi u vanjskom objektu i nije uhvaćena, bit će izazvana u potprogramu koji je pozvao taj vanjski objekt.

Objekt MyClassA može obraditi pogrešku (što je preporučljivo), ili ju obnoviti. Sučelje određuje da svaki objekt koji obnavlja pogrešku izazvanu u pozivanom objektu ne smije jednostavno slati pogrešku (proslijediti kod pogreške), nego umjesto toga treba pretvoriti broj pogreške u nešto smisleno. Kad tako preslikate pogrešku, broj može biti ili broj određen Visual Basicom koji naznačuje stanje pogreške, ako vaš rukovatelj može ustanoviti da je pogreška slična pogreškama određenim Visual Basicom (na primjer, prekoračenje ili dijeljenje nulom), ili broj neodređene pogreške. Dodajte novi broj konstanti vbObjectError ugrađenoj u Visual Basic kako bi naznačili ostalim rukovateljima da je ta pogreška izazvana vašim objektom.

Uvijek kad je moguće, modul klase trebao bi pokušati obraditi svaku pogrešku koja se pojavi unutar samog modula, i trebao ti također pokušati obraditi pogreške koje se pojave u objektu na koji ukazuje, a koje nisu obrađene tim objektom. Međutim, postoje neke pogreške koje modul klase ne može obraditi jer ih ne može predvidjeti. Postoje također slučajevi kad je prikladnije da pogrešku obradi objekt koji upućuje, nego objekt na kojeg se upućuje.

Kad se pogreška pojavi u modulu forme, Visual Basic izaziva jedan od predodređenih brojeva pogreške Visual Basica.

Napomena Ako stvarate javnu klasu, obavezno dokumentirajte značenje svakog rukovatelja pogreškama koji nije iz Visual Basica (javne klase ne mogu biti stvorene u verziji Learning). Ostali programeri koji upućuju na vaše javne klase trebaju znati kako rukovati pogreškama koje izazovu vaši objekti.

Kad obnovite pogrešku, nemojte mijenjati ostala svojstva objekta Err. Ako izazvana pogreška nije uhvaćena, svojstva Source i Description mogu biti prikazana kako bi pomogla korisniku da poduzme akciju ispravljanja.

Rukovanje pogreškama u objektima

Modul klase treba sadržavati sljedeći rukovatelj pogreškama kako bi prilagodio svaku pogrešku koju može uhvatiti, obnavljajući one koje ne može riješiti:

```
MojPoslužiteljRukovatelja:
  Select Case BrojPogreške
    Case 7               ‘ Rukovanje pogreškom Nedovoljno memorije.
           .
           .
           .
    Case 440             ‘ Rukovanje pogreškom vanjskog objekta.
           Err.Raise Number:=vbObjectError + 9999
    ‘ pogre{ka iz drugog objekta Visual Basica.
    Case Is > vbObjectError And Is < vbObjectError + 65536
           ObjectError = BrojPogreške
    Select Case ObjectError
           ‘ Ovaj objekt rukuje pogreškom, temeljeno
           ‘ na dokumentaciji koda pogre{ke za objekt.
           Case vbObjectError + 10
                  .
                  .
                  .
           Case Else
                  ‘ Preslikavanje pogreške kao opće pogreške objekta
                  ‘ i obnavljanje pogreške.
                  Err.Raise Number:=vbObjectError + 9999
           End Select
    Case Else
           ‘ Preslikavanje pogreške kao opće pogreške objekta
           ‘ i obnavljanje pogreške.
           Err.Raise Number:=vbObjectError + 9999
  End Select
  Err.Clear
  Resume Next
```

Izraz `Case 440` hvata pogreške koje se pojavljuju u objektu na koji se upućuje izvan aplikacije Visual Basica. U ovom primjeru, pogreška je jednostavno prenesena korištenjem vrijednosti 9999, budući da je ovom tipu centralnog rukovatelja teško ustanoviti uzrok pogreške. Kad je pogreška izazvana, ona je općenito rezultat kobne pogreške automatizacije (one koja će uzrokovati kraj izvođenja sastavnog dijela), ili se pojavljuje zato što objekt nije ispravno obradio uhvaćenu pogrešku. pogreška 440 ne bi trebala biti prenesena osim ako je kobna pogreška. Ako je ta zamka napisana za ugrađeni rukovatelj kao što je raspravljeno u odlomku "Ugrađeno rukovanje pogreškama", ranije u ovom poglavlju, moguće je ustanoviti uzrok pogreške i ispraviti ga.

Ovaj izraz

```
Case Is > vbObjectError And Is < vbObjectError + 65536
```

hvata pogreške koje su uzrokovane u objektu unutar aplikacije Visual Basica, ili u samom objektu koji sadrži tog rukovatelja. Samo pogreške uzrokovane od objekata bit će u opsegu objekta vbObjectError.

Dokumentacija koda pogreške pružena za objekt treba odrediti moguće kodove pogrešaka i njihovo značenje, tako da taj dio rukovatelja treba biti napisan tako da inteligentno riješi unaprijed očekivane pogreške. Stvarni kodovi pogrešaka mogu biti dokumentirani unutar pomaka objekta vbObjectError, ili mogu biti dokumentirani nakon što su dodani pomaku, a u tom slučaju izraz Case Else treba oduzeti objekt vbObjectError, umjesto da ga dodaje. S druge strane, pogreške objekata mogu biti konstante, prikazane u tipskoj biblioteci objekta, kao što je prikazano u pretraživaču objekata. U tom slučaju, upotrijebite konstantu pogreške u izrazu Case Else, umjesto koda pogreške.

Svaka neobrađena pogreška treba biti obnovljena s novim brojem, kao što je prikazano u izrazu Case Else. Unutar vaše aplikacije, možete oblikovati rukovatelja tako da preduhitri te nove brojeve koje ste odredili. Ako je to bila javna klasa (nedostupno u verziji Learning), trebali bi također uključiti objašnjenje novog koda rukovanja pogreškom u dokumentaciji vaše aplikacije.

Posljednji izraz Case Else hvata i obnavlja sve ostale pogreške koje nisu uhvaćene drugdje u rukovatelju. Budući da će ovaj dio zamke hvatati pogreške mogu, ali i ne moraju imati dodane konstante u objektu vbObjectError, trebate jednostavno preslikati te pogreške u opći kod "neriješene pogreške". Taj kod bi trebao biti dodan objektu vbObjectError, ukazujući svakom rukovatelju da je ta pogreška uzrokovana u upućenom objektu.

Traženje pogrešaka rukovateljima u ActiveX sastavnim dijelovima

Kad ispravljate pogreške u aplikaciji koja ima pokazivač na objekt stvoren u Visual Basicu ili klasi određenoj u modulu klase, možda će vas zbuniti ustanovljavanje koji je objekt stvorio pogrešku. Kako bi to olakšali, možete odabrati opciju Break in Class Module kartice General u dijaloškom okviru Options (dostupnom iz izbornika Tools). Kad je ta opcija potvrđena, pogreška u modulu klase ili objekt u drugoj aplikaciji ili projektu koji se izvodi u Visual Basicu će potaknuti tu klasu da uđe u mod prekida, omogućujući vam da analizirate pogrešku. Pogreška koja se pojavi u prevedenom objektu neće prikazati prozor za neposredan upis naredbi u modu prekida; umjesto toga, takve pogreške će biti obrađene rukovateljem pogrešaka samog objekta, ili će biti uhvaćene pozivajućim modulom.

Za više informacija Za temeljitu raspravu o opciji Break in Class Module, pogledajte “Ispravljanje modula klase” u 9. poglavlju “Programiranje objektima”.

Pristupi otkrivanju pogrešaka

Tehnike otkrivanja pogrešaka predstavljene u ovom poglavlju upotrijebljavaju alate za analiziranje koje pruža Visual Basic. Visual Basic ne može raspoznati niti ispraviti pogreške umjesto vas, ali pruža alate koji vam pomažu u analiziranju kako izvođenje teče od jednog dijela potprograma do drugog, te kako se mijenjaju varijable i postavke svojstava dok se izvode naredbe. Alati otkrivanja pogrešaka omogućuju vam pogled iznutra u vašu aplikaciju kako bi vam pomogli ustanoviti što se i zašto događa.

Podrška otkrivanju pogrešaka Visual Basica uključuje točke prekida, izraze prekida, izraze promatranja, prolaz kroz kod jedan po jedan izraz ili potprogram, te prikazivanje vrijednosti varijabli ili svojstava. Visual Basic također sadrži posebne osobine otkrivanja pogrešaka, kao što su sposobnost editiraj-i-nastavi, određivanje idućeg izraza koji će biti izveden, te ispitivanje potprograma dok je aplikacija u modu prekida.

Za više informacija Za brz pregled otkrivanja pogrešaka u Visual Basicu, pogledajte odlomak “Savjeti za otkrivanje pogrešaka”, kasnije u ovom poglavlju.

Vrste pogrešaka

Kako bi razumjeli koliko je korisno otkrivanje pogrešaka, razmotrite tri vrste pogrešaka koje možete otkriti:

- Pogreške prevođenja
- Pogreške tijekom izvođenja
- Logičke pogreške

Pogreške prevođenja

Pogreške prevođenja rezultat su neispravno sastavljenog programskog koda. Ako tijekom izrade aplikacije netočno napišete ključnu riječ, izostavite neki obavezan znak interpunkcije, ili upotrijebite naredbu Next bez pripadajuće naredbe For, Visual Basic će otkriti te pogreške kad prevedete aplikaciju.

Pogreške prevođenja uključuju pogreške u sintaksi. Na primjer, možete imati ovakvu naredbu:

```
Left
```

Left je valjana riječ u jeziku Visual Basica, ali bez objekta, ne zadovoljava zahtjeve sintakse za tu riječ (*objekt*.Left). Ako ste potvrdili opciju Auto Syntax Check na kartici Editor dijaloškog okvira Options, Visual Basic će prikazati poruku pogreške odmah čim napravite sintaksnu pogrešku u kodnom prozoru.

Kako odrediti opciju Auto Syntax Check

1. U izborniku **Tools**, odaberite **Options**, te kliknite karticu **Editor** u dijaloškom okviru **Options**.
2. Odaberite **Auto Syntax Check**.

Za više informacija Pogledajte dio "Izbjegavanje pogrešaka", kasnije u ovom poglavlju, za ostale tehnike koje možete upotrijebiti za izbjegavanje pogrešaka u svem kodu.

pogreške tijekom izvođenja

pogreške tijekom izvođenja pojavljuju se dok se aplikacija izvodi (i otkrije ih Visual Basic) kad izraz pokuša operaciju koju je nemoguće ostvariti. Primjer toga je dijeljenje s nulom. Pretpostavimo da imate ovaj izraz:

```
Brzina = Kilometara / Sati
```

Ako varijabla Sati sadržava nulu, dijeljenje je neispravna operacija, iako je sam izraz sintaksno ispravan. Aplikacija se mora izvoditi prije nego što može otkriti takvu pogrešku.

Za više informacija U vašu aplikaciju možete uključiti kod koji će hvatati i obrađivati pogreške tijekom izvođenja kad se pojave. Za informacije o upravljanju pogreškama tijekom izvođenja, pogledajte odlomak "Kako rukovati pogreškama", ranije u ovom poglavlju.

Logičke pogreške

Logičke pogreške se pojavljuju kad se aplikacija ne izvodi na način na koji bi trebala. Aplikacija može imati sintaksno valjan programski kod, izvoditi se bez obavljanja bilo koje neispravne operacije, pa ipak proizvesti neispravne rezultate. Samo ispitivanjem aplikacije i analiziranjem rezultata možete provjeriti da li se aplikacija ispravno izvodi.

Kako pomažu alati otkrivanja pogrešaka

Alati za otkrivanje pogrešaka oblikovani su kako bi vam pomogli s:

- Logičkim pogreškama i pogreškama tijekom izvođenja.
- Promatranjem ponašanja koda koji nema pogrešaka.

Na primjer, neispravan rezultat može biti proizveden na kraju dugog niza proračuna. U otkrivanju pogrešaka, zadatak je ustanoviti što je i gdje pošlo krivo. Možda ste zaboravili inicijalizirati varijablu, odabrali ste krivi operator ili ste upotrijebili netočnu formulu.

Nema čarobnih trikova za otkrivanje pogrešaka, i nema nepromjenjivog niza postupaka koji radi svaki put. Temeljno, otkrivanje pogrešaka vam pomaže u razumijevanju što se događa dok se izvodi vaša aplikacija. Alati otkrivanja pogrešaka daju vam brzu snimku trenutnog stanja vaše aplikacije, uključujući:

- Izgled korisničkog sučelja (user interface, UI).
- Vrijednosti varijabli, izraza i svojstava.
- Aktivne pozive potprograma.

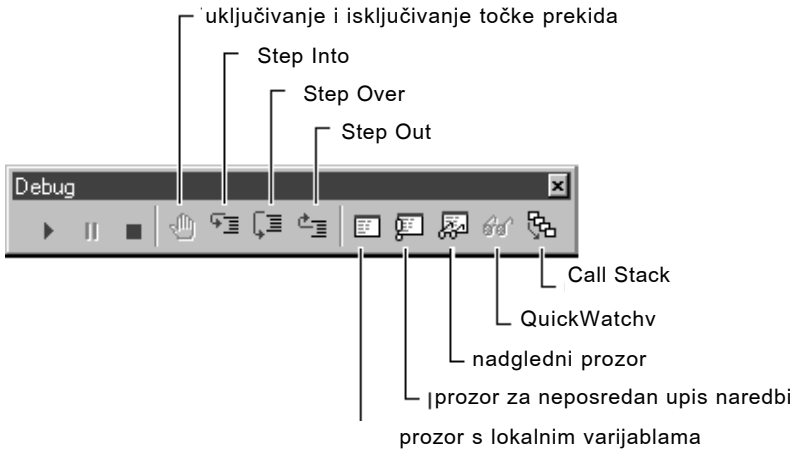
Što bolje shvatite kako radi vaša aplikacija, to ćete brže pronaći pogreške.

Za više informacija Za više detalja o pregledu i ispitivanju varijabli, izraza, svojstava i aktivnih poziva potprograma, pogledajte odlomke “Ispitivanje podataka i potprograma prozorom za neposredan upis naredbi” i “Nadziranje stoga poziva”, kasnije u ovom poglavlju.

Alatna traka Debug

Među njegovim raznim alatima za otkrivanje pogrešaka, Visual Basic pruža nekoliko gumbâ na dodatnoj alatnoj traci Debug koji su vrlo korisni. Slika 13.5 prikazuje te alate. Kako bi prikazali alatnu traku Debug, kliknite desnom tipkom miša na alatnu traku Visual Basica i odaberite opciju Debug.

Slika 13.5 Alatna traka Debug



Sljedeća tablica ukratko opisuje namjenu svakog alata. Teme o ovom poglavlju raspravljaju situacije gdje vam svaki od ovih alata može pomoći u učinkovitijem otkrivanju pogrešaka ili analiziranju aplikacije.

alat	namjena
Breakpoint	Određuje liniju u kodnom prozoru gdje Visual Basic obustavlja izvođenje aplikacije.
Step Into	Izvodi sljedeću izvodljivu liniju koda u aplikaciji i ulazi u potprograme.
Step Over	Izvodi sljedeću izvodljivu liniju koda u aplikaciji bez ulaza u potprograme.
Step Out	Izvodi ostatak trenutnog potprograma i prekida izvođenje na sljedećoj liniji pozivnog potprograma.
Locals Window	Prozor s lokalnim varijablama, prikazuje trenutne vrijednosti lokalnih varijabli.
Immediate Window	Prozor za neposredan upis naredbi, omogućuje vam izvođenje koda ili traženje vrijednosti dok je aplikacija u modu prekida.
Watch Window	Nadgledni prozor, prikazuje vrijednosti određenih izraza.
Quick Watch	Dijaloški okvir za brzi pregled, ispisuje trenutne vrijednosti izraza dok je aplikacija u modu prekida.
Call Stack	U modu prekida, predstavlja dijaloški okvir koji pokazuje sve potprograme koji su pozivani, ali još nisu završili rad.

Za više informacija Alati za otkrivanje pogrešaka su neophodni samo ako u vašoj aplikaciji postoje pogreške. Pogledajte sljedeći odlomak “Izbjegavanje pogrešaka”.

Izbjegavanje pogrešaka

Postoji nekoliko načina za izbjegavanje stvaranja pogrešaka u vašoj aplikaciji:

- Pažljivo oblikujte svoje aplikacije zapisivanjem važnijih događaja i načina na koje će vaš programski kod odgovoriti na svakog od njih. Dajte svakom potprogramu događaja i svakom općem potprogramu određenu, dobro određenu namjenu.
- Uključite brojne komentare. Kad se vratite na analiziranje svog koda, puno ćete ga bolje shvatiti ako u komentarima navedete namjenu svakog potprograma.
- Jasno se upućujte na objekte uvijek kad je to moguće. Odredite objekte kao što su izlistani u okviru Classes/Modules pretraživača objekata, prije nego da upotrijebljavate vrijednosti tipa Variant ili općenite podatke tipa Object.

- Razvijte dosljednu shemu davanja imena varijablama i objektima u vašoj aplikaciji. Za više informacija, pogledajte Dodatak B “Pravila programiranja Visual Basica”.
- Jedan od najčešćih izvora pogrešaka je neispravno pisanje imena varijable ili miješanje jedne kontrole drugom. Možete upotrijebiti izraz Option Explicit za izbjegavanje pogrešnog pisanja imena varijabli. Za više informacija o zahtijevanju izričitog određivanja varijable, pogledajte “Uvod u varijable, konstante i tipove podataka” u 5. poglavlju “Osnove programiranja”.

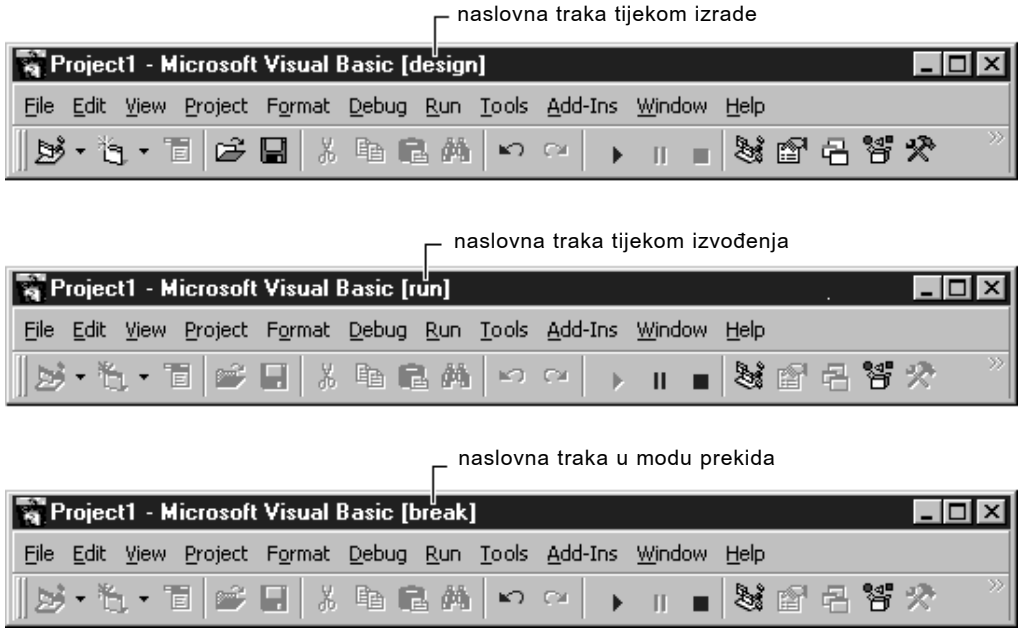
Vrijeme izrade, vrijeme izvođenja i mod prekida

Kako bi ispitali aplikaciju i otkrili pogreške, trebate razumjeti u kojem ste od tri moda u bilo kojem trenutku. Upotrijebljavate Visual Basic tijekom izrade kako bi stvorili aplikaciju, a tijekom izvođenja kako bi ju izveli. Ovo poglavlje predstavlja *mod prekida* (*break mode*), koji obustavlja izvođenje aplikacije tako da možete pregledati i promijeniti podatke.

Prepoznavanje trenutnog moda

Naslovna traka Visual Basica vam uvijek pokazuje trenutan mod. Slika 13.6 pokazuje naslovnu traku za vrijeme izrade, vrijeme izvođenja i mod prekida.

Slika 13.6 Prepoznavanje trenutnog moda s naslovnom trakom Visual Basica



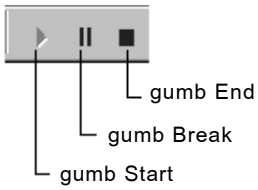
U sljedećoj tablici ispisane su karakteristike sva tri moda.

mod	opis
Vrijeme izrade	Većina posla pri stvaranju aplikacije obavlja se u vremenu izrade. Možete oblikovati forme, stvarati kontrole, pisati kod i upotrijebljavati prozor s svojstvima za određivanje ili pregled postavki svojstava. Ne možete koristiti alate otkrivanja greški, osim za postavljanje točaka prekida i stvaranja nadglednih izraza. U izborniku Run, odaberite Start, ili kliknite gumb Run za prebacivanje na vrijeme izvođenja. Ako vaša aplikacija sadrži kod koji se izvodi kad se aplikacija pokrene, odaberite Step Into iz izbornika Run (ili pritisnite F8) za postavljanje aplikacije u mod prekida na prvom izvodivom izrazu.
Vrijeme izvođenja	Kad aplikacija preuzme kontrolu, surađujete s aplikacijom na isti način kao što će korisnik. Možete vidjeti kod, ali ga ne možete mijenjati. U izborniku Run, odaberite End, ili kliknite gumb End kako bi se prebacili natrag na vrijeme izrade.
Mod prekida	U izborniku Run, odaberite Break, kliknite gumb Break, ili pritisnite CTRL + BREAK za prebacivanje u mod prekida. Izvođenje se obustavlja dok se aplikacija izvodi. Možete vidjeti i editirati kod (odaberite Code iz izbornika View, ili pritisnite F7), pregledati ili promijeniti podatke, ponovno pokrenuti aplikaciju, završiti izvođenje, ili nastaviti izvođenje iz iste točke. Tijekom izrade možete postaviti točke prekida i nadgledne izraze, ali ostali alati otkrivanja pogrešaka rade samo u modu prekida. Pogledajte “Korištenje moda prekida”, kasnije u ovom poglavlju.

Korištenje alatne trake za promjenu modova

Alatna traka pruža tri gumba koji vam omogućuju brzo prebacivanje s jednog moda na drugi. Ti gumbi su prikazani na slici 13.7.

Slika 13.7 Gumbi Start, Break i End na alatnoj traci



Koji će od tih gumbâ biti dostupan, ovisi o tome je li Visual Basic u modu izvođenja, modu izrade ili modu prekida. Sljedeća tablica ispisuje dostupne gumbe u različitim modovima.

mod	dostupni gumbi alatne trake
Vrijeme izrade	Start
Vrijeme izvođenja	Break, End
Prekid	Continue, End (u modu prekida, gumb Start postaje gumb Continue)

Korištenje prozora za otkrivanje pogrešaka

Ponekad ćete moći pronaći uzrok problema izvođenjem dijela programskog koda. Međutim, češće ćete također trebati analizirati što se događa s podacima. Možete izdvojiti problem u varijabli ili svojstvu s neispravnom vrijednošću, te zatim ustanoviti kako su i zašto ta varijabla ili svojstvo dobili neispravnu vrijednost.

S prozorima za otkrivanje pogrešaka, možete nadzirati vrijednosti izraza i varijabli dok prolazite kroz naredbe u vašoj aplikaciji. Postoje tri prozora za otkrivanje pogrešaka: prozor za neposredan upis naredbi (Immediate window), nadgledni prozor (Watch window) i prozor s lokalnim varijablama (Locals window).

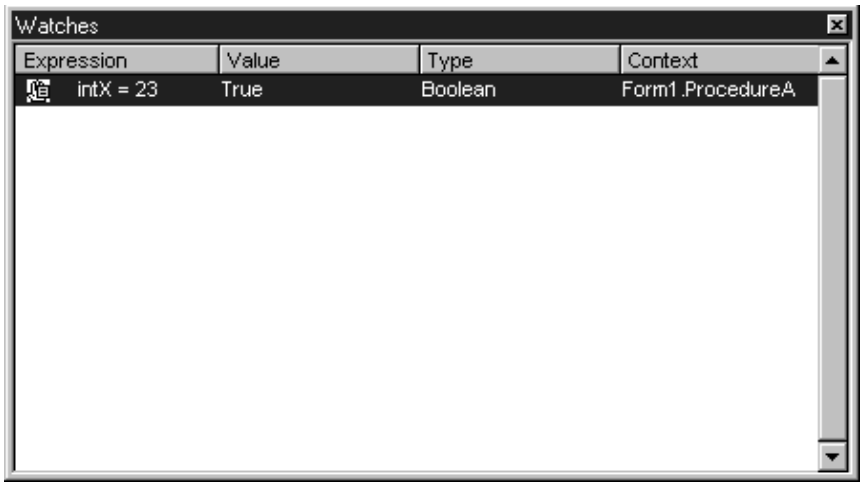
- *Prozor za neposredan upis naredbi* prikazuje informacije koje su rezultat naredbi otkrivanja pogrešaka u vašem kodu, ili koje zahtijevate pisanjem naredbi izravno u prozor.

Slika 13.8 Prozor za neposredan upis naredbi

Za više informacija Kako bi naučili više o prozoru za neposredan upis naredbi, pogledajte "Ispitivanje podataka i potprograma prozorom za neposredan upis naredbi", kasnije u ovom poglavlju.

- *Nadgledni prozor* pokazuje trenutne *nadgledne izraze*, izraze za koje vi odlučujete koje će vrijednosti nadgledati dok se izvodi kod. *Izraz prekida* je nadgledni izraz koji će potaknuti Visual Basic da uđe u mod prekida kad određeno stanje koje ste odredili postane točno. U nadglednom prozoru, stupac Context naznačuje potprogram, modul ili module u kojima se proračunava svaki nadgledni izraz. Nadgledni prozor može prikazati vrijednost za nadgledni izraz samo ako je trenutan izraz u određenom sklopu. Inače, stupac Value prikazuje poruku koja naznačuje da izraz nije u sklopu. Kako bi pristupili nadglednom prozoru, odaberite Watch Window u izborniku View. Slika 13.9 prikazuje nadgledni prozor.

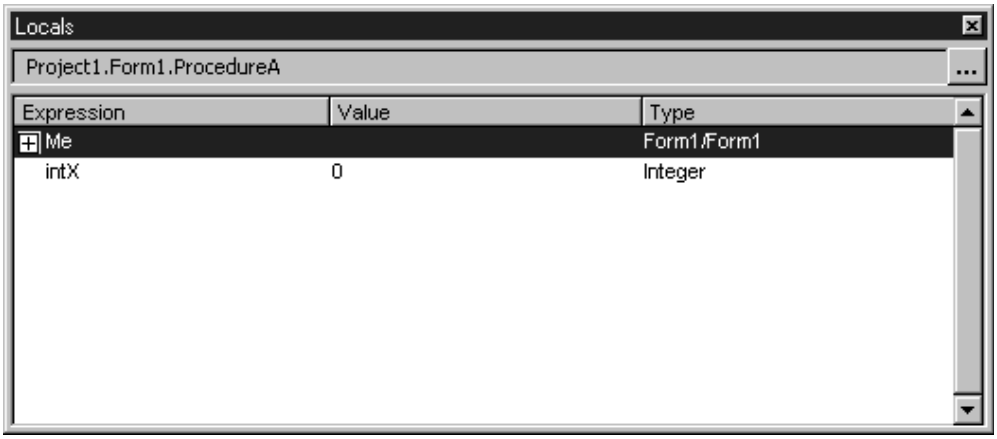
Slika 13.9 Nadgledni prozor



Za više informacija Kako bi naučili više o nadglednom prozoru, pogledajte "Nadziranje podataka nadglednim izrazima" kasnije u ovom poglavlju.

- *Prozor s lokalnim varijablama* pokazuje vrijednost svake varijable unutar područja trenutnog potprograma. Kako se izvođenje prebacuje s potprograma na potprogram, sadržaj prozora s lokalnim varijablama se mijenja kako bi odražavao samo varijable primjenjive za trenutan potprogram. Kako bi pristupili prozoru s lokalnim varijablama, odaberite Locals Window u izborniku View. Slika 13.10 prikazuje prozor s lokalnim varijablama.

Slika 13.10 Prozor s lokalnim varijablama



Trenutan potprogram i forma (ili modul) određuju koje će varijable biti prikazane prema pravilima područja predstavljenim u odlomku "Razumijevanje područja varijabli" u 5. poglavlju "Osnove programiranja". Na primjer, pretpostavimo da prozor za neposredan upis naredbi naznačuje da je forma Form1 trenutna forma. U tom slučaju, možete prikazati svaku od varijabli na razini forme u formi Form1. Možete također upotrijebiti naredbu Debug.Print za pregledavanje lokalnih varijabli potprograma prikazanih u kodnom prozoru (uvijek možete ispitati vrijednost javne varijable). Za više informacija o ispisu podataka u prozor za neposredan upis naredbi, pogledajte "Ispitivanje podataka i potprograma prozorom za neposredan upis naredbi", kasnije u ovom poglavlju.

Korištenje moda prekida

Tijekom izrade, možete promijeniti izgled ili kod svoje aplikacije, ali ne možete vidjeti kako vaše promjene utječu na način izvođenja aplikacije. Tijekom izvođenja, možete promatrati kako se aplikacija ponaša, ali ne možete izravno mijenjati kod.

Mod prekida zaustavlja operacije aplikacije i daje vam brzu snimku njezinog stanja u bilo kojem trenutku. Varijable i postavke svojstava su sčuvane, tako da možete analizirati trenutno stanje aplikacije i unijeti promjene koje utječu na način izvođenja aplikacije. Kad je aplikacija u modu prekida, možete:

- Mijenjati kod u aplikaciji.
- Promatrati stanje sučelja aplikacije.
- Nadzirati vrijednosti varijabli, svojstava i izraza.
- Mijenjati vrijednosti varijabli i svojstava.
- Vidjeti ili odrediti koju će naredbu aplikacija izvesti kao sljedeću.

- Trenutno izvesti naredbe Visual Basica.
- Ručno nadzirati operacije aplikacije.

Napomena Točke prekida i nadzorne izraze možete postaviti i tijekom izrade, ali ostali alati otkrivanja pogrešaka rade samo u modu prekida.

Ulaz u mod prekida na problematičnom izrazu

Kod otkrivanja pogrešaka, možete trebati da se aplikacija zaustavi na mjestu u kodu gdje mislite da bi problem mogao započeti. To je jedan razlog zašto Visual Basic pruža točke prekida i naredbe Stop. *Točka prekida* (*breakpoint*) određuje izraz ili skup stanja na kojima Visual Basic automatski zaustavlja izvođenje i stavlja aplikaciju u mod prekida bez izvođenja izraza koji sadržava točku prekida. Pogledajte "Korištenje naredbi Stop", kasnije u ovom poglavlju, za usporedbu naredbi Stop i točaka prekida.

Možete ući u mod prekida ručno ako učinite bilo što od sljedećeg dok se aplikacija izvodi:

- Pritisak na CTRL + BREAK.
- Odabir naredbe Break u izborniku Run.
- Klik na gumb Break na alatnoj traci.

Moguće je prekinuti izvođenje ako je aplikacija nezaposlena (kad je između obrade svojih događaja). Kad se to dogodi, izvođenje se ne zaustavlja na određenom izrazu, ali se Visual Basic svejedno prebacuje u mod prekida.

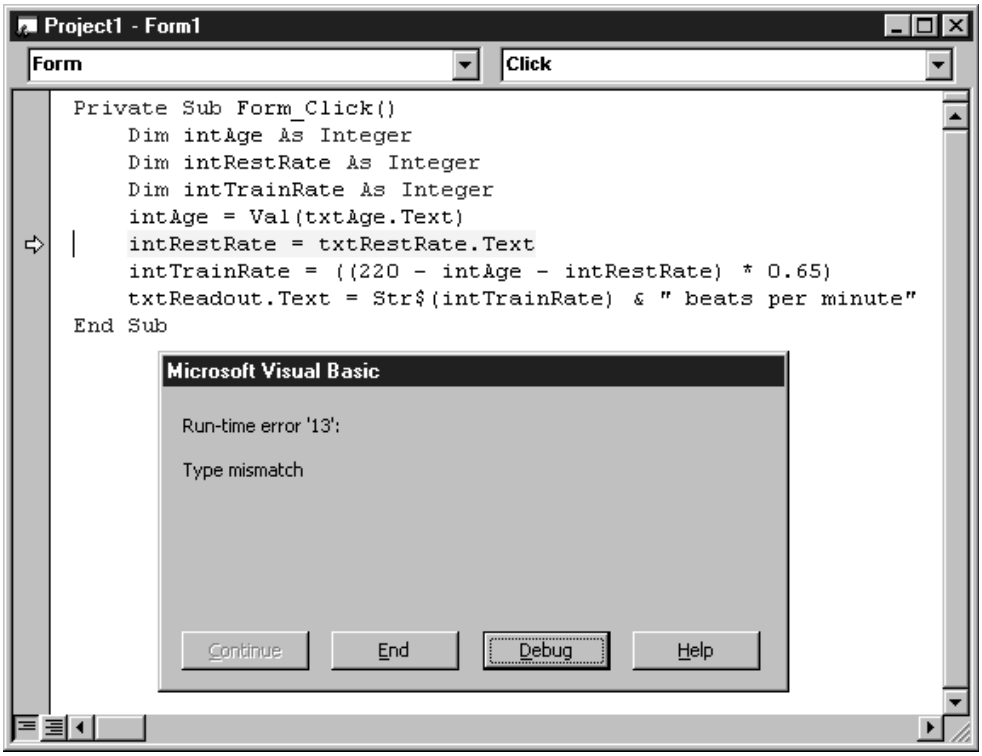
Možete također ući u mod prekida automatski ako se dogodi bilo što od sljedećeg:

- Izraz je stvorio neuhvaćenu pogrešku tijekom izvođenja.
- Izraz je stvorio pogrešku tijekom izvođenja, a opcija hvatanja pogrešaka Break on All Errors je potvrđena.
- Izraz prekida određen u dijaloškom okviru Add Watch promijenio se ili je postao istinit, ovisno o tome kako ste ga odredili.
- Izvođenje je dosegnulo liniju s točkom prekida.
- Izvođenje je dosegnulo naredbu Stop.

Ispravljanje pogreške tijeka izvođenja i nastavljanje

Neke pogreške tijeka izvođenja rezultat su jednostavnih omaški kod upisivanja koda; te pogreške su lako ispravljive. Česte pogreške uključuju pogrešno napisana imena i neodgovarajuća svojstva ili postupke s objektima – na primjer, pokušaj korištenja postupka Clear na okviru s tekstom, ili svojstva Text s okvirom s popisom datoteka. Slika 13.11 prikazuje poruku pogreške tijeka izvođenja.

Slika 13.11 Pogreške tijeka izvođenja zaustavljaju izvođenje



Često možete unijeti ispravak i nastaviti izvođenje aplikacije unutar istog izraza koji je zaustavio aplikaciju, čak i ako ste promijenili dio programskog koda. Jednostavno odaberite Continue iz izbornika Run ili kliknite gumb Continue na alatnoj traci. Kad nastavite izvođenje aplikacije, možete provjeriti je li problem ispravljen.

Ako potvrdite opciju Break on All Errors iz grupe opcija Default Error Trapping State na kartici General dijaloškog okvira Options (dostupnog iz izbornika Tools), Visual Basic onemogućuje rukovatelje pogreškama u kodu, tako da kad izraz stvori pogrešku tijekom izvođenja, Visual Basic ulazi u mod prekida. Ako opcija Break on All Errors nije potvrđena, i ako postoji rukovatelj pogreškama, on će presresti kod i poduzeti akciju ispravljanja.

Napomena Kad promijenite opcije grupe Default Error Trapping State kroz dijaloški okvir Options, te postavke postaju podrazumijevane za sve sljedeće termine rada s Visual Basicom. Kako bi promijenili rukovanje pogreškama samo za trenutan rad, odaberite Toggle u pomoćnom izborniku kodnog prozora za otvaranje podizbornika koji omogućuje odabir moda prekida.

Neke promjene (najčešće, promjena određivanja varijable ili dodavanje novih varijabli ili potprograma) traže od vas ponovno pokretanje aplikacije. Kad se to dogodi, Visual Basic predstavlja poruku koja vas pita želite li ponovno pokrenuti aplikaciju.

Nadziranje podataka nadglednim izrazima

Dok otkrivate pogreške u svojoj aplikaciji, neki proračun možda neće proizvesti rezultat koji želite ili se mogu pojaviti problemi kad određena varijabla ili svojstvo pretpostave određenu vrijednost ili opseg vrijednosti. Većina problema pri otkrivanju pogrešaka ne vodi trenutno do jednog izraza, tako da ćete trebati promatrati ponašanje varijable ili izraza kroz potprogram.

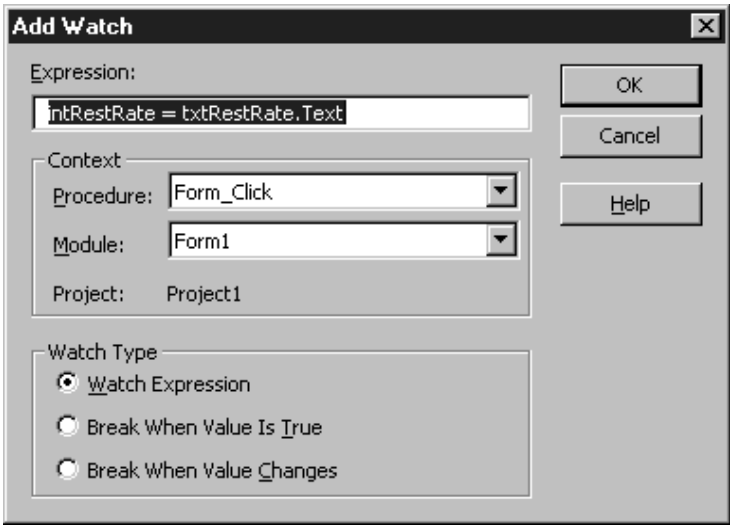
Visual Basic automatski nadzire nadgledne izraze – izraze koje vi određujete – za vas. Kad aplikacija uđe u mod prekida, ti se nadgledni izrazi pojavljuju u nadglednom prozoru, gdje možete promatrati njihove vrijednosti.

Možete također usmjeriti nadgledne izraze da postave aplikaciju u mod prekida uvijek kad se vrijednost izraza promijeni ili bude jednaka određenoj vrijednosti. Na primjer, umjesto prolaza kroz možda desetine ili stotine petlji jednog izraza, možete upotrijebiti nadgledni izraz kako bi postavili aplikaciju u mod prekida kad brojač petlje dosegne određenu vrijednost. Možda ćete trebati ući u mod prekida svaki put kad zastavica u potprogramu promijeni vrijednost.

Dodavanje nadglednog izraza

Nadgledni izraz možete dodati tijekom izrade ili u modu prekida. Upotrijebite dijaloški okvir Add Watch (prikazan na slici 13.12) za dodavanje nadglednog izraza.

Slika 13.12 Dijaloški okvir Add Watch



Sljedeća tablica opisuje dijaloški okvir Add Watch.

dio	opis
Okvir Expression	Mjesto gdje upisujete izraz koji se proračunava nadglednim izrazom. Izraz je varijabla, svojstvo, poziv funkcije, ili svaki drugi valjan izraz.
Grupa opcija Context	Određuje područje varijabli nadgledanih izrazom. Upotrijebite ove opcije ako imate varijable istog imena s različitim područjima. Možete također ograničiti područje varijabli u nadglednim izrazima na određeni potprogram, određenu formu ili modul, ili ih možete primijeniti za cijelu aplikaciju odabirom stavki All Procedures i All Modules. Visual Basic može brže proračunati varijablu u tjesnijem sklopu.
Grupa opcija Watch Type	Određuje kako Visual Basic odgovara na nadgledni izraz. Visual Basic može nadgledati izraz i prikazati njegovu vrijednost u nadglednom prozoru kad aplikacija uđe u mod prekida. Aplikacija može automatski ući u mod prekida kad izraz bude proračunat kao točan (različit od nule) izraz ili svaki put kad se promijeni vrijednost izraza.

Kako dodati nadgledni izraz

1. U izborniku **Debug** odaberite stavku **Add Watch**.
2. Trenutan izraz (ako postoji) iz kodnog prozora će se pojaviti u okviru **Expression** dijaloškog okvira **Add Watch**. Ako to nije izraz koji želite nadgledati, upišite izraz za proračunavanje u okvir **Expression**.
3. Ako je potrebno, odredite područje varijabli za nadgledanje.

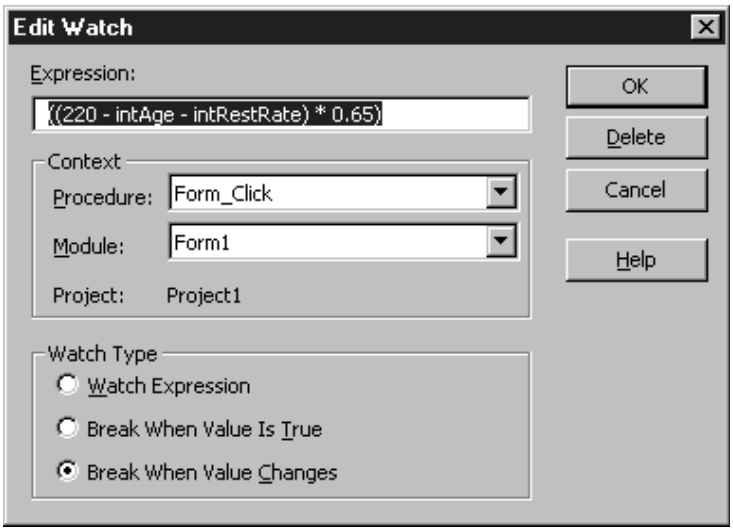
   Ako odaberete opciju **Procedure** ili **Module** u grupi **Context**, odaberite ime potprograma, forme ili modula iz odgovarajućeg okvira s popisom.
4. Ako je potrebno, odaberite gumb izbora u grupi **Watch Type** za određivanje kako želite da Visual Basic odgovori na nadgledni izraz.
5. Odaberite gumb **OK**.

   Napomena Možete također dodati izraz povlačenjem iz kodnog prozora i ispuštanjem u nadgledni prozor.

Editiranje ili brisanje nadglednog izraza

Dijaloški okvir Edit Watch, prikazan na slici 13.13, ispisuje sve trenutne nadgledne izraze. Možete editirati i obrisati svaki nadgledni izraz ispisan u nadglednom prozoru.

Slika 13.13 Dijaloški okvir Edit Watch



Kako editirati nadgledni izraz

1. U nadglednom prozoru, dvaput kliknite na nadgledni izraz kojeg želite editirati.

   - ili -

   Označite nadgledni izraz koje želite editirati i odaberite stavku **Edit Watch** u izborniku **Debug**.

2. Prikazuje se dijaloški okvir **Edit Watch** i jednak je dijaloškom okviru **Add Watch** osim drugačije naslovne trake i dodanog gumba Delete.
3. Napravite sve promjene za izraz, područje računanja varijabli ili tip nadgledanja.
4. Odaberite gumb **OK**.

Kako obrisati nadgledni izraz

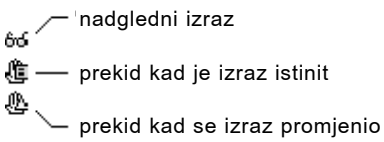
U **nadglednom prozoru** odaberite nadgledni izraz kojeg želite obrisati.

- Pritisnite tipku DELETE.

Prepoznavanje tipova nadgledavanja

Na lijevoj strani svakog nadglednog izraza u nadglednom prozoru je ikona koja označuje tip nadzora tog izraza. Slika 13.14 određuje ikonu za svaki od tri tipa nadzora.

Slika 13.14 Ikone tipa nadzora



Korištenje dijaloškog okvira Quick Watch

Dok ste u modu prekida, možete provjeriti vrijednost svojstva, varijable ili izraza za koje niste odredili nadzorni izraz. Kako bi provjerili takve izraze, upotrijebite dijaloški okvir Quick Watch, prikazan na slici 13.15.

Slika 13.15 Dijaloški okvir Quick Watch

Dijaloški okvir Quick Watch pokazuje vrijednost izraza kojeg ste odabrali u kodnom prozoru. Kako bi nastavili nadgledati taj izraz, kliknite gumb Add; bit će prikazan nadgledni prozor s već unesenim bitnim informacijama iz dijaloškog okvira Quick Watch. Ako Visual Basic ne može proračunati vrijednost trenutnog izraza, gumb Add je onemogućen.

Kako prikazati dijaloški okvir Quick Watch

1. Označite nadgledni izraz u kodnom prozoru.
2. Kliknite gumb **Quick Watch** na alatnoj traci **Debug** (kako bi prikazali alatnu traku Debug, kliknite desnom tipkom miša na alatnu traku Visual Basica i odaberite opciju **Debug**).

   - ili -

   Pritisnite SHIFT + F9.

   - ili -

   U izborniku **Debug** odaberite **Quick Watch**.
3. Ako želite dodati nadgledni izraz temeljen na izrazu u dijaloškom okviru **Quick Watch**, kliknite gumb **Add**.

Korištenje točke prekida za selektivno zaustavljanje izvođenja

Tijekom izvođenja, točka prekida kazuje Visual Basicu da se zaustavi točno prije izvođenja određene linije koda. Kad Visual Basic izvodi potprogram i otkrije liniju koda s točkom prekida, prebacuje se u mod prekida.

Možete postaviti ili maknuti točku prekida u modu prekida ili tijekom izrade, ili tijekom izvođenja kad je aplikacija nezaposlena.

Kako postaviti ili maknuti točku prekida

1. U kodnom prozoru, pomaknite točku prekida na liniju koda gdje želite postaviti ili maknuti točku prekida.

   - ili -

   Kliknite na marginu na lijevom rubu kodnog prozora pored linije gdje želite postaviti ili maknuti točku prekida.
2. U izborniku **Debug** odaberite **Toggle Breakpoint**.

   - ili -

   Kliknite gumb **Toggle Breakpoint** na alatnoj traci **Debug** (kako bi prikazali alatnu traku Debug, kliknite desnom tipkom miša na alatnu traku Visual Basica i odaberite opciju **Debug**).

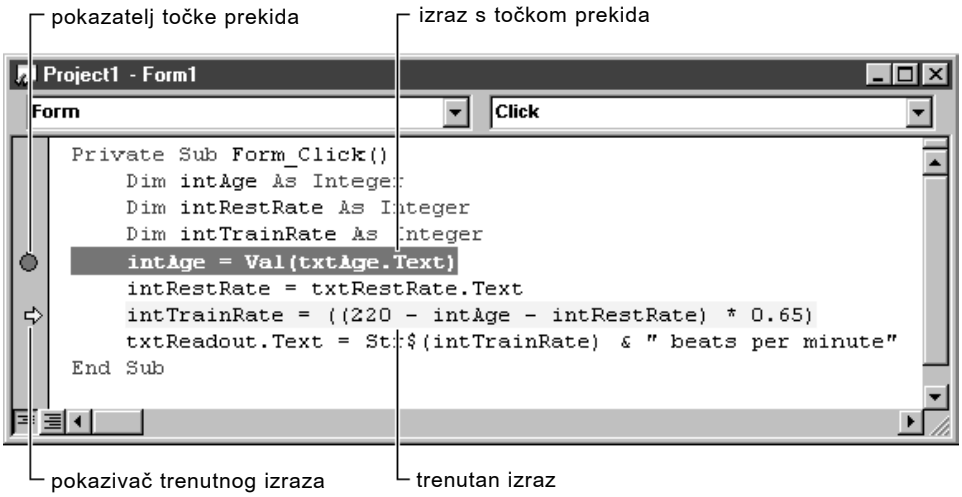
   - ili -

   Pritisnite F9.

Kad postavite točku prekida, Visual Basic osvjetljava odabranu liniju i podebljava pismo, koristeći boje koje ste odredili na kartici Editor Format dijaloškog okvira Options, dostupnog iz izbornika Tools.

Na primjer, slika 13.16 pokazuje potprogram s točkom prekida na petoj liniji. U kodnom prozoru, Visual Basic naznačuje točku prekida prikazivanjem teksta u toj liniji podebljanim pismom i bojama određenim za točku prekida.

Slika 13.16 Potprogram zaustavljen točkom prekida



Prepoznavanje trenutnog izraza

Na slici 13.16, osvijetljeni pravokutnik okružuje sedmu liniju programskog koda. Taj obris naznačuje *trenutan izraz*, ili idući izraz koji će biti izveden. Kad trenutan izraz također sadrži točku prekida, liniju koda osvjetljava samo pravokutni obris. Kad se jednom trenutan izraz pomakne na neku drugu liniju, linija s točkom prekida se ponovno prikazuje s podebljanim pismom i u boji.

Kako odrediti boju teksta trenutne linije

1. U izborniku **Tools**, odaberite **Options** i kliknite karticu **Editor Format** u dijaloškom okviru **Options**.
2. Ispod natpisa **Code Colors**, odaberite **Execution Point Text**, te odredite boje ispisa (**Foreground**), pozadine (**Background**) i pokazivača (**Indicator**).

Pregledavanje aplikacije na točki prekida

Jednom kad dosegnete točku prekida i aplikacija se zaustavi, možete pregledati trenutno stanje aplikacije. Provjera rezultata aplikacije je laka, jer možete pomicati fokus među formama i modulima svoje aplikacije, kodnom prozoru i prozorima za otkrivanje pogrešaka.

Točka prekida zaustavlja aplikaciju upravo prije izvođenja linije koja sadržava točku prekida. Ako želite promotriti što se događa kad se izvede linija s točkom prekida, morate izvesti još najmanje jedan izraz. Kako bi to napravili, upotrijebite naredbe Step Into ili Step Over.

Napomena Iako je moguće postaviti točku prekida u potprogramu događaja MouseMove ili u događaju Timer, to može prouzročiti neočekivane rezultate. Normalan tijek događaja je prekinut kod ulaza u mod prekida; prolaz korak po korak kroz kod u ovim potprogramima može pokazati drugačije ponašanje od onog koje će se pojaviti u modu izvođenja.

Za više informacija Pogledajte odlomak "Izvođenje odabranih dijelova vaše aplikacije", kasnije u ovom poglavlju.

Kad pokušavate izdvojiti problem, zapamtite da izraz može biti posredno stvoriti pogrešku jer dodjeljuje neispravnu vrijednost varijabli. Kako bi pregledali vrijednosti varijabli i svojstava dok ste u modu prekida, upotrijebite prozor s lokalnim varijablama, dijaloški okvir Quick Watch, nadgledne izraze ili prozor za neposredan upis naredbi.

Za više informacija Da bi naučili kako koristiti prozor za neposredan upis naredbi za ispitivanje vrijednosti svojstava i varijabli, pogledajte "Ispitivanje podataka i potprograma prozorom za neposredan upis naredbi", kasnije u ovom poglavlju. Kako bi naučili više o nadglednim izrazima, pogledajte "Nadziranje podataka nadglednim izrazima", ranije u ovom poglavlju.

Korištenje naredbi Stop

Postavljanje naredbe Stop u potprogram je alternativa postavljanu točke prekida. Uvijek kad Visual Basic otkrije naredbu Stop, zaustavlja izvođenje i prebacuje se u mod prekida. Iako naredbe Stop djeluju kao točke prekida, ne postavljaju se i ne miču na isti način.

Oprez Obavezno uklonite sve naredbe Stop prije stvaranja izvršne datoteke. Ako samostalna aplikacija Visual Basica (.exe) otkrije naredbu Stop, obrađuje ju kao naredbu End i trenutno završava izvođenje, bez pojavljivanja bilo kojih događaja QueryUnload ili Unload.

Napomena Obično je bolje koristiti postupak Assert od naredbe Stop. Postupak Assert zaustavlja izvođenje samo kad određeni uvjet nije zadovoljen; za razliku od naredbe Stop, pozivi postupka Assert se automatski miču kad se prevodi aplikacija. Za više informacija, pogledajte odlomak "Provjera vašeg koda zahtjevima", kasnije u ovom poglavlju.

Zapamtite da naredba Stop ne radi ništa više osim privremenog zaustavljanja izvođesnja, dok naredba End zaustavlja izvođenje, ponovno postavlja varijable, i vraća u vrijeme izrade. Uvijek možete odabrati naredbu Continue iz izbornika Run za nastavak izvođenja aplikacije.

Za više informacija Pogledajte "Kako rukovati pogreškama", ranije u ovom poglavlju, za primjer koji koristi naredbu Stop.

Izvođenje odabranih dijelova vaše aplikacije

Ako možete prepoznati izraz koji je uzrokovao pogrešku, jedna točka prekida može vam pomoći da pronađete problem. Međutim, češće ćete znati samo opće područje koda koje je uzrokovalo pogrešku. Točka prekida vam pomaže da izdvojite to problematično područje. Nakon toga možete upotrijebiti naredbe Step Into i Step Over kako bi promotrili učinak svakog izraza. Ako je potrebno, možete također preskočiti izraze ili se vratiti pokretanjem izvođenja na novoj liniji.

mod koraka	opis
Step Into	Izvodi trenutan izraz i prekida na sljedećoj liniji, čak i ako je ona u drugom potprogramu.
Step Over	Izvodi cijeli potprogram pozvan trenutnom linijom i prekida na liniji koja je iza trenutne linije.
Step Out	Izvodi ostatak trenutnog potprograma i prekida na izrazu koji se nalazi iza onog koji je pozvao potprogram.

Napomena Morate biti u modu prekida kako bi mogli upotrijebiti ove naredbe. One nisu dostupne tijekom izrade ili izvođenja.

Korištenje naredbe Step Into

Možete upotrijebiti naredbu Step Into za izvođenje koda jedan po jedan izraz (to je također znano kao korak po korak). Nakon takvog prolaza kroz svaki izraz, možete vidjeti njihove učinke pogledom na forme vaše aplikacije ili u prozore za otkrivanje pogrešaka.

Kako prolaziti kroz kod jedan po jedan izraz

- U izborniku **Debug**, odaberite **Step Into**.

  - ili -

  Kliknite gumb **Step Into** na alatnoj traci **Debug** (kako bi prikazali alatnu traku Debug, kliknite desnom tipkom miša na alatnu traku Visual Basica i odaberite opciju **Debug**).

  - ili -

  Pritisnite F8.

Kad upotrijebite naredbu Step Into za prolaz kroz kod jedan po jedan izraz, Visual Basic se privremeno prebacuje na vrijeme izvođenja, izvodi trenutan izraz, te prelazi na idući izraz. Zatim se prebacuje natrag u mod prekida.

Napomena Visual Basic vam omogućuje prolaz kroz pojedine izraze, čak i ako su u istoj liniji. Linija koda može sadržavati dva ili više izraza, razdvojene dvotočkom (:). Visual Basic koristi pravokutni obris za naznačivanje koji će od izraza biti izveden kao sljedeći. Točke prekida vrijede samo za prvi izraz u liniji s više izraza.

Korištenje naredbe Step Over

Naredba Step Over je jednaka naredbi Step Into, osim kad trenutan izraz sadrži poziv potprograma. Za razliku od naredbe Step Into, koja ulazi u potprogram koji se poziva, naredba Step Over izvodi ga kao cjelinu i zatim prelazi na idući izraz u trenutnom potprogramu. Pretpostavimo, na primjer, da izraz poziva potprogram PostaviAlarm:

```
PostaviAlarm 11, 30, 0
```

Ako odaberete naredbu Step Into, kodni prozor pokazuje potprogram PostaviAlarm i postavlja trenutan izraz na početak tog potprograma. To je bolji izbor samo ako želite analizirati kod unutar potprograma PostaviAlarm.

Ako upotrijebite naredbu Step Over, kodni prozor nastavlja prikazivati trenutan potprogram. Izvođenje napreduje na naredbu odmah iza poziva potprograma PostaviAlarm, osim ako potprogram PostaviAlarm sadrži točku prekida ili naredbu Stop. Upotrijebite naredbu Step Over ako želite ostati na istoj razini koda i ne trebate analizirati potprogram PostaviAlarm.

Možete slobodno mijenjati između naredbi Step Onto i Step Over. Naredba koju ćete upotrijebiti ovisi o tome koje dijelove koda želite analizirati u određeno vrijeme.

Kako koristiti naredbu Step Over

- U izborniku Debug, odaberite Step Over.

  - ili -

  Kliknite gumb **Step Over** na alatnoj traci **Debug** (kako bi prikazali alatnu traku Debug, kliknite desnom tipkom miša na alatnu traku Visual Basica i odaberite opciju **Debug**).

  - ili -

  Pritisnite SHIFT + F8.

Korištenje naredbe Step Out

Naredba Step Out je slična naredbama Step Into i Step Over, osim što napreduje kroz ostatak koda u trenutnom potprogramu. Ako je potprogram pozvan iz drugog potprograma, ova naredba napreduje do izraza odmah iza onog koji je pozvao potprogram.

Kako koristiti naredbu Step Out

- U izborniku **Debug**, odaberite **Step Out**.

  - ili -

  Kliknite gumb **Step Out** na alatnoj traci **Debug** (kako bi prikazali alatnu traku Debug, kliknite desnom tipkom miša na alatnu traku Visual Basica i odaberite opciju **Debug**).

  - ili -

  Pritisnite CTRL + SHIFT + F8.

Zaobilaženje dijelova koda

Kad je vaša aplikacija u modu prekida, možete upotrijebiti naredbu Run To Cursor kako bi odabrali izraz dalje prema dolje u vašem kodu gdje želite zaustaviti izvođenje. To vam omogućuje "prekoračivanje" nezanimljivih dijelova koda, kao što su velike petlje.

Kako koristiti naredbu Run To Cursor

1. Postavite svoju aplikaciju u mod prekida.
2. Postavite pokazivač tamo gdje želite zaustaviti izvođenje.
3. Pritisnite CTRL + F8.

   - ili -

   U izborniku **Debug**, odaberite **Run To Cursor**.

Određivanje idućeg izraza koji će biti izveden

Dok otkrivate pogreške ili eksperimentirate s vašom aplikacijom, možete upotrijebiti naredbu Set Next Statement kako bi preskočili određen dio koda – na primjer, dio koji sadrži poznatu pogrešku – tako da možete nastaviti s praćenjem ostalih problema. Možete se također vratiti na raniji izraz kako bi ispitali dio aplikacije koristeći drugačije vrijednosti za svojstva ili varijable.

S Visual Basicom, možete odrediti drukčiju liniju koda koja će se izvesti kao iduća, pod uvjetom da spada u isti potprogram. Učinak je sličan korištenju naredbe Step Into, osim što naredba Step Into izvodi samo iduću liniju koda u potprogramu. Određivanjem iduće linije izvođenja, odabirete koja će se linija izvesti sljedeća.

Kako odrediti iduću liniju koja će biti izvedena

1. U modu prekida pomaknite točku ubacivanja (pokazivač) na liniju koda za koju želite da se izvede kao sljedeća.
2. U izborniku **Debug** odaberite **Set Next Statement**.
3. Za nastavak izvođenja, u izborniku **Run**, odaberite **Continue**.

   - ili -

   U izborniku **Debug** odaberite **Run To Cursor**, **Step Into**, **Step Over** ili **Step Out**.

Pokazivanje idućeg izraza koji će biti izveden

Naredbu Show Next Statement možete upotrijebiti za postavljanje pokazivača na liniju koja će se izvesti kao sljedeća. Ova osobina je prikladna ako ste izvodili kod u rukovatelju pogreškom i niste sigurni gdje će se izvođenje nastaviti. Naredba Show Next Statement dostupna je samo u modu prekida.

Kako prikazati iduću liniju koja će biti izvedena

1. Dok ste u modu prekida, u izborniku **Debug**, odaberite **Show Next Statement**.
2. Za nastavljanje izvođenja, u izborniku **Run**, odaberite **Continue**.

   - ili -

   U izborniku **Debug**, odaberite **Run To Cursor**, **Step Into**, **Step Over** ili **Step Out**.

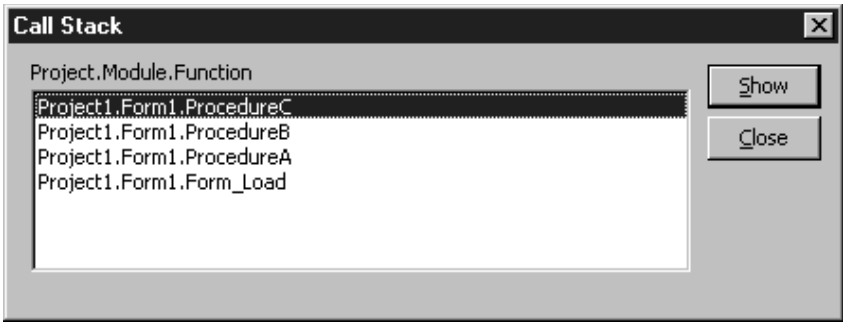
Nadziranje stoga poziva

Dijaloški okvir Call Stack prikazuje popis svih aktivnih poziva potprograma. *Aktivni pozivi potprograma* su potprogrami u aplikaciji koji su pokrenuti, ali nisu završeni.

Dijaloški okvir Call Stack pomaže vam u praćenju operacije aplikacije dok se izvodi kao niz ugniježđenih potprograma. Na primjer, potprogram događaja može pozvati drugi potprogram, koji može pozvati treći potprogram – sve prije nego što se završi potprogram događaja koji je pokrenuo taj lanac. Pozive takvih ugniježđenih potprograma može biti teško pratiti i mogu zamrsiti postupak otkrivanja pogrešaka. Slika 13.17 prikazuje dijaloški okvir Call Stack.

Napomena ako postavite aplikaciju u mod prekida tijekom petlje nezaposlenosti, u dijaloškom okviru Call Stack neće se pojaviti nijedan unos.

Slika 13.17 Dijaloški okvir Call Stack



Dijaloški okvir Call Stack možete prikazati samo kad je aplikacija u modu prekida.

Kako prikazati dijaloški okvir Call Stack

- U izborniku **View** odaberite **Call Stack**.

  - ili -

  Kliknite gumb **Call Stack** na alatnoj traci **Debug** (kako bi prikazali alatnu traku Debug, kliknite desnom tipkom miša na alatnu traku Visual Basica i odaberite opciju **Debug**).

  - ili -

  Pritisnite CTRL + L.

  - ili -

  Kliknite gumb pored okvira Procedure u **prozoru s lokalnim varijablama**.

Praćenje ugniježđenih potprograma

Dijaloški okvir Call Stack ispisuje sve aktivne pozive potprograma u nizu ugniježđenih poziva. On postavlja najraniji aktivni poziv potprograma na dno popisa i dodaje sljedeće pozive potprograma prema vrhu popisa.

Informacija dana za svaki potprogram počinje imenom modula ili forme, iza kojeg slijedi ime pozivanog potprograma. Budući da dijaloški okvir Call Stack ne naznačuje varijablu dodijeljenu primjeru forme, ne razlikuje višestruke primjere formi ili klasa. Za više informacija o višestrukim primjerima forme, pogledajte 9. poglavlje “Programiranje objektima” i odlomak “Aplikacije s više dokumenata (MDI)” u 6. poglavlju “Stvaranje korisničkog sučelja”.

Dijaloški okvir Call Stack možete upotrijebiti za prikazivanje izraza u potprogramu koji prosljeđuje nadzor nad aplikacijom sljedećem potprogramu u popisu.

Kako prikazati izraz koji poziva drugi potprogram u dijaloškom okviru Call Stack

1. U dijaloškom okviru **Call Stack**, odaberite poziv potprograma kojeg želite prikazati.
2. Odaberite gumb **Show**.

Dijaloški okvir se zatvara i u kodnom prozoru se pojavljuje potprogram.

Položaj pokazivača u kodnom prozoru naznačuje izraz koji poziva sljedeći potprogram u dijaloškom okviru Call Stack. Ako odaberete trenutan potprogram u dijaloškom okviru Call Stack, pokazivač će se pojaviti na trenutnom izrazu.

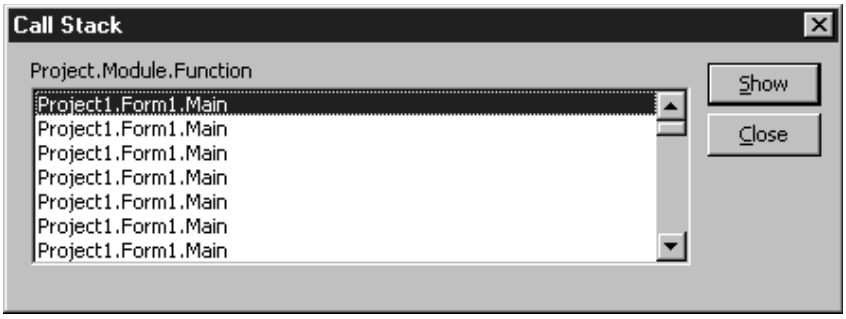
Provjera rekurzivnih potprograma

Dijaloški okvir Call Stack može biti koristan u otkrivanju jesu li pogreške “Nedovoljno prostora za stog” (Out of stack space) uzrokovane rekurzijom. *Rekurzija* je sposobnost rutine da poziva sama sebe. Možete to ispitati dodavanjem sljedećeg koda formi u novom projektu:

```
Sub Main()
  Static intX As Integer
  intX = intX + 1
  Main
End Sub

Private Sub Form_Click()
  Main
End Sub
```

Pokrenite aplikaciju, kliknite formu, i čekajte na poruku pogreške "Nedovoljno prostora za stog". Odaberite gumb Debug, te zatim odaberite naredbu Call Stack u izborniku View. Vidjet ćete višestruke pozive potprograma Main, kao što je prikazano na slici 13.18.

Slika 13.18 Dijaloški okvir Call Stack ispisuje rekurzivni potprogram



Za dvostruku provjeru, osvijetlite intX u kodnom prozoru, te odaberite naredbu Quick Watch u izborniku Debug. Vrijednost varijable intX je broj koji pokazuje koliko se puta izveo potprogram Main prije prekida.

Ispitivanje podataka i potprograma prozorom za neposredan upis naredbi

Ponekad kad otkrivate pogreške ili eksperimentirate s aplikacijom, trebat ćete izvesti pojedine potprograme, proračunati izraze, ili dodijeliti nove vrijednosti varijablama ili svojstvima. Za ostvarivanje tih zadataka možete upotrijebiti prozor za neposredan upis naredbi. Izraze proračunavate ispisom njihovih vrijednosti u tom prozoru.

Ispis informacija u prozoru za neposredan upis naredbi

Postoje dva načina ispisa u prozor za neposredan upis naredbi:

- Uključivanje izraza Debug.Print u kodu aplikacije.
- Upis postupaka Print izravno u prozor za neposredan upis naredbi.

Ove tehnike ispisivanja nude nekoliko prednosti nad nadglednim izrazima:

- Ne trebate prekinuti izvođenje kako bi dobili povratne informacije o izvođenju aplikacije. Možete vidjeti prikazane podatke ili druge poruke dok izvodite aplikaciju.
- Povratna informacija se ispisuje u odvojenom području (prozoru za neposredan upis naredbi), tako da se ne miješa s izlaznim rezultatima koje vidi korisnik.
- Budući da možete snimiti taj kod kao dio forme, ne trebate ponovno određivati te izraze idući put kad budete radili s aplikacijom.

Ispis iz koda aplikacije

Postupak Print šalje izlazne rezultate u prozor za neposredan upis naredbi uvijek kad uključite prefiks s objektom Debug:

Debug.Print [*stavke*][;]

Na primjer, sljedeći izraz ispisuje vrijednost varijable Plaća u prozor za neposredan upis naredbi svaki put kad se izvede:

```
Debug.Print "Plaća = "; Plaća
```

Ova tehnika radi najbolje kad postoji posebno mjesto u kodu vaše aplikacije na kojem je poznato da varijabla (u ovom slučaju, Plaća) mijenja vrijednost. Na primjer, možete postaviti prethodni izraz u petlju koja stalno mijenja vrijednost varijable Plaća.

Napomena Kad prevodite svoju aplikaciju u datoteku tipa .exe, izrazi Debug.Print se uklanjaju. Stoga, ako vaša aplikacija samo koristi izraze Debug.Print s stringovima ili jednostavnim tipovima varijabli kao argumentima, neće imati nijedan izraz Debug.Print. Međutim, Visual Basic neće ukloniti pozive funkcija koji se pojavljuju kao argumenti u izrazu Debug.Print. Prema tome, svi popratni efekti tih funkcija će se i dalje događati u prevedenoj .exe datoteci, iako se rezultati tih funkcija neće ispisivati.

Za više informacija Pogledajte "Objekt Debug" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Ispis iz prozora za neposredan upis naredbi

Kad uđete u mod prekida, možete pomaknuti fokus na prozor za neposredan upis naredbi kako bi pregledali podatke.

Kako pregledati podatke u prozoru za neposredan upis naredbi

1. Kliknite prozor za neposredan upis naredbi (ako je vidljiv).

   - ili -

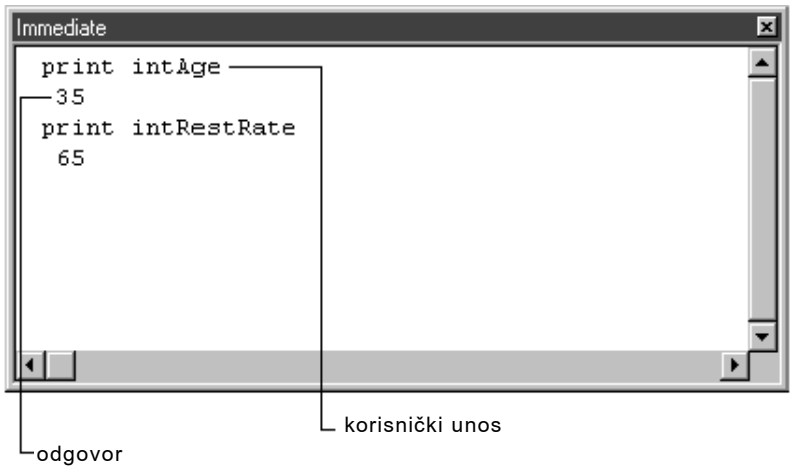
   U izborniku **View** odaberite **Immediate Window**.

   Kad ste pomaknuli fokus na prozor za neposredan upis naredbi, možete upotrijebiti postupak Print bez objekta Debug.

2. Upišite ili ulijepite izraz u prozor za neposredan upis naredbi, pa pritisnite ENTER.

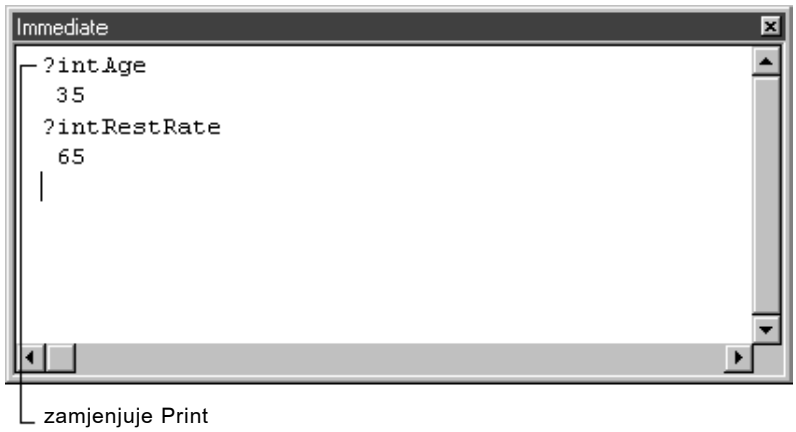
   Prozor za neposredan upis naredbi odgovara izvođenjem izraza, kao što je prikazano na slici 13.19.

Slika 13.19 Korištenje postupka Print za ispis u prozoru za neposredan upis naredbi



Upitnik (?) je korisna kratica za postupak Print. Upitnik znači isto što i riječ Print, i može biti upotrijebljen u svakom sklopu gdje se koristi riječ Print. Na primjer, izrazi na slici 13.19 mogli bi biti upisani kako je prikazano na slici 13.20.

Slika 13.20 Korištenje upitnika umjesto postupka Print



Ispis vrijednosti svojstava

Možete proračunati svaki valjan izraz u prozoru za neposredan upis naredbi, uključujući izraze koji sadržavaju svojstva. Trenutno aktivna forma ili modul određuju područje. Ako se izvođenje zaustavi unutar koda koji je dodan formi ili klasi, možete ukazati na svojstva te forme (ili neke od njezinih kontrola) te napraviti izričit pokazivač na formu izrazima poput sljedećih:

```
? BackColor
? Text1.Height
```

Prvi izraz ispisuje brojčanu vrijednost trenutne boje pozadine forme u prozor za neposredan upis naredbi. Uz pretpostavku da je Text1 kontrola na trenutno aktivnoj formi, drugi izraz ispisuje visinu kontrole Text1.

Ako je izvođenje zaustavljeno u modulu ili drugoj formi, morate izričito odrediti ime forme kako slijedi:

```
? Form1.BackColor
? Form1.Text1.Height
```

Napomena Upućivanje na neučitanu formu u prozoru za neposredan upis naredbi (ili svagdje drugdje) učitava tu formu.

Za više informacija Kako bi naučili o promjenama svojstava i vrijednosti u prozoru za neposredan upis naredbi, pogledajte sljedeći odlomak “Dodjela vrijednosti varijablama i svojstvima”.

Dodjela vrijednosti varijablama i svojstvima

Kad počnete izdvajati mogući uzrok pogreške, možete poželjeti ispitati učinke vrijednosti određenih podataka. Kad ste u modu prekida, u prozoru za neposredan upis naredbi možete postaviti vrijednosti izrazima poput sljedećih:

```
BackColor = 255
VScroll1.Value = 100
MaxRedova = 50
```

Prvi izraz mijenja svojstvo trenutno aktivne forme, drugi mijenja svojstvo `VScroll1`, a treći dodjeljuje vrijednost varijabli.

Nakon što postavite vrijednosti za jedno ili više svojstava i varijabli, možete nastaviti izvođenje kako bi vidjeli rezultate. Također možete ispitati učinke na potprogramima, kao što je opisano u sljedećem odlomku “Ispitivanje podataka s prozorom za neposredan upis naredbi”.

Ispitivanje podataka s prozorom za neposredan upis naredbi

Prozor za neposredan upis naredbi proračunava svaki valjani izvodivi izraz Visual Basica, ali ne prihvaća određivanje podataka. Međutim, možete unijeti pozive potprograma tipa Sub ili Function, što vam omogućuje ispitivanje mogućeg učinka potprograma s bilo kojim danim skupom argumenata. Jednostavno upišite izraz u prozor za neposredan upis naredbi (dok ste u modu prekida) kao što bi ga upisali u kodni prozor. Na primjer:

```
X = Kvadrat(2, 8, 8)
PrilažiGrafikon 50, Arr1
Form_MouseDown 1, 0, 100, 100
```

Kad pritisnete tipku ENTER, Visual Basic se prebacuje u vrijeme izvođenja kako bi izveo izraz, i zatim se vraća u mod prekida. U tom trenutku, možete vidjeti rezultate i ispitati sve moguće učinke na varijablama ili vrijednostima svojstava.

Za pozive potprograma primjenjuje se područje djelovanja kao i za varijable. Možete pozvati svaki potprogram unutar trenutno aktivne forme. Uvijek možete pozvati potprogram u modulu, osim ako ga niste odredili kao privatnog (Private), u tom slučaju možete pozvati taj potprogram samo dok se izvodi modul.

Za više informacija Područje djelovanja je raspravljeno u odlomku "Uvod u varijable, konstante i tipove podataka" u 5. poglavlju "Osnove programiranja".

Pregled i ispitivanje višestrukih primjera potprograma

Prozor za neposredan upis naredbi možete upotrijebiti za ponavljano izvođenje potprograma, ispitujući učinak raznih uvjeta. Svaki zasebni poziv potprograma održava se kao odvojen primjer u Visual Basicu. To vam omogućuje odvojeno ispitivanje varijabli i postavki svojstava u svakom primjeru potprograma. Da bi vidjeli kako to radi, otvorite novi projekt i dodajte sljedeći kod u modul forme:

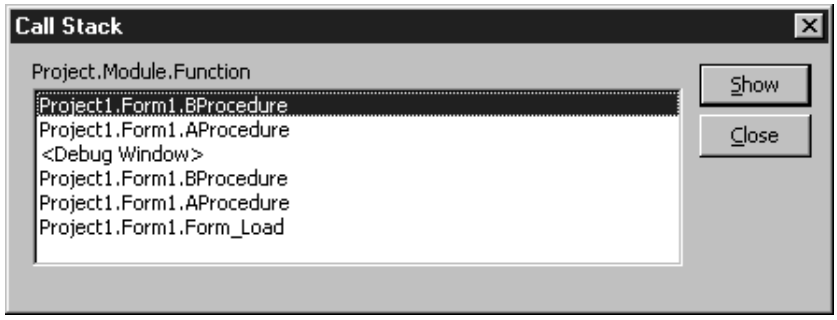
```
Private Sub Form_Click()
  AProcedure
End Sub

Sub AProcedure()
  Dim intX As Integer
  intX = 10
  BProcedure
End Sub

Sub BProcedure()
  Stop
End Sub
```

Pokrenite aplikaciju i kliknite formu. Naredba Stop postavlja Visual Basic u mod prekida i prikazuje se prozor za neposredan upis naredbi. Promijenite vrijednost varijable intX na 15 u potprogramu "AProcedure", prebacite se na prozor za neposredan upis naredbi, i upišite sljedeće:

```
AProcedure
```

To poziva potprogram "AProcedure" i ponovno pokreće aplikaciju. Ako se prebacite na prozor za neposredan upis naredbi i ponovno pokrenete potprogram "AProcedure", te zatim otvorite dijaloški okvir Call Stack, vidjet ćete ispis sličan onom na slici 13.21. Ispisano je svako zasebno pokretanje aplikacije, odvojeno ispisom `[<Debug Window>]`.

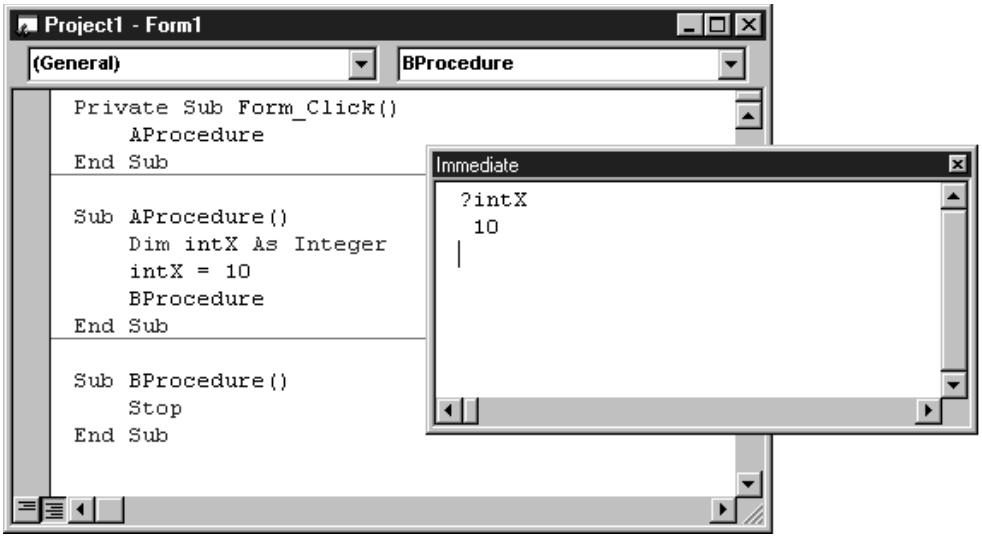
Slika 13.21 Dijaloški okvir Call Stack prikazuje višestruke primjere potprograma



Visual Basic održava ispis potprograma izvedenih svakom naredbom iz prozora za neposredan upis naredbi. Noviji ispisi su na vrhu popisa. Dijaloški okvir Call Stack možete upotrijebiti za odabir svakog primjera potprograma, te zatim ispisati vrijednosti varijabli iz tog potprograma u prozor za neposredan upis naredbi.

Na primjer, ako dvaput kliknete na najraniji primjer potprograma "AProcedure" i upotrijebite prozor za neposredan upis naredbi kako bi ispisali vrijednost varijable intX, on će vratiti vrijednost 10, kao što je prikazano na slici 13.22. Ako promijenite vrijednost varijable intX na 15 za iduće izvođenje potprograma "AProcedure", ta vrijednost se sprema s drugim primjerom potprograma.

Slika 13.22 Ispis vrijednosti varijabli u prozoru za neposredan upis naredbi



Napomena Iako prozor za neposredan upis naredbi podržava većinu izraza, struktura kontrola je valjana samo ako može biti u potpunosti izražena jednom linijom koda; upotrijebite dvotočke za razdvajanje izraza koji čine strukturu kontrole. Sljedeća petlja For je valjana u prozoru za neposredan upis naredbi:

```
For I = 1 To 20: Print 2 * I: Next I
```

Provjera brojeva pogrešaka

Prozor za neposredan upis naredbi možete upotrijebiti za prikazivanje poruka povezanih s određenim brojem pogreške. Na primjer, upišite ovaj izraz u prozor za neposredan upis naredbi:

```
error 58
```

Pritisnite ENTER za izvođenje izraza. Prikazat će se odgovarajuća poruka pogreške, kao što je prikazano na slici 13.23.

Slika 13.23 Prikaz poruke pogreške iz prozora za neposredan upis naredbi

Savjeti za korištenje prozora za neposredan upis naredbi

Ovdje su neke prečice koje možete upotrijebiti u prozoru za neposredan upis naredbi:

- Možete upotrijebiti osobinu Savjeti o podacima (Data Tips) za ispitivanje vrijednosti varijable ili objekta. Savjeti o podacima su slični plutajućim savjetima (ToolTips) osim što prikazuju trenutnu vrijednost kad se pokazivač miša drži iznad varijable ili svojstva objekta u kodnom prozoru tijekom moda prekida. Prikaz savjeta o podacima je ograničen na varijable i objekte koji su trenutno u području.
- Savjeti o podacima su također dostupni u prozoru za neposredan upis naredbi tijekom moda prekida. Za razliku od editora koda, prozor za neposredan upis naredbi će prikazati vrijednosti za svojstva objekta neovisno o području ako je pruženo potpuno označeno ime objekta. Na primjer, savjet o podacima će uvijek biti prikazan za izraz Form1.Text1.Width, ali neće za izraz Text1.Width osim ako je kontrola Text1 trenutno nije u području.
- Kad jednom upišete izraz, možete ga ponovno izvesti pomicanjem točke ubacivanja natrag na taj izraz i pritiskom na tipku ENTER bilo gdje na toj liniji.
- Prije pritiska tipke ENTER, možete editirati trenutan izraz kako bi promijenili njegov učinak.
- Možete upotrijebiti miša ili kursorske tipke za pomicanje u prozoru za neposredan upis naredbi. Ne pritiskajte tipku ENTER osim ako niste na izrazu kojeg želite izvesti.
- Kombinacija tipki CTRL + HOME će vas odvesti na vrh prozora za neposredan upis naredbi; kombinacija tipki CTRL + END će vas odvesti na njegovo dno.
- Tipke HOME i END pomiču pokazivač na početak i kraj trenutne linije.

  Napomena Iako ćete uglavnom upotrijebljavati prozor za neposredan upis naredbi u modu prekida, također je moguće upotrijebljavati ga u modu oblikovanja. Ta sposobnost je pružena kako bi se mogle otkrivati pogreške u ActiveX sastavnim dijelovima unutar razvojnog okruženja. Korištenje prozora za neposredan upis naredbi tijekom izrade za standardne projekte može uzrokovati neočekivane rezultate.

Razmatranje posebnog otkrivanja pogrešaka

Određeni događaji koji su uobičajeni dio korištenja Microsoft Windowsa mogu postaviti posebne probleme kod otkrivanja pogrešaka u aplikaciji. Važno je biti svjestan tih posebnih problema tako da oni ne zbunjuju ili kompliciraju postupak otkrivanja pogrešaka.

Ako ostanete svjesni načina kako mod prekida može nametnuti događaje koji se razlikuju od onog što očekuje vaša aplikacija, obično možete naći rješenja. U nekim potprogramima događaja, možda ćete trebati upotrijebiti izraze Debug.Print kako bi nadzirali vrijednosti varijabli i svojstava umjesto korištenja nadglednih izraza ili točaka prekida. Možda ćete također trebati promijeniti vrijednosti varijabli koje ovise o nizu događaja. To je raspravljeno u sljedećim odlomcima.

Prekid izvođenja tijekom događaja MouseDown

Ako prekinete izvođenje tijekom potprograma događaja MouseDown, možete otpustiti tipku miša ili upotrijebiti miša za obavljanje drugih zadataka. Međutim, kad nastavite izvođenje, aplikacija pretpostavlja da je tipka miša i dalje pritisnuta. Nećete dobiti događaj MouseUp sve dok ponovno ne pritisnete tipku miša i zatim ju otpustite.

Kad pritisnete tipku miša tijekom izvođenja, ponovno ćete prekinuti izvođenje u potprogramu događaja MouseDown, uz pretpostavku da tu imate točku prekida. Na taj način, nikad nećete dobiti događaj MouseUp. Obično je rješenje micanje točke prekida iz potprograma MouseDown.

Prekid izvođenja tijekom događaja KeyDown

Ako prekinete izvođenje tijekom potprograma KeyDown, primjenjuju se slična razmišljanja. Ako potprogram KeyDown sadrži točku prekida, možda nikad nećete dobiti događaj KeyUp (događaji KeyDown i KeyUp su opisani u 11. poglavlju “Odgovaranje na događaje miša i tipkovnice”).

Prekid izvođenja tijekom događaja GotFocus ili LostFocus

Ako prekinete izvođenje tijekom potprograma događaja GotFocus ili LostFocus, vremensko usklađivanje sistemskih poruka može uzrokovati besmislene rezultate. Upotrijebite izraz Debug.Print umjesto točke prekida u potprogramima događaja GotFocus ili LostFocus.

Modalni dijalozi i poruke pogrešaka obustavljaju događaje

Razvojno okruženje ne može izazvati događaje dok je prikazana modalna forma ili okvir s porukom, jer u dijelu za otkrivanje pogrešaka postoje mogući sukobi. Zbog toga, događaji su obustavljeni sve dok se ne otpusti modalna forma ili okvir s porukom.

Važno Obustavljanje događaja pojavljuje se samo u razvojnom okruženju. Kad je projekt preveden događaji će biti izazvani čak i ako je prikazana modalna forma ili okvir s porukom.

Ovo su neki primjeri u kojim se to može pojaviti:

- Forma s kontrolom mjerača vremena se izvodi u razvojnom okruženju. Odabir stavke Options iz izbornika Tools će otvoriti dijaloški okvir Options, koji je modalan. Sve dok se taj dijalog ne otpusti, događaj Timer kontrole mjerača vremena neće biti izazvan.
- Primjer korisničke kontrole s kontrolom mjerača vremena na njoj je postavljen na formu tijekom izrade (mjerač vremena se može koristiti za stvaranje animirane kontrole; taj učinak se može pojaviti čak i u modu izrade, budući da kontrole mogu izvoditi kod tijekom izrade). Odabir stavke Add Class Module iz izbornika Project će otvoriti dijalog Add Class Module, koji je modalan. Događaj Timer kontrole mjerača vremena biti će obustavljen sve dok se ne otpusti dijalog.
- Korisnički dokument sadrži kontrolu mjerača vremena, i naredbeni gumb koji prikazuje okvir s porukom. Ako se pogreške u korisničkom dokumentu otkrivaju korištenjem Internet Explorera, pritisak naredbenog gumba za prikaz okvira s porukom će uzrokovati obustavljanje događaja Timer kontrole mjerača vremena sve dok se ne otpusti okvir s porukom.

Ispitivanje i korištenje argumenata naredbene linije

Možete odabrati da imate aplikaciju koja koristi argumente naredbene linije, koji pružaju podatke vašoj aplikaciji kod pokretanja. Korisnik ih može unijeti odabirom naredbe Run operativne okoline, te zatim može upisati argumente nakon imena aplikacije. Argumente naredbene linije možete također upotrijebljavati kad stvarate ikonu za aplikaciju.

Na primjer, pretpostavimo da stvarate aplikaciju alarmnog sata. Jedna od tehnika postavljanja vremena alarma je dopuštanje korisniku da izravno unese odabrano vrijeme. Korisnik može unijeti sljedeći string u dijaloški okvir Run:

```
Alarm 11:00:00
```

Funkcija Command vraća sve argumente upisane nakon imena aplikacije (u ovom slučaju, ime aplikacije je Alarm). Aplikacija Alarm ima samo jedan argument, pa u kodu aplikacije možete dodijeliti taj argument izravno stringu koji sprema odabrano vrijeme:

```
VrijemeAlarma = Command
```

Ako funkcija Command vrati prazan string, nema argumenata naredbene linije. Tada aplikacija mora izravno pitati za informaciju ili odabrati podrazumijevanu akciju.

Kako bi ispitali kod koji koristi funkciju Command, možete odrediti primjer argumenata naredbene linije iz okruženja Visual Basica. Aplikacija proračunava jednostavan unos naredbene linije na isti način kao i kad korisnik upiše argument.

Kako postaviti primjer argumenata naredbene linije

1. U izborniku **Project** odaberite **Properties**.
2. Kliknite karticu **Make** u dijaloškom okviru **Project Properties**.
3. Upišite primjer argumenata u polje **Command Line Arguments** (ne upisujte ime aplikacije).
4. Odaberite **OK**.
5. Pokrenite aplikaciju.

Za više informacija Pogledajte odlomak "Funkcija Command" u priručniku *Microsoft Visual Basic 6.0 Language Reference* biblioteke *Microsoft Visual Basic 6.0 Reference Library*.

Uklanjanje informacija otkrivanja pogrešaka prije prevođenja

Ako ne želite da izrazi otkrivanja pogrešaka budu sadržani u aplikaciji koju distribuirate korisnicima, upotrijebite uvjetno prevođenje kako bi prikladno obrisali te izraze kad se upotrijebi naredba Make EXE File.

Na primjer:

```
Sub Assert(Izraz As Boolean, Por As String)
  If Not Izraz Then
    MsgBox Por
  End If
End Sub

Sub APotprogram(intX As Integer)
  #If fDebug Then
    Assert intX < 10000 and intX > 0, _
    "Argument izvan opsega"
  #End If
    ' Kod sad može primiti ispravnu vrijednost.
End Sub
```

Budući da se poziv potprograma Assert uvjetno prevodi, uključen je u izvršnu datoteku samo ako je varijabla fDebug postavljena na True. Kad prevodite distribucijsku verziju aplikacije, postavite varijablu fDebug na False. Kao rezultat, izvršna .exe datoteka će biti što je moguće manja.

Napomena Od verzije 5.0 Visual Basica, više nije potrebno stvarati vlastite potprograme tipa Assert. Izraz Debug.Assert izvodi isto djelovanje i automatski se izbacuje iz prevedenog koda. Pogledajte sljedeći odlomak "Provjera vašeg koda zahtjevima", za više informacija.

Provjera vašeg koda zahtjevima

Zahtjevi su prikladan način ispitivanja uvjeta koji trebaju postojati na određenim mjestima u vašem kodu. Razmislite o izrazu Assert kao o stvaranju pretpostavke. Ako je vaša pretpostavka točna (True), zahtjev će biti zanemaren; ako je vaša pretpostavka netočna (False), Visual Basic će vam ju pokazati.

U Visual Basicu, zahtjevi primaju formu ili postupak: postupak Assert objekta Debug. Postupak Assert prima jedan argument tipa Boolean koji navodi uvjet koji će biti proračunat. Sintaksa postupka Assert je sljedeća:

Debug.Assert(*izraz tipa Boolean*)

Izraz Debug.Assert se nikad neće pojaviti u prevedenoj aplikaciji, ali kad izvodite aplikaciju u razvojnom okruženju on će potaknuti aplikaciju da uđe u mod prekida s linijom koja sadrži označen izraz (uz pretpostavku da je izraz proračunat u False). Sljedeći primjer pokazuje izraz Debug.Assert:

```
Debug.Assert Trim(ImeKorisnika) <> "Pero Perić"
```

U ovom slučaju, ako varijabla ImeKorisnika sadrži string "Pero Perić", aplikacija će ući u mod prekida; inače će se izvođenje redovno nastaviti. Upotreba izraza Debug.Assert je slična postavljanju nadzora s potvrđenom opcijom Break When Value Is True, osim što će se izvođenje prekinuti kad je vrijednost netočna.

Korištenje prevođenja uz zahtjev

Prevođenje uz zahtjev (Compile on Demand) i prevođenje u pozadini (Background Compile) su srodne osobine koje omogućuju vašoj aplikaciji da se brže izvodi u razvojnom okruženju. Moguće je da upotreba ovih osobina sakrije pogreške prevođenja u vašem kodu sve dok ne napravite izvršnu datoteku za vaš cijeli projekt. Obje osobine su u pravilu uključene, i mogu se isključiti ili uključiti na kartici General dijaloškog okvira Options dostupnog iz izbornika Tools.

Prevođenje uz zahtjev omogućuje vašoj aplikaciji, u razvojnom okruženju, da prevede programski kod samo kad je potrebno. Kad je prevođenje uz zahtjev uključeno i odaberete naredbu Start u izborniku Run (ili pritisnete tipku F5), prevest će se samo kod potreban za pokretanje aplikacije. Nakon toga, dok primjenjujete ostatak sposobnosti aplikacije u razvojnom okruženju, prevodi se više koda kad je potrebno.

Prevođenje u pozadini omogućuje Visual Basicu tijekom izvođenja u razvojnom okruženju da nastavi prevoditi kod ako se ne događaju druge akcije.

S ovim uključenim osobinama, dio koda možda neće biti preveden kad se projekt izvodi u razvojnom okruženju. Nakon toga, kad odaberete naredbu Make EXE File (ili isključite prevođenje uz zahtjev), možda ćete vidjeti nove i neočekivane pogreške dok se taj kod iznova prevodi.

Postoje tri tehnike koje možete upotrijebiti u okruženju u redovnim razmacima, ili u svako vrijeme, kako bi ispraznili sve pogreške skrivene korištenjem prevođenja uz zahtjev.

- Isključite prevođenje uz zahtjev i zatim pokrenite aplikaciju. To će prisiliti Visual Basic da provjeri postoje li pogreške prevođenja u cijeloj aplikaciji.
- Napravite izvršnu datoteku s svojim projektom. To će također prisiliti Visual Basic da potraži pogreške prevođenja u cijeloj aplikaciji.
- Odaberite naredbu Start With Full Compile (pokretanje s punim prevođenjem) u izborniku Run.

Savjeti za otkrivanje pogrešaka

Postoji nekoliko načina za pojednostavljivanje otkrivanja pogrešaka:

- Kad vaša aplikacija ne proizvodi ispravne rezultate, pretražite cijeli programski kod i pokušajte naći izraze koji mogu uzrokovati problem. Postavite točke prekida na tim izrazima i ponovno pokrenite aplikaciju.
- Kad se aplikacija zaustavi, ispitajte vrijednosti važnih varijabli i svojstava. Upotrijebite dijaloški okvir Quick Watch ili postavite nadgledne izraze za nadziranje tih vrijednosti. Upotrijebite prozor za neposredan upis naredbi kako bi ispitali varijable i izraze.
- Upotrijebite opciju Break on All Errors za otkrivanje gdje se pojavila pogreška. Kako bi privremeno promijenili tu opciju, odaberite stavku Toggle u pomoćnom izborniku kodnog prozora, te promijenite opciju u podizborniku. Prođite kroz vaš kod, koristeći nadgledne izraze i prozor s lokalnim varijablama za nadzor kako se mijenjaju vrijednosti dok se izvodi programski kod.
- Ako se pogreška pojavljuje u petlji, odredite izraz prekida kako bi ustanovili gdje se pojavljuje problem. Upotrijebite prozor za neposredan upis naredbi zajedno s naredbom Set Next Statement kako bi ponovno izveli petlju nakon ispravljanja.
- Ako ustanovite da varijabla ili svojstvo uzrokuju problem u vašoj aplikaciji, upotrijebite izraz Debug.Print za zaustavljanje izvođenja kad se pogrešna vrijednost dodijeli varijabli ili svojstvu.
- Kako bi postavili stanje hvatanja pogrešaka koje je podrazumijevano za Visual Basic na početku svakog otkrivanja pogrešaka, otvorite dijaloški okvir Options (dostupan iz izbornika Tools), odaberite karticu General i postavite opciju Default Error Trapping State. Visual Basic će koristiti tu postavku kad ga idući put pokrenete, čak i ako je postavka unesena za drugi projekt.

Ponekad možete otkriti pogrešku koju je posebno teško otkriti. Ne paničarite – evo nekih stvari koje možete napraviti:

- Prvo i najvažnije, napravite rezervnu kopiju. To je točka na kojoj čak i iskusni programeri često gube puno sati rada. Kad eksperimentirate, vrlo je lako slučajno prebrisati ili obrisati nužne dijelove koda.

- Upotrebljavajte objekte za otkrivanje pogrešaka koji su ugrađeni u Visual Basic. Pokušajte prepoznati liniju ili linije koda koje stvaraju pogrešku. Izdvojite taj kod. Ako možete izdvojiti problem na jedan blok koda, pokušajte ponovno stvoriti isti problem s tim blokom koda odvojenim od ostatka vaše aplikacije. Odaberite kod, kopirajte ga, pokrenite novi projekt, ulijepite kod u novi projekt, pokrenite novi projekt, i pogledajte pojavljuje li se pogreška i dalje.
- Stvorite evidencijsku datoteku. Ako ne možete izdvojiti kod, ili je problem lutajući, ili se problem pojavljuje samo kad je preveden, tada će vještina otkrivanja pogrešaka Visual Basica biti manje učinkovita. U takvim situacijama možete stvoriti evidencijsku datoteku koja će bilježiti aktivnost vaše aplikacije. To će vam omogućiti da postupno izdvojite položaj sumnjivog koda. Pozovite sljedeći potprogram iz raznih mjesta u vašoj aplikaciji. Trebate mu proslijediti tekstualni string koji ukazuje trenutan položaj koda koji se izvodi u vašoj aplikaciji.

```
Sub EvidenDato(Poruka As String)
  Dim EviDato As Integer
  EviDato = FreeFile
  Open "C:\VB\EvidDato.log" For Append As #EviDato
  Print #EviDato, Por
  Close #EviDato
End Sub

Sub Sub1()

  ' ...

  Call EvidenDato("Ovdje sam u Sub1")

  ' ...

End Sub
```

- Pojednostavite problem. Ako je moguće, udaljite iz svog projekta sve kontrole treće strane i korisničke kontrole. Zamijenite ih standardnim kontrolama Visual Basica. Uklonite sav kod koji ne izgleda povezan s problemom.
- Smanjite područje traženja. Ako ne možete riješiti problem ni s jednim od gornjih postupaka, tada je vrijeme da iz područja traženja problema uklonite sve ostale uzroke koji nisu povezani s Visual Basicom. Kopirajte svoje datoteke AUTOEXEC.BAT i CONFIG.SYS u rezervne datoteke. Izbacite iz te dvije datoteke sve pogonitelje i aplikacije koje bezuvjetno nisu neophodne za izvođenje vaše aplikacije pod Windowsima. Promijenite vaš slikovni pogonitelj na standardni pogonitelj Windows VGA. Zatvorite Windowse i ponovno podignite sustav. To će ukloniti mogućnost da postoji neka druga aplikacija ili pogonitelj koji se miješaju s vašom aplikacijom.
- Ako ne možete pronaći rješenje i nemoćni ste izdvojiti ili riješiti problem s bilo kojim od ovih postupaka, vrijeme je da potražite pomoć. Pogledajte dokumentaciju u tehničkoj podršci.

Za više informacija Točke prekida su opisane u odlomku "Korištenje točke prekida za selektivno zaustavljanje izvođenja" ranije u ovom poglavlju. Pročitajte više o nadglednim izrazima u odlomku "Nadziranje podataka nadglednim izrazima". Prozor za neposredan upis naredbi je raspravljen u odlomku "Ispitivanje podataka i potprograma prozorom za neposredan upis naredbi". Pogledajte odlomak "Provjera vašeg koda zahtjevima" za više o postupku Assert objekta Debug.

14. POGLAVLJE

Obrada pogona, mapa i datoteka

Kad programirate u Windowsima, vrlo je važno imati sposobnost dodavanja, premještanja, mijenjanja, stvaranja ili brisanja mapa (direktorija) i datoteka, te upravljati pogonima i dobivati informacije o njima.

Visual Basic vam omogućuje obradu pogona, mapa i datoteka na dva različita načina: kroz uobičajene postupke kao što su naredbe Open, Write # i tako dalje, te kroz novi skup alata, model objekta File System Object (FSO).

Sadržaj

- Uvod u model objekata datotečnog sustava
- Programiranje u modelu FSO objekata
- Obrada datoteka starim naredbama i funkcijama

Uvod u model objekata datotečnog sustava

Nova osobina Visual Basica je model objekta File System Object (FSO), koji pruža alat temeljen na objektima za rad s mapama i datotekama. To vam omogućuje upotrebu poznate sintakse objekt.postupak s bogatim skupom svojstava, postupaka i događaja za obradu mapa i datoteka, kao dodatak korištenju uobičajenih izraza i naredbi Visual Basica.

Model FSO objekata (model objekata datotečnog sustava) daje vašim aplikacijama sposobnost stvaranja, mijenjanja, premještanja i brisanja mapa, ili otkrivanja postoji li određena mapa, te ako postoji, gdje. On vam također omogućuje dobivanje informacija o mapama, kao što su njihova imena, datum kad su stvorene ili posljednji put mijenjane, i tako dalje.

Model FSO objekata čini također puno lakšim obradu datoteka. Kad obrađujete datoteke, vaš je prvi cilj spremiti podatke u lako pristupačnom obliku djelotvornom kao izvor i po zauzeću prostora. Trebate biti sposobni stvarati datoteke, ubacivati i mijenjati podatke, i proizvoditi (čitati) podatke. Iako podatke možete spremati u baze podataka, kao baze tipa Jet ili SQL, to dodaje značajnu količinu nadgradnje vašoj aplikaciji. Zbog puno razloga, nećete željeti takvu nadgradnju, ili zahtjevi pristupa vašim podacima neće trebati sve dodatne osobine pridružene s potpuno opremljenom bazom podataka. U tom slučaju, spremanje vaših podataka u binarnu ili tekstualnu datoteku je najučinkovitije rješenje.

Model FSO objekata, koji je sadržan u tipskoj biblioteci Scripting (Scrrun.dll), podržava stvaranje i upravljanje tekstualnom datotekom kroz objekt TextStream. Međutim, za sada ne podržava stvaranje ili upravljanje binarnim datotekama. Kako bi upravljali binarnim datotekama, upotrijebite naredbu Open sa zastavicom Binary. Pune informacije o upravljanju binarnim datotekama nalaze se u odlomku “Korištenje binarnog pristupa datotekama”, kasnije u ovom poglavlju.

Objekti datotečnog sustava

Model FSO objekata ima ove objekte:

objekt	opis
Drive	Omogućuje vam skupljanje informacija o pogonima dodijeljenim sustavu, kao što su koliko je prostora raspoloživo, koje je njihovo ime dijela i tako dalje. Uočite da “pogon” nije nužno tvrdi disk. To može biti pogon CD-ROM-a, RAM disk i tako dalje. Također, pogoni ne moraju biti fizički povezani sa sustavom; mogu također biti logički povezani kroz lokalnu mrežu (LAN).
Folder	Omogućuje vam stvaranje, brisanje ili premještanje mapa, te ispitivanje sustava o njihovim imenima, stazama i tako dalje.
File	Omogućuje vam stvaranje, brisanje ili premještanje datoteka, te ispitivanje sustava o njihovim imenima, stazama i tako dalje.
FileSystemObject	Glavni objekt grupe, pun postupaka koji vam omogućuju stvaranje, brisanje, dobivanje popratnih informacija, te općenito upravljanje pogonima, mapama i datotekama. Većina postupaka povezanih s ovim objektom jednaka je postupcima u ostalim objektima.
TextStream	Omogućuje vam čitanje i zapisivanje tekstualnih datoteka.

Za informacije o raznim svojstvima, postupcima i događajima u modelu objekata datotečnog sustava, upotrijebite pretraživač objekata u Visual Basicu (pritisnite F2) i pogledajte tipsku biblioteku Scripting.

Programiranje u modelu FSO objekata

Programiranje u modelu FSO objekata sadržava tri glavna zadatka:

- Korištenje postupka CreateObject ili dimenzioniranje varijable kao objekta tipa FileSystemObject za stvaranje objekta tipa FileSystemObject.
- Korištenje odgovarajućeg postupka na novostvorenom objektu.
- Pristup svojstvima objekta.

Model FSO objekata je sadržan u tipskoj biblioteci nazvanoj Scripting, koja se nalazi u datoteci Scrrun.dll. Ako već nemate pokazivač na nju, potvrdite stavku Microsoft Scripting Runtime u dijalogu References dostupnom iz izbornika Properties. Nakon toga možete koristiti pretraživač objekata kako bi vidjeli njezine objekte, zbirke, svojstva, postupke i događaje, kao i njezine konstante.

Stvaranje objekta tipa FileSystemObject

Prvi korak je stvaranje objekta tipa FileSystemObject s kojim ćete raditi. Možete to napraviti na dva načina:

- Dimenzionirajte varijablu s tipom objekta FileSystemObject:

  ```
  Dim fso As New FileSystemObject
  ```

- Upotrijebite postupak CreateObject za stvaranje objekta tipa FileSystemObject:

  ```
  Set fso = CreateObject("Scripting.FileSystemObject")
  ```

  U gornjoj sintaksi, Scripting je ime tipske biblioteke, a FileSystemObject je ime objekta, od kojega stvarate primjer.

  Napomena Prvi postupak radi samo u Visual Basicu, do drugi postupak radi i u Visual Basicu i u VBScriptu.

Korištenje odgovarajućeg postupka

Idući korak je korištenje odgovarajućeg postupka objekta FileSystemObject. Na primjer, ako želite *stvoriti* novi objekt, možete upotrijebiti postupke CreateFolder ili CreateTextFile (model FSO objekata ne podržava stvaranje ili brisanje pogona).

Ako želite obrisati objekte, možete upotrijebiti postupke DeleteFile i DeleteFolder objekta FileSystemObject, ili postupak Delete objekata File i Folder.

Upotrebom odgovarajućih postupaka, možete također kopirati i premještati datoteke i mape.

Uočite da su neke djelotvornosti u objektu tipa FileSystemObject suvišne. Na primjer, možete kopirati datoteku korištenjem postupka CopyFile objekta FileSystemObject, ili možete upotrijebiti postupak Copy objekta File. Oba postupka rade isto. Oba postoje kako bi vam pružili maksimalnu fleksibilnost u programiranju.

Pristup postojećim pogonima, datotekama i mapama

Kako bi dobili pristup *postojećem* pogonu, datoteci ili mapi, upotrijebite odgovarajući postupak "dobivanja" objekta FileSystemObject:

- GetDrive
- GetFolder
- GetFile

Na primjer:

```
Dim fso As New FileSystemObject, dato As File
Set dato = fso.GetFile("c:\test.txt")
```

Zapamtite, međutim, da ne trebate upotrebljavati postupke "dobivanja" za novostvorene objekte, jer su funkcije "stvaranja" već vratile hvataljku za novostvorene objekte. Na primjer, ako stvorite novu mapu korištenjem postupka CreateFolder, ne trebate zatim upotrijebiti postupak GetFolder za pristup njezinim svojstvima, kao što su Name, Path, Size, i tako dalje. Samo postavite varijablu na funkciju CreateFolder kako bi dobili hvataljku za novostvorenu mapu, i zatim pristupite njezinim svojstvima, postupcima i događajima:

```
Private Sub Stvaranje_Mape()
  Dim fso As New FileSystemObject, mapa As Folder
  Set mapa = fso.CreateFolder("c:\MojTest")
  MsgBox "Stvorena mapa: " & mapa.Name
End Sub
```

Pristup svojstvima objekta

Kad ste jednom stvorili hvataljku za objekt, možete pristupati njezinim svojstvima. Na primjer, recimo da želite dohvatiti ime određene mape. Najprije ćete stvoriti primjer objekta, zatim ćete dohvatiti hvataljku za njega odgovarajućim postupkom (u ovom slučaju, postupkom GetFolder, jer mapa već postoji):

```
Set mapa = fso.GetFolder("c:\")
```

Sad kad imate hvataljku za objekt Folder, možete provjeriti njegovo svojstvo Name:

```
Debug.Print "Ime mape je: "; mapa.Name
```

Ako želite pronaći kad je datoteka posljednji put mijenjana, upotrijebite sljedeću sintaksu:

```
Dim fso As New FileSystemObject, dato As File
' Dohvat objekta File za ispitivanje.
Set dato = fso.GetFile("c:\detlog.txt")
' Ispis informacije.
Debug.Print "Datoteka je mijenjana: "; fil.DateLastModified
```

Rad s pogonima i mapama

S modelom FSO objekata možete raditi s pogonima i mapama programirajući, isto kako to interaktivno možete u Windows Exploreru. Možete kopirati i premještati mape, dobivati informacije o pogonima i mapama, i tako dalje.

Dobivanje informacije o pogonima

Objekt Drive omogućuje vam dobivanje informacije o raznim pogonima dodijeljenima sustavu, fizički ili putem mreže. Njegova svojstva vam omogućuju dobivanje informacija o:

- ukupnoj veličini pogona u bajtovima (svojstvo TotalSize)
- koliko je prostora raspoloživo na pogonu u bajtovima (svojstva AvailableSpace ili FreeSpace)
- koje je slovo dodijeljeno pogonu (svojstvo DriveLetter)
- koji je to tip pogona, je li izmjenjivi, tvrdi, mrežni, CD-ROM ili RAM disk (svojstvo DriveType)
- serijskom broju pogona (svojstvo SerialNumber)
- tip datotečnog sustava kojeg koristi pogon, kao FAT, FAT32, NTFS, i tako dalje (svojstvo FileSystem)
- je li pogon na raspolaganju za upotrebu (svojstvo IsReady)
- imenu dijela i/ili volumena (svojstva ShareName i VolumeName)
- Stazi ili korijenskoj mapi pogona (svojstva Path i RootFolder)

Primjer korištenja objekta Drive

Sljedeći primjer pokazuje kako upotrijebiti objekt Drive za sakupljanje informacija o pogonu. Zapamtite da u sljedećem kodu nećete vidjeti pokazivač na stvarni objekt Drive; umjesto toga, upotrijebit ćete postupak GetDrive za dobivanje pokazivača na postojeći objekt Drive (u ovom slučaju, pog):

```
Private Sub Command3_Click()
  Dim fso As New FileSystemObject, pog As Drive, s As String
  Set pog = fso.GetDrive(fso.GetDriveName("c:"))
  s = "Pogon " & UCase("c:") & " - "
  s = s & drv.VolumeName & vbCrLf
  s = s & "Ukupno prostora: " & _
  FormatNumber(pog.TotalSize / 1024, 0)
  s = s & " Kb" & vbCrLf
  s = s & "Slobodnog prostora: " &_
  FormatNumber(pog.FreeSpace / 1024, 0)
  s = s & " Kb" & vbCrLf
  MsgBox s
End Sub
```

Korištenje funkcije CurDir, te izraza ChDrive, ChDir i App.Path

Ako koristite funkciju CurDir, izraze ChDrive i ChDir, ili svojstvo Path (App.Path), budite svjesni da oni mogu vratiti stazu UNC tipa (znači, \\Poslužitelj\Dio...) prije nego stazu pogona (kao E:\Mapa), ovisno o tome kako izvodite vašu aplikaciju ili projekt.

App.Path vraća UNC stazu:

- Kad izvodite projekt nakon što ste ga učitali s dijela mreže, čak i ako je dijelu mreže dodijeljeno slovo kao oznaka pogona.
- Kad izvodite prevedenu izvršnu datoteku s dijela mreže (ali samo ako se ona izvodi koristeći UNC stazu).

Izraz ChDrive ne može rukovati UNC stazama, i zbog toga izaziva pogrešku kad izraz App.Path vrati takav tip. Možete rukovati tom pogreškom dodavanjem izraza On Error Resume Next prije izraza ChDrive, ili ispitivanjem prva dva karaktera vrijednosti izraza App.Path kako bi vidjeli jesu li oni *backslash* karakteri (\ ):

```
On Error Resume Next
ChDrive App.Path
ChDir App.Path
```

Ovakva promjena obrađuje sve slučajeve u kojima se aplikacija pokreće iz Windowsa korištenjem UNC staze (na primjer, u dijalogu Run kojem se pristupa iz izbornika Start), jer Windowsi postavljaju trenutan direktorij u UNC stazu. Izraz ChDir ispravno rukuje mijenjanjem među UNC stazama (neuspjeh izraza ChDrive može se zanemariti, jer za UNC stazu ne postoji slovčana oznaka pogona).

Međutim, prethodni kod neće raditi ako ste pokrenuli aplikaciju upisivanjem UNC staze u naredbenu liniju MS-DOS-a. Razlog tomu je što naredbena linija uvijek ima stazu pogona za trenutan direktorij, pa je funkcija CurDir uvijek postavljena na stazu pogona. Izraz ChDir ne izaziva pogrešku, ali ne uspijeva promijeniti direktorij iz staze pogona u UNC stazu. Jedino rješenje za takvu situaciju je pronalaženje lokalnog pogona koji je pridružen dijelu određenom u UNC stazi, ili upotreba mrežnih naredbi za stvaranje takvog preslikavanja.

Ako je projekt učitan u razvojnu okolinu Visual Basica iz dijela mreže – ili UNC staze ili preslikane staze pogona – izraz App.Path vraća UNC stazu kad se projekt pokrene i izraz ChDrive ne uspije ili izazove pogrešku. Izraz ChDir ne izaziva pogrešku, ali direktorij se ne mijenja. Jedino rješenje je ručno određivanje pogona i direktorija:

```
Const DIOPROJEKTA = “E:\MOJPROJEKT”
#Const Debug = True
#If Debug Then
  ChDrive DIOPROJEKTA
  ChDir DIOPROJEKTA
#Else
  On Error Resume Next
  ChDrive App.Path
  ChDir App.Path
# End If
```

Ako više od jedne osobe može otvoriti projekt na dijelu mreže, varijabla DOS okruženja se može upotrijebiti za omogućavanje svakoj osobi da ima svoj vlastiti preslikani dio:

```
#Const Debug = True
#If Debug Then
  ChDrive Environ("DIRMOGPROJEKTA")
  ChDir Environ("DIRMOGPROJEKTA")
#Else
  On Error Resume Next
  ChDrive App.Path
  ChDir App.Path
# End If
```

Vrijednost izraza DIRMOGPROJEKTA određuje slovčanu oznaku i stazu preslikanog dijela, na primjer:

```
SET DIRMOGPROJEKTA=M:\VBProj\MojProjekt
```

Rad s mapama

Sljedeća popis pokazuje uobičajene poslove s mapama i postupke za njihovo ostvarivanje:

posao	postupak
Stvaranje mape	FileSystemObject.CreateFolder
Brisanje mape	Folder.Delete ili FileSystemObject.DeleteFolder
Premještanje mape	FolderMove ili FileSystemObject.MoveFolder
Kopiranje mape	FolderCopy ili FileSystemObject.CopyFolder
Dohvaćanje imena mape	Folder.Name
Pronalaženje postoji li mapa na pogonu	FileSystemObject.FolderExists
Dobivanje primjera postojećeg objekta Folder	FileSystemObject.GetFolder
Pronalaženje imena roditeljske mape	FileSystemObject.GetParentFolderName
Pronalaženje staze sistemskih mapa	FileSystemObject.GetSpecialFolder

Primjer

Ovaj primjer pokazuje upotrebu objekata Folder i FileSystemObject za upravljanje mapama i dobivanje informacija o njima:

```
Private Sub CommandO_Click()
  ‘ Dobivanje primjera objekta FileSystemObject.
  Dim fso As New FileSystemObject, mapa As Folder, s As String
  ‘ Dobivanje objekta Drive.
  Set mapa = fso.GetFolder(“c:”)
  ‘ Ispis imena roditeljske mape.
  Debug.Print “Ime roditeljske mape je: “ & mapa
  ‘ Ispis imena pogona.
  Debug.Print “Sadržana je na pogonu “ & mapa.Drive
  ‘ Ispis imena korijenske datoteke.
  If mapa.IsRootFolder = True Then
    Debug.Print “Ova mapa je korijenska mapa.”
  Else
    Debug.Print “ Ova mapa nije korijenska mapa.”
  End If
  ‘ Stvaranje nove mape objektom FileSystemObject.
  fso.CreateFolder (“c:\Lažna”)
  Debug.Print “Stvorena je mapa C:\Lažna.”
  ‘ Ispis osnovnog imena mape.
  Debug.Print “Osnovno ime = “ & fso.GetBaseName(“c:\Lažna”)
  ‘ Brisanje novostvorene mape.
  fso.DeleteFolder (“c:\Lažna”)
  Debug.Print “Obrisana je mapa C:\Lažna.”
End Sub
```

Rad s datotekama

Možete raditi s datotekama u Visual Basicu upotrebom novih objektno usmjerenih FSO objekata, kao što su Copy, Delete, Move i OpenAsTextStream, među ostalima, ili upotrebom starijih postojećih funkcija kao što su Open, Close, FileCopy, GetAttr i tako dalje. Zapamtite da možete premještati, kopirati i brisati datoteke neovisno o njihovom tipu.

Za više informacija o korištenju starijih postojećih funkcija, pogledajte odlomak “Obrada datoteka starim naredbama i funkcijama” u ovom poglavlju. Ostatak ovog dijela opisuje upotrebu novih FSO objekata, postupaka i svojstava za rad s datotekama.

Postoje dvije glavne kategorije upravljanja datotekama:

- Stvaranje, dodavanje ili brisanje podataka, i čitanje datoteka
- Premještanje, kopiranje i brisanje datoteka

Stvaranje datoteka i dodavanje podataka sa FSO objektima

Postoje tri načina stvaranja sekvencijalne datoteke teksta (ponekad nazvane kao "tijek teksta"). Jedan način je korištenje postupka CreateTextFile. Praznu tekstualnu datoteku možete stvoriti ovako:

```
Dim fso As New FileSystemObject, dato As File
Set dato = fso.CreateTextFile("c:\testdato.txt", True)
```

> **Napomena** Model FSO objekata još ne podržava stvaranje binarnih datoteka i datoteka s izravnim pristupom. Kako bi stvorili takve datoteke, upotrijebite naredbu Open sa zastavicom Binary ili Random. Pune informacije o upravljanju binarnim datotekama i datotekama s izravnim pristupom nalaze se u odlomcima "Korištenje binarnog pristupa datotekama" i "Korištenje izravnog pristupa datotekama", kasnije u ovom poglavlju.

Drugi način je upotreba postupka OpenTextFile objekta FileSystemObject s postavljenom zastavicom ForWriting:

```
Dim fso As New FileSystemObject, ts As New TextStream
Set ts = fso.OpenTextFile("c:\testdato.txt", ForWriting)
```

Također možete upotrijebiti postupak OpenAsTextStream s postavljenom zastavicom ForWriting:

```
Dim fso As New FileSystemObject, dato As File, ts As TextStream
Set fso = CreateObject("Scripting.FileSystemObject")
fso.CreateTextFile("test1.txt")
Set dato = fso.GetFile("test1.txt")
Set ts = OpenAsTextStream(ForWriting)
```

Dodavanje podataka datoteci

Kad je stvorena tekstualna datoteka, možete joj dodati podatke u tri koraka:

1. Otvorite tekstualnu datoteku za zapisivanje podataka.
2. Zapišite podatke.
3. Zatvorite datoteku.

Kako bi otvorili datoteku, možete upotrijebiti jedan od dva postupka: postupak OpenAsTextStream objekta File, ili postupak OpenTextFile objekta FileSystemObject.

Kako bi zapisali podatke u otvorenu tekstualnu datoteku, upotrijebite postupak Write ili postupak WriteLine, oba od objekta TextStream. Jedina razlika između ova dva postupka je što postupak WriteLine dodaje karakter oznake nove linije na kraj određenog stringa.

Ako želite dodati novu liniju tekstualnoj datoteci, upotrijebite postupak WriteBlankLines.

Kako bi zatvorili otvorenu datoteku, upotrijebite postupak Close objekta TextStream.

Slijedi primjer otvaranja datoteke, upotrebe sva tri postupka zapisivanja podataka u datoteku, i zatvaranja datoteke:

```
Sub Stvaranje_Datoteke()
  Dim fso, txtdatoteka
  Set fso = CreateObject("Scripting.FileSystemObject")
  Set txtdatoteka = fso.CreateTextFile("c:\testdato.txt", True)
  txtdatoteka.Write("Ovo je test. ")            ' Zapisivanje linije.
  ' Zapisivanje linije s oznakom nove linije.
  txtdatoteka.WriteLine("Proba 1, 2, 3.")
  ' Zapisivanje tri oznake nove linije u datoteku.
  txtdatoteka.WriteBlankLines(3)
  txtdatoteka.Close
End Sub
```

Čitanje podataka sa FSO objektima

Kako bi pročitali podatke iz tekstualne datoteke, upotrijebite postupke Read, ReadLine ili ReadAll objekta TextStream:

posao	postupak
Čitanje određenog broja karaktera iz datoteke	Read
Čitanje cijele linije (sve do, ali ne uključujući, karaktera oznake nove linije)	ReadLine
Čitanje cijelog sadržaja tekstualne datoteke	ReadAll

Ako koristite postupke Read ili ReadLine i želite preskočiti do određenog dijela podataka, možete upotrijebiti postupke Skip ili SkipLine.

Rezultirajući tekst postupaka čitanja spremljen je u stringu koji može biti prikazan u kontroli, dijeljen operatorima stringa (kao što su Left, Right i Mid), ulančan i tako dalje.

Napomena Konstanta vbNewLine sadrži karakter ili karaktere (ovisno o operativnom sustavu) koji pomiču pokazivač na početak nove linije (oznaka kraja reda/novi redak). Budite svjesni da krajevi nekih stringova mogu imati takve karaktere koji se ne ispisuju.

Primjer

```
Sub Čitanje_Datoteka()
  Dim fso As New FileSystemObject, txtdatoteka, _
  dat1 As File, ts As TextStream
  Set txtdatoteka = fso.CreateTextFile("c:\testdato.txt", True)
  MsgBox "Zapisivanje datoteke"
  ' Zapisivanje linije.
  Set dat1 = fso.GetFile("c:\testdato.txt")
```

```
  Set ts = dat1.OpenAsTextStream(ForWriting)
  ts.Write "Pozdrav svijetu"
  ts.Close
  ' Čitanje sadržaja datoteke.
  Set ts = dat1.OpenAsTextStream(ForReading)
  s = ts.ReadLine
  MsgBox s
  ts.Close
End Sub
```

Premještanje, kopiranje i brisanje datoteka

Model FSO objekata ima po dva postupka za premještanje, kopiranje i brisanje datoteka:

posao	postupak
premještanje datoteke	File.Move ili FileSystemObject.MoveFile
kopiranje datoteke	File.Copy ili FileSystemObject.CopyFile
brisanje datoteke	File.Delete ili FileSystemObject.DeleteFile

Primjer

Ovaj primjer stvara tekstualnu datoteku u korijenskom direktoriju pogona C, zapisuje neke informacije u nju, premješta je u direktorij imena \tmp, stvara njezinu kopiju u direktoriju imena \temp, te zatim briše kopije iz oba direktorija.

Kako bi izveli ovaj primjer, provjerite imate li direktorije s imenima \tmp i \temp u korijenskom direktoriju pogona C:

```
Sub Upravljanje_datotekama()
  Dim fso As New FileSystemObject, txtdatoteka, dat1, dat2
  Set txtdatoteka = fso.CreateTextFile("c:\testdato.txt", True)
  MsgBox "Zapisivanje datoteke"
  ' Zapisivanje linije.
  txtdatoteka.Write("Ovo je test.")
  ' Zatvaranje datoteke za zapisivanje.
  txtdatoteka.Close
  MsgBox "Premještanje datoteke u c:\tmp"
  ' Dobivanje hvataljke za datoteku u korijenskom dir-u c:\.
  Set dat1 = fso.GetFile("c:\testdato.txt")
  ' Premještanje datoteke u direktorij \tmp.
  dat1.Move("c:\tmp\testdato.txt")
  MsgBox "Kopiranje datoteke u c:\temp"
  ' Kopiranje datoteke u \temp.
  dat1.Copy("c:\temp\testdato.txt")
```

```
    MsgBox “Brisanje datoteke”
    ‘ Dobivanje hvataljki za trenutne položaje datoteka.
    Set dat1 = fso.GetFile(“c:\tmp\testdato.txt”)
    Set dat2 = fso.GetFile(“c:\temp\testdato.txt”)
    ‘ Brisanje datoteka.
    dat1.Delete
    dat2.Delete
    MsgBox “Sve je obavljeno!”
End Sub
```

Obrada datoteka starim naredbama i funkcijama

Još od prve verzije Visual Basica, datoteke se obrađuju korištenjem naredbe Open i ostalih povezanih naredbi i funkcija (ispisanih u nastavku). Ovi mehanizmi će na kraju biti izbačeni u korist modela FSO objekata, ali su potpuno podržani u Visual Basicu 6.0.

Ako možete oblikovati svoju aplikaciju tako da koristi datoteke baze podataka, nećete trebati pružiti izravan pristup datotekama u svojoj aplikaciji. Kontrola podataka i kontrole povezivanja omogućuju vam čitanje i zapisivanje podataka u i iz baze podataka, što je puno lakše od korištenja tehnika izravnog pristupanja datotekama.

Međutim, postoje trenuci kad trebate čitati iz i zapisivati u datoteke koje nisu iz baze podataka. Sljedeći skup tema pokazuje kako obrađivati datoteke za izravno stvaranje, upravljanje i spremanje teksta i drugih podataka.

Tipovi pristupa datotekama

Sama po sebi, *datoteka* se sastoji samo od niza povezanih bajtova spremljenih na disku. Kad vaša aplikacija pristupi datoteci, mora pretpostaviti što bi ti bajtovi trebali predstavljati (karaktere, zapise podataka, cijele brojeve, stringove i tako dalje).

Ovisno o tome koju vrstu podataka sadrži datoteka, upotrijebit ćete odgovarajući tip pristupa datoteci. U Visual Basicu, postoje tri tipa pristupa datotekama:

- Sekvencijalni – Za čitanje i zapisivanje tekstualnih datoteka u neprekidnim blokovima.
- Izravan – Za čitanje i zapisivanje tekstualnih ili binarnih datoteka sastavljenih od zapisa nepromjenjive duljine.
- Binaran – Za čitanje i zapisivanje neodređeno sastavljenih datoteka.

Sekvencijalni pristup (*sequential access*) je oblikovan za korištenje s jednostavnim tekstualnim datotekama. Za svaki karakter u datoteci pretpostavlja se da predstavlja karakter teksta ili karakter niza za oblikovanje teksta, kao što je karakter nove linije (NL). Podaci su spremljeni kao ANSI karakteri. Pretpostavlja se da je datoteka otvorena za *izravan pristup* (*random access*) sastavljena od niza *zapisa* jednake duljine.

Možete primijeniti korisnički određene tipove za stvaranje zapisa sačinjenih od puno polja – svaki može imati različite tipove podataka. Podaci su spremljeni kao binarne informacije.

Binarni pristup (*binary access*) vam dopušta korištenje datoteka za spremanje podataka kako god želite. Sličan je izravnom pristupu, osim što ovdje nema pretpostavki o tipu podataka i duljini zapisa. Međutim, morate točno znati kako su podaci zapisani u datoteku kako bi ih mogli ispravno dohvatiti.

Za više informacija Kako bi naučili više o tipovima pristupa podacima, pogledajte odlomke "Korištenje sekvencijalnog pristupa datotekama", "Korištenje izravnog pristupa datotekama" i "Korištenje binarnog pristupa datotekama" , kasnije u ovom poglavlju.

Funkcije i naredbe pristupanja datotekama

Sljedeće funkcije se koriste sa sva tri tipa pristupa datotekama:

Dir	FileLen	LOF
EOF	FreeFile	Seek
FileCopy	GetAttr	SetAttr
FileDateTime	Loc	

Sljedeća tablica ispisuje sve naredbe i funkcije pristupanja datotekama, dostupne za svaki od tri tipa pristupa datotekama.

naredbe i funkcije	sekvencijalni	izravni	binarni
Close	X	X	X
Get		X	X
Input( )	X		X
Input #	X		
Line Input #	X		
Open	X	X	X
Print #	X		
Put		X	X
Type...End Type		X	
Write #	X		

Za više informacija Za dodatne informacije o naredbama i funkcijama pristupa datotekama, potražite tu funkciju ili naredbu u kazalu stalne pomoći.

Korištenje sekvencijalnog pristupa datotekama

Preporučeno je da upotrebljavate objekte FSO modela za stvaranje tekstualnih datoteka, ali sljedeće informacije su pružene u slučaju da trebate upotrijebiti starije postupke stvaranja tekstualnih datoteka.

Sekvencijalni pristup radi najbolje kad želite obraditi datoteke koje se sastoje samo od teksta, kao datoteke stvorene sa tipičnim editorom teksta – znači, datotekama u kojima podaci *nisu* podijeljeni u nizove zapisa. Sekvekcijalni pristup možda neće biti najpogodniji za spremanje dugih nizova brojeva, jer se svaki broj sprema kao tekstualni karakter. Četveroznamenkasti broj trebat će 4 bajta za spremanje umjesto 2 bajta koja su potrebna za spremanje istog broja kao cjelobrojne vrijednosti.

Otvaranje datoteka za sekvencijalni pristup

Kad otvorite datoteku za sekvencijalni pristup, otvarate ju za izvođenje jedne od sljedećih operacija:

- Ulaz karaktera iz datoteke (Input)
- Izlaz karaktera u datoteku (Output)
- Dodavanje karaktera datoteci (Append)

Kako bi otvorili datoteku za sekvencijalni pristup, upotrijebite sljedeću sintaksu za naredbu Open:

Open *stazaime* **For** [**Input** | **Output** | **Append**] **As** *brojdatoteke* [**Len** = *veličinaspremnika*]

Kad otvorite sekvencijalnu datoteku za ulaz, datoteka već mora postojati; inače će se pojaviti greška. Međutim, kad pokušate otvoriti nepostojeću datoteku za izlaz ili dodavanje, naredba Open će prvo stvoriti datoteku i zatim ju otvoriti.

Neobavezni argument Len određuje broj karaktera u međuspremniku kod kopiranja podataka između datoteke i vaše aplikacije.

Nakon otvaranja datoteke za operacije Input, Output ili Append, morate ju zatvoriti, koristeći naredbu Close, prije nego što ju ponovno otvorite za drugi tip operacije.

Editiranje datoteka otvorenih za sekvencijalni pristup

Ako želite editirati datoteku, najprije pročitajte njezin sadržaj u varijable aplikacije, zatim promijenite varijable, i na kraju, zapišite varijable natrag u datoteku. Sljedeći odlomci raspravljaju o načinu editiranja zapisa otvorenih za sekvencijalni pristup.

Čitanje stringova iz datoteka

Kako bi dohvatili sadržaj tekstualne datoteke, otvorite datoteku za sekvencijalni ulaz. Zatim upotrijebite naredbe Line Input #, Input( ), ili Input # za kopiranje datoteke u varijable aplikacije.

Visual Basic pruža naredbe i funkcije koje će pročitati i zapisati jedan po jedan karakter ili jednu po jednu liniju iz sekvencijalnih datoteka.

Na primjer, sljedeći dio koda čita datoteku jednu po jednu liniju:

```
Dim LinijeIzDato, IdućaLinija As String

Do Until EOF(BrojDato)
  Line Input #BrojDato, IdućaLinija
  LinijeIzDato = LinijeIzDato + IdućaLinija + Chr(13) + Chr(10)
Loop
```

Iako izraz Line Input # prepoznaje kraj linije kad dođe do oznake kraj-reda/novi-redak, ne uključuje tu oznaku kad pročita liniju u varijablu. Ako želite zadržati oznaku kraj-reda/novi-redak, vaš kod ju mora dodati.

Možete također upotrijebiti naredbu Input #, koja čita popis izraza brojeva i/ili izraza stringova zapisanih u datoteku. Na primjer, kako bi pročitali liniju iz datoteke sa poštanskom popisom, mogli bi upotrijebiti sljedeći izraz:

```
Input #BrojDato, ime, ulica, grad, država, pošbroj
```

Funkciju Input možete upotrijebiti za kopiranje bilo kojeg broja karaktera iz datoteke u varijablu, pružajući varijablu koja je dovoljno velika. Na primjer, sljedeći kod upotrebljava funkciju Input za kopiranje određenog broja karaktera u varijablu:

```
LinijeIzDato = Input(n, BrojDato)
```

Kako bi kopirali cijelu datoteku u varijablu, upotrijebite funkciju InputB za kopiranje bajtova iz datoteke u varijablu. Budući da funkcija InputB vraća ANSI string, morate upotrijebiti funkciju StrConv za pretvaranje ANSI stringa u UNICODE string, kako slijedi:

```
LinijeIzDato = StrConv(InputB(LOF(BrojDato), BrojDato), vbUnicode)
```

Zapisivanje stringova u datoteke

Kako bi spremili sadržaje varijabli u sekvencijalnu datoteku, otvorite ju za sekvencijalne operacije Output ili Append, i zatim upotrijebite naredbu Print #. Na primjer, editor teksta mogao bi koristiti sljedeću liniju koda za kopiranje sadržaja okvira s tekstom u datoteku:

```
Print #BrojDato, Okvir.Text
```

Visual Basic također podržava naredbu Write #, koja zapisuje popis izraza brojeva i/ili izraza stringova u datoteku. Ova naredba automatski razdvaja svaki izraz zarezom i postavlja znakove navodnika oko tekstualnih izraza:

```
Dim NekiString As String, NekiBroj As Integer

NekiString = “NekiKarakteri”
NekiBroj = 23445
Write #BrojDato, NekiString, NekiBroj
```

Ovaj dio koda zapisuje dva izraza u datoteku određenu podatkom BrojDato. Prvi izraz sadrži string, a drugi izraz sadrži broj 23445. Stoga, Visual Basic zapisuje sljedeće karaktere (uključujući sve znakove interpunkcije) u datoteku:

```
"NekiKarakteri",23445
```

Napomena Ako upotrebljavate naredbe Write # i Input # sa sekvencijalnim pristupom, razmislite o upotrebi izravnog ili binarnog pristupa umjesto toga, jer su oni bolje opremljeni za rad sa zapisima podataka.

Za više informacija Za dodatne informacije o sekvencijalnom pristupu datotekama, pogledajte odlomak "Naredba Open" u priručniku *Microsoft Visual Basic 6.0 Language Reference* biblioteke *Microsoft Visual Basic 6.0 Reference Library*.

Korištenje izravnog pristupa datotekama

Model FSO objekata ne pruža postupke stvaranja datoteke za izravan pristup ni postupke izravnog pristupanja. Ako trebate stvoriti ili čitati datoteke s izravnim pristupom, ove informacije će vam u tome pomoći.

Bajtovi u datotekama za izravan pristup oblikuju jednake zapise, svaki od njih sadrži jedno ili više polja. Zapis s jednim poljem prikladan je za svaki standardni tip podatka, kao što je cijeli broj ili string nepromjenjive duljine. Zapis saviše od jednog polja prikladan je za korisnički određen tip podatka. Na primjer, tip Djelatnik određen u nastavku stvara 19-bajtne zapise koji se sastoje od tri polja:

```
Type Djelatnik
  Prezime As String * 10
  Titula As String * 7
  Rang As String * 2
End Type
```

Određivanje varijabli

Prije nego što vaša aplikacija otvori datoteku za izravan pristup, treba odrediti sve varijable potrebne za rukovanje podacima iz datoteke. To uključuje korisnički određene tipove, koji odgovaraju zapisima u datoteci, kao i standardne tipove za ostale varijable koje sadrže podatke koji se odnose na obrađivanje datoteke otvorene za izravan pristup.

Određivanje tipova zapisa

Prije otvaranja datoteke za izravan pristup, odredite tip koji odgovara zapisima koje stvara datoteka ili će ih sadržavati. Na primjer, datoteka Podaci Djelatnika mogla bi odrediti korisnički određen tip podatka nazvanog Osoba na ovaj način:

```
Type Osoba
  ID                          As Integer
  MjesečnaPlaća               As Currency
  DatumZadnjegPregleda        As Long
  Ime                         As String * 15
  Prezime                     As String * 15
```

```
  Titula                          As String * 15
  Napomene                As String * 150
End Type
```

Određivanje varijabli polja u određivanju tipa

Budući da svi zapisi u datoteci za izravan pristup moraju imati istu duljinu, često je korisno da tekstualni elementi u korisnički određenom tipu imaju nepromjenjivu duljinu, kao što je prikazano u određivanju tipa Osoba gore, gdje, na primjer, elementi `Ime` i `Prezime` imaju nepromjenjivu duljinu od 15 karaktera.

Ako stvarni string sadrži manje karaktera od nepromjenjive duljine tekstualnog elementa u kojeg se zapisuje, Visual Basic preostala mjesta u zapisu popunjava karakterima razmaka (kod karaktera 32). Također, ako je string dulji od veličine polja, višak se odrezuje. Ako upotrebljavate stringove duljine varijable, ukupna veličina svakog zapisa spremljenog naredbom Put ili dohvaćenog naredbom Get ne smije biti veća od duljine zapisa određene u dodatku Len naredbe Open.

Određivanje ostalih varijabli

Nakon određivanja tipa koji odgovara tipičnom zapisu, odredite sve ostale varijable koje vaša aplikacija treba za obrađivanje datoteke otvorene za izravan pristup. Na primjer:

```
‘ Varijabla zapisa.
Public Djelatnik As Osoba
‘ Praćenje trenutnog zapisa.
Public Položaj As Long
‘ Broj zadnjeg zapisa u datoteci.
Public PosljednjiZapis As Long
```

Otvaranje datoteka za izravan pristup

Kako bi otvorili datoteku za izravan pristup, upotrijebite sljedeću sintaksu za naredbu Open:

Open *stazaime* [**For Random**] **As** *brojdatoteke* **Len** = *duljinazapisa*

Budući da je tip Random podrazumijevani tip pristupa, ključne riječi For Random su neobavezne.

Izraz Len = *duljinazapisa* određuje veličinu svakog zapisa u bajtovima. Zapamtite da svaka varijabla stringa u Visual Basicu sprema string tipa Unicode i da trebate odrediti duljinu u bajtovima za taj string tipa Unicode. Ako je vrijednost *duljinazapisa* manja od stvarne duljine zapisa zapisanog u datoteci, stvorit će se pogreška. Ako je vrijednost *duljinazapisa* veća od stvarne duljine zapisa, zapis će biti zapisan, iako će dio prostora na disku biti uzaludno potrošen.

Možete upotrijebiti sljedeći kod za otvaranje datoteke:

```
Dim BrojDato As Integer, DuljinaZapisa As Long, Djelatnik As Osoba

‘ Izračunavanje duljine svakog zapisa.
DuljinaZapisa = LenB(Djelatnik)
‘ Dobivanje idu}eg raspoloživog broja datoteke.
BrojDatoteke = FreeFile
‘ Otvaranje nove datoteke naredbom Open.
Open “MOJADATO.DAT” For Random As BrojDato Len = DuljinaZapisa
```

Editiranje datoteka otvorenih za izravan pristup

Ako želite editirati datoteku za izravan pristup, najprije pročitajte zapise iz datoteke u varijable aplikacije, zatim promijenite vrijednosti u varijablama, i na kraju, zapišite varijable natrag u datoteku. Sljedeći odlomci raspravljaju kako editirati datoteke otvorene za izravan pristup.

Čitanje zapisa u varijable

Upotrijebite naredbu Get za kopiranje zapisa u varijable. Na primjer, kako bi kopirali zapis iz datoteke Podaci Djelatnika u varijablu `Djelatnik`, možete upotrijebiti sljedeći kod:

```
Get BrojDato, Položaj, Djelatnik
```

U ovoj liniji koda, varijabla `BrojDato` sadrži broj koji naredba Open koristi za otvaranje datoteke; varijabla `Položaj` sadrži broj zapisa kojeg treba kopirati; a varijabla `Djelatnik`, određena kao korisnički određen tip Osoba, prihvaća sadržaj zapisa.

Zapisivanje varijabli u zapise

Upotrijebite naredbu Put za dodavanje ili zamjenu zapisa u datotekama otvorenim za izravan pristup.

Zamjena zapisa

Kako bi zamijenili zapise, upotrijebite naredbu Put, određujući položaj zapisa kojeg želite zamijeniti; na primjer:

```
Put #BrojDato, Polo`aj, Djelatnik
```

Ovaj kod će zamijeniti zapis broja određenog varijablom Položaj, s podacima u varijabli Djelatnik.

Dodavanje zapisa

Kako bi dodali nove zapise na kraj datoteke otvorene za izravan pristup, upotrijebite naredbu Put na način prikazan u prethodnom dijelu koda. Postavite vrijednost varijable Položaj tako da bude za jedan veća od broja zapisa u datoteci. Na primjer, kako bi dodali zapis datoteci koja sadrži pet zapisa, postavite varijablu Položaj na 6.

Sljedeći izraz dodaje zapis na kraj datoteke:

```
PosljednjiZapis = PosljednjiZapis + 1
Put #BrojDato, PosljednjiZapis, Djelatnik
```

Brisanje zapisa

Mogli bi obrisati zapis čišćenjem njegovih polja, ali bi zapis i dalje postojao u datoteci. Obično ne želite prazne zapise u vašoj datoteci, jer troše prostor i dolaze u sukob sa sekvencijalnim operacijama. Bolje je kopirati preostale zapise u novu datoteku, te zatim obrisati staru datoteku.

Kako maknuti obrisani zapis iz datoteke za izravan pristup

1. Stvorite novu datoteku.
2. Kopirajte sve valjane zapise iz izvorne datoteke u novu datoteku.
3. Zatvorite izvornu datoteku i upotrijebite naredbu Kill kako bi ju obrisali.
4. Upotrijebite naredbu Name za primjenu imena nove datoteke u ime izvorne datoteke.

Za više informacija Za dodatne informacije o datotekama s izravnim pristupom, pogledajte odlomak "Naredba Open" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Korištenje binarnog pristupa datotekama

Model FSO objekata ne pruža postupke stvaranja datoteke za binarni pristup ni postupke binarnog pristupanja. Ako trebate stvoriti ili čitati datoteke s binarnim pristupom, ove informacije će vam u tome pomoći.

Binarni pristup daje vam potpuni nadzor nad datotekom, jer bajtovi u datoteci mogu predstavljati bilo što. Na primjer, možete uštedjeti prostor na disku gradnjom zapisa duljine varijable. Upotrijebite binarni pristup kad je važno sačuvati malu veličinu datoteke.

Napomena Kad zapisujete binarne podatke u datoteku, upotrijebite varijablu podataka tipa Byte, umjesto varijable tipa String. Stringovi trebaju sadržavati karaktere, a binarni podaci možda neće biti odgovarajuće spremljeni u varijablama tipa String.

Otvaranje datoteke za binarni pristup

Kako bi otvorili datoteku za binarni pristup, upotrijebite sljedeću sintaksu za naredbu Open:

Open *stazaime* **For Binary As** *brojdatoteke*

Kao što možete vidjeti, sintaksa naredbe Open za binarni pristup razlikuje se sintakse naredbe Open za izravan pristup po tome što argument Len = *duljinazapisa* nije određen. Ako uključite duljinu zapisa u naredbu Open za binarni pristup, ona se zanemaruje.

Spremanje informacija u polja duljine varijable

Kako bi najbolje cijenili binarni pristup, razmotrite teoretsku datoteku Podaci Djelatnika. Ova datoteka koristi zapise nepromjenjive duljine i polja za spremanje informacija o djelatnicima.

```
Type Osoba
  ID                    As Integer
  MjesečnaPlaća         As Currency
  DatumZadnjegPregleda  As Long
  Ime                   As String * 15
  Prezime               As String * 15
  Titula                As String * 15
  Napomene              As String * 150
End Type
```

Neovisno o stvarnom sadržaju polja, svaki zapis u toj datoteci zauzima 209 bajtova.

Korištenjem binarnog pristupa možete smanjiti upotrebu prostora na disku. Pošto takav pristup ne zahtjeva polja nepromjenjive duljine, u određivanju tipa mogu se izostaviti parametri duljine stringova.

```
Type Osoba
  ID                    As Integer
  MjesečnaPlaća         As Currency
  DatumZadnjegPregleda  As Long
  Ime                   As String
  Prezime               As String
  Titula                As String
  Napomene              As String
End Type

Public Djel As Osoba            ‘ Određivanje zapisa.
```

Zapis svakog djelatnika u datoteci Podaci Djelatnika sada sprema samo točan broj potrebnih bajtova jer su polja duljine varijable. Loša strana binarnog ulaza/izlaza s poljima duljine varijable je što ne možete izravno pristupiti zapisima – morate pristupati zapisima po redu kako bi saznali duljinu svakog zapisa. Možete izravno tražiti položaj određenog bajta u datoteci, ali nema izravnog načina da znate koji je zapis na kojem položaju bajta ako su zapisi raznih duljina.

Za više informacija Za dodatne informacije o datotekama sa binarnim pristupom, pogledajte odlomak “Naredba Open” u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

15. POGLAVLJE

Oblikovanje u korist izvođenja i sukladnosti

U idealnom svijetu, svaki korisnik vaših aplikacija trebao bi imati računalo s najbržim mogućim procesorom, obilje memorije, neograničen prostor na disku i vatreno brzu mrežnu povezanost. Stvarnost određuje da će za većinu korisnika, stvarno izvođenje aplikacije biti ograničeno na jedan ili više gore navedenih čimbenika. Kako stvarate sve veće i usavršenije aplikacije, količina memorije koju troši aplikacija i brzina kojom se izvodi postaje sve značajnija. Možete odlučiti da trebate optimizirati svoju aplikaciju tako da ju napravite manjom te ubrzavajući proračunavanja i prikazivanja.

Dok oblikujete i programirate svoju aplikaciju, postoji niz tehnika koje mogu biti upotrebljene za optimizaciju izvođenja. Neke tehnike mogu vam pomoći u stvaranju vrže aplikacije; druge će vam pomoći da ju napravite manjom. U ovom poglavlju naučit ćete neke od uobičajenih trikova optimizacije koje možete upotrijebiti u svojoj vlastitoj aplikaciji.

Visual Basic dijeli većinu svojih osobina programskog jezika sa Visual Basicom za aplikacije (Visual Basic for Applications, VBA), koji je uključen u programski paket Microsoft Office te puno drugih aplikacija. Visual Basic, verzija Scripting (VBScript), jezik skripti za Internet, je također podskup programskog jezika Visual Basic. Ako također razvijate nešto u Visual Basicu za aplikacije ili VBScriptu, vjerojatno ćete htjeti podijeliti dio vašeg koda između tih jezika.

Ovo poglavlje raspravlja o razlikama između tri verzije Visual Basica i pruža neke savjete za stvaranje prenosivog koda.

Sadržaj

- Razumijevanje optimizacije
- Optimiziranje brzine
- Optimiziranje veličine
- Optimiziranje objekata
- Prevedene protiv tumačenih aplikacija
- Sukladnost s ostalim Microsoft aplikacijama

Primjer aplikacije: Optimize.vbp

Većina tehnika optimizacije iz ovog poglavlja pokazana je u primjeru aplikacije Optimize.vbp koja se nalazi u direktoriju Samples.

Razumijevanje optimizacije

Optimizacija može biti shvaćena i kao znanost i kao umjetnost. Znanost su tehnike optimizacije; umjetnost je određivanje gdje i kad optimizacija treba biti primijenjena. U pravilu, optimizacija je "postupak proizvodnje učinkovitijih (manjih i/ili bržih) aplikacija kroz odabir i oblikovanje građe podataka, algoritama i nizova instrukcija".

Uobičajena je zabluda da je optimizacija postupak koji se poduzima na kraju ciklus razvoja. Kako bi stvorili stvarno optimiziranu aplikaciju, morate ju optimizirati dok ju razvijate. Trebate pažljivo odabirati vaše algoritme, odvagujući brzinu i veličinu te ostala ograničenja; trebate oblikovati teorije o tome koji će dijelovi vaše aplikacije biti brzi ili spori, veliki ili sažeti; trebate isprobati te teorije tijekom programiranja.

Prvi korak u procesu optimizacije je određivanje vašeg cilja. Možete optimizirati svoju aplikaciju za puno različitih osobina:

- Stvarna brzina (kako brzo vaša aplikacija stvarno proračunava ili izvodi druge operacije).
- Brzina prikazivanja (kako brzo vaša aplikacija iscrtava ekran).
- Zamijećena brzina (kako brzo vaša aplikacija izgleda da radi; to je često povezano s brzinom prikazivanja, ali ne uvijek i s stvarnom brzinom).
- Veličina u memoriji.
- Veličina grafike (ovo izravno utječe na veličinu u memoriji, ali često ima dodatna grananja kod rada u Microsoft Windowsima).

Rijetko, međutim, možete optimizirati za više osobina. Tipično, pristup koji optimizira veličinu utječe na brzinu; slično tome, aplikacija koja je optimizirana za brzinu često je veća od njenog sporijeg duplikata. Zbog tog razloga, preporučene tehnike optimizacije u jednom području mogu izravno biti suprotne savjetima za drugo područje.

Važno je zapamtiti da optimizacija nije uvijek potpuno korisna. Ponekad će promjene koje napravite kako bi ubrzali ili usporili vašu aplikaciju rezultirati kodom kojeg je teže održavati ili u njemu otkrivati pogreške. Neke tehnike optimizacije u suprotnosti su s praksom strukturiranog programiranja, što može uzrokovati probleme kad pokušate proširiti vašu aplikaciju u budućnosti ili ju pokušate uklopiti u druge aplikacije.

Pri oblikovanju strategije optimizacije za vašu aplikaciju postoje tri stvari koje treba razmotriti: treba znati što optimizirati, gdje optimizirati i kada prestati.

Znati što optimizirati: razumijevanje stvarnog problema

Ako ne počnete s jasnim ciljem na umu, možete izgubiti puno vremena optimizirajući pogrešne stvari. Vaš cilj treba biti temeljen na potrebama i očekivanjima korisnika. Na primjer, brzina može biti glavna preokupacija kod proračunavanja poreza u aplikaciji na mjestu prodaje, dok će veličina aplikacije biti najvažnija za aplikaciju koja će biti preuzeta putem Interneta. Ključ razvijanja dobre strategije optimizacije je razumijevanje stvarnog problema na koji se odnosi optimizacija.

Iako će vaša strategija optimizacije postići određeni cilj, pomaže ako o optimizaciji razmišljate tijekom postupka razvijanja. Kad pišete programski kod, možete puno naučiti jednostavnim prolazom kroz vaš kod korak po korak te pažljivim razmišljanjem o tome što se stvarno događa. Možete zaboraviti da postavljanje svojstava uzrokuje pojavljivanje događaja, te ako u tim potprogramima događaja postoji puno koda, bezazlena linija koda može uzrokovati strahovito kašnjenje u vašoj aplikaciji. Čak i ako je vaš temeljni cilj veličina, optimiziranje brzine može ponekad biti ostvareno bez povećanja veličine koda.

Znati gdje optimizirati: velika korist s malim naporom

Ako ste poput većine programera, ne možete si priuštiti vrijeme za optimizaciju svega u svojoj aplikaciji. Ponekad je korisno razmišljati o posjedovanju “proračuna za optimizaciju”. Na kraju krajeva, dodatno vrijeme jednako je dodatnim troškovima razvoja. Gdje možete utrošiti vaše vrijeme kako bi dobili najveći povrat vašeg ulaganja? Očito ćete se željeti usredotočiti na područja koja izgledaju najsporija ili najbrža, ali kako bi povećali rezultate vaših napora, trebate se usredotočiti na kod gdje će malo posla napraviti velike razlike.

Na primjer, ako je brzina vaš temeljni cilj, tijela petlji su obično dobro mjesto za početak. Uvijek kad ubrzavate operacije unutar petlje, poboljšanje se množi brojem koliko je puta petlja izvedena. Za petlje s velikim brojem ponavljanja, samo jedna operacija s stringom manje u tijelu petlje može napraviti veliku razliku. Isto načelo također vrijedi za često pozivane potprograme.

Znati gdje prestati: odmjeravanje rezultata

Ponekad stvari nisu vrijedne optimiziranja. Na primjer, pisanje razrađene i brze rutine sortiranja je beskorisno ako sortirate samo desetak stavki. Moguće je sortirati stvari dodavajući ih sortiranom okviru s popisom te ih zatim čitati natrag po redu. S velikim brojem stavki to je jezivo neučinkovito, ali ako nema puno stavki tada je jednako brzo kao i svaki drugi postupak, a kod je zadivljujuće jednostavan (iako pomalo nejasan).

Postoje drugi slučajevi kad je optimizacija uzaludan napor. Ako je vaša aplikacija konačno ograničena brzinom vašeg diska ili mreže, postoji malo toga što možete napraviti u vašem kodu kako bi ubrzali stvari. Umjesto toga trebate razmišljati o načinima kako ta kašnjenja učiniti manje problematičnima za vaše korisnike; trake napretka im kazuju da vaš kod nije jednostavno srušen, pripremanje podataka tako da rjeđe vide kašnjenja, dopuštanje korištenja drugih aplikacija dok čekaju i tako dalje.

Za više informacija Pogledajte odlomak "Prekidanje pozadinske obrade" u 11. poglavlju "Odgovaranje na događaje miša i tipkovnice".

Optimiziranje brzine

Brzina je često glavni odlučujući čimbenik u općem utisku korisnika te zadovoljstvu aplikacijom. Na žalost, većina stvari koje utječu na brzinu aplikacije su izvan vašeg nadzora kao programera: brzina procesora, manjak odgovarajuće memorije, ili brzina podatkovnih veza. Zbog tog razloga, često je neophodno optimizirati vašu aplikaciju tako da se brže izvodi (ili barem izgleda da se brže izvodi).

Optimizacije brzine mogu biti podijeljene u tri opće kategorije: stvarna brzina (stvarno vrijeme potrošeno na izvođenje proračunavanja i izvođenje koda), brzina prikazivanja (vrijeme potrošeno na prikazivanje grafike ili iscrtavanje ekrana) i zamijećena brzina (kako brzo vaša aplikacija izgleda da radi). Tipovi optimizacije koje ćete stvarno upotrijebiti ovise o tipu i namjeni aplikacije – nisu sve optimizacije prikladne ili korisne u svim slučajevima.

s svakim tipom optimizacije, trebate odvagnuti moguće prednosti i cijenu postizanja. Nema puno smisla potrošiti sate optimizirajući potprogram koji je rijetko pozivan. Ustanovite područja gdje će poboljšanja brzine utjecati (i biti uočena) na većinu korisnika, kao što je početno vrijeme učitavanja za aplikaciju.

Optimiziranje koda

Osim ako ne radite zadatke kao što je stvaranje fraktala, vaše aplikacije najvjerojatnije neće biti ograničene stvarnom brzinom obrađivanja vašeg koda. Obično su drugi čimbenici – kao brzina slike, kašnjenja mreže ili aktivnosti diska – ograničavajući čimbenik u vašim aplikacijama. Na primjer, kad se forma sporo učitava, uzrok će prije biti broj kontrola i grafika na formi nego spori programski kod u događaju Form_Load. Međutim, možete pronaći mjesta u vašoj aplikaciji gdje je brzina vašeg programskog koda ograničavajući čimbenik, posebno kod potprograma koji su često pozivani. Kad je to slučaj, postoji nekoliko tehnika koje možete upotrijebiti kako bi povećali stvarnu brzinu vaših aplikacija:

- Izbjegavajte korištenje varijabli tipa Variant.
- Upotrebljavajte varijable tipa Long integer i cjelobrojnu matematiku.
- Spremajte često korištena svojstva u varijable.
- Upotrebljavajte varijable na razini modula umjesto varijabli tipa Static.

- Zamijenite pozive potprograma s ugrađenim kodom.
- Upotrebljavajte konstante kad god je to moguće.
- Proslјeđujte argumente izrazom ByVal umjesto izrazom ByRef.
- Upotrebljavajte tipske neobavezne argumente.
- Iskoristite prednost zbirki.

Čak i ako ne optimizirate vaš programski kod za brzinu, pomaže ako ste svjesni ovih tehnika i njihovih pratećih načela. Ako vam prijeđe u naviku odabirati učinkovitije algoritme dok programirate, rastući dobitak može na kraju iznositi prilično sveobuhvatno poboljšanje u brzini.

Izbjegavajte korištenje varijabli tipa Variant

Podrazumijevan tip podatka u Visual Basicu je tip Variant. To je spretno za programere početnike i za aplikacije gdje brzina obrade nije u pitanju. Ako pokušavate optimizirati stvarnu brzinu vaše aplikacije, međutim, trebate izbjegavati varijable tipa Variant. Pošto Visual Basic pretvara varijable tipa Variant u odgovarajući tip podatka tijekom izvođenja aplikacije, operacije koje sadrže ostale jednostavne tipove podataka uklanjaju taj dodatni korak i brže su od njihovih istoznačnih varijabli tipa Variant.

Dobar način izbjegavanja varijabli tipa Variant je korištenje izraza Option Explicit, koji vas prisiljava da odredite sve svoje varijable. Kako bi upotrijebili izraz Option Explicit, potvrdite kontrolnu kućicu Require Variable Declaration na kartici Editor dijaloškog okvira Options, dostupnog iz izbornika Tools.

Budite oprezni kad određujete višestruke varijable: ako ne upotrijebite uvjet As *tip*, one će zapravo biti određene kao tip Variant. Na primjer, u sljedećem određivanju, varijable X i Y su tipa Variant:

```
Dim X, Y, Z As Long
```

U prerađenom određivanju, sve tri varijable su tipa Long:

```
Dim X As Long, Y As Long, Z As Long
```

Za više informacija Kako bi naučili više o tipovima podataka u Visual Basicu, pogledajte odlomak “Tipovi podataka” u 5. poglavlju “Osnove programiranja”.

Upotrebljavajte varijable tipa Long integer i cjelobrojnu matematiku

Za aritmetičke operacije izbjegavajte varijable tipa Currency, Single i Double. Upotrebljavajte varijable tipa Long integer kad god možete, posebno u petljama. Tip Long integer je izvorni tip podatka 32-bitne centralne procesorske jedinice (CPU), tako da su operacije s tim tipom vrlo brze; ako ne možete upotrijebiti varijablu tipa Long, tipovi podataka Integer ili Byte su sljedeći najbolji izbor. U većini slučajeva, možete upotrijebiti cjele brojeve tipa Long kad bi inače mogla biti zatražena vrijednost s plivajućim

zarezom. Na primjer, ako svojstvo ScaleMode uvijek postavljate na twipove ili piksele za sve vaše forme i kontrole slike, možete upotrebljavati cijele brojeve tipa Long za sve vrijednosti veličine i položaja, za kontrole i grafičke postupke.

Kod izvođenja dijeljenja, upotrijebite operator cjelobrojnog dijeljenja (/) ako ne trebate decimalni rezultat. Matematika cijelih brojeva je uvijek brža od matematike s plivajućim zarezom jer ne zahtjeva prebacivanje operacije matematičkom koprocesoru. Ako trebate matematiku s decimalnim vrijednostima, tip podatka Double je brži od tipa podatka Currency.

U sljedećoj tablici poredani su tipovi brojčanih podataka po brzini računanja.

tip brojčanog podatka	brzina
Long	najbrži
Integer	
Byte	
Single	
Double	
Currency	najsporiji

Spremajte često korištena svojstva u varijable

Vrijednost varijable možete dobiti i postaviti brže od vrijednosti svojstva. Ako često dohvaćate vrijednost svojstva (kao u petlji), vaš programski kod će se izvoditi brže ako svojstvo dodijelite varijabli izvan petlje te zatim upotrebljavate varijablu umjesto svojstva. Varijable su općenito 10 do 20 puta brže od svojstava istog tipa.

Nikad ne uzimajte vrijednost svakog danog svojstva više od jednom unutar potprograma osim ako znate da je vrijednost promijenjena. Umjesto toga, dodijelite vrijednost svojstva varijabli i upotrijebite varijablu u cijelom nastavku koda. Na primjer, kod poput ovog je vrlo spor:

```
For i = 0 To 10
  picIkona(i).Left = picPaleta.Left
Next i
```

Prerađen, ovaj kod je puno brži:

```
picLijevo = picPaleta.Lijevo
For i = 0 To 10
  picIkona(i).Left = picLijevo
Next i
```

Slično tome, kod poput ovog...

```
Do Until EOF(F)
  Line Input #F, idućaLinija
  Text1.Text = Text1.Text + idućaLinija
Loop
```

... je puno sporiji od ovog:

```
Do Until EOF(F)
  Line Input #F, idućaLinija
  spremnikVar = spremnikVar & idu}aLinija & vbCrLf
Loop
Text1.Text = spremnikVar
```

Međutim, ovaj kod izvodi jednak posao i još je brži:

```
Text1.Text = Input(F, LOF(F))
```

Kao što možete vidjeti, postoji nekoliko postupaka za postizanje istog zadatka; najbolji algoritam je također i najbolja optimizacija.

Ista tehnika može se primijeniti za vraćanje vrijednosti iz funkcija. Spremanjem povratnih vrijednosti funkcije izbjegavaju se česti pozivi dinamički povezive biblioteke (DLL) izvođenja, Msvbvm60.dll.

Upotrebljavajte varijable na razini modula umjesto varijabli tipa Static

Dok su varijable određene tipom Static korisne za spremanje vrijednosti kod višestrukih izvođenja potprograma, sporije su od lokalnih varijabli. Spremanjem iste vrijednosti u varijablu na razini modula vaš će se potprogram izvoditi brže. Zapamtite, međutim, kako ćete trebati osigurati da je samo jednom potprogramu dozvoljeno mijenjati varijablu na razini modula. Loša strana toga je da će vaš kod biti manje čitljiv i teži za održavanje.

Zamijenite pozive potprograma s ugrađenim kodom

Iako upotreba potprograma čini vaš programski kod modularnijim, izvođenje svakog poziva potprograma uvijek uključuje ponešto dodatnog posla i vremena. Ako imate petlju koja poziva potprogram puno puta, možete odstraniti tu nadgradnju uklanjanjem poziva potprograma i postavljanjem sadržaja potprograma izravno unutar petlje. Ako postavite isti programski kod u nekoliko petlji, međutim, umnoženi kod povećava veličinu vaše aplikacije. On također povećava mogućnost da nećete zapamtiti ažurirati svaki primjer umnoženog koda kad radite promjene.

Slično tome, pozivanje potprograma koji se nalazi u istom modulu je brže od pozivanja istog potprograma u odvojenom modulu tipa .BAS; ako isti potprogram treba biti pozivan iz raznih modula, dobitak je poništen.

Upotrebljavajte konstante kad god je to moguće

Korištenje konstanti čini izvođenje vaše aplikacije bržim. Konstante također čine vaš kod čitljivijim i lakšim za održavanje. Ako u vašem kodu postoje stringovi ili brojevi koji se ne mijenjaju, odredite ih kao konstante. Konstante se rješavaju jednom kad se vaša aplikacija prevodi, s prikladnom vrijednosti zapisanom u kodu. S varijablama, međutim, svaki put kad se aplikacija izvodi i pronađe varijablu, treba dobiti trenutnu vrijednost varijable.

Uvijek kad je moguće, upotrijebite ugrađene konstante ispisane u pretraživaču objekata umjesto stvaranja svojih vlastitih. Ne trebate se brinuti o uključivanju modula koji sadrže neupotrebljene konstante u vašoj aplikaciji; kad napravite izvršnu datoteku, neupotrebljene konstante se uklanjaju.

Prosljeđujte nepromijenjene argumente izrazom ByVal umjesto izrazom ByRef

Kad pišete potprograme tipa Sub ili Function koji uključuju nepromijenjene argumente, brže je proslijediti argumente vrijednošću (ByVal) nego upućivanjem (ByRef). Argumenti u Visual Basicu se u pravilu prosljeđuju izrazom ByRef, ali relativno malo potprograma stvarno mijenja vrijednosti svojih argumenata. Ako ne trebate mijenjati argumente unutar potprograma, odredite ih izrazom ByVal, kao u sljedećem primjeru:

```
Private Sub NapraviNešto(ByVal strIme As String, _
ByVal intStarost As Integer)
```

Upotrebljavajte tipske neobavezne argumente

Tipski neobavezni argumenti mogu poboljšati brzinu poziva potprograma tipa Sub ili Function. U prethodnim verzijama Visual Basica, neobavezni argumenti su morali biti tipa Variant. Ako vaš potprogram ima argumente tipa ByVal, kao u sljedećem primjeru, 16 bajtova tipa Variant će biti postavljeno na stog.

```
Private Sub NapraviNešto(ByVal strIme As String, _
Optional ByVal vntStarost As Variant, _
Optional ByVal vntTežina As Variant)
```

Vaša funkcija koristi manje prostora stoga po pozivu, i manje se podataka pomiče u memoriju, ako upotrijebite tipske neobavezne argumente:

```
Private Sub NapraviNešto(ByVal strIme As String, _
Optional ByVal intStarost As Integer, _
Optional ByVal intTežina As Integer)
```

Tipski neobavezni argumenti su brži za pristup od argumenata tipa Variant, a kao dodatak, dobit ćete poruku greške tijekom prevođenja ako pribavite informacije pogrešnog tipa podatka.

Iskoristite prednost zbirki

Sposobnost određivanja i korištenja zbirki objekata je moćna osobina Visual Basica. Iako zbirke mogu biti vrlo korisne, za najbolje izvođenje trebate ih ispravno upotrebljavati:

- Upotrebljavajte izraz For Each...Next radije nego izraz For...Next.
- Izbjegavajte korištenje argumenata Before i After kad dodajete objekte zbirci.
- Upotrebljavajte zbirke po ključu prije nego matrice za grupe objekata istog tipa.

Zbirke vam omogućuju ponavljanje kroz njih korištenjem cjelobrojne petlje For...Next. Međutim, konstrukcija For Each...Next je čitljivija i u većini slučajeva brža. Ponavljanje For Each...Next je ostvareno kreatorom zbirke, pa će se stvarna brzina razlikovati među raznim objektima zbirke. Međutim, izraz For Each...Next će rijetko biti sporiji od izraza For...Next jer je najjednostavnije ostvarivanje linearan stil ponavljanja For...Next. U nekim slučajevima izvršitelj može upotrijebiti usavršenija ostvarivanja od linearnog ponavljanja, pa će izraz For Each...Next biti puno brži.

Brže je dodati objekte zbirci ako ne upotrebljavate argumente Before i After. Ti argumenti zahtijevaju od Visual Basica da pronađe drugi objekt u zbirci prije nego što može dodati novi objekt.

Kad imate grupu objekata istog tipa, vaš uobičajeni odabir je upravljanje njima u zbirci ili matrici (ako su različitih tipova, zbirka je vaš jedini izbor). S gledišta brzine, koji ćete pristup trebati odabrati ovisi o tome kako namjeravate pristupati objektima. Ako možete pridružiti jedinstven ključ svakom objektu, tada je zbirka najbrži izbor. Korištenje ključa za dohvaćanje objekta iz zbirke je brže od uzastopnog pretraživanja matrice. Međutim, ako nemate ključeve i zbog toga ćete uvijek pretraživati objekte, matrica je bolji izbor. Matrice su brže za uzastopno pretraživanje od zbirki.

Za mali iznos objekata, matrice koriste manje memorije i često mogu biti brže pretražene. Stvaran broj objekata kad zbirke postaju učinkovitije od matrica je oko 100 objekata; međutim, to se može razlikovati ovisno o brzini procesora i raspoloživoj memoriji.

Za više informacija Pogledajte “Korištenje zbirki umjesto matrica” u 8. poglavlju “Više o programiranju”.

Mjerenje izvođenja

Određivanje najboljeg algoritma za neku situaciju nije uvijek očigledno. Ponekad ćete željeti ispitati vaše teorije; to se može lako napraviti stvaranjem jednostavne aplikacije za mjerenje izvođenja, kao što je pokazano u nastavku. Primjer aplikacije Optimize.vbp također sadrži primjere nekoliko raznih primjera ispitivanja.

Kako stvoriti aplikaciju za ispitivanje izvođenja

1. Otvorite novi .exe projekt.
2. Stvorite formu s dva naredbena gumba: Command1 i Command2.
3. U događaj Command1_Click dodajte sljedeći programski kod:

```
Private Sub Command1_Click()
  Dim dblPočetak As Double
  Dim dblKraj As Double
  Dim i As Long

  dblPočetak = Timer           ‘ Dobivanje vremena početka.
```

```
    For i = 0 To 9999
       Rutina za ispitivanje            ‘ Ovdje dodajte vašu rutinu.
    Next

    dblEnd = Timer                      ‘ Dobivanje vremena kraja.

    Debug.Print dblEnd - dblStart       ‘ Prikaz proteklog vremena.
End Sub
```

4. Dodajte isti kod u događaj Command2_Click, zamjenjujući drugu verziju vaše rutine unutar petlje.
5. Pokrenite aplikaciju i pratite rezultate u prozoru za neposredan upis naredbi.

Ovaj primjer podrazumijevano svojstvo klase Timer Visual Basica za mjerenje izvođenja rutine unutar petlje. Postavljanjem vašeg programskog koda unutar petlje za svaki naredbeni gumb, možete brzo usporediti izvođenje dva algoritma. Programski kod može biti unutar petlje ili se može pozivati od drugih potprograma.

Možete trebati eksperimentirati s različitim vrijednostima za gornje granice brojača petlje, posebno za brze rutine. Obavezno izvedite svaku verziju nekoliko puta kako bi dobili prosjek; rezultati se mogu razlikovati od jednog do drugog izvođenja.

Vašu aplikaciju možete također optimizirati povećanjem brzine pristupa podacima.

Optimiziranje brzine prikaza

Zbog grafičke prirode Microsoft Windowsa, brzina grafike i ostalih operacija prikazivanja može biti presudna za *zamijećenu brzinu* aplikacije. Što se brže forme pojavljuju i iscrtavaju, to će vaša aplikacija izgledati brža korisniku. Postoji nekoliko tehnika koje možete upotrebljavati za ubrzavanje vidljive brzine vaše aplikacije, uključujući:

- Postavite svojstvo ClipControls spremnika na False.
- Upotrebljavajte svojstvo AutoRedraw prikladno.
- Upotrebljavajte kontrole slike umjesto kontrola okvira za sliku.
- Sakrijte kontrole kad postavljate svojstva kako bi izbjegli višestruka iscrtavanja.
- Koristite postupak Line umjesto postupka PSet.

Postavite svojstvo ClipControls spremnika na False

Osim ako ne koristite grafičke postupke (Line, PSet, Circle i Print), trebali bi postaviti svojstvo ClipControls na False za formu te sve kontrole okvira i okvira za sliku (to može uzrokovati nepredvidljive rezultate ako vaš kod sadrži grafičke postupke koji crtaju iza ostalih kontrola). Kad je svojstvo ClipControls postavljeno na False, Visual Basic ne precrtava kontrole pozadinom prije ponovnog iscrtavanja samih kontrola. Na formama koje sadrže puno kontrola, rezultirajuća poboljšanja brzine su značajna.

Za više informacija Pogledajte "Slojevitost grafike s svojstvima AutoRedraw i ClipControls " u 12. poglavlju "Rad s tekstom i grafikom".

Upotrebljavajte svojstvo AutoRedraw prikladno

Kad je svojstvo AutoRedraw postavljeno na True za formu ili kontrolu, Visual Basic održava sliku tako da ponovno iscrtava tu formu ili kontrolu. Iako to poboljšava brzinu jednostavnih ponovnih iscrtavanja (na primjer, kad je forma ili kontrola otkrivena nakon što je pomaknut prozor koji ju je pokrivao), usporava grafičke postupke. Visual Basic mora izvesti grafičke postupke na slici svojstva AutoRedraw te zatim kopirati cijelu sliku na ekran. Ovaj postupak također troši značajnu količinu memorije.

Ako vaša aplikacija stvara složenu grafiku, ali ju ne mijenja često, postavljanje svojstva AutoRedraw na True je prikladan postupak. Međutim, ako vaša aplikacija crta grafiku koje je često mora mijenjati, postići ćete bolje izvođenje ako postavite svojstvo AutoRedraw na False i izvedete grafičke postupke za formu ili kontrolu u događaju Paint.

Za više informacija Pogledajte "Slojevitost grafike s svojstvima AutoRedraw i ClipControls " u 12. poglavlju "Rad s tekstom i grafikom".

Upotrebljavajte kontrole slike umjesto kontrola okvira za sliku

Ova optimizacija povećava brzinu i smanjuje veličinu vaše aplikacije; upotrijebite ju kad god možete. Kad jednostavno prikazujete slike i reagirate na događaje klika i akcije miša nad njima, upotrijebite kontrolu slike umjesto okvira za sliku. Ne upotrebljavajte okvir za sliku osim ako trebate sposobnosti koje pruža samo okvir za sliku, kao što su grafički postupci, sposobnost sadržavanja drugih kontrola ili dinamička razmjena podataka (dynamic data exchange, DDE).

Sakrijte kontrole kad postavljate svojstva kako bi izbjegli višestruka iscrtavanja

Svako ponovno iscrtavanje košta. Što manje ponovnih iscrtavanja Visual Basic treba obaviti, to će vaša aplikacija izgledati bržom. Jedan način smanjivanja broja ponovnih iscrtavanja je stvaranje kontrola nevidljivima dok njima upravljate. Na primjer, pretpostavimo da želite promijeniti veličinu nekoliko okvira s tekstom u događaju Resize forme:

```
Sub Form_Resize()
  Dim i As Integer, sVisina As Integer
  sVisina = ScaleHeight / 4
  For i = 0 To 3
    lstPrikaz(i).Move 0, i * sVisina, _
    ScaleWidth, sVisina
  Next
End Sub
```

Ovo će stvoriti četiri odvojena ponovna iscrtavanja, po jedno za svaki okvir s tekstom. Možete smanjiti broj ponovnih iscrtavanja postavljanjem svih okvira s tekstom unutar okvira za sliku, i sakrivanjem okvira za sliku prije nego što pomaknete okvire s tekstom i promijenite im veličinu. Zatim, kad ponovno učinite vidljivim okvir za sliku, svi okviri s tekstom će biti iscrtani u jednom prolazu:

```
Sub Form_Resize()
  Dim i As Integer, sVisina As Integer
  picSpremnik.Visible = False
  picSpremnik.Move 0, 0, ScaleWidth, ScaleHeight
  sVisina = ScaleHeight / 4
  For i = 0 To 3
    lstPrikaz(i).Move 0, i * sVisina, _
    ScaleWidth, sVisina
  Next
  picSpremnik.Visible = True
End Sub
```

Uočite da ovaj primjer upotrebljava postupak Move umjesto postavljanja svojstava Top i Left. Postupak Move postavlja oba svojstva u jednoj operaciji, uštedujući dodatna ponovna iscrtavanja.

Koristite postupak Line umjesto postupka PSet

Postupak Line je brži od niza postupaka PSet. Izbjegavajte korištenje postupka PSet i gomilajte točke u jednom postupku Line. Kontrole lika i linije su prikladne za jednostavne grafičke elemente koji se rijetko mijenjaju; složenom grafikom, ili grafikom koja se brzo mijenja, općenito se bolje rukuje s grafičkim postupcima.

Optimiziranje zamijećene brzine

Često subjektivna brzina vaše aplikacije ima malo zajedničkog s stvarnom brzinom izvođenja koda. Za korisnika, aplikacija koja se hitro pokrene, brzo iscrtava, i pruža stalne povratne informacije izgleda "poletnije" od aplikacije koja se samo "objesi" kad se uzburka tijekom rada. Možete upotrijebiti niz raznih tehnika kako bi vašoj aplikaciji dali "polet":

- Držite forme skrivenima, ali učitanima.
- Učitavajte podatke unaprijed.
- Koristite mjerače vremena za rad u pozadini.
- Upotrebljavajte pokazivače napretka.
- Ubrzajte početak vaše aplikacije.

Držite forme skrivenima, ali učitanima

Skrivanje formi umjesto njihovog izbacivanja je trik koji postoji od ranih dana Visual Basica 1.0, ali je i dalje učinkovit. Očigledan nedostatak ove tehnike je količina memorije koju troše učitane forme, ali to može biti prevladano ako si možete priuštiti trošak memorije i ako je stvaranje brzog pojavljivanja formi od najveće važnosti.

Učitavajte podatke unaprijed

Vidljivu brzinu vaše aplikacije možete također poboljšati dohvaćanjem podataka unaprijed. Na primjer, ako trebate otići do diska kako bi učitali prvu od nekoliko datoteka, zašto ne bi učitali onoliko datoteka koliko možete? Osim ako datoteke nisu izuzetno male, korisnik će vidjeti kašnjenje u svakom slučaju. Povećano vrijeme potrošeno na učitavanje dodatnih datoteka će vjerojatno proći neprimjećeno, i nećete ponovno trebati odgađati rad korisnika.

Koristite mjerače vremena za rad u pozadini

U nekim aplikacijama možete napraviti značajan posao dok čekate na akciju korisnika. Najbolji način ostvarivanja toga je kroz kontrolu mjerača vremena. Upotrijebite statičke varijable (ili na razini modula) kako bi pratili vaš napredak, i napravite vrlo mali dio posla svaki put kad istekne interval mjerača vremena. Ako očuvate vrlo malu količinu obavljenog posla u svakom događaju mjerača vremena, korisnik neće vidjeti nikakav utjecaj na pristupačnost aplikacije i možete unaprijed dohvaćati podatke ili raditi druge stvari koje će dodatno ubrzati vašu aplikaciju.

Za više informacija Kako bi naučili više o kontroli mjerača vremena, pogledajte "Korištenje kontrole mjerača vremena" u 7. poglavlju "Korištenje standardnih kontrola Visual Basica". Za raspravu o pozadinskoj obradi, pogledajte "Prekidanje pozadinske obrade" u 11. poglavlju "Odgovaranje na događaje miša i tipkovnice".

Upotrebljavajte pokazivače napretka

Kad ne možete izbjeći dug zastoj u vašoj aplikaciji, trebate dati korisniku neki znak da se vaša aplikacija nije jednostavno objesila. Windowsi 95/98 upotrebljavaju standardnu traku napretka kako bi to naznačili korisniku. Možete upotrijebiti kontrolu trake napretka (ProgressBar) u dodatku Microsoft Windows Common Controls uključenom u verzije Professional i Enterprise Visual Basica. Upotrijebite funkciju DoEvents na strateškim mjestima, posebno svaki put kad ažurirate vrijednost trake napretka, kako bi dozvolili vašoj aplikaciji da se ponovno iscrta dok korisnik radi druge stvari.

Na samom kraju, trebali bi prikazivati pokazivač čekanja kako bi naznačili zastoj, postavljanjem svojstva MousePointer forme na vrijednost vbHourglass (11).

Ubrzajte početak vaše aplikacije

Vidljiva brzina je najvažnija kad se pokreće vaša aplikacija. Prvi dojam korisnika o brzini aplikacije mjeri se kako brzo može vidjeti nešto nakon što klikne na ime aplikacije u izborniku Start. s raznim dinamičkim bibliotekama tijeka izvođenja koje trebaju biti učitane za Visual Basic za aplikacije, ActiveX kontrole, i tako dalje, neizbježan je određeni zastoj s svakom aplikacijom. Međutim, postoje neke stvari koje možete napraviti kako bi korisniku dali odgovor što je prije moguće:

- Upotrebljavajte postupak Show u događaju Form_Load.
- Pojednostavite vašu početnu formu.
- Ne učitavajte module koje ne trebate.
- Pokrenite manju Visual Basic aplikaciju na početku za učitavanje dinamičkih biblioteka tijeka izvođenja unaprijed.

Upotrebljavajte postupak Show u događaju Form_Load

Kad se forma učita prvi put, sav programski kod u događaju Form_Load se izvodi prije nego što se prikaže forma. Možete promijeniti takvo ponašanje korištenjem postupka Show u kodu događaja Form_Load, dajući korisniku nešto na što će gledati dok se izvodi ostatak koda u događaju. Iza postupka Show postavite funkciju DoEvents kako bi osigurali iscrtavanje forme:

```
Sub Form_Load()
  Me.Show                ‘ Prikaz početne forme.
  DoEvents               ‘ Osiguravanje iscrtavanja početne forme.
  Load GlavnaForma       ‘ Učitavanje glavne forme aplikacije.
  Unload Me              ‘ Izbacivanje početne forme.
  GlavnaForma.Show       ‘ Prikaz glavne forme.
End Sub
```

Pojednostavite vašu početnu formu

Što je forma složenija, treba više vremena da se učita. Održavate vašu početnu formu jednostavnom. Većina aplikacija za Microsoft Windowse prikazuje jednostavan ekran s autorskim pravima (također znan kao uvodni ekran, splash screen) na početku; vaša aplikacija može napraviti isto. Što je manje kontrola na početnoj formi, i što manje koda one sadržavaju, to će se brže učitati i prikazati. Čak i ako se nakon toga odmah učitava druga, složenija forma, korisnik će znati da je aplikacija pokrenuta.

Za velike aplikacije možete trebati unaprijed učitati najčešće korištene forme na početku tako da one mogu biti prikazane odmah kad je potrebno. Zadovoljavajući način da to napravite je prikaz trake napretka na vašoj početnoj formi i njeno ažuriranje kad učitate svaku od ostalih formi. Pozovite funkciju DoEvents nakon učitavanja svake forme tako da se vaša početna forma može ponovno iscrtati. Kad su sve važne forme učitane, početna forma može prikazati prvu formu i izbaciti se. Svakako, svaka forma koju unaprijed učitate će izvesti kod u svojem događaju Form_Load, pa vodite računa da to ne uzrokuje probleme ili pretjerane zastoje.

Ne učitavajte module koje ne trebate

Visual Basic učitava module koda na zahtjev, prije nego sve odjednom na početku. To znači da ako nikad ne pozivate potprogram u modulu, taj modul nikad neće biti učitan. Suprotno tome, ako vaša početna forma poziva potprograme u raznim modulima, tada će svi ti moduli biti učitani kad se pokrene vaša aplikacija, što usporava stvari. Zbog toga bi trebali izbjegavati pozivati potprograme u drugim modulima iz vaše početne forme.

Pokrenite manju Visual Basic aplikaciju na početku za učitavanje dinamičkih biblioteka tijeka izvođenja unaprijed

Velik dio vremena potrebnog za pokretanje Visual Basic aplikacije troši se na učitavanje raznih dinamičkih biblioteka (DLL) tijeka izvođenja za Visual Basic, ActiveX i ActiveX kontrole. Svakako, ako su one već učitane, ništa od tog vremena neće biti potrošeno. Stoga, korisnici će vidjeti brže pokretanje vaše aplikacije ako već postoji druga aplikacija koja se izvodi i koristi neke ili sve te dinamičke biblioteke.

Jedan način za značajno poboljšanje izvođenja pokretanja vaših aplikacija je pružanje druge male, korisne aplikacije koju korisnik uvijek pokreće. Na primjer, možete napisati malu aplikaciju za prikaz kalendara i instalirati ju u početnu grupu Windowsa. Ona će se zatim automatski učitati pri podizanju sustava, i iako je korisna sama za sebe, također osigurava učitavanje raznih dinamičkih biblioteka izvođenja Visual Basica.

Na kraju, s verzijama Professional i Enterprise Visual Basica možete podijeliti vašu aplikaciju u aplikaciju glavnog kostura i nekoliko izvodivih sastavnih dijelova ili dinamičkih biblioteka. Manja glavna aplikacija će se brže učitati, i zatim može učitavati ostale dijelove prema potrebi.

Optimiziranje veličine

U prošlosti, raspoloživa memorija i sistemski izvori su često bili ograničavajući čimbenici u oblikovanju aplikacije. s 32-bitnim operativnim sustavima, kao što su Windows 95/98 i Windows NT, ti čimbenici su rijetko izvor brige za većinu programera u Visual Basicu. Međutim, postoji niz primjera gdje je smanjivanje veličine aplikacije i dalje važno.

Veličina je izuzetno važna za aplikacije koje će biti preuzete s Interneta ili prenesene kao dodaci u e-pošti. Za one koji nisu dovoljno sretni da imaju najbrža podatkovna povezivanja, prijenos datoteke od 1 megabajta može potrajati 1 sat ili više. Kao dodatak izvršnim .exe datotekama, većina aplikacija će zahtijevati dodatne .dll ili .ocx datoteke, povećavajući veličinu (i vrijeme) preuzimanja. U takvim primjerima, trebat ćete optimizirati veličinu vaše aplikacije na disku.

Čak i ako korisnici neće preuzimati aplikacije na taj način, obično je dobra ideja napraviti vašu aplikaciju što je moguće sažetijom. Manje aplikacije se brže učitavaju, i pošto troše manje memorije, možete istovremeno izvoditi dodatne aplikacije. Možete često poboljšati izvođenje optimiziranjem veličine vaše aplikacije u memoriji.

Smanjivanje veličine koda

Kad je smanjivanje veličine aplikacije važno, postoji niz tehnika koje možete primijeniti kako bi vaš kod napravili još sažetijim. Kao dodatak smanjenju veličine aplikacije u memoriji, većina tih optimizacija će također smanjiti veličinu izvršne datoteke. Kao dodatna korist, manja aplikacija će se brže učitavati.

Većina tehnika optimizacije veličine uključuje uklanjanje nepotrebnih elemenata iz vašeg koda. Visual Basic automatski uklanja određene elemente kad prevodite vašu aplikaciju. Nema razloga za ograničavanje duljine ili broja sljedećih elemenata:

- Imena identifikatora
- Komentari
- Prazne linije

Ni jedan od tih elemenata ne utječe na veličinu vaše aplikacije u memoriji kad se izvodi kao izvršna datoteka.

Ostali elementi, kao što su varijable, forme i potprogrami, zauzimaju prostor u memoriji. Obično je najbolje pojednostaviti ih. Postoji nekoliko tehnika koje možete upotrijebiti za smanjivanje memorije koju zauzima vaša aplikacija kad se izvodi kao izvršna datoteka. Sljedeće tehnike mogu smanjiti veličinu koda:

- Smanjite broj učitanih formi.
- Smanjite broj kontrola.
- Upotrebljavajte natpise umjesto okvira s tekstom.
- Čuvajte podatke u datotekama na disku i učitavajte ih samo kad je potrebno.
- Organizirajte svoje module.
- Razmislite o zamjenama za tipove podatka Variant.
- Upotrebljavajte dinamičke matrice i brišite ih za povratak memorije.
- Vratite prostor koji koriste stringovi ili varijable objekata.
- Uklonite mrtav kod i neupotrijebljene varijable.

Smanjite broj učitanih formi

Svaka učitana forma, vidljiva ili ne, troši značajnu količinu memorije (koja ovisi o broju i tipu kontrola na formi, veličini slika na formi i tako dalje). Učitavajte forme samo kad ih trebate prikazati, i izbacite ih (radije nego da ih sakrijete) kad ih više ne trebate. Zapamtite da svaki pokazivač na svojstva, postupke ili kontrole forme, ili varijabla forme određena s ključnom riječi New, uzrokuje učitavanje forme od strane Visual Basica.

Kad izbacite formu korištenjem postupka Unload, otpušta se samo dio memorije koju je zauzimala forma. Kako bi oslobodili cijelu memoriju, poništite sve pokazivače na formu korištenjem ključne riječi Nothing:

```
Set Forma = Nothing
```

Smanjite broj kontrola

Kad oblikujete vašu aplikaciju, pokušajte na formu postaviti što je moguće manje kontrola. Stvarno ograničenje ovisi o tipu kontrola kao i o raspoloživom sustavu, ali u praksi, svaka forma s velikim brojem kontrola će se izvoditi sporo. Srodna tehnika je korištenje matrica kontrola gdje je moguće, radije od postavljanja velikog broja kontrola istog tipa na formu tijekom izrade aplikacije.

Za više informacija Kako bi naučili više o matricama kontrola, pogledajte "Rad s matricama kontrola" u 7. poglavlju "Korištenje standardnih kontrola Visual Basica".

Upotrebljavajte natpise umjesto okvira s tekstom

Kontrole natpis troše manje izvora Windows od okvira s tekstom, pa bi trebali upotrebljavati natpise umjesto okvira s tekstom gdje je to moguće. Na primjer, ako na formi trebate skrivenu kontrolu za spremanje teksta, učinkovitije je upotrijebiti kontrolu natpisa.

Čak i forma za unos podataka koja zahtjeva brojna tekstualna polja može biti optimizirana korištenjem ove tehnike. Možete stvoriti natpis za svako polje i upotrijebiti jedan okvir s tekstom za unos, pomičući ga na položaj idućeg natpis u događaju LostFocus:

```
Private Sub Label1_LostFocus()
  ' Ažuriranje kontrole Label1
  Label1.Caption = Text1.Text
  ' Pomicanje okvira za tekst iznad idućeg natpisa
  Text1.Move Label2.Left, Label2.Top
  ' Ažuriranje sadržaja okvira s tekstom
  Text1.Text = Label2.Caption
End Sub
```

Kontrola natpis može izgledati kao okvir za tekst ako joj odredite svojstva BackColor i BorderStyle. Iako ovakva tehnika zahtjeva više koda, može značajno smanjiti korištenje sistemskih izvora za formu koja sadrži brojna polja.

Čuvajte podatke u datotekama na disku i učitavajte ih samo kad je potrebno

Podaci koje postavljate izravno u vašu aplikaciju tijekom izrade (kao svojstva ili konstantni stringovi i brojevi u vašem kodu) povećavaju memoriju koju vaša aplikacija troši tijekom izvođenja. Možete smanjiti količinu potrošene memorije učitavanjem

podataka iz datoteke s diska ili iz izvora tijekom izvođenja. To je posebno vrijedno za velike slike i stringove.

Za više informacija Za informacije o dodavanju izvora vašoj aplikaciji, pogledajte "Rad s datotekama izvora" u 8. poglavlju "Više o programiranju".

Organizirajte vaše module

Visual Basic učitava module na zahtjev – znači, učitava modul u memoriju samo kad vaš kod pozove neki od potprograma u tom modulu. Ako nikad ne pozovete potprogram u određenom modulu, Visual Basic nikad neće učitati taj modul. Postavljanjem srodnih potprograma u isti modul, Visual Basic će učitati module samo kad je potrebno.

Razmislite o zamjenama za tipove podatka Variant

Tip podatka Variant je izuzetno fleksibilan, ali je također veći od svih ostalih tipova podataka. Kad morate istisnuti svaki pojedini bit iz vaše aplikacije, razmislite o zamjeni varijabli tipa Variant, te posebno matrica varijabli tipa Variant, drugim tipovima podataka.

Svaka varijabla tipa Variant troši 16 bajtova, u usporedbi s 2 bajta za tip Integer ili 8 bajtova za tip Double. Varijable tipa String promjenjive duljine koriste 4 bajta plus 1 bajt po karakteru u stringu, ali svaka varijabla tipa Variant koja sadrži string troši 16 bajtova plus 1 bajt po karakteru u stringu. Pošto su tako velike, varijable tipa Variant su posebno problematične kad se koriste kao lokalne varijable ili argumenti potprograma, jer brzo troše prostor stoga.

U nekim slučajevima, međutim, korištenje ostalih tipova podataka prisiljava vas da dodate više koda kako bi nadoknadili gubitak fleksibilnosti koju pružaju tipovi podataka Variant, rezultat čega može biti nikakvo smanjivanje veličine.

Upotrebljavajte dinamičke matrice i brišite ih za povratak memorije

Razmislite o korištenju dinamičkih matrica umjesto nepromjenjivih matrica. Kad više ne trebate podatke u dinamičkoj matrici, upotrijebite izraze Erase ili ReDim Preserve kako bi odbacili nepotrebne podatke, i vratili memoriju koju je koristila matrica. Na primjer, možete vratiti prostor koji je koristila dinamička matrica sljedećim kodom:

```
Erase MojaMatrica
```

Dok izraz Erase potpuno uklanja matricu, izraz ReDim Preserve čini matricu manjom bez gubljenja njenog sadržaja:

```
ReDim Preserve MojaMatrica(10, manjibroj)
```

Brisanje matrice nepromjenjive veličine neće vratiti memoriju te matrice – jednostavno će izbrisati vrijednosti svakog elementa matrice. Ako je svaki element bio string, ili tip Variant koji sadrži string ili matricu, tada će brisanje matrice vratiti memoriju od tih stringova ili tipova Variant, a ne memoriju od same matrice.

Vratite prostor koji koriste stringovi ili varijable objekata

Prostor koji koriste (nestatički) lokalne varijable stringa i matrica se automatski vraća kad se završi potprogram. Međutim, varijable stringa i matrica koje su opće ili na razini modula ostaju postojati sve dok se izvodi vaša aplikacija. Ako pokušavate održati vašu aplikaciju što je moguće manjom, trebate vratiti prostor koji koriste te varijable što prije možete. Prostor stringa vraćate dodjeljivanjem praznog stringa toj varijabli:

```
VarijablaNekogStringa = ""        ‘ Vraćanje prostora.
```

Slično tome, možete vratiti dio prostora (ali ne cijeli prostor) kojeg koristi varijabla objekta tako da ju postavite na Nothing. Na primjer, ovako možete maknuti varijablu objekta Recordset:

```
Private rs As New Recorset

  ...                     ‘ Ovdje ide kod za pokretanje i
  ...                     ‘ korištenje skupa zapisa.
  rs.Close                ‘ Zatvaranje skupa zapisa.
  Set rs = Nothing        ‘ Postavljanje pokazivača objekta na Nothing.
```

Ako ne postavite pokazivač objekta izričito na Nothing, pokazivač na objekt će ostati u memoriji sve dok se aplikacija ne završi; za aplikaciju koja koristi puno objekata to može brzo potrošiti vašu raspoloživu memoriju i usporiti aplikaciju.

Možete također vratiti prostor izbacivanjem formi i postavljajući ih na Nothing radije nego ih samo sakriti kad više nisu potrebne.

Uklonite mrtav kod i neupotrijebljene varijable

Dok razvijate i mijenjate vaše aplikacije, iza vas može ostati *mrtav kod* – cijeli potprogrami koji se ne pozivaju ni s jednog mjesta u vašem kodu. Možda također imate određene varijable koje se više ne koriste. Iako Visual Basic uklanja neupotrebljene konstante, ne uklanja neupotrebljene varijable i mrtav programski kod kad stvarate izvršnu datoteku. Ponovno pregledajte vaš kod kako bi pronašli i uklonili neupotrebljene potprograme i varijable. Na primjer, izrazi Debug.Print, iako se zanemaruju tijekom izvođenja, ponekad su nazočni u izvršnoj datoteci.

Izrazi Debug.Print s stringovima ili varijablama kao argumentima se ne prevode kad stvarate izvršnu datoteku. Međutim, tamo gdje izrazi Debug.Print imaju poziv funkcije kao argument, prevoditelj zanemaruje sam izraz Debug.Print, ali prevodi poziv funkcije. Zatim, kad se izvodi aplikacija, funkcija se poziva, ali se zanemaruje povratni rezultat. Pošto će funkcije koje se pojavljuju kao argumenti u izrazima Debug.Print zauzeti prostor i vrijeme ciklus u izvršnoj datoteci, može biti korisno obrisati te izraze prije nego što napravite izvršnu datoteku.

Upotrijebite naredbu Find izbornika Edit za traženje upućivanja na određenu varijablu. Ili, ako u svakom od vaših modula imate izraze Option Explicit, možete brzo otkriti upotrebljava li se varijabla u vašoj aplikaciji ako uklonite njena određivanja ili ih označite kao komentare i pokrenete aplikaciju. Ako se varijabla upotrebljava, Visual Basic će stvoriti grešku. Ako ne vidite pogrešku, varijabla se ne upotrebljava.

Za više informacija Kako bi naučili više o izrazu Debug.Print, pogledajte "Ispis informacija u prozoru za neposredan upis naredbi" u 13. poglavlju "Traženje i obrada grešaka".

Redukcija u grafici

Grafike (slike i grafički postupci) mogu potrošiti puno memorije. Do određene mjere, to je neizbježno: grafika sadrži puno informacija, tako da teži veličini. Međutim, u puno slučajeva, možete smanjiti utjecaj koji grafika ima na veličinu vaših aplikacija primjenjivanjem nekih od sljedećih tehnika:

- Upotrijebite kontrolu slike za prikaz slika.
- Učitajte slike iz datoteka kad su potrebne i dijelite slike.
- Upotrijebite postupak PaintPicture.
- Oslobodite memoriju koju koristi grafika.
- Upotrebljavajte oblik .rle slika ili metadatoteke.

Upotrijebite kontrolu slike za prikaz slika

Kontrole okvira za sliku u puno aplikacija Visual Basica postoje samo da bi bile kliknute ili povučene i ispuštene. Ako je to sve što činite s kontrolom okvira za sliku, uzalud trošite puno izvora Windowsa. Za te namjene, kontrole slike su nadmoćnije od kontrola okvira za sliku. Svaka kontrola okvira za sliku je stvaran prozor i koristi značajne sistemske izvore. Kontrola slike je kontrola "lake kategorije", a ne prozor i ne koristi ni približno toliko izvora. Zapravo, tipično možete upotrijebiti pet do deset puta više kontrola slike nego kontrola okvira za sliku. Nadalje, kontrole slike se iscrtavaju brže od kontrola okvira za sliku. Upotrebljavajte kontrole okvira za sliku samo kad trebate osobine koje pružaju, kao što su dinamička razmjena podataka (DDE), grafički postupci, ili sposobnost sadržavanja drugih kontrola.

Učitajte slike iz datoteka kad su potrebne i dijelite slike

Kad postavite svojstvo Picture tijekom izrade, dodajete sliku formi i time povećavate memoriju koju troši forma tijekom izvođenja. Možete smanjiti potrošnju memorije spremanjem slika u datoteku izvora te korištenjem funkcije LoadResPicture za njezino učitavanje tijekom izvođenja. Ako nikad u isto vrijeme ne koristite sve slike pridružene formi, ova tehnika čuva memoriju kod spremanja svih slika u kontrolama na formi. Ona može ubrzati učitavanje forme jer ne trebaju biti učitane sve slike prije nego što forma može biti prikazana.

Možete dijeliti iste slike među više kontrola okvira za sliku, kontrola slike, i formi. Ako upotrijebite programski kod poput sljedećeg, možete održavati samo jednu kopiju slike:

```
Slika = LoadPicture("C:\Windows\Chess.bmp")
Image1.Picture = Slika           ' Upotreba iste slike.
Picture1.Picture = Slika         ' Upotreba iste slike.
```

Usporedite to s sljedećim kodom, koji uzrokuje učitavanje tri kopije slike, trošeći više memorije i vremena:

```
Slika = LoadPicture("C:\Windows\Chess.bmp")
Image1.Picture = LoadPicture("C:\Windows\Chess.bmp")
Picture1.Picture = LoadPicture("C:\Windows\Chess.bmp")
```

Slično tome, ako učitate istu sliku u više formi ili kontrola tijekom izrade aplikacije, kopija te slike snima se s svakom formom ili kontrolom. Umjesto toga, trebali bi postaviti sliku u jednu formu i zatim ju podijeliti s ostalim formama i kontrolama, kako je opisano gore. To čini vašu aplikaciju manjom (jer ne sadrži suvišne kopije slike) i bržom (jer se slika ne treba učitavati s diska više puta).

Upotrijebite postupak PaintPicture

Umjesto postavljanja slika u kontrole, možete upotrijebiti postupak PaintPicture za prikazivanje slika bilo gdje na formama. To je posebno korisno kad želite popločati formu ponavljajućom slikom: trebate učitati sliku samo jednom, ali možete upotrijebiti postupak PaintPicture za njeno iscrtavanje više puta.

Oslobodite memoriju koju koristi grafika

Kad više ne upotrebljavate sliku u svojstvu Picture forme, okvira za sliku ili kontrole slike, postavite svojstvo Picture na Nothing kako bi ga ispraznili:

```
Set Picture1.Picture = Nothing
```

Ako upotrijebite svojstvo Image okvira za sliku ili forme, Visual Basic stvara sliku svojstva AutoRedraw (čak i ako je svojstvo AutoRedraw te forme ili okvira za sliku postavljeno na False). Kad ste završili s korištenjem svojstva Image, možete vratiti memoriju koju je koristila ta slika korištenjem postupka Cls prije postavljanja svojstva AutoRedraw na False. Na primjer, sljedeći kod vraća memoriju koju je koristilo svojstvo Image kontrole nazvane mojaslika:

```
' Uključivanje slike svojstva AutoRedraw.
mojaslika.AutoRedraw = True
mojaslika.Cls                          ' Brisanje te slike.
mojaslika.AutoRedraw = False           ' Isključivanje slike.
```

Upotrebljavajte oblik .rle slika ili metadatoteke

Iako je podrazumijevani oblik slike bitmapirana slika (.bmp), Visual Basic također može iskoristiti ostale oblike grafičkih datoteka. Pojedine aplikacije za crtanje i grafike vam omogućuju snimanje bitmapiranih slika u standardni oblik sažete bitmape nazvan Run Lenght Encoded (.rle). Bitmapirane slike tipa .rle mogu biti nekoliko puta manje od njihovih nesažetih duplikata, posebno kod slika koje sadrže velike dijelove popunjene istom bojom, i nisu zamjetno sporije za učitavanje ili prikazivanje. Upotreba metadatoteka (.wmf) može proizvesti još veće uštede – 10 puta ili više u nekim slučajevima. Pokušajte upotrebljavati metadatoteke s njihovom normalnom veličinom: one se puno sporije iscrtavaju kad trebaju biti razvučene na veću ili manju veličinu.

Možete također upotrebljavati oblike .gif i .jpg. Oni su općenito puno manji; međutim, postoji određeni gubitak u kvaliteti slike i brzini učitavanja.

Razdijeljene aplikacije

Visual Basic vam omogućuje da razmišljate o arhitekturi vaše aplikacije na nove načine. Umjesto jedne, monolitne izvršne aplikacije, možete napisati aplikaciju koja se sastoji od središnje pristupne izvršne aplikacije podržane nizom ActiveX sastavnih dijelova. Takav pristup nudi nekoliko značajnih dobitaka optimizacije:

- Sastavni dijelovi se učitavaju na zahtjev i mogu biti izbačeni kad više nisu potrebni.
- Sastavni dijelovi unakrsne obrade mogu biti 32-bitni izvršni dijelovi na Windowsima 95/98 ili Windowsima NT, čak i ako su ostali dijelovi aplikacije 16-bitni sastavni dijelovi.
- Udaljeni sastavni dijelovi mogu koristiti izvore ostalih uređaja na mreži.

Kao dodatak, u sastavnim dijelovima greške se mogu otkrivati neovisno i mogu biti ponovno upotrebljeni u drugim aplikacijama. To možda neće poboljšati brzinu vaše aplikacije, ali može poboljšati brzinu kojom ćete stvoriti iduću aplikaciju.

Da bi odredili kako najbolje optimizirati vašu aplikaciju razdjeljivanjem, morate procijeniti vrste sastavnih dijelova koje možete stvoriti i kako će se oni uklopiti u vašu aplikaciju. Postoje tri vrste sastavnih dijelova koje možete stvoriti s verzijama Professional i Enterprise Visual Basica:

- Unakrsna obrada
- U obradi
- Udaljeni

Ove tri vrste nisu isključive: možete upotrijebiti sve tri u jednoj aplikaciji. s gledišta optimiziranja vaše aplikacije, svaka od njih ima vrlo različite osobine.

Za više informacija Stvaranje sastavnih dijelova je raspravljeno unutar vodiča *Microsoft Visual Basic 6.0 Component Tools Guide* biblioteke *Microsoft Visual Basic 6.0 Reference Library* uključene s verzijama Professional i Enterprise Visual Basica.

Sastavni dijelovi unakrsne obrade

Sastavni dio unakrsne obrade je izvršna aplikacija koja nudi svoje usluge drugim aplikacijama. Kao i sve izvršne aplikacije, pokreće se i izvodi s svojim vlastitim stogom u svojem vlastitom prostoru obrade; prema tome, kad aplikacija koja djeluje kao klijent koristi jedan od objekata pruženih od sastavnog dijela, operacija prelazi iz klijentskog prostora obrade u prostor obrade sastavnog dijela – odatle ime. Sastavni dijelovi unakrsne obrade nude neke vrijedne osobine kad se uspoređuju s ostalim tipovima:

- Neusklađena operacija (“nizanje”)
- Neuhvaćene pogreške u sastavnom dijelu neće uzrokovati rušenje aplikacije koja ga je pozvala.
- Međuoperativnost između 16-bitnih i 32-bitnih aplikacija.

Od ovih osobina, prva i posljednja točka su od posebnog interes s gledišta optimizacije.

Budući da je sastavni dio unakrsne obrade zasebna aplikacija, on može djelovati neusklađeno s sastavnim dijelom koji djeluje kao klijent. On ima odvojeno “nizanje” koje višezadaćno radi s klijentskom aplikacijom (tehnički rečeno, to nije nizanje nego odvojena obrada; međutim, pojmovno je to dvoje jednako). Dvije aplikacije mogu biti u vezi i dijeliti objekte, ali se izvode neovisno. To je posebno korisno kad vaša aplikacija treba izvesti neku operaciju koja uzima puno vremena. Klijent može pozvati sastavni dio da izvede operaciju i zatim nastavi odgovarati korisniku.

Čak i ako će se vaša aplikacija izvoditi na 32-bitnom sustavu, možda nećete biti u mogućnosti napraviti ju odmah kao 32-bitnu ako se oslanjate na ostavštinu 16-bitnih aplikacija ili sastavnih dijelova. Međutim, ako razdijelite vašu aplikaciju korištenjem sastavnih dijelova unakrsne obrade, možete miješati i udružiti 16-bitne i 32-bitne sastavne dijelove. To vam omogućuje povećano iskorištavanje prednosti 32-bitnih osobina i izvođenja dok održavate vaše ulaganje u 16-bitne sastavne dijelove.

Uz sve svoje prednosti, sastavni dijelovi unakrsne obrade imaju značajan nedostatak: izvođenje. To se očituje na nekoliko načina:

- Brzina pokretanja
- Nadgradnja pozivima unakrsne obrade

Sastavni dio unakrsne obrade je izvršni dio stvoren Visual Basicom, tako da se također primjenjuju ista pitanja vezana uz pokretanje aplikacije. Dobra vijest je da ako pozivate sastavni dio unakrsne obrade napisan u Visual Basicu iz druge aplikacije Visual Basica, gotovo sve dinamičke biblioteke za podršku će već biti učitane. To znatno smanjuje vrijeme potrebno za pokretanje sastavnog dijela. Većina sastavnih dijelova je manja od vaše prosječne Visual Basic aplikacije, s nekoliko ili ni jednom formom za učitavanje, što ponovno poboljšava vrijeme učitavanja. Unatoč tome, sastavni dio unakrsne obrade će uvijek biti sporiji kod pokretanja od sastavnog dijela u obradi.

Kad se izvodi, sastavni dio unakrsne obrade trpi zbog svoje prirode: svaka interakcija s sastavnim dijelom je poziv unakrsne obrade. Granice unakrsnih obrada uzimaju puno od ciklus glavnog procesora. Svaki pokazivač na objekt iz sastavnog dijela unakrsne obrade je tako puno skuplji od jednakog pokazivača na objekt u samoj klijentskoj aplikaciji ili u sastavnom dijelu u obradi. Smanjivanje broja poziva unakrsne obrade u vašem kodu može smanjiti utjecaj nadgradnje pozivima unakrsne obrade.

Sastavni dijelovi u obradi

Sastavni dio u obradi nudi svoje usluge ostalim aplikacijama unutar njihovog prostora obrade. U usporedbi s sastavnim dijelovima unakrsne obrade, sastavni dijelovi u obradi nude dvije prednosti:

- Poboljšano vrijeme učitavanja
- Nema nadgradnje unakrsne obrade

s sastavnim dijelom u obradi, ne treba biti stvorena nova obrada i ne trebaju biti učitane dinamičke biblioteke tijeka izvođenja. To može učiniti sastavni dio u obradi značajno bržim za učitavanje u usporedbi s jednakim sastavnim dijelom unakrsne obrade.

Pošto je u obradi, nema nadogradnje unakrsne obrade kod upućivanja na postupke ili svojstva objekta kojeg pruža sastavni dio. Objekti iz sastavnog dijela rade s istim učinkom kao objekti iz same klijentske aplikacije.

Udaljeni sastavni dijelovi

Verzija Enterprise Visual Basica omogućuje vam stvaranje udaljenih sastavnih dijelova koji se izvode na odvojenim uređajima negdje na mreži. Iako će nadogradnja mreže neminovno zahtijevati danak u izvođenju aplikacije, možete ga nadoknaditi korištenjem izvora dodatnih procesora. To je posebno istinito kad radite s udaljenim sastavnim dijelom koji radi s podacima koji su lokalni na uređaju koji sadržava sastavni dio. Pošto će podaci u svakom slučaju biti poslani preko mreže, sastavni dio koji radi lokalno s podacima i vraća samo rezultate preko mreže može stvarno biti učinkovitiji.

Na primjer, možete napisati objekt u sastavnom dijelu koji može tražiti datoteke koje odgovaraju određenom uvjetu na lokalnom tvrdom disku. Ako to napravite udaljenim sastavnim dijelom i postavite kopiju na svaki uređaj na mreži, možete napisati raspodijeljenu aplikaciju traženja datoteka koja paralelno pretražuje sve dijelove mreže, koristeći sve izvore svih tih procesora.

Optimiziranje objekata

Kako ćete upotrebljavati sve više i više objekata u vašim Visual Basic aplikacijama, optimiziranje vaše upotrebe tih objekata postajat će sve važnije. Postoji nekoliko ključnih tehnika za stvaranje najučinkovitije upotrebe objekata:

- Upotrebljavajte rano povezivanje.

- Smanjite upotrebu točaka.
- Upotrebljavajte izraze Set i With...End With.
- Smanjite pozive unakrsne obrade.

U Visual Basicu, upućivanje na objekt druge aplikacije u vašem kodu (dobivanjem ili postavljanjem svojstva objekta, ili izvođenjem nekog od njegovih postupaka) osniva poziv unakrsne obrade. Pozivi unakrsne obrade su skupi i trebate ih pokušati izbjegavati ako razmišljate o optimiziranju vaše aplikacije.

Rano povezivanje protiv kasnog povezivanja

Visual Basic može učinkovitije upotrijebiti objekte ako ih rano povežete. Objekt može biti rano povezan ako pribavite pokazivač na tipsku biblioteku koja sadrži objekt, i odredite tip objekta:

```
Dim X As New MojObjekt
```

Ili, jednako vrijedno:

```
Dim X As MojObjekt
Set X = New MojObjekt
```

Rano povezivanje omogućuje Visual Basicu da napravi najviše posla u razlučivanju određivanja objekta tijekom prevođenja umjesto tijekom izvođenja, kad to utječe na izvođenje. To također omogućuje Visual Basicu da provjeri sintaksu svojstava i postupaka korištenih s objektom te da izvijesti o svakoj greški.

Ako Visual Basic ne može rano povezati objekt, mora ga povezati kasno. Kasno povezivanje objekata je skupo: tijekom prevođenja nemate provjeru greške, a svako upućivanje tijekom izvođenja traži najmanje 50% više rada Visual Basica.

Općenito, trebali bi uvijek rano povezivati objekte kad je to moguće. Jedina prilika kad bi trebali odrediti varijablu s As Object je ako nemate tipsku biblioteku za objekt ispitivanja, ili trebate biti u mogućnosti proslijediti svaku vrstu objekta kao argument potprogramu.

Za više informacija Kako bi naučili više o ranom povezivanju, pogledajte “Ubrzavanje pokazivača objekata” u 10. poglavlju “Programiranje sastavnim dijelovima”.

Smanjite upotrebu točaka

Kad upućujete na objekte drugih aplikacija iz Visual Basica, upotrebljavate sintaksu s točkama “ . “ kako bi se kretali kroz hijerarhiju objekta sastavljenu od zbirki, objekata, svojstava i postupaka. Nije neuobičajeno stvoriti vrlo duge stringove kretanja. Na primjer:

```
‘ Upućivanje na ćeliju A1 na popisu Sheet1 u prvoj
‘ radnoj knjizi proračunske tablice Microsoft Excela.
Application.Workbooks.Item(1).Worksheets.Item_
(“Sheet1”).Cells.Item(1, 1)
```

Osim što je ova linija prilično dugačak niz za tipkanje, zapravo je teška za čitanje – i vrlo neučinkovita.

Kad iz Visual Basica pozivate objekt, svaka "točka" od Visual Basica zahtijeva da napravi višestruke pozive.

Kako bi napisali najučinkovitije aplikacije, smanjite korištenje točaka kad upućujete na objekt.

Obično možete smanjiti broj točaka analiziranjem vama dostupnih objekata i postupaka. Na primjer, gornja linija programskog koda može biti skraćena micanjem postupka Item (to je ionako podrazumijevan postupak za zbirke, pa ćete ga rijetko upotrebljavati u kodu) te upotrebom učinkovitijeg postupka Range:

```
‘ Upućivanje na ćeliju A1 na popisu Sheet1 u prvoj
‘ radnoj knjizi proračunske tablice Microsoft Excela.
Application.Workbooks(1).Worksheets(“Sheet1”)
.Range(“A1”)
```

Ovo možete još više skratiti ponovnim pisanjem koda tako da upućuje na aktivan list aktivne radne knjige, umjesto određenog lista u određenoj radnoj knjizi:

```
‘ Upućivanje na ćeliju A1 na aktivnom popisu
‘ aktivne radne knjige.
Range(“A1”)
```

Naravno, gornji primjer pretpostavlja da u redu upućivati na ćeliju A1 svakog lista koji je trenutno aktivan.

Upotrebljavajte izraze Set i With...End With

Korištenje izraza Set vam također omogućuje skraćene nizove kretanja i daje vam nešto više nadzora nad vašim kodom. Sljedeći primjer koristi izraze Dim i Set za stvaranje varijabli koje upućuju na često korištene objekte:

```
Dim xlOpseg As Object
Set xlOpseg = Application.ActiveSheet.Cells(1, 1)
xlOpseg.Font.Bold = True
xlOpseg.Width = 40
```

Visual Basic pruža sastav With...End With za postavljanje podrazumijevanog objekta unutar koda:

```
With Application.ActiveSheet.Cells(1, 1)
  .Font.Bold = True
  .Width = 40
End With
```

Smanjite pozive unakrsne obrade

Ako upotrebljavate ActiveX sastavni dio unakrsne obrade, ne možete potpuno izbjeći stvaranje poziva unakrsne obrade. Međutim, postoji nekoliko načina smanjivanja broja poziva unakrsne obrade koje trebate napraviti. Ako je moguće, ne upućujte na objekte unutar petlje tipa For...Next. Spremajte vrijednosti u varijable i upotrebljavajte varijable u petljama. Ako trebate pozvati velik broj postupaka iz objekta, možete značajno poboljšati izvođenje vaše aplikacije premještanjem koda u sastavni dio. Na primjer, ako je sastavni dio aplikacija Word ili Microsoft Excel, možete postaviti makro naredbu s petljom u predložak Worda ili potprogram s petljom u modul Microsoft Excela. Zatim možete pozvati makro ili potprogram iz Visual Basica, što je jedan poziv koji pokreće operaciju petlje unutar sastavnog dijela.

Ako pišete sastavne dijelove, možete oblikovati objekte u sastavnom dijelu tako da budu učinkoviti smanjivanjem poziva unakrsne obrade potrebnih za izvođenje operacije. Na primjer, kad imate nekoliko međusobno povezanih svojstava, ostvarite postupak s nekoliko argumenata – svaki za jedno svojstvo. Pozivanje postupka zahtjeva jedan poziv unakrsne obrade neovisno o tome koliko ima argumenata, dok postavljanje svakog svojstva zahtjeva odvojen poziv unakrsne obrade. Slično tome, ako pretpostavljate da će sastavni dio koji radi kao klijent željeti pozvati vaš sastavni dio u petlji (na primjer, kako bi zbrojio vrijednosti ili izračunao prosjek u svojstvu popisa), možete poboljšati izvođenje pružanjem postupaka koji čine petlje unutar vašeg objekta i vraćaju odgovarajuću vrijednost.

Za više informacija Stvaranje sastavnih dijelova je raspravljeno unutar vodiča *Microsoft Visual Basic 6.0 Component Tools Guide* uključenog s verzijama Professional i Enterprise Visual Basica.

Prevedene protiv tumačenih aplikacija

U pravilu, aplikacije stvorene u Visual Basicu se prevode kao tumačene datoteke ili izvršne datoteke p-koda. Tijekom izvođenja, instrukcije u izvršnim datotekama se prevode ili tumače dinamički povezivom datotekom (DLL) tijeka izvođenja. Verzije Professional i Enterprise Visual Basica uključuju mogućnost prevođenja u izvršnu datoteku izvornog koda. U većini slučajeva, prevođenje u izvorni kod može pružiti stvaran dobitak u brzini u odnosu na tumačene verzije iste aplikacije; međutim, to nije uvijek slučaj. Slijede neke općenite smjernice koje se tiču prevođenja u izvorni kod.

- Programski kod koji izvodi puno jednostavnih operacija na čvrsto napisanim, neslovčanim varijablama će postići maksimalan omjer stvorenog izvornog koda prema premještenim operacijama u p-kodu. Složena financijska proračunavanja ili stvaranje fraktala će, stoga, imati koristi iz izvornog koda.
- Aplikacije s intenzivnim izračunavanjima, ili aplikacije koje miješaju puno bitova i bajtova unutar lokalnih struktura podataka, će vrlo vidljivo zaraditi izvornim kodom.

- Za većinu aplikacija, posebno onih koje imaju puno Windows API poziva, poziva postupka COM, i upravljanja stringovima, izvorni kod neće biti puno brži od p-koda.
- Aplikacije koje se uglavnom sastoje od funkcija iz biblioteke izvođenja Visual Basica za aplikacije neće imati puno prednosti iz izvornog koda, ako će je uopće i imati, jer je kod u biblioteci izvođenja Visual Basica za aplikacije već visoko optimiziran.
- Programski kod koji sadrži puno poziva potprograma koji se odnose na ugrađene potprograme također će malo vjerojatno postati bitno brži s izvornim kodom. Razlog tome je što sav posao postavljanja okvira stoga, pokretanja varijabli i čišćenja pri izlazu uzima isto vrijeme i s mehanizmom p-koda i s stvorenim izvornim kodom.

Zapamtite da svi pozivi objekata, dinamičkih biblioteka ili funkcija izvođenja Visual Basica za aplikacije poništavaju prednosti izvođenja u izvornom kodu. Razlog tomu je što se relativno malo vremena potroši izvodeći kod – većina vremena (obično oko 90 – 95%) se potroši unutar formi, objekata podataka, dinamičkih biblioteka Windows ili izvođenja Visual Basica za aplikacije, uključujući rukovanje unutarnjim stringovima i varijablama.

U testovima stvarnog svijeta, klijentske aplikacije obično potroše oko 5% svog ukupnog vremena izvođenja izvodeći p-kod. Stoga, ako je izvorni kod trenutačan, korištenje izvornog koda za te aplikacije će pružiti najviše 5% poboljšanja u vremenu izvođenja.

Ono što izvorni kod radi je omogućavanje programerima da pišu dijelove koda ili algoritme s intenzivnim proračunavanjima u Basicu koji nikad prije nisu bili mogući zbog pitanja izvođenja. Omogućavanje puno bržeg izvođenja takvih "svaštica" može također poboljšati pristupačnost određenih dijelova aplikacije, što poboljšava zamijećeno izvođenje cijele aplikacije.

Za više informacija Kako bi naučili više o prevođenju u izvorni kod, pogledajte "Prevođenje vašeg projekta u izvorni kod" u 8. poglavlju "Više o programiranju".

Sukladnost s ostalim Microsoft aplikacijama

Visual Basic je stariji član porodice Visual Basic proizvoda koja uključuje Visual Basic za aplikacije (Visual Basic for Applications) i Visual Basic, verziju Scripting (VBScript). Iako većina koda kojeg napišete u Visual Basicu može biti dijeljena s aplikacijama napisanim u Visual Basicu za aplikacije ili verziji Scripting, postoje neke iznimke.

Sukladnost s Visual Basicom za aplikacije

Visual Basic za aplikacije je zaseban, opći aplikacijski skriptni jezik i okolina na koju korisnici i programeri mogu utjecati kroz njihovu radnu površinu Windowsa. Visual Basic za aplikacije je uključen u programski paket Microsoft Office i ostale aplikacije Microsofta. Također je odobren drugim proizvođačima softvera i uključen je u širok raspon proizvoda.

Visual Basic za aplikacije, sadržan u biblioteci Vba6.dll, je mehanizam programskog jezika podređen Visual Basicu. Ova biblioteka sadrži sve elemente jezika koji se dijele s Visual Basicom za aplikacije i Visual Basicom. Njene elemente možete vidjeti odabirom stavke VBA u okviru Library pretraživača objekata. Programski kod napisan u Visual Basicu za aplikacije je prenosiv u Visual Basic uz sljedeća ograničenja: programski kod Visual Basica za aplikacije koji upućuje na specifične elemente aplikacija (kao što je radni list Microsoft Excela) može biti prenesen, pod uvjetom da sadrži potpuno označeno upućivanje i pod uvjetom da upućena aplikacija postoji na ciljnom uređaju.

Elementi specifični za Visual Basic, kao što su forme i ugrađene kontrole, sadržani su u tipskoj biblioteci Vb6.olb (koja je također vidljiva u pretraživaču objekata). Općenito, programski kod napisan u Visual Basicu može se prenijeti u Visual Basic za aplikacije sve dok ne upućuje na takve elemente.

Za više informacija Kako bi naučili više o Visual Basicu za aplikacije, posjetite Microsoftovu Web stranicu na http://www.microsoft.com. Kako bi naučili više o upućenim objektima, pogledajte "Stvaranje pokazivača na objekt" u 10. poglavlju "Programiranje sastavnim dijelovima". Kako bi naučili više o prevođenju u izvorni kod, pogledajte "Prevođenje vašeg projekta u izvorni kod" u 8. poglavlju "Više o programiranju".

Sukladnost s Visual Basicom, verzija Scripting

Visual Basic, verzija Scripting (VBScript) je oblikovan kako bi bio udomaćen unutar pretraživača Interneta, kao što je Microsoft Internet Explorer ili drugi pretraživači treće strane. VBScript je vrlo brz jezični mehanizam lake kategorije oblikovan posebno za okruženja kao što su Internet, intraneti, ili World Wide Web. VBScript utječe na snagu Visual Basica i omogućuje programerima korištenje njihovog znanja razvijanja Visual Basica kako bi brzo stvorili rješenja za Internet ili World Wide Web.

VBScript podržava podskup jezične sintakse Visual Basica za aplikacije. Visual Basic, verzija Scripting, ne uključuje sučelje razvojnog okruženja kakvo se može pronaći u Microsoft Visual Basicu, jer je oblikovan kako bi bio jezični mehanizam lake kategorije koji može biti dijeljen na raznim platformama. Programski kod VBScripta možete napisati u kodnom editoru Visual Basica, ali ne možete izvesti ni ispitati aplikaciju u razvojnom okruženju Visual Basica.

Pošto je VBScript razvojni jezik unakrsne platforme, nisu uključeni neki od elemenata jezika Visual Basic za aplikacije. Tu spadaju sve funkcije datotečnog ulaza/izlaza, ugrađene konstante, ugrađeni tipovi podataka, i tako dalje. Kad prenosite programski kod iz Visual Basica u VBScript, važno je da u vašem kodu potražite nepodržane elemente.

Za više informacija Kako bi naučili više o Visual Basicu, verzija Scripting, uključujući popis podržanih elemenata tog jezika, posjetite Microsoftovu Web stranicu na http://www.microsoft.com.

16. POGLAVLJE

Međunarodna izdanja

Ako namjeravate distribuirati vaše Visual Basic aplikacije na međunarodno tržište, možete smanjiti iznos vremena i programskog koda neophodnog za izradu vaše aplikacije kao djelotvorne na stranom tržištu kao što je i na domaćem tržištu. Ovo poglavlje predstavlja ključne pojmove i definicije za razvijanje međunarodnih aplikacija sa Visual Basicom, predstavlja model lokalizacije i ističe prednosti oblikovanja softvera za međunarodno tržište.

Ovo poglavlje također raspravlja smjernice za pisanje programskog koda Visual Basica koji rezultira kao fleksibilna, prenosiva i stvarno međunarodna aplikacija. Dio je posvećen pisanju koda Visual Basica koji rukuje posebnim gledištima dvobajtnog skupa znakova (double-byte character set, DBCS) koji se upotrebljava u verzijama Windowsa za istočnu Aziju.

Sadržaj

- Definicije međunarodnog softvera
- Oblikovanje međunarodnog softvera
- Korištenje datoteka izvora za lokalizaciju
- Oblikovanje međunarodnog korisničkog sučelja
- Općenita razmišljanja za pisanje međunarodnog koda
- Pisanje međunarodnog koda u Visual Basicu
- Izdanja specifična za dvobajtni skup znakova (DBCS)
- Dvosmjerne osobine Visual Basica

Primjer aplikacije: Atm.vbp

Neki od primjera koda u ovom poglavlju uzeti su iz primjera Automated Teller Machine (Atm.vbp), koji se nalazi u direktoriju Samples.

Definicije međunarodnog softvera

Prije nego što počnete razvijati međunarodni softver, trebali bi znati neke osnovne pojmove.

Međunarodni softver

Međunarodni softver je softver koji se može prodati širom svijeta. Softverski proizvod je međunarodan samo ako je djelotvoran na stranom tržištu kao što je na domaćem tržištu. Za više informacija o tome kako lokalizirati vaše aplikacije, pogledajte "Oblikovanje međunarodnog softvera", kasnije u ovom poglavlju.

Poprište

Poprište opisuje korisnikovo okruženje – lokalna pravila, kulturu i jezik zemljopisnog područja korisnika. Poprište je sačinjeno od jedinstvene kombinacije *jezika* i *države*. Dva primjera poprišta su: engleski/SAD (English/U.S.) i francuski/Belgija (French/Belgium).

Jezik može biti govorni u više od jedne države; na primjer, francuski je govorni jezik u Francuskoj, Belgiji, Kanadi i kod puno afričkih naroda. Iako te države dijele zajednički jezik, određena nacionalna pravila (kao valuta) se razlikuju među tim državama. Zbog toga, svaka država predstavlja jedinstveno poprište. Slično tome, jedna država može imati više od jednog službenog jezika. Belgija ima tri – francuski, danski i njemački. Prema tome, Belgija ima tri različita poprišta. Za više informacija o postavkama specifičnim za područja, pogledajte "Općenita razmišljanja za pisanje međunarodnog koda", kasnije u ovom poglavlju.

Lokalizacija

Lokalizacija je postupak kojim se aplikacija prilagođuje poprištu. On uključuje više od samo doslovnog, riječ-za-riječ prevođenja tih izvora – to je značenje koje mora komunicirati sa korisnikom. Za više informacija o tome kako lokalizirati vašu aplikaciju, pogledajte "Oblikovanje međunarodnog softvera", kasnije u ovom poglavlju.

Izvori tekstova

Izvori tekstova upućuju na sav tekst koji se pojavljuje u korisničkom sučelju aplikacije. Oni uključuju, ali nisu ograničeni samo na, izbornike, dijaloške okvire i poruke obavješćivanja, upozoravanja i pogrešaka. Ako će aplikacija biti korištena na poprištu koje je drugačije od onog gdje je razvijena, ovi izvori moraju biti prevedeni, ili lokalizirani.

Za više informacija Za definicije terminologije istočne Azije, pogledajte odlomak "ANSI, DBCS i Unicode: definicije" kasnije u ovom poglavlju. Za više informacija o izvorima tekstova i datotekama izvora, pogledajte "Korištenje datoteka izvora za lokalizaciju", kasnije u ovom poglavlju.

Oblikovanje međunarodnog softvera

Puno je učinkovitije oblikovati vašu aplikaciju sa lokalizacijom na umu, sljedeći pristup koji odvaja izvore tekstova od koda, nego ponovno pregledavati vašu završenu aplikaciju kako bi ju učinili međunarodnom kasnije u postupku razvoja.

Prednosti oblikovanja međunarodnog softvera

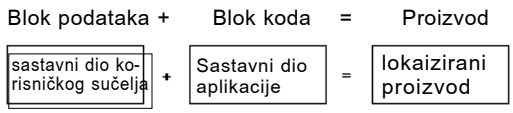
Postoje četiri glavne prednosti oblikovanja i ostvarivanja vaše Visual Basic aplikacije tako da je osjetljiva i prikladna međunarodnim pravilima, stranim podacima, i obradi oblikovanja:

- Vašu Visual Basic aplikaciju možete brže izbaciti na tržište. Nije potrebno dodatno međunarodno razvijanje jednom kad je završena početna verzija projekta.
- Možete učinkovitije upotrebljavati izvore. Dodavanje međunarodne podrške vašoj završenoj Visual Basic aplikaciji može ju učiniti nestabilnijom. Biti će potrebno više razvijanja i ispitivanja izvora nego kad je ista podrška ostvarena kao dio originalnog postupka razvijanja.
- Ako je međunarodna verzija vaše Visual Basic aplikacije izgrađena iz istog skupa izvora kao i verzija u kojoj ste izvorno razvili vašu aplikaciju, samo izdvojeni moduli će trebati međunarodno prilagođivanje, što čini održavanje koda lakšim i jeftinijim kod uključivanja međunarodne podrške. Pogledajte “Korištenje datoteka izvora za lokalizaciju”, kasnije u ovom poglavlju.
- Razvijanje međunarodne verzije vaše aplikacije postaje lako. Na primjer, možete razviti verziju vaše aplikacije na engleskom jeziku koja će se izvoditi na njemačkom operativnom okruženju bez ponovnog pisanja koda. Sve što ćete trebati napraviti je prilagođivanje korisničkog sučelja. Pogledajte “Oblikovanje međunarodnog korisničkog sučelja”, kasnije u ovom poglavlju.

Model lokalizacije

Svaka aplikacija koja će biti lokalizirana predstavlja dva koncepcijska bloka: *blok koda* i *blok podataka*. Slika 16.1 prikazuje blok podataka kao “sastavni dio korisničkog sučelja” i blok koda kao “sastavni dio aplikacije”.

Slika 16.1 Blok podataka i blok koda čine lokalizirani proizvod



Blok podataka sadrži sve izvore tekstova korisničkog sučelja, ali ne i kod. Suprotno tome, blok koda sadrži samo programski kod aplikacije koji se izvodi za sva poprišta. Programski kod Visual Basica rukuje izvorima tekstova i postavkama specifičnim za poprište. Odlomak “Pisanje međunarodnog koda u Visual Basicu”, kasnije u ovom poglavlju, pruža detalje o načinu pisanja koda Visual Basica koji rukuje postavkama specifičnim za poprište, kao što su datumi, valute, i brojčane vrijednosti sa razdvojnicma.

Teoretski, možete razviti lokaliziranu verziju vaše Visual Basic aplikacije mijenjanjem samo bloka podataka. Blok koda bi za sva poprišta trebao biti isti. Rezultat spajanja bloka podataka sa blokom koda je lokalizirana verzija vaše aplikacije. Ključ uspješnog oblikovanja međunarodnog softvera je razdvajanje blokova koda i podataka te sposobnost vaše Visual Basic aplikacije da točno pročita podatke, neovisno o poprištu.

Iako će biti više posla za osobu koja piše Visual Basic aplikaciju, ni jedan element korisničkog sučelja ne bi trebao biti u kodu Visual Basica. Umjesto toga, izvori tekstova trebali bi biti postavljeni u odvojenu datoteku, iz koje će biti učitani tijekom izvođenja aplikacije. Ta datoteka se naziva *datoteka izvora* (*resource file*, .res), i to je datoteka koja sadrži sve tekstove, slike i ikone koje su lokalizirane. Za više informacija o datotekama izvora, pogledajte "Korištenje datoteka izvora za lokalizaciju", kasnije u ovom poglavlju.

Timovi koji rade na lokaliziranju aplikacije trebali bi raditi isključivo na datoteci izvora kako bi razvili različite jezične verzije aplikacije. Takav pristup ima sljedeće prednosti:

- **Učinkovitost**. Razvijanje nove međunarodne verzije aplikacije uključuje samo stvaranje nove međunarodne datoteke izvora jer svaka verzija ima isti blok koda. To olakšava stvaranje višestrukih jezičnih verzija vaše Visual Basic aplikacije.
- **Veća sigurnost**. Bez obzira odlučite li lokalizirati vašu aplikaciju sami ili to povjeriti vanjskoj tvrtki, nećete trebati pristupati izvornom kodu kako bi razvili međunarodne verzije vaše aplikacije. Takav pristup će također smanjiti vrijeme ispitivanja potrebnog za međunarodnu verziju.
- **Bolja lokalizacija**. Postavljanjem svih izvora tekstova u jednu datoteku, osiguravate učinkovitiji postupak lokalizacije i smanjujete mogućnost ostavljanja nekih tekstova nelokaliziranih.

Sljedeća tablica ispisuje neke čimbenike o kojima treba voditi računa pri oblikovanju vaše Visual Basic aplikacije.

čimbenik	stavka
Jezik	Tekstovi u korisničkom sučelju (izbornici, dijaloški okviri, i poruke grešaka)
	Tiskana i ugrađena dokumentacija
Postavke specifične za poprište	Oblici datuma/vremena (razdvojnici, redoslijed dan/mjesec/godina)
	Oblici brojeva (razdvojnici tisuća i decimala)
	Oblici valute (simbol i oblik ispisa)
	Redoslijed sortiranja i uspoređivanje teksta

Prvi čimbenik, jezik, se odnosi uglavnom na fazu oblikovanja vaše Visual Basic aplikacije. Pogledajte "Oblikovanje međunarodnog korisničkog sučelja" za više informacija. Drugi čimbenik, postavke specifične za poprište, raspravljen je u odlomcima "Pisanje međunarodnog koda u Visual Basicu" i "Međunarodni redoslijed sortiranja i uspoređivanje teksta", kasnije u ovom poglavlju.

Korištenje datoteka izvora za lokalizaciju

Datoteka izvora je koristan mehanizam za razdvajanje lokaliziranih informacija iz koda u Visual Basicu.

Napomena U vašem projektu možete imati samo jednu datoteku izvora. Ako pokušate dodati više od jedne datoteke izvora, Visual Basic će stvoriti poruku pogreške.

Prednosti spremanja tekstova u datotekama izvora

Kad pišete programski kod Visual Basica, možete upotrijebiti funkcije LoadResString, LoadResPicture i LoadResData umjesto pokazivača na tekstove, slike i podatke. Spremanje takvih elemenata u datoteku izvora nudi dvije prednosti:

- Poboljšano je izvođenje i sadržaj jer tekstovi, slike, ikone i podaci mogu biti učitani iz datoteke izvora na zahtjev, umjesto da se svi učitaju odjednom kad se učita forma ili modul.
- Izvori koji trebaju biti prevedeni na drugi jezik, odvojeni su u jednoj datoteci izvora. Nema potrebe za pristup izvornom kodu ili ponovno prevođenje aplikacije.

Kako stvoriti datoteku izvora

1. Odaberite naredbu **Add New Resource File** iz izbornika **Project**.

   Napomena Ova naredba je dostupna samo kad je učitan dodatak Resource Editor. Kako bi učitali dodatak Resource Editor, odaberite **Add-In Manager** iz izbornika **Add-Ins**. U dijaloškom okviru **Add-In Manager**, odaberite **VB6 Resource Editor** i potvrdite kontrolnu kućicu **Loaded/Unloaded**.

2. U dijaloškom okviru **Open, a Resource File**, upišite ime datoteke izvora. Datoteka izvora će biti dodana čvoru **Related Documents** u **projektnom prozoru**.

   Napomena Vašem projektu možete također dodati postojeću datoteku izvora. Međutim, zapamtite da sve promjene koje napravite u postojećoj datoteci izvora mogu utjecati na drugu aplikaciju koja koristi tu datoteku.

Visual Basic prepoznaje datoteke izvora po nastavku imena datoteke .res. Ako datoteka izvora nema prikladan nastavak imena datoteke, Visual Basic ju neće učitati. Suprotno tome, ako bilo koja datoteka upotrebljava nastavak imena datoteke .res, Visual Basic pretpostavlja da je to datoteka izvora kad ju dodaje projektu. Ako takva datoteka ne poštuje oblik standardan za datoteku izvora, Visual Basic će stvoriti grešku kad prvi put pokušate upotrijebiti funkcije podrške datoteci izvora (LoadResString, LoadResPicture i LoadResData), ili kad pokušate napraviti izvršnu datoteku. Visual Basic će stvoriti istu poruku greške ako projektu pokušate dodati 16-bitnu datoteku izvora.

Datoteka .res je, prije i nakon što prevedete izvršnu datoteku, standardna *datoteka izvora Windowsa*, što znači da izvori sadržani u datoteci mogu biti učitani u svaki standardni editor izvora temeljen na Windowsima.

Kako editirati datoteku izvora

1. Odaberite naredbu **Resource Editor** iz izbornika **Tools**.

   Napomena Ova naredba je dostupna samo kad je učitan dodatak Resource Editor. Kako bi učitali dodatak Resource Editor, odaberite **Add-In Manager** iz izbornika **Add-Ins**. U dijaloškom okviru **Add-In Manager**, odaberite **VB6 Resource Editor** i potvrdite kontrolnu kućicu **Loaded/Unloaded**.

2. Odaberite gumb iz **alatne trake editora izvora** kako bi editirali postojeći izvor ili dodali novi. Kako bi naučili više o editiranju izvora, pogledajte dokumentaciju dodatka Resource Editor.

Zaključavanje datoteka izvora

Visual Basic upotrebljava zaključavanje datoteke na datoteci izvora kako bi spriječio probleme kad više aplikacija pokuša koristiti datoteku u isto vrijeme. Visual Basic će zaključati datoteku izvora uvijek:

- Kad je Visual Basic u modu izvođenja ili u modu prekida.
- Kad stvarate izvršnu datoteku.

Za više informacija Za primjer kako datoteka izvora može biti upotrebljena za stvaranje aplikacije koja radi na više poprišta, pogledajte sljedeći odlomak "Primjer aplikacije Automated Teller Machine". Za dodatne informacije o programiranju s datotekama izvora, pogledajte "Rad s datotekama izvora" u 8. poglavlju, "Više o programiranju".

Primjer aplikacije Automated Teller Machine

Ovaj primjer aplikacije je oblikovan kako bi prikazao podršku za datoteke izvora u Visual Basicu. Aplikacija sadrži tri forme, standardni modul i datoteku izvora. Kad pokrenete primjer aplikacije Automated Teller Machine (Atm.vbp), uvodni ekran vam dozvoljava izvođenje bankarske transakcije na jednom od nekoliko jezika, uključujući njemački, francuski, talijanski i španjolski.

Sljedeći kod iz datoteke FrmInput.frm učitava izvore spremljene u datoteci Atm32.res, koja sadrži lokalizirane tekstove za sve jezike.

```
Sub Form_Load()
  imgFlag = LoadResPicture(I, vbResBitmap)
  Caption = LoadResString(I)
  lblPINCode = LoadResString(1 + I)
  fraAccount = LoadResString(2 + I)
  optChecking.Caption = LoadResString(3 + I)
  optSavings.Caption = LoadResString(4 + I)
  lblAmount = LoadResString(5 + I)
  cmdOK.Caption = LoadResString(6 + I)
  SetCursor cmdOK
End Sub
```

```
Sub cmdOK_Click()
   ‘ Prikaz poruke obrade.
   MsgBox LoadResString(7 + I)
   frmAboutWithdrawn.Show vbModal
   Unload Me
End Sub
```

Tijekom izvođenja, ovaj programski kod čita odgovarajući dio datoteke izvora, temeljen na pomaku koji je postavljen kad korisnik odabere jezik na uvodnom ekranu. *Pomak* (*offset*) je javna varijabla određena u standardnom modulu koja ukazuje koliko se daleko od početne točke nalazi određena stavka. U primjeru aplikacije ATM, varijabla pomaka je I.

U datoteci izvora, identifikatori izvora od 16 do 47 su rezervirani za engleski, od 48 do 79 su rezervirani za francuski, od 80 do 111 su rezervirani za njemački i tako dalje. Svaki jezik sadrži lokalizirane unose koji čine blok podataka ove aplikacije. Ovaj blok trenutno sadržava jedanaest izvora koji su osobiti za svaki jezik.

Ovaj primjer aplikacije, koji sadrži nekoliko blokova podataka, uvodi izbor u datoteku izvora specifičnu za određeni jezik koristeći samo jedan blok podataka. Ovisno o prirodi aplikacije koju razvijate, možete razmisliti o korištenju po jedne datoteke izvora za svaku jezičnu verziju vaše aplikacije ili jedne datoteke izvora koja sadrži sve lokalizirane blokove podataka.

Oblikovanje primjera aplikacije Automated Teller Machine predstavlja nekoliko prednosti iznad onih istaknutih ranije u ovom poglavlju:

- Aplikacija se može povećavati u području pružanja usluga na više jezika. Jednostavno dodajte isti blok podataka datoteci izvora i lokalizirajte ga prema potrebi. Ako odlučite dodati podršku za novi jezik, morat ćete dodati gumb na uvodni ekran.
- Aplikacija se može povećavati u veličini ako želite proširiti vašu aplikaciju, na primjer, omogućavanjem korisnicima aplikacije ATM da naprave pologe. Jednostavno omogućite širi opseg identifikatora (na primjer, 160) za svaki jezik u datoteci izvora. Trenutno, opseg identifikatora je od 16 do 47, od 48 do 79, i tako dalje.

Za više informacija Pogledajte odlomke “Funkcija LoadResString”, “Funkcija LoadResPicture” i “Funkcija LoadResData” u priručniku *Microsoft Visual Basic 6.0 Language Reference* biblioteke *Microsoft Visual Basic 6.0 Reference Library*. Za informacije o datotekama izvora, pogledajte “Rad s datotekama izvora” u 8. poglavlju “Više o programiranju”, i 15. poglavlje “Oblikovanje u korist izvođenja i sukladnosti”.

Oblikovanje međunarodnog korisničkog sučelja

Budući da je tekst sklon povećavanju kad lokalizirate aplikaciju, trebali bi obratiti posebnu pažnju kad oblikujete sljedeće dijelove korisničkog sučelja:

- Poruke
- Izbornici i dijaloški okviri
- Ikone i slike
- Pristupne tipke i prečice kombinacijom tipki

Poruke

Stringovi engleskog teksta su obično kraći od lokaliziranih tekstualnih stringova u drugim jezicima. Sljedeća tablica pokazuje dodatni prosjek rasta za stringove, temeljen na početnoj veličini. Ovi podaci su izvučeni iz proteklih projekata lokalizacije Visual Basica i opisuju prosječan stupanj rasta.

duljina engleskog teksta (u znakovima)	dodatna veličina za lokalizirane stringove
1 do 4	100%
5 do 10	80%
11 do 20	60%
21 do 30	40%
31 do 50	20%
iznad 50	10%

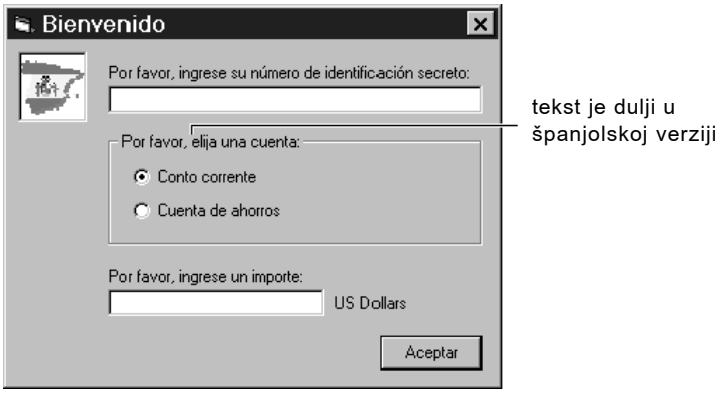
Kad oblikujete sučelje, razmotrite ove stupnjeve rasta i omogućite lomljenje teksta u više linija ako tekstovi poruka postanu dulji.

Izbornici i dijaloški okviri

Kao i kod poruka, izbornici i dijaloški okviri mogu se povećati kad se aplikacija lokalizira. Razmotrite sljedeća dva identična dijaloška okvira u primjeru aplikacije Automated Teller Machine. Možete vidjeti da je u dijaloškom okviru dodijeljen dodatni prostor kako bi omogućio širenje teksta. Slika 16.2 prikazuje dijaloški okvir za engleski jezik, dok slika 16.3 prikazuje dijaloški okvir za španjolski jezik. Znate li da tekst može rasti, planirajte vaše sučelje tako da se tim kontrolama ne treba mijenjati veličina ili se ostali elementi ne trebaju ponovno uobličivati kod lokaliziranja.

Slika 16.2 Dijaloški okvir unos na engleskom u primjeru aplikacije ATM

Slika 16.3 Dijaloški okvir unos na španjolskom u primjeru aplikacije ATM



U izbornicima i dijaloškim okvirima, izbjegavajte natrpane naslovne trake. Čak i kratice mogu biti dulje ili čak ne moraju postojati u drugim jezicima.

Ikone i slike

Ikone i slike se obično koriste za prikazivanje određene funkcionalnosti bez upotrebe teksta. Razmotrite sljedeća pravila kad radite s ikonama i slikama:

- Izbjegavajte korištenje slika koje nisu međunarodni standard. Sljedeća slika predstavlja poštanski sandučić u Sjedinjenim Američkim Državama, ali većina korisnika iz drugih područja ju neće prepoznati.

- Izbjegavajte korištenje slika koje sadrže tekst. Za ponovno iscrtavanje takvih slika treba vremena, a povećanje teksta također može postati prepreka, kao što je prikazano na sljedećim ikonama:

- Neka vaše slike ili ikone vode računa o lokalnoj kulturi. Ono što je prihvatljivo na jednom popništu možda će na drugom biti neprikladno ili uvredljivo.

Pristupne tipke i prečice kombinacijom tipki

Različita popništa imaju različite rasporede tipkovnice. Ne postoje svi znakovi na svim rasporedima tipaka. Kad razvijate svoju Visual Basic aplikaciju, budite sigurni da sve kombinacije pristupnih tipki i prečica kombinacijom tipki koje dodijelite mogu biti proizvedene na međunarodnim tipkovnicama. Jedan jednostavan postupak za provjeravanje hoće li vaša pridruživanja tipaka ispravno raditi za popršte koje ciljate je da odaberete željen raspored tipaka iz vašeg kontrolnog panela Windowsa, zajedno s slikama rasporeda tipaka (što sadržavaju neki priručnici s uputama), i isprobate kombinacije pristupnih tipki i prečica kombinacijom tipki.

Budući da neke kombinacije pristupnih tipki i prečice kombinacijom tipki nisu dostupne na određenim popništima ili su rezervirane za sistemsku upotrebu od nekih verzija Windowsa, najbolje je izbjegavati ih kad razvijate svoju Visual Basic aplikaciju. Ovo su neki primjeri znakova koje treba izbjegavati:

@ $ { } [ ] \ ~ | ^ ‘ < >

Jedan način rada bez ovakvih ograničenja je korištenje brojeva i funkcijskih tipaka (F1, F2, itd.) umjesto slova u prečicama kombinacijom tipki. To može biti manje intuitivno, ali neće zahtijevati nikakvu lokalizaciju, jer zapravo svi rasporedi tipaka sadrže brojke i funkcijske tipke.

> **Napomena** DBCS znakovi ne mogu biti upotrijebljeni kao pristupne tipke i prečice kombinacijom tipki.

Općenita razmišljanja za pisanje međunarodnog koda

Kad razvijate aplikaciju koja će biti lokalizirana – bez obzira programirate li u Visual Basicu ili s drugim alatom – morate uzeti u obzir razlike među jezicima. Trebate prepoznati tekstove koji moraju biti lokalizirani, omogućujući tekstovima da se prošire, uz izbjegavanje zamke povezivanja tekstova.

Tvrdo kodirani lokalizirani tekstovi

Model lokalizacije predstavljen u odlomku “Oblikovanje međunarodnog softvera” predstavio je pojmove bloka podataka i bloka koda. Kad gradite datoteke izvora koje sadržavaju sve lokalizirane tekstove, važno je uključiti *samo* tekstove koji trebaju biti lokalizirani. Svaki dio koji ne treba biti djelomično ili potpuno lokaliziran može biti tvrdo kodiran. Suprotno tome, također je osnovna stvar osigurati da su svi izvori koji trebaju biti lokalizirani, stvarno nazočni u tim datotekama izvora.

Veličina međuspremnika

Ako određujete veličinu međuspremnika na temelju očekivane duljine teksta ili riječi, osigurajte da taj međuspremnik može prihvatiti dulje riječi i tekstove. Pogledajte "Oblikovanje međunarodnog korisničkog sučelja", za postotke prosječnog rasta prevedenih riječi. Veličina međuspremnika koju odredite u vašem kodu Visual Basica treba računati na takvo povećanje.

Razmotrite sljedeći primjer. Vaš Visual Basic određuje 2-bajtnu veličinu međuspremnika za riječ "OK". Međutim, u španjolskom, ista riječ se prevodi s "Aceptar", što će uzrokovati prekoračenje u vašoj aplikaciji. Jednaka razmišljanja se mogu primijeniti za dvobajtne znakove. Pogledajte "Izdanja specifična za dvobajtni skup znakova (DBCS)", kasnije u ovom poglavlju za više informacija o sistemu DBCS.

Povezivanje tekstova

Kad pokušate smanjiti veličinu teksta, jedan mogući pristup je povezivanje tekstova. Taj postupak vam omogućuje korištenje istog izvora u nekoliko tekstova. Međutim, postoje neke opasnosti kod upotrebe ovakvog pristupa. Razmotrite sljedeći primjer:

engleski	francuski
String1: one after the other.	String1: un après l'autre.
String2: The controls will be deleted	String2: Les contrôles seront supprimés
String3: The forms will be deleted	String3: Les feuilles seront supprimées

Obrađeni zasebno, tekstovi String1, String2 i String3 mogu lako biti lokalizirani. Ako vaš kod izvodi String2 + String1 ili String3 + String1, rezultirajući engleski tekst će izgledati u redu. Međutim, lokalizirani tekstovi će vjerojatno biti pogrešni. U francuskom stupcu, na primjer, String3 + String1 će biti pogrešni jer u francuskom jeziku, gramatički, forme (feuilles) je riječ ženskog roda, pa bi String1 trebao biti "une après l'autre", a ne "un après l'autre". Ista situacija će se dogoditi u puno drugih stranih jezika. Jedini način da to izbjegnete je čuvanje izraza String2 i String1, te String3 i String1, zajedno u datoteci izvora.

U gornjem primjeru, redoslijed riječi koje čine rečenicu je isti u engleskom i francuskom jeziku. Međutim, redoslijed riječi nije općenito isti u ta dva jezika, niti u puno ostalih stranih jezika (na primjer, u njemačkom i japanskom jeziku glagol se obično pojavljuje na kraju rečenice). Sljedeći primjer prikazuje takvu situaciju:

engleski	francuski
String1: DLL	String1: DLL
String2: Missing Data Access	String2: Accès aux données manquante
String3: Missing OLE	String3: OLE manquante

Ako vaš kod izvodi String2 + String1 i String3 + String1, lokalizirane verzije će biti isprekidane jer redoslijed ta dva teksta proizvodi poruku koja nema smisla. Jedno moguće rješenje je jednostavno dodati String1 tekstovima String2 i String3 izravno u datoteci izvora i ukloniti String1.

Drugo moguće rješenje je predstavljeno u sljedećoj tablici:

engleski	francuski
String2: Missing Data Access '|1'	String2: '|1' d'accès aux données manquant
String3: Missing OLE '|1'	String3: '|1' OLE manquant

U ovom slučaju, osoba koja lokalizira aplikaciju može prepoznati '|1' kao oznaku mjesta i napraviti potrebne promjene u datoteci izvora kako bi odrazila prikladan način za izgradnju rečenice za lokaliziran jezik.

Na kraju, također je važno znati da riječi ili rečenice koje izgledaju isto u engleskom mogu trebati biti prevedene u drugačije riječi ili rečenice kod lokalizacije. Razmotrite sljedeći primjer:

engleski	francuski
String1: Setup program	String1: Programme d'installation
String2: String1 did not complete successfully.	String2: String1, a échoué.

U engleskoj verziji, String1 se upotrebljava kao natpis aplikacije podešavanja. On se također upotrebljava kao dio poruke pogreške u tekstu String2. U francuskoj verziji, String1 radi savršeno kao samostalni tekst natpisa. Međutim, treba biti promijenjen u "Le programme d'installation" kako bi bio upotrijebljen s tekstom String2.

Pisanje međunarodnog koda u Visual Basicu

Pripremanje proizvoda za upotrebu u drugim poprištima sadržava više od samog prevođenja tekstova poruka. Proizvod mora podržavati nacionalna pravila i pružiti podršku za brojeve određenu posebnostima države. Kako bi znali raditi s različitim datumima, valutama i brojčanim vrijednostima te razdvojnicima, morate razumjeti razlike koje Visual Basic čini između poprišta sustava i poprišta koda.

Poprište sustava protiv poprišta koda

Poprište sustava je poprište korisnika koji izvodi vašu aplikaciju – koristi se kao pokazivač za korisnički ulaz i izlaz podataka i upotrebljava postavke kontrolnog panela koje pruža operativni sustav. *Poprište koda* je uvijek English/U.S. u Visual Basicu, neovisno o međunarodnoj verziji koju upotrebljavate. Poprište koda određuje programski jezik i sve postavke specifične za poprište.

Datum

U Visual Basicu nikad ne upisujte datume kao tekst u vašem kodu. Unos datuma u kod u obliku #mjesec/dan/godina# osigurava ispravno tumačenje datuma na svakom popršitu sustava. Budući da Visual Basic dozvoljava samo postavku English/U.S. kao poprište programiranja, korisniku će datum biti isti bez obzira gdje se izvodi vaša aplikacija.

Na primjer, ako korisnik upiše 8/2/97 u ulazni dijaloški okvir, izraz

```
CDate ("8/2/97")
```

vraća sljedeće rezultate, temeljene na popršitu sustava:

operativni sustav	izlazni rezultat
French/France	08/02/97 (= 8. veljače 1997.)
English/U.S.	8/2/97 (= 2. kolovoza 1997.)

Suprotno tome, ako u kodu unesete 8/2/97, izraz

```
CDate (#8/2/97#)
```

vraća rezultate iz sljedeće tablice, temeljene na popršitu koda:

operativni sustav	izlazni rezultat
French/France	02/08/97 (= 2. kolovoza 1997.)
English/U.S.	8/2/97 (= 2. kolovoza 1997.)

Ako je korisnik u Francuskoj i unese 8/2/97, aplikacija će protumačiti taj datum kao 8. veljače 1997., jer je oblik datuma u Francuskoj dan/mjesec/godina. Ako korisnik u Sjedinjenim Američkim Državama unese isti tekst, aplikacija će to razumjeti kao 2. kolovoza 1997., jer je oblik datuma mjesec/dan/godina.

Valuta

Izbjegavajte pisanje valute kao teksta u svem kodu. Na primjer, sljedeći kod neće se izvesti ni na jednom području osim onih gdje je znak dolara ($) simbol valute.

```
Novac = "$1.22"
NoviNovac = CCur(Novac)
```

Ako izvedete ovaj primjer koda u popršitu French/France, gdje je "F" simbol valute, Visual Basic će stvoriti poruku pogreške "Pogrešan tip podatka" (Type mismatch), jer se znak $ ne prepoznaje kao simbol valute u tom popršitu. Umjesto toga, jednostavno upotrebljavajte brojeve, kao što je pokazano u sljedećem primjeru. Upotrijebite točku kao decimalni razdvojnik, jer je poprište koda za Visual Basic uvijek English/U.S. Sljedeći kod će se ispravno izvoditi, neovisno o popršitu korisnika.

```
Novac = 1.22
NoviNovac = CCur(Novac)
```

Brojčane vrijednosti i razdvojnici

U Sjedinjenim Američkim Državama, točka (.) se upotrebljava kao decimalni razdvojnik. U nekim europskim državama, međutim, kao decimalni razdvojnik se upotrebljava zarez (,). Slično tome, u Sjedinjenim Američkim Državama, zarez se upotrebljava kao razdvojnik tisuća kako bi razdvojio grupe od po tri znamenke lijevo od decimalnog razdvojnika. U nekim europskim državama, za tu namjenu se upotrebljavaju točka ili razmak. Sljedeća tablica ispisuje neke primjere različitih oblika ispis brojeva:

države	oblici ispis brojeva
SAD	1,234,567.89 1,234.56 .123
Francuska	1 234 567,89 1 234,56 0,123
Italija	1.234.567,89 1.234,56 0,123

Napomena U Visual Basicu, funkcije Str i Val uvijek pretpostavljaju da je točka decimalni razdvojnik. U većini poprišta, ta pretpostavka nije valjana. Umjesto toga, upotrebljavajte funkcije CStr, CDbl, CSng, CInt i CLng za pružanje međunarodnih pretvaranja iz bilo kojeg tipa podatka u tip podatka kojeg trebate. Ove funkcije upotrebljavaju poprište sustava za određivanje decimalnog razdvojnika.

Za više informacija Pogledajte idući odlomak "Funkcije svjesne poprišta", za više informacija o funkcijama Print i Format. Pogledajte odlomke "Funkcija CStr" "FunkcijaCSng" i "FunkcijaCInt", te, u Dodatku E, odlomak "Funkcije pretvaranja tipova". Svi ovi odlomci nalaze se u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Funkcije svjesne poprišta

Svako poprište ima različita pravila za prikazivanje datuma, vremena, brojeva, valute i ostalih informacija. Nije potrebno znati sva pravila poprišta vaših korisnika. U Visual Basicu, puno funkcija upotrebljava poprište sustava korisnika, koje upotrebljava postavke kontrolnog panela pružene od operativnog sustava za automatsko određivanje pravila tijekom izvođenja. Ove funkcije se nazivaju *funkcije svjesne poprišta*.

Postupak Print

Iako postupak Print pruža malu fleksibilnost za različite izlazne oblike, upotrebljava poprište korisničkog sustava. U sljedećem primjeru, datumi se ispisuju korištenjem ispravnog kratkog oblika datuma, brojevi se ispisuju s ispravnim decimalnim razdvojnikom, a valute se ispisuju s ispravnim simbolom:

```
MojDatum = #11/24/1997#
MojBroj = 26.5
Novac = 1636.32
MojNovac = Format(Novac "###,###.##")
Debug.Print MojDatum, MojBroj, MojNovac
```

Kad se ovaj kod pokrene u poprištu English/U.S., sljedeći izlazni rezultat će se pojaviti u prozoru za neposredan upis naredbi:

```
11/24/1997  26.5    1,636.32
```

Kad se taj kod pokrene u poprištu German/Germany, sljedeći izlazni rezultat će se pojaviti u prozoru za neposredan upis naredbi:

```
24/11/1997  26,5    1.636,32
```

Za više informacija Pogledajte odlomak "Postupak Print" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Funkcija Format

Funkcija Format može prihvatiti kod oblikovanja, ali kod oblikovanja uvijek proizvodi isti tip izlaznog rezultata neovisno o poprištu korisnika. Na primjer, kod oblikovanja "mm-dd-yy" nije prikladan za korisnika u Belgiji, gdje se dan piše prije mjeseca.

Za veću fleksibilnost, funkcija Format također pruža imenovane oblike koji će automatski ustvrditi će koja pravila primijeniti tijekom izvođenja, uključujući oblike General Date, Long Date, Short Date i Long Time. Upotreba imenovanih oblika proizvodi izlazni rezultat koji je temeljen na poprištu korisničkog sustava. Imenovani formati mogu čak stvoriti izlazni rezultat u domaćem jeziku korisnika, uključujući imena mjeseci i dana u tjednu. Sljedeći primjer to prikazuje:

```
MojDatum = #8/22/1997 5:22:20 PM#
NoviDatum1 = Format(MojDatum "Medium Date")
NoviDatum2 = Format(MojDatum "Short Date")
NoviDatum3 = Format(MojDatum "Long Date")
NoviDatum4 = Format(MojDatum "General Date")
Debug.Print NoviDatum1, NoviDatum2, NoviDatum3, NoviDatum4
```

Kad se ovaj kod pokrene u poprištu English/U.S., sljedeći izlazni rezultat će se pojaviti u prozoru za neposredan upis naredbi:

```
22-Aug-97  8/22/97  Monday, August 22, 1997     8/22/97 5:22:20 PM
```

Kad se taj kod pokrene u području French/France, sljedeći izlazni rezultat će se pojaviti u prozoru za neposredan upis naredbi:

```
22-août-97 22/08/97       lundi 22 août 1997    22/08/97 17:22:20
```

Za više informacija Pogledajte odlomak "Funkcija Format" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Međunarodni redoslijed sortiranja i uspoređivanje teksta

Uspoređivanje teksta se često koristi u Visual Basicu. Upotreba te funkcionalnosti, međutim, može proizvesti neispravne rezultate ako previdite određene zahtjeve programiranja.

Sortiranje teksta

Sortiranje teksta znači raspoređivanje teksta prema pravilima jezika. Oblik i pismo su nebitni za postupak sortiranja jer oboje uključuje način predstavljanja, a ne sadržaj. Na prvi pogled, sortiranje teksta izgleda jednostavno: a dolazi prije *b*, *b* dolazi prije *c* i tako dalje. Međutim, postoji puno jezika koji imaju složenija pravila za sortiranje. Ispravno međunarodno sortiranje nije uvijek jednostavno proširenje sortiranja engleskog teksta, i zahtijeva drugačije shvaćanje postupka sortiranja.

Ispravno međunarodno sortiranje može uključivati *sortiranje osjetljivo na sklop*. Sažimanje i širenje znakova su dva važna područja sortiranja osjetljivog na sklop.

- *Sažimanje znaka* se pojavljuje kad se kombinacija dva znaka obrađuje kao samostalno, jedinstveno slovo. Na primjer, u španjolskom jeziku, kombinacija dva znaka *ch* je samostalno, jedinstveno slovo i u sortiranju se postavlja između *c* i *d*.
- *Širenje znaka* se pojavljuje u slučajevima kad jedno slovo predstavlja jedan znak, ali se taj jedan znak sortira kao da su dva. Na primjer, znak *ß* (oštro s) je jednak *ss* u oba područja, German/Germany i German/Switzerland. Međutim, *ß* je jednak *sz* u području German/Austria.

Prije ostvarivanja redoslijeda sortiranja, morate razmotriti kodne stranice. *Kodna stranica* je poredani skup znakova koji ima brojčane pokazatelje (kodne točke) pridružene svakom znaku. Pošto postoje razne kodne stranice, jedna kodna točka može predstavljati različite znakove u različitim kodnim stranicama. Dok većina kodnih stranica dijeli kodne točke od 32 do 127 (ASCII skup znakova), razlikuju se izvan tog opsega. Tipično, redanje svih dodatnih slova u tim kodnim stranicama nije abecedno.

Za više informacija Pogledajte odlomak "Redoslijed sortiranja i uspoređivanje tekstova s DBCS znakovima", kasnije u ovom poglavlju, za više informacija o radu s jezicima istočne Azije.

Uspoređivanje teksta u Visual Basicu

Pravila uspoređivanja teksta se razlikuju za svako područje. Visual Basic pruža niz alata, kao što su operator Like i funkcija StrComp, koji su svjesni područja. Kako bi ih učinkovito koristili, međutim, izraz Option Compare najprije mora biti jasno objašnjen.

Uspoređivanje tekstova s izrazom Option Compare

Kad upotrebljavate taj izraz, morate odrediti postupak uspoređivanja tekstova: ili Binary ili Text za dani modul. Ako odredite postupak Binary, uspoređivanja se rade u skladu s redoslijedom sortiranja izvedenim iz ugrađenog binarnog predstavljanja znakova. Ako odredite postupak Text, uspoređivanja se rade u skladu s tekstualnim redoslijedom sortiranja neosjetljivim na velika i mala slova, kojeg određuje poprište korisničkog sustava. Podrazumijevan postupak uspoređivanja tekstova je Binary.

U sljedećem primjeru koda, korisnik unosi informacije u dva okvira za upis podataka. Informacije se zatim uspoređuju i sortiraju po odgovarajućem abecednom redoslijedu.

```
Private Sub Form_Click()
Dim ime1 As String, ime2 As String
  ime1 = InputBox("Ovdje upišite prvo ime hardvera:")
  ime2 = InputBox("Ovdje upišite drugo ime hardvera:")
If ime1 < ime2 Then
  por = " ' " & ime1 & " ' dolazi prije ' " & _
  ime2 & " ' "
Else
  por = " ' " & ime2 & " ' dolazi prije ' " & _
  ime1 & " ' "
End If
  MsgBox por
End Sub
```

Ako se ovaj kod izvede na poprištu English/U.S., okvir s porukom će sadržavati sljedeći izlazni rezultat ako korisnik upiše printer i Screen:

```
'Screen' dolazi prije 'printer'
```

Rezultat se temelji na činjenici da je podrazumijevan postupak uspoređivanja teksta Binary. Budući da je ugrađeno binarno predstavljanje velikog slova *S* manje od onog za malo slovo *p*, uvjetni izraz Screen < printer je valjan. Kad dodate izraz Option Compare Text u odjeljak Declarations modula, Visual Basic uspoređuje ova dva teksta na temelju osjetljivom na velika i mala slova, što rezultira sljedećim izlaznim rezultatom:

```
'printer' dolazi prije 'Screen'
```

Ako se ovaj kod izvede na poprištu French/Canada, okvir s porukom će sadržavati sljedeći izlazni rezultat ako korisnik upiše imprimante i écran:

```
'imprimante' dolazi prije 'écran'
```

Slično tome, ako dodate izraz Option Compare Text u vaš kod, dva izraza će se pojaviti u pravom redoslijedu – znači, écran će biti ispred imprimante. Kao dodatak osjetljivosti na mala i velika slova, uspoređivanje uzima u obzir naglašene znakove, kao što je *é* u francuskom jeziku, i postavlja ih odmah iza standardnog znaka – u ovom slučaju, *e*, u redoslijedu sortiranja.

Ako je korisnik upisao ecran i écran, izlazni rezultat će biti:

```
'ecran' dolazi prije 'écran'
```

Za više informacija Pogledajte odlomak "Izraz Option Compare" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Uspoređivanje tekstova operatorom Like

Možete upotrijebiti operator Like za uspoređivanje dva teksta. Možete također upotrijebiti njegove sposobnosti uspoređivanja uzoraka. Kad pišete međunarodni softver, morate biti svjesni funkcija za uspoređivanje uzoraka. Kad se s operatorom Like koriste rasponi znakova, određen uzorak naznačuje raspon redoslijeda sortiranja. Na primjer, s postupkom Binary za uspoređivanje tekstova (standardno ili dodavanjem izraza Option Compare Binary u vaš kod), u rasponu [A – C] će nedostajati naglašeni veliki znakovi, a i svi znakovi malih slova. Uklopit će se samo tekstovi koji počinju s A, B i C. To neće biti prihvatljivo u puno jezika. U njemačkom jeziku, na primjer, u tom rasponu će nedostajati svi tekstovi koji počinju s *Ä*. U francuskom jeziku, neće biti uključen ni jedan od tekstova koji počinju s *Â*.

s postupkom Text za uspoređivanje tekstova, svi naglašeni znakovi, a i, a biti će sadržani u rasponu. U području French/France, međutim, tekstovi koji počinju s ̧ ili ̧ neće biti sadržani, jer se znakovi ̧ i ̧pojavljuju poslije znakova *C* i *c* u redoslijedu sortiranja.

Korištenje raspona [A – Z] za provjeru svih tekstova koji počinju abecednim znakom nije valjan pristup u nekim područjima. s postupkom Text za uspoređivanje tekstova, tekstovi koji počinju znakovima *Ø* i *ø* neće biti uključeni u taj raspon ako se vaša aplikacija izvodi na području Danish/Denmark. Ta dva znaka su dio danske abecede, ali se pojavljuju iza znaka Z. Zbog toga, trebate dodati slova iza slova Z. Na primjer, izraz Print "øl" Like "[A-Z]*" će vratiti vrijednost False, ali će izraz Print "øl" Like "[A-ZØ]*" vratiti vrijednost True s izrazom Option Compare Text.

Uspoređivanje tekstova s funkcijom StrComp

Funkcija StrComp je korisna kad želite usporediti tekstove. Ona vraća vrijednost koja vam kazuje je li jedan tekst manji, jednak ili već od drugog teksta. Povratna vrijednost se također temelji na postupku uspoređivanja tekstova (Binary ili Text) koju određujete izrazom Option Compare. Funkcija StrComp može dati različite rezultate s tekstovima koje uspoređujete, ovisno o tome koji postupak uspoređivanja tekstova odredite.

Za više informacija Pogledajte "Redoslijed sortiranja i uspoređivanje tekstova s DBCS znakovima", kasnije u ovom poglavlju, za više informacija o uspoređivanju tekstova iz jezika istočne Azije. Pogledajte također odlomak "Funkcija StrComp" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Ulaz/izlaz kod međunarodnih datoteka

Područje je također važno uzeti u obzir kod rada s ulazom/izlazom datoteka u Visual Basicu. Izrazi Print # i Write # mogu biti upotrijebljeni za rad s datotekama podataka, ali imaju različite namjene.

Izraz Print

Izraz Print # postavlja podatke u datoteku onako kako se podaci prikazuju na ekranu, u obliku lokalnog poprišta. Na primjer, izlaz podataka upotrebljava sistemski oblik datuma Short Date, a brojčane vrijednosti upotrebljavaju sistemski decimalni razdvojnik.

Izraz Input # ne može čitati podatke svjesne poprišta u Visual Basicu koji su zapisani u datoteku izrazom Print #. Kako bi zapisali podatke neovisne o poprištu koji mogu biti pročitani u Visual Basicu na svakom poprištu, upotrijebite izraz Write # umjesto izraza Print #.

Izraz Write

Slično izrazu Print #, izraz Write # postavlja podatke u datoteku u nepromjenjivom obliku, što osigurava mogućnost čitanja podataka iz datoteke na svakom poprištu kad se koristi izraz Input #. Na primjer, datumi se zapisuju u datoteku korištenjem univerzalnog oblika datuma, a brojčane vrijednosti se zapisuju u datoteku korištenjem točke kao decimalnog razdvojnika. U sljedećem primjeru koda, datum i brojčana vrijednost se zapisuju u datoteku izrazom Write #. Ista datoteka se kasnije ponovno otvara, njezin sadržaj se čita izrazom Input #, i rezultati se ispisuju u prozor za neposredan upis naredbi. Informacija tipa Long Date se uzima iz poprišta sustava:

```
Dim MojDatum As Date, NoviDatum As Date
Dim MojBroj As Variant
  MojDatum = #8/2/67#
  MojBroj = 123.45
Open “Testdato” For Output As #1
  Write #1, MojDatum, MojBroj
Close #1

Open “Testdato” For Input As #1
  Input #1, MojDatum, MojBroj
  NoviDatum = Format(MojDatum “Long Date”)
Debug.Print NoviDatum, MojBroj
Close #1
```

Kad pokrenete ovaj kod na poprištu English/U.S., u prozoru za neposredan upis naredbi će se pojaviti sljedeći izlazni rezultat:

```
Wednesday, August 02, 1967        123.45
```

Kad pokrenete ovaj kod na poprištu French/France, u prozoru za neposredan upis naredbi će se pojaviti sljedeći izlazni rezultat:

```
mercedi 2 août 1967                      123,45
```

U oba poprišta, izlazni rezultat je točan – znači, informacija se ispravno sprema i dohvaća korištenjem izraza Write # i Input #.

Za više informacija Za dodatne informacije o obradi datoteka, pogledajte odlomak "Rad s datotekama" u 14. poglavlju "Obrada pogona, mapa i datoteka". Također pogledajte odlomke "Izraz Print #" i "Izraz Write #" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Upiti tipa SQL svjesni poprišta

Kao što je objašnjeno u odlomku "Pisanje međunarodnog koda u Visual Basicu", različite države imaju različite oblike datuma. Ako vaša aplikacija izvodi uspoređivanje između dva datuma, izrazi datuma moraju biti spremljeni u jedinstvenom obliku kako bi se osiguralo pouzdano uspoređivanje, neovisno o poprištu korisnika. U Visual Basicu, mehanizam baze podataka sprema vrijednost datuma/vremena kao vrijednost tipa DateSerial, koja je predstavljena 8-bajtnim brojem s plivajućim zarezom, s datumom kao cjelobrojnim dijelom i vremenom kao decimalnim dijelom. Takav pristup je potpuno neovisan o poprištu i dopustit će vam izvođenje uspoređivanja datuma/vremena korištenjem međunarodnih oblika datuma/vremena.

Strukturirani jezik upita (Structured Query Language, SQL) je standard tipa ANSI kojeg se pridržava Visual Basic. Datumi se spremaju u tablice i baze podataka koristeći oblik poprišta English/U.S. (mjesec/dan/godina). Taj oblik je također prihvaćen od mehanizma baze podataka Microsoft Jet. Upiti koji upotrebljavaju takva polja mogu vratiti pogrešne zapise ili ih uopće neće vratiti ako se upotrijebi oblik datuma različitog od ovog tipa.

Ovo ograničenje također vrijedi za svojstvo Filter, postupke FindFirst, FindNext, FindPrevious i FindLast objekta Recordset, i za uvjet WHERE izraza tipa SQL.

Korištenje funkcija DateSerial i DateValue

Postoje dvije funkcije koje možete upotrijebiti za rukovanje ograničenjima SQL standarda. Izbjegavajte upotrebu izraza datum/vrijeme u svom kodu. Umjesto toga, razmislite o korištenju funkcija DateValue ili DateSerial za stvaranje datuma kojeg želite. Funkcija DateValue koristi sistemsku postavku Short Date za tumačenje teksta kojeg pribavite; funkcija DateSerial koristi skup argumenata koji će se izvoditi na svakom poprištu. Ako upotrebljavate izraze datum/vrijeme u vašem SQL upitu ili s svojstvom Filter, nemate izbora osim korištenja oblika poprišta English/U.S. za datum i vrijeme.

Sljedeći primjeri pokazuju kako izvesti upit temeljen na datumu. U prvom primjeru, koristi se oblik datuma različit od američkog. Vraćeni objekt Recorset je prazan jer u izrazu datuma postoji sintaksna pogreška:

```
Dim mojdb As Database
Dim mojds As Recordset

Set mojdb = OpenDatabase("Mojabaza.mdb")
' Tablica koja sadrži polje datum/vrijeme.
Set mojds = mojdb.OpenRecordset("Mojatab,dbopenDynaset")
' Oblik datuma je dd/mm/gg.
mojds.FindFirst "DateFiled > #30/03/97#"
' Kontrola podataka je povezana s mojdb.
Data1.Recordset.Filter = "DateFiled = #30/03/97#"
```

```
mojdb.Close
mojds.Close
```

Sljedeći primjer, međutim, radit će prikladno na svakom popristu jer datum u odgovarajućem obliku:

```
Dim mojdb As Database
Dim mojds As Recordset

Set mojdb = OpenDatabase("Mojabaza.mdb")
' Tablica koja sadrži polje datum/vrijeme.
Set mojds = mojdb.OpenRecordset("Mojatab,dbopenDynaset")
mojds.FindFirst "DateFiled > #03/30/97#"        ' Oblik datuma
                                                ' je mm/dd/gg.

' Kontrola podataka je povezana s mojdb.
Data1.Recordset.Filter = "DateFiled = _
DateValue(""" & DateString & """)"

mojdb.Close
mojds.Close
```

Izdanja specifična za dvobajtni skup znakova (DBCS)

Dvobajtni skup znakova (*doble-byte character set*, DBCS) je stvoren za obradu jezika istočne Azije koji upotrebljavaju simbolične znakove, koji zahtijevaju više od 256 znakova podržanih ANSI standardom. Znakovi u DBCS skupu su adresirani 16-bitnim označavanjem, upotrebom 2 bajta. s 16-bitnim označivanjem možete predstaviti 65536 znakova, iako je za jezike istočne Azije određeno daleko manje znakova. Na primjer, japanski skupovi znakova danas određuju oko 12 000 znakova.

U popristima gdje se upotrebljava DBCS skup – uključujući Kinu, Japan, Tajvan i Koreju – u skup znakova uključeni su i jednobajtni i dvobajtni znakovi. Jednobajtni znakovi korišteni u tim popristima usklađeni su s 8-bitnim nacionalnim standardima za svaku državu i blisko se podudaraju s ASCII skupom znakova. Određeni rasponi kodova u tim jednobajtnim skupovima znakova (single-byte character set, SBCS) su oblikovani kao *vodeći bajtovi* DBCS znakova. Uzastopan par sačinjen od vodećeg bajta i pratećeg bajta predstavlja jedan dvobajtni znak. Raspon kodova upotrijebljen za vodeći bajt ovisi o popristu.

Napomena Skup znakova tipa DBCS je različit od skupa znakova tipa Unicode. Pošto Visual Basic interno predstavlja sve tekstove u Unicode obliku, znakovi tipa ANSI i DBCS se pretvaraju u tip Unicode, a znakovi skupa Unicode se pretvaraju u ANSI znakove ili DBCS znakove automatski uvijek kad je potrebna pretvorba. Možete također ručno pretvarati znakove među tipa Unicode i tipa ANSI/DBCS. Za više informacija o pretvaranju između različitih skupova znakova, pogledajte odlomak "Funkcije upravljanja DBCS tekstovima", kasnije u ovom poglavlju.

Kad s Visual Basicom razvijate aplikaciju koja podržava standard DBCS, trebate razmišljati o sljedećim stvarima:

- Razlike među standardima Unicode, ANSI i DBCS.
- Redoslijed sortiranja i uspoređivanje tekstova s DBCS znakovima.
- Funkcije upravljanja DBCS tekstovima.
- Pretvaranje DBCS tekstova.
- Kako ispravno prikazati i ispisati pisma u DBCS okruženju.
- Kako obrađivati datoteke koje uključuju dvobajtne znakove.
- DBCS identifikatori.
- Događaji koji podržavaju skup DBCS.
- Kako pozivati funkcije Windows API-ja.

Savjet Razvijanje aplikacije koja podržava DBCS je dobra praksa, bez obzira hoće li se aplikacija izvoditi na poprištu gdje se upotrebljava DBCS. Takav pristup će vam pomoći u razvijanju fleksibilne prijenosne i potpuno međunarodne aplikacije. Ni jedna od osobina Visual Basica koje podržavaju standard DBCS neće ometati ponašanje vaše aplikacije u okruženjima koja upotrebljavaju isključivo jednobajtne skupove znakova (SBCS), a veličina vaše aplikacije se neće povećati jer oba standarda, DBCS i SBCS, interno koriste standard Unicode.

Za više informacija Za ograničenja kod upotrebe standarda DBCS za pristupne tipke i prečice kombinacijom tipki, pogledajte odlomak “Oblikovanje međunarodnog korisničkog sučelja”, ranije u ovom poglavlju.

ANSI, DBCS i Unicode: definicije

Visual Basic upotrebljava skup Unicode za spremanje i upravljanje tekstovima. Unicode je skup znakova gdje se upotrebljavaju 2 bajta za predstavljanje svakog znaka. Neke druge aplikacije, kao što je Windows 95 API, upotrebljavaju skupove ANSI (American National Standards Institute) ili DBCS za spremanje ili upravljanje tekstovima. Kad prebacite tekst izvan Visual Basica, možete otkriti razlike između skupa Unicode i skupova ANSI/DBCS. Sljedeća tablica prikazuje skupove znakova ANSI, DBCS i Unicode u različitim okruženjima.

okruženje	skup(ovi) znakova
Visual Basic	Unicode
32-bitne biblioteke objekata	Unicode
16-bitne biblioteke objekata	ANSI i DBCS
Windows NT API	Unicode
Automatizacija u Windowsima NT	Unicode
Windows 95 API	ANSI i DBCS
Automatizacija u Windowsima 95/98	Unicode

ANSI

Standard ANSI je najpopularniji standard kojeg koriste osobna računala. Budući da standard ANSI upotrebljava samo jedan bajt za predstavljanje svakog znaka, ograničen je na maksimum od 256 kodova za znakove i interpunkcije. Iako je to prikladno za engleski jezik, ne podržava u potpunosti puno drugih jezika.

DBCS

Standard DBCS se koristi na sustavima Microsoft Windows koji se distribuiraju u većini dijelova Azije. Pruža podršku za puno različitih abeceda jezika istočne Azije, kao što su kineska, japanska i korejska. DBCS upotrebljava brojeve od 0 do 128 za predstavljanje skupa znakova po standardu ANSI. Neki brojevi veći od 128 djeluju kao *znakovi vodećeg bajta*, i to nisu stvarni znakovi već jednostavno pokazivači koji naznačuju da je iduća vrijednost znak iz nelatinskog skupa znakova. U standardu DBCS, ASCII znakovi su dugi samo jedan bajt, dok su japanski, korejski i drugi istočnoazijski znakovi dugi 2 bajta.

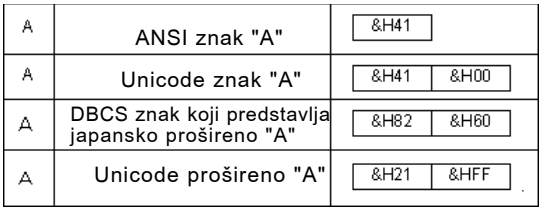
Unicode

Standard Unicode je shema kodiranja znakova koja upotrebljava 2 bajta za *svaki* znak. Međunarodna organizacija za standarde (International Standards Organization, ISO) određuje broj u rasponu od 0 do 65535 (216-1) za gotovo svaki znak i simbol u svakom jeziku (uz dodatne prazne prostore za daljnje proširivanje). Na svim 32-bitnim verzijama Windowsa, standard Unicode se upotrebljava od modela objekata sastavnih dijelova (Component Object Model, COM), temelja za OLE i ActiveX tehnologije. Standard Unicode je potpuno podržan od Windows NT. Iako standardi Unicode i DBCS imaju dvobajtne znakove, sheme kodiranja su potpuno različite.

Primjeri koda znaka

Slika 16.4 prikazuje primjer koda znaka u svakom skupu znakova. Uočite drugačije kodove u svakom bajtu dvobajtnih znakova.

Slika 16.4 Kodovi znaka “A” u standardima ANSI, Unicode i DBCS



Redoslijed sortiranja i uspoređivanje tekstova s DBCS znakovima

Trebate biti svjesni problema kad sortirate i uspoređujete DBCS tekst, jer izraz Option Compare Text ima posebno ponašanje kad se koristi s DBCS tekstovima. Kad koristite izraz Option Compare Binary, uspoređivanja se izvode prema redoslijedu sortiranja izvučenom iz ugrađenih binarnih predstavljanja znakova. Kad koristite izraz Option Compare Text, uspoređivanja se izvode prema tekstualnom redoslijedu sortiranja neosjetljivom na mala i velika slova, kojeg određuje poprište korisničkog sustava.

Izraz "neosjetljiv na mala i velika slova" znači zanemarivanje razlika između velikih slova i malih slova. U DBCS okruženju, to ima dodatni smisao. Na primjer, neki skupovi znakova tipa DBCS (uključujući japanski, tradicionalni kineski i korejski) imaju dva predstavljanja za isti znak: slovo prirodne širine i prošireno slovo. Na primjer, postoji jednobajtni znak "A" i dvobajtni znak "A". Iako se oni prikazuju s različitom širinom znaka, izraz Option Compare Text ih obrađuje kao isti znak. Slična pravila postoje za svaki skup znakova po standardu DBCS.

Trebate biti pažljivi kad uspoređujete dva teksta. Čak i ako su dva teksta proračunati kao isti korištenjem operatora Like ili funkcije StrComp, identični znakovi u tekstovima mogu biti različiti, a također može biti različita i duljina teksta.

Za više informacija Za opće informacije o uspoređivanju tekstova izrazom Option Compare, pogledajte odlomak "Međunarodni redoslijed sortiranja i uspoređivanje teksta", ranije u ovom poglavlju.

Funkcije upravljanja DBCS tekstovima

Iako se dvobajtni znak sastoji od vodećeg bajta i pratećeg bajta, te zahtijeva dva uzastopna bajta za spremanje, mora biti obrađivan kao jedna cjelina u svakoj operaciji koja uključuje znakove i tekstove. Neke funkcije upravljanja tekstovima ispravno rukuju takvim tekstovima, uključujući DBCS znakove, na temelju znakova.

Te funkcije imaju verziju ANSI/DBCS i binarnu verziju i/ili verziju Unicode, kao što je prikazano u sljedećoj tablici. Upotrebljavajte prikladne funkcije, ovisno o svrsi upravljanja tekstom.

Verzije funkcija označene s "B" u sljedećoj tablici namijenjene su posebno za korištenje s stringovima binarnih podataka. Verzije "W" namijenjene su korištenju s Unicode stringovima.

funkcija	opis
Asc	Vraća ANSI ili DBCS kod znaka za prvi znak u stringu.
AscB	Vraća vrijednost prvog bajta u danom stringu koji sadrži binarne podatke.
AscW	Vraća Unicode kod znaka za prvi znak u stringu.
Chr	Vraća string koji sadrži određeni ANSI ili DBCS kod znaka.
ChrB	Vraća binarni string koji sadrži određeni bajt.

funkcija	opis
ChrW	Vraća string koji sadrži određeni Unicode kod znaka.
Input	Vraća određen broj ANSI ili DBCS znakova iz datoteke.
InputB	Vraća određen broj bajtova iz datoteke.
InStr	Vraća prvo pojavljivanje jednog stringa unutar drugog.
InStrB	Vraća prvo pojavljivanje bajta unutar binarnog stringa.
Left, Right	Vraća određen broj znakova s desne ili lijeve strane stringa.
LeftB, RightB	Vraća određen broj bajtova s lijeve ili desne strane binarnog stringa.
Len	Vraća duljinu stringa izraženu brojem znakova.
LenB	Vraća duljinu stringa izraženu brojem bajtova.
Mid	Vraća određen broj znakova iz stringa.
MidB	Vraća određen broj bajtova iz binarnog stringa.

Funkcije bez "B" ili "W" u ovoj tablici ispravno rukuju DBCS i ANSI znakovima. Kao dodatak gornjim funkcijama, funkcija String rukuje DBCS znakovima. To znači da sve te funkcije smatraju DBCS znak kao jedan znak čak i ako se znak sastoji od 2 bajta.

Ponašanje ovih funkcija je različito kad obrađuju SBCS i DBCS znakove. Na primjer, funkcija Mid se u Visual Basicu koristi za vraćanje određenog broja znakova iz stringa. U poprištima koja upotrebljavaju standard DBCS, broj *znakova* i broj *bajtova* nisu neophodno jednaki. Funkcija Mid će vratiti samo broj znakova, a ne broj bajtova.

U većini slučajeva, upotrebljavajte funkcije koje su temeljene na znakovima kad rukujete podacima u stringovima jer te funkcije mogu ispravno rukovati ANSI stringovima, DBCS stringovima i Unicode stringovima.

Funkcije upravljanja stringovima temeljene na bajtovima, kao što su funkcije LenB i LeftB, pružene su za rukovanje podacima u stringu kao binarnim podacima. Kad spremite znakove u varijablu tipa String ili dohvatite znakove iz varijable tipa String, Visual Basic automatski vrši pretvorbu između Unicode i ANSI znakova. Kad rukujete binarnim podacima, upotrebljavajte matricu tipa Byte umjesto varijable tipa String, te funkcije upravljanja stringovima temeljene na bajtovima.

Za više informacija Pogledajte odgovarajuću funkciju u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Ako želite rukovati stringovima binarnih podataka, možete preslikati znakove u stringu u matricu tipa Byte korištenjem sljedećeg koda:

```
Dim MojBajtString() As Byte
‘ Preslikavanje stringa u matricu tipa Byte.
MojBajtString = “ABC”
‘ Prikaz binarnih podataka.
```

```
For i = LBound(MojBajtString) To UBound(MojBajtString)
  Print Right(“ “ + Hex(MojBajtString(i)), 2) + “ ,”;
Next
Print
```

Pretvaranje DBCS tekstova

Visual Basic pruža nekoliko funkcija pretvaranja stringova koje su korisne za DBCS znakove; to su funkcije StrConv, UCase i LCase.

Funkcija StrConv

Općenita mogućnost funkcije StrConv je pretvaranje velikih slova u mala slova, i obratno. Kao dodatak tim mogućnostima, funkcija ima nekoliko mogućnosti specifičnih za tip DBCS. Na primjer, možete pretvoriti prirodno široka slova u proširena slova određivanjem postavke vbWide kao drugog argumenta ove funkcije. Možete pretvoriti jedan tip znaka u drugi, kao što su tipovi hiragana i katakana u japanskom pismu. Funkcija StrConv vam omogućuje određivanje svojstva LocaleID za string, ako je drugačije od vrijednosti sistemskog svojstva LocaleID.

Funkciju StrConv možete također upotrijebiti za pretvaranje Unicode znakova u ANSI/DBCS znakove, i obratno. Obično se tekst u Visual Basicu sastoji od Unicode znakova. Kad trebate rukovati stringovima tipa ANSI/DBCS (na primjer, kako bi izračunali broj bajtova u stringu prije nego što zapišete string u datoteku), možete upotrijebiti tu djelotvornost funkcije StrConv.

Pretvaranje veličine slova kod proširenih slova

Veličinu slova možete promijeniti korištenjem funkcije StrConv s argumentima vbUpperCase ili vbLoverCase, ili korištenjem funkcija UCase ili LCase. Kad upotrebljavate te funkcije, veličina proširenih slova engleske abecede se mijenja u DBCS standardu kao kod ANSI znakova.

Značenja pisma, prikaza i ispisa u DBCS okruženju

Kad upotrebljavate pismo oblikovano samo za SBCS znakove, DBCS znakovi možda neće biti prikazani ispravno u DBCS verziji Windowsa. Trebate promijeniti svojstvo Name objekta Font kad razvijate aplikaciju osposobljenu za DBCS poprišta s engleskom verzijom Visual Basica ili s svakom verzijom jezika SBCS poprišta. Svojstvo Name određuje pismo koje se upotrebljava za prikaz teksta u kontroli, kod iscrtavanja tijekom izvođenja, ili tijekom operacije ispisa. Podrazumijevana postavka ovog svojstva je MS Sans Serif u engleskoj verziji Visual Basica. Kako bi tekst prikazali ispravno u DBCS okruženju, morate promijeniti to svojstvo na pismo prikladno za DBCS okruženje gdje će se vaša aplikacija izvoditi. Možda ćete također trebati promijeniti veličinu pisma mijenjanjem svojstva Size objekta Font. Obično će tekst u vašoj aplikaciji biti najbolje prikazan u pismu veličine 9 točaka na većini platformi istočne Azije, do je pismo veličine 8 točaka tipično za europske platforme.

Ova razmišljanja primjenjuju se također kod ispis DBCS znakova s vašom aplikacijom.

Kako izbjeći mijenjanje postavki pisma

Ako nemate ni jedno pismo osposobljeno za DBCS standard ili ne znate koje je pismo prikladno za ciljnu platformu, postoji nekoliko mogućnosti kako bi radili s primjerima pisama.

U tradicionalnoj kineskoj, pojednostavljenoj kineskoj i korejskoj verziji Windowsa, postoji sposobnost sustava nazvana *udruživanje pisama* (*Font Association*). S korejskim Windowsima, na primjer, udruživanje pisama automatski preslikava svako englesko pismo iz vaše aplikacije u korejsko pismo. Prema tome, i dalje možete vidjeti prikazane korejske znakove, čak i ako vaša aplikacija koristi engleska pisma. Udruženo pismo se određuje postavkom u registru \HKEY_LOCAL_MACHINE\System\CurrentControlSet\control \fontassoc\Associated DefaultFonts u sistemskim registrima platforme izvođenja. S podrškom udruživanju pisama na vašem sustavu, možete izvoditi vaše engleske aplikacije na kineskoj ili korejskoj platformi bez mijenjanja bilo koje postavke pisma. Udruživanje pisama mije dostupno na drugim platformama, kao što su japanski Windowsi.

Druga mogućnost je korištenje pisama tipa System ili FixedSys. Ta pisma su dostupna na svakoj platformi. Zapamtite da pisma tipa System ili FixedSys imaju nekoliko varijacija u veličini. Ako veličina pisma koju ste postavili tijekom izrade (svojstvom Size objekta Font) za neko od tih pisama ne odgovara veličini pisma na računalu korisnika, postavka će možda biti zanemarena, a prikazan tekst će biti odrezan.

Kako promijeniti pismo tijekom izvođenja

Čak i ako imate prije spomenute mogućnosti, takvi izbori imaju ograničenja. Slijedi primjer općeg rješenja za mijenjanje pisma u vašoj aplikaciji tijekom izvođenja. Sljedeći programski kod, koji radi u svakoj jezičnoj verziji Windowsa, primjenjuje pismo specifično za objekt Font određen u argumentu.

```
Private Const DEFAULT_CHARSET = 1
Private Const SYMBOL_CHARSET = 2
Private Const SHIFTJIS_CHARSET = 128
Private Const HANGEUL_CHARSET = 129
Private Const CHINESEBIG5_CHARSET = 136
Private Const CHINESESIMPLIFIED_CHARSET = 134
Private Declare Function GetUserDefaultLCID Lib “kernel32” () As Long

Public Sub SetProperFont(obj As Object)
  On Error GoTo ErrorSetProperFont
  Select Case GetUserDefaultLCID
  Case &H404              ‘ Tradicionalni kineski
    obj.Charset = CHINESEBIG5_CHARSET
    obj.Name = ChrW(&H65B0) + ChrW(&H7D30) + ChrW(&H660E) _
    + ChrW(&H9AD4)        ‘ Novi Ming-Li
    obj.Size = 9
```

```
  Case &H411              ‘ Japan
    obj.Charset = SHIFTJIS_CHARSET
    obj.Name = ChrW(&HFF2D) + ChrW(&HFF33) + ChrW(&H20) + _
    ChrW(&HFF30) + ChrW(&H30B4) + ChrW(&H30B7) + _
    ChrW(&H30C3) + ChrW(&H30AF)
    obj.Size = 9
  Case &H412              ‘ Korejski UserLCID
    obj.Charset = HANGEUL_CHARSET
    obj.Name = ChrW(&HAD74) + ChrW(&HB9BC)
    obj.Size = 9
  Case &H804              ‘ Pojednostavljeni kineski
    obj.Charset = CHINESESIMPLIFIED_CHARSET
    obj.Name = ChrW(&H5B8B) + ChrW(&H4F53)
    obj.Size = 9
  Case Else               ‘ Ostale države
    obj.Charset = DEFAULT_CHARSET
    obj.Name = “”               ‘ Dobivanje podrazumijevanog pisma
                                ‘ korisničkog sučelja.
    obj.Size = 8
  End Select
  Exit Sub
ErrorSetProperFont:
  Err.Number = Err
End Sub
```

Ovaj primjer koda možete preinačiti kako bi se pismo moglo primijeniti s ostalim postavkama pisma, kao što su opcije ispisa.

Obrada datoteka koje koriste dvobajtne znakove

U poprištima gdje se upotrebljava standard DBCS, datoteka može sadržavati dvobajtne i jednobajtne znakove. Budući da je DBCS znak predstavljen s dva bajta, vaš kod Visual Basica mora izbjeći njegovo dijeljenje. U sljedećem primjeru, pretpostavlja se da tekstualna datoteka Testdato sadržava DBCS znakove.

```
‘ Otvaranje datoteke za čitanje podataka.
Open “TESTDATO” For Input As #1

‘ Čitanje svih znakova u datoteci.
Do While Not EOF(1)
  MojZnak = Input(1, #1)         ‘ Čitanje znaka.
  ‘ Izvođenje operacije korištenjem varijable MojZnak.
Loop
Close #1                         ‘ Zatvaranje datoteke.
```

Kad čitate nepromjenjivu duljinu bajtova iz binarne datoteke, upotrijebite matricu tipa Byte umjesto varijable tipa String kako bi spriječili pretvaranje tipa ANSI u tip Unicode u Visual Basicu.

```
Dim MojBajtString(0 To 4) As Byte

Get #1, MojBajtString
```

Kad upotrebljavate varijablu tipa String s funkcijama Input ili InputB za čitanje bajtova iz binarne datoteke, pojavljuje se pretvaranje u tip Unicode kad je rezultat neispravan.

Imajte na umu da imena datoteka i direktorija također mogu sadržavati DBCS znakove.

Za više informacija Za dodatne informacije o obradi datoteka, pogledajte odlomak "Rad s datotekama" u 14. poglavlju "Obrada pogona, mapa i datoteka". Za informacije o tipu podatka Byte, pogledajte odlomak "Tipovi podataka" u 5. poglavlju "Osnove programiranja".

Identifikatori u DBCS okruženju

Znakovi tipa DBCS nisu podržani ni u jednom od sljedećih identifikatora:

- Imena javnih potprograma
- Javne varijable
- Javne konstante
- Ime projekta (određeno u dijaloškom okviru Project Properties)
- Imena klas (svojstvo Name modula klase, korisnička kontrola, stranica sa svojstvima ili korisnički dokument)

Događaj KeyPress osposobljen za DBCS

Događaj KeyPress može obrađivati dvobajtni kod znaka kao jedan događaj. Viši bajt argumenta *keyascii* predstavlja vodeći bajt dvobajtnog znaka, a niži bajt predstavlja prateći bajt koda znaka.

U sljedećem primjeru, događaj KeyPress možete proslijediti okviru s tekstom, bez obzira je li znak kojeg unesete jednobajtan ili dvobajtan.

```
Sub Text1_KeyPress(KeyAscii As Integer)
  MojZnak = Chr(KeyAscii)
  ‘ Izvođenje operacije korištenjem varijable MojZnak.
End Sub
```

Za više informacija Pogledajte odlomak "Događaj KeyPress" u priručniku *Microsoft Visual Basic 6.0 Language Reference*.

Pozivanje funkcija Windows API-ja

Puno funkcija Windows API-ja i DLL-ova vraća veličinu u bajtovima. Ta povratna vrijednost predstavlja veličinu vraćenog stringa. Visual Basic pretvara vraćen string u tip Unicode čak i kad povratna vrijednost i dalje predstavlja veličinu stringa tipa ANSI ili DBCS. Zbog toga, možda nećete biti u mogućnosti upotrijebiti tu vraćenu veličinu kao veličinu stringa. Sljedeći kod ispravno dobiva vraćen string:

```
spremnik = String(145, Chr(" "))
  vra = GetPrivateProfileString(dio,     unos, _
  standardno, spremnik, Len(spremnik) - 1, imedatoteke)
  vrastring = Left(spremnik, Instr(spremnik, Chr(0)) - 1))
```

Za više informacija Za više informacija, pogledajte odlomak "Pristupanje API-ju Microsoft Windowsa" u 4. dijelu "Pristupanje DLL-ovima i Windows API-ju" vodiča *Microsoft Visual Basic 6.0 Component Tools Guide* biblioteke *Microsoft Visual Basic 6.0 Reference Library*, dostupne u verzijama Professional i Enterprise.

Dvosmjerne osobine Visual Basica

Visual Basic je osposobljen za dvosmjernost, bidirekcionalnost (također znano kao "BiDi"). Izraz "dvosmjernost" je opći izraz koji se koristi za opis softverskih proizvoda koji podržavaju arapski i druge jezike u kojima se piše sdesna nalijevo. Preciznije rečeno, dvosmjernost upućuje na sposobnost proizvoda da upravlja tekstom i prikazuje ga za jezike u kojima se piše s lijeva prema desno i jezike u kojima se piše s desna prema lijevo. Na primjer, prikazivanje rečenica s riječima napisanim i na engleskom i na arapskom zahtijeva sposobnost dvosmjernosti.

Microsoft Visual Basic sadržava standardne osobine za stvaranje i izvođenje Windows aplikacija s punom podrškom za dvosmjerne jezike. Međutim, te osobine su operativne samo kad je Visual Basic instaliran u dvosmjernom 32-bitnom okruženju Microsoft Windowsa, kao što su arapski Microsoft Windowsi 95/98. Također su dostupna i druga 32-bitna okruženja Microsoft Windowsa.

Svojstvo RightToLeft je dodano formama, kontrolama i drugim objektima Visual Basica kako bi pružilo jednostavan mehanizam stvaranja objekata s dvosmjernim osobinama. Iako je svojstvo RightToLeft dio svake instalacije Microsoft Visual Basica, operativno je samo kad je Microsoft Visual Basic instaliran u dvosmjerno 32-bitno okruženje Microsoft Windowsa.

Dokumentacija Bidirectional Features opisuje sve dvosmjerne osobine Microsoft Visual Basica. Te informacije možete pronaći u MSDN pomoći ako na kartici Contents otvorite dio Visual Basic Documentation i u dijelu Reference otvorite dio Additional Information. Kliknite također vezu See Also ispod naslova odlomaka kako bi izravno došli do pregleda povezanih tema.

Zbog sukladnosti s Microsoft Visual Basicom 4.0, dvije verzije 32-bitne kontrole Grid (Grid32.ocx) su uključene s Microsoft Visual Basicom 6.0, ali nisu instalirane. Obje se nalaze u mapi \Tools medija proizvoda. Standardna i dvosmjerna verzija se nalaze u podmapama \Controls i \Controls\Bidi, respektivno.

17. POGLAVLJE

Distribuiranje vaših aplikacija

Nakon što stvorite Visual Basic aplikaciju, moći ćete je distribuirati drugima. Možete slobodno distribuirati svaku aplikaciju koju stvorite s Visual Basicom, svakome tko upotrebljava Microsoft Windowse. Vaše aplikacije možete distribuirati na disketama, CD-ima, putem mreže, ili čak intraneta ili Interneta.

Kad distribuirate aplikaciju, postoje dva koraka kroz koje morate proći:

- Pakiranje – morate zapakirati datoteke vaše aplikacije u jedan ili više datoteka tipa .cab koje mogu biti raspakirane na mjestu koje odaberete, i morate stvoriti aplikaciju podešavanja za određene tipove pakiranja. *Datoteka tipa .cab* je sažeta datoteka koja je dobro opremljena za distribuciju na diskovima ili putem Interneta.
- Raspoređivanje – vašu zapakiranu aplikaciju morate premjestiti na mjesto od kuda ju korisnici mogu instalirati. To može značiti kopiranje paketa na diskete ili na lokalni ili mrežni pogon, ili postavljanje paketa na Web stranicu.

Možete koristiti dva alata za pakiranje i raspoređivanje vaših aplikacija: čarobnjaka za pakiranje i raspoređivanje (Package and Deployment Wizard, prije Setup Wizard), ili alat za podešavanje (Setup Toolkit), koji dolaze s vašom instalacijom Visual Basica. Čarobnjak za pakiranje i raspoređivanje automatizira puno postupaka uključenih u distribuiranje aplikacija predstavljajući vam mogućnosti s kojima možete oblikovati vaše .cab datoteke. Alat za podešavanje dopušta vam prilagođivanje dijela onog što se događa tijekom postupka instalacije.

Sadržaj

Čarobnjak za pakiranje i raspoređivanje

Čarobnjak za pakiranje i raspoređivanje (Package and Deployment Wizard) Visual Basica pomaže vam pri stvaranju .cab datoteka za vašu aplikaciju, njihovom grupiranju u cjelinu, ili *paket*, koji sadrži sve informacije potrebne za instalaciju, te isporuku tih paketa krajnjim korisnicima. Čarobnjaka za pakiranje i raspoređivanje možete upotrijebiti za stvaranje paketa koji se distribuiraju na disketama, CD-ima, lokalnom ili mrežnom pogonu, ili Internetom. Čarobnjak za pakiranje i raspoređivanje automatizira većinu posla upletenog u stvaranje i raspoređivanje tih datoteka.

Čarobnjak za pakiranje i raspoređivanje nudi sljedeće mogućnosti:

- Opcija Package pomaže vam pri pakiranju datoteka projekta u .cab datoteku koja zatim može biti raspoređena. U nekim slučajevima ova opcija stvara program podešavanja koji instalira .cab datoteke. Čarobnjak ustanovljava koje datoteke trebate pakirati i vodi vas kroz izbore koji trebaju biti napravljeni kako bi se stvorila jedna ili više .cab datoteka za vaš projekt.
- Opcija Deploy pomaže vam pri isporuci vaših zapakiranih aplikacija na prikladan medij distribucije, kao što su diskete, dio mreže, ili Web stranica.
- Opcija Manage Scripts omogućuje vam pregled i upravljanje skriptima koje ste snimili u prethodnim radovima pakiranja i raspoređivanja s čarobnjakom. Svaki put kad koristite čarobnjaka, snimate skripta koja sadrže sve izbore koje ste napravili. Ta skripta možete ponovno upotrijebiti u kasnijim upotrebama čarobnjaka ako želite upotrijebiti slične postavke i napraviti iste odabire kao i prošli put.

Za više informacija Pogledajte “Pakiranje aplikacije s čarobnjakom”, kasnije u ovom poglavlju, za više informacija o pakiranju vaših projekata te za više informacija o raspoređivanju aplikacije na Internet ili drugo odredište. Pogledajte “Upravljanje skriptima čarobnjaka”, kasnije u ovom poglavlju, za više informacija o stvaranju i korištenju skripata.

Čarobnjak za pakiranje i raspoređivanje nasuprot projekta alata za podešavanje

Čarobnjak za pakiranje i raspoređivanje vodi vas kroz stvaranje i distribuiranje profesionalnih programa podešavanja za vaše Visual Basic aplikacije. Osim stvaranja .cab datoteka za vašu aplikaciju, čarobnjak također stvara program podešavanja aplikacije prevođenjem projekta alata za podešavanje (Setup Toolkit) instaliranog s Visual Basicom. Program podešavanja se naziva setup1.exe.

U većini slučajeva, čarobnjak za pakiranje i raspoređivanje je najbolji način stvaranja i distribuiranja programa setup1.exe za vaše aplikacije. Međutim, ako želite da vaš program podešavanja aplikacije upotrebljava osobine koje ne pruža čarobnjak za pakiranje i raspoređivanje, možete to napraviti mijenjanjem projekta Setup Toolkit. Kao i kod svakog drugog projekta Visual Basica, mogu se promijeniti forme, kod i djelotvornost tog projekta.

Napomena Čarobnjak za pakiranje i raspoređivanje i alat za podešavanje stvaraju programe za podešavanje i medij distribucije samo za aplikacije Visual Basica. Kako bi stvorili programe za podešavanje drugih aplikacija temeljenih na Windowsima, upotrijebite alat za podešavanje pribavljen s tim razvojnim proizvodom ili iz Microsoft Windows SDK-a.

Za više informacija Za više informacija o korištenju alatnog projekta za pakiranje i raspoređivanje vaših aplikacija, pogledajte "Alat za podešavanje", kasnije u ovom poglavlju.

Pokretanje čarobnjaka za pakiranje i raspoređivanje

Čarobnjak za pakiranje i raspoređivanje Visual Basica olakšava vam stvaranje potrebnih .cab datoteka i programa podešavanja za vašu aplikaciju. Kao i ostali čarobnjaci, čarobnjak za pakiranje i raspoređivanje od vas traži informacije tako da može stvoriti postavu točno kako želite.

Postoje tri načina na koja možete pokrenuti čarobnjaka za pakiranje i raspoređivanje:

- Možete ga pokrenuti iz Visual Basica kao dodatak. Ako pokrećete čarobnjaka kao dodatak, najprije morate postaviti potrebna upućivanja u upravitelju dodataka kako bi učitali čarobnjaka. Kad upotrebljavate čarobnjaka kao dodatak, Visual Basic pretpostavlja da želite raditi s projektom kojeg trenutno imate otvorenog. Ako želite raditi s drugim projektom, morate otvoriti taj projekt prije pokretanja čarobnjaka, ili morate upotrijebiti čarobnjaka kao samostalan dio.
- Možete ga pokrenuti kao samostalan dio izvan razvojnog okruženja. Kad pokrenete čarobnjaka kao samostalan dio, od vas se traži da odaberete projekt s kojim želite raditi.
- Možete ga pokrenuti u tihom modu tako da ga pokrenete iz naredbene linije. Pogledajte "Pokretanje čarobnjaka u tihom modu" u ovom dijelu poglavlja za više informacija.

Nakon što pokrenete čarobnjaka, niz ekrana će od vas tražiti informacije o vašem projektu i omogućit će vam biranje opcija za pakiranje. Svaki ekran objašnjava kako se upotrebljava, uključujući neobavezne informacije, te informacije koje moraju biti upisane prije nego što se možete pomaknuti na idući ekran. Ako trebate više informacija o bilo kojem ekranu, pritisnite F1 ili kliknite gumb Help.

Napomena Trebali bi snimiti i prevesti svoj projekt prije nego što pokrenete čarobnjaka za pakiranje i raspoređivanje.

U većini slučajeva, čarobnjak za pakiranje i raspoređivanje je sve što trebate za stvaranje paketa koji je spreman za raspoređivanje. Međutim, ako želite prilagoditi vaš postupak pakiranja za nadalje ili pružiti djelotvornost koja nije podržana čarobnjakom za pakiranje i raspoređivanje, možete promijeniti projekt Setup Toolkit.

Kako pokrenuti čarobnjaka za pakiranje i raspoređivanje iz Visual Basica

1. Otvorite projekt kojeg želite zapakirati ili rasporediti korištenjem čarobnjaka.

   Napomena Ako radite u projektnoj grupi ili imate više istovremeno učitanih projekata, projekt kojeg želite zapakirati ili rasporediti mora biti trenutan projekt prije nego što pokrenete čarobnjaka.

2. Upotrijebite upravitelja dodacima (Add-In Manager) za učitavanje čarobnjaka za pakiranje i raspoređivanje, ako je potrebno: odaberite stavku **Add-In Manager** u izborniku **Add-Ins**, odaberite stavku **Package and Deployment Wizard** s popisa, i kliknite **OK**.

3. Odaberite stavku **Package and Deployment Wizard** iz izbornika **Add-Ins** za pokretanje čarobnjaka.

4. Na glavnom ekranu odaberite jednu od sljedećih opcija:

   - Ako želite stvoriti standardno pakiranje, pakiranje za Internet ili datoteku ovisnosti za projekt, kliknite **Package**.
   - Ako želite rasporediti projekt, kliknite **Deploy**.
   - Ako želite pregledati, editirati ili obrisati skripta, kliknite **Manage Scripts**.

   Za uvod u ove opcije, pogledajte odlomak “Čarobnjak za pakiranje i raspoređivanje”.

5. Nastavite kroz ekrane čarobnjaka.

Kako pokrenuti čarobnjaka za pakiranje i raspoređivanje kao samostalan dio

1. Ako je projekt kojeg želite zapakirati otvoren, snimite ga i zatvorite Visual Basic.

2. Kliknite gumb **Start**, pa kliknite stavku **Package and Deployment Wizard** u podizborniku Visual Basica.

3. U popisu **Project** na uvodnom ekranu, odaberite projekt kojeg želite zapakirati.

   Napomena Možete kliknuti gumb **Browse** ako vaš projekt nije na popisu.

4. Na glavnom ekranu, odaberite jednu od sljedećih opcija:

   - Ako želite stvoriti standardno pakiranje, pakiranje za Internet ili datoteku ovisnosti za projekt, kliknite **Package**.
   - Ako želite rasporediti projekt, kliknite **Deploy**.
   - Ako želite pregledati, editirati ili obrisati skripte, kliknite **Manage Scripts**.

5. Nastavite kroz ekrane čarobnjaka.

Pokretanje čarobnjaka u tihom modu

Korištenjem skripti, vaše projekte možete zapakiravati i raspoređivati u *tihom modu*. U tihom modu, čarobnjak se izvodi bez potrebe da ga pratite kako bi odabirali mogućnosti i pomicali se kroz ekrane. Čarobnjak pakira i raspoređuje vaš projekt korištenjem postavki sadržanih u skriptima.

Tihi mod je posebno koristan ako pakirate i raspoređujete neki projekt kao dio skupne obrade. Na primjer, rano u razvijanju vašeg projekta, možete upotrijebiti čarobnjaka za pakiranje i raspoređivanje za pakiranje vašeg projekta i raspoređivanje na mjesto ispitivanja. Kasnije možete stvoriti datoteku skupne obrade za periodično izvođenje istog pakiranja i raspoređivanja dok ažurirate svoj projekt.

Kako pakirati i raspoređivati u tihom modu

1. Otvorite prozor MS-DOS-a.
2. Upišite ime izvršne datoteke čarobnjaka, pdcmdln.exe, zajedno s stazom i imenom datoteke vašeg Visual Basic projekta, te odgovarajuće argumente naredbene linije, kao što je pokazano u sljedećem primjeru:

```
PDCmdln.exe C:\Projekt1\Projekt1.vbp /p "Pakiranje za Internet"
/d Raspored1 /l "C:\Projekt1\Tihi mod.log"
```

   Napomena Pakiranje i raspoređivanje možete izvesti istovremeno ako odredite oba argumenta, /p i /d, kao što je pokazano u gornjem primjeru. Inače, upotrijebite ili
   /p ili /d.

argument	opis
/p *skriptapakiranja*	Upišite **/p** s imenom prethodno snimljenih skripata pakiranja kako bi tiho zapakirali projekt prema određenim skriptima.
/d *skriptaraspoređivanja*	Upišite **/d** s imenom prethodno snimljenih skripti raspoređivanja kako bi tiho rasporedili projekt prema određenim skriptima.
/l *staza*	Određuje da čarobnjak treba spremiti sve izlazne rezultate, kao poruke pogrešaka i izvješća uspješnosti, u datoteku umjesto njihovog prikazivanja na ekranu. Upišite **/l** s stazom i imenom datoteke u koju će biti spremljeni izlazni rezultati. Ako datoteka ne postoji, čarobnjak će ju stvoriti.

   Napomena Svako ime datoteke ili skripta koje sadrži razmake treba biti zatvoreno u navodnike, kao što je pokazano u gornjem primjeru.

Za više informacija Pogledajte idući odlomak "Pakiranje aplikacije s čarobnjakom", za upute kako upotrijebiti čarobnjaka za pakiranje vašeg projekta. Pogledajte odlomak "Raspoređivanje aplikacije s čarobnjakom" za upute kako upotrijebiti čarobnjaka za raspoređivanje vaših projekata. Pogledajte odlomak "Mijenjanje projekta za podešavanje", kasnije u ovom poglavlju, za više informacija o prilagođavanju postupka instalacije.

Pakiranje aplikacije s čarobnjakom

Pakiranje aplikacije je radnja stvaranja paketa koji može instalirati vašu aplikaciju na računalo korisnika. *Paket* se sastoji od .cab datoteke ili datoteka koje sadrže vaše sažete projektne datoteke i sve ostale potrebne datoteke koje korisnik treba za instaliranje i izvođenje vaše aplikacije. Te datoteke mogu sadržavati programe za podešavanje, pomoćne .cab datoteke, ili ostale potrebne datoteke. Dodatne datoteke razlikuju se ovisno o tipu paketa koji stvarate.

Možete stvoriti dvije vrste paketa – standardne pakete i pakete namijenjene Internetu. Ako namjeravate distribuirati na diskovima, disketama, ili putem mreže, trebali bi stvoriti *standardni paket* za svoju aplikaciju. Ako namjeravate distribuirati putem stranice intraneta ili Interneta, trebali bi stvoriti *Internet paket*.

U većini slučajeva, vašu aplikaciju ćete pakirati korištenjem čarobnjaka za pakiranje i raspoređivanje, koji je pribavljen s Visual Basicom. Aplikacije možete zapakirati ručno, ali čarobnjak pruža korisne prečice i automatizira neke od poslova koje bi sami trebali izvesti u postupku ručnog pakiranja.

> Napomena Kao dodatak, možete zajedno koristiti alat podešavanja i čarobnjaka za pakiranje i raspoređivanje. Projekt Setup Toolkit možete promijeniti kako bi prilagodili vaše programe podešavanja i dodali osobine koje ne pruža čarobnjak za pakiranje i raspoređivanje, te zatim upotrijebili čarobnjaka za pakiranje i raspoređivanje aplikacije.

Kao dodatak stvaranju standardnog i Internet pakiranja, možete također upotrijebiti dio za pakiranje u čarobnjaku za pakiranje i raspoređivanje za stvaranje datoteka ovisnosti. Datoteke ovisnosti ispisuju sastavne dijelove izvođenja koji moraju biti distribuirani s datotekama projekta vaše aplikacije.

> Važno Svaki put kad stvorite paket, trebate postaviti broj verzije vašeg projekta na kartici Make dijaloškog okvira Project Properties. To je posebno važno ako distribuirate nove verzije postojeće aplikacije: bez odgovarajuće promjene brojeva verzije, računalo krajnjeg korisnika može ustanoviti da kritične datoteke ne trebaju biti ažurirane.

Za više informacija Za detaljnije informacije o standardnim paketima i njihovim sadržajima, pogledajte odlomak “Standardni paketi”. Pogledajte odlomak “Internet paketi” za više informacija o Internet paketima i njihovim sadržajima. Pogledajte “Korištenje čarobnjaka za pakiranje i raspoređivanje s alatom za podešavanje”, za više informacija o korištenju ta dva alata zajedno za stvaranje prilagođenih alata podešavanja. Pogledajte odlomak “Datoteke ovisnosti” za objašnjenje sadržaja datoteke ovisnosti. Svi ovi odlomci su kasnije u ovom poglavlju.

Datoteke koje smijete distribuirati

Možete slobodno distribuirati svaku aplikaciju ili sastavni dio kojeg ste stvorili s Visual Basicom. Uz izvršne (.exe) datoteke, vaša aplikacija može zahtijevati druge datoteke, kao što su dinamičke biblioteke (DLL), ActiveX kontrole (.ocx datoteke) ili bitmapirane slike (.bmp datoteke).

Možete legalno distribuirati datoteke primjera aplikacija i sve datoteke koje su izvorno kopirane u poddirektorij \Icons direktorija \Visual Studio\Common\Graphics kad prvi put instalirate Visual Basic na svoj sustav. Microsoft ne daje nikakvo jamstvo, izričito ili posredno, što se tiče sposobnosti prodavanja ili prikladnosti takvih aplikacija, niti preuzima bilo kakve obaveze ili odgovornosti za njihovo korištenje.

Ako ste nabavili verzije Professional ili Enterprise Visual Basica, možete također distribuirati sve datoteke izvorno kopirane u poddirektorije:
\Visual Studio\Common\Graphics i \Program Files\Common Files\ODBC.

Napomena Možda ćete također smjeti distribuirati ostale ActiveX kontrole, izvršne datoteke, i dinamičke biblioteke koje ste kupili. Potražite suglasnost u proizvođačevom ovlaštenju za svaku od datoteka koje namjeravate distribuirati kako bi ustanovili imate li pravo distribuirati datoteku s vašom aplikacijom.

Opći koraci u postupku pakiranja

Neovisno o tipu pakiranja kojeg stvarate ili alatu kojeg upotrebljavate kako bi ga stvorili, postoje neki koraci koji moraju biti poduzeti.

Napomena Čarobnjak za pakiranje i raspoređivanje automatski izvodi puno ovih koraka umjesto vas.

1. **Odredite tip paketa kojeg želite stvoriti.** Možete stvoriti standardni paket za aplikacije temeljene na Windowsima koje će biti distribuirane na disketama, CD-ima ili putem mreže; umjesto toga, možete stvoriti Internet paket za aplikacije koje će biti distribuirane putem Weba. Možete također odabrati stvaranje samo datoteke ovisnosti.
2. **Odredite datoteke koje trebate distribuirati.** Čarobnjak mora odrediti projektne datoteke i datoteke ovisnosti za vašu aplikaciju prije nego što može stvoriti paket. *Projektne datoteke* su datoteke koje su uključene u sam projekt – na primjer, .vbp datoteka i njezin sadržaj. *Datoteke ovisnosti* su datoteke ili sastavni dijelovi izvođenja koje vaša aplikacija zahtijeva za izvođenje. Informacije o ovisnosti su spremljene u datoteci VB6dep.ini, ili u raznim .dep datotekama koje odgovaraju sastavnim dijelovima vašeg projekta.
3. **Odredite gdje na računalu korisnika instalirati datoteke.** Programske datoteke i datoteke podešavanja se obično instaliraju u poddirektorij direktorija Program Files, dok se sistemske datoteke i datoteke ovisnosti obično instaliraju u direktorije \Windows\System ili \Winnt\System32. Vaš program podešavanja to mora uzeti u obzir i odrediti gdje instalirati svaku datoteku.
4. **Stvorite svoj paket.** Čarobnjak stvara paket i program podešavanja (setup1.exe) za njega, upućujući na sve potrebne datoteke. Krajnji rezultat ovog koraka je jedna ili više .cab datoteka i sve potrebne datoteke podešavanja.
5. **Rasporedite svoj paket.** Postupak raspoređivanja sadržava stvaranje vašeg medija distribucije i kopiranje svih potrebnih datoteka na mjesto gdje mu korisnici mogu pristupiti. Za informacije o raspoređivanju, pogledajte “Raspoređivanje aplikacije s čarobnjakom”, kasnije u ovom poglavlju.

Za više informacija Za puni popis uobičajenih datoteka za izvođenje, podešavanje te datoteka ovisnosti, pogledajte odlomak “Datoteke koje trebate distribuirati”, kasnije u ovom poglavlju. Pogledajte odlomak “Datoteke ovisnosti”, kasnije u ovom poglavlju, za više informacija o stvaranju .dep datoteke.

Osobine pakiranja

Korištenjem čarobnjaka za pakiranje i raspoređivanje, lako možete stvoriti profesionalni program podešavanja za vaše aplikacije ili rasporediti Internet aplikaciju na Web. Čarobnjak izvodi sljedeće korake tijekom postupka pakiranja:

- **Automatsko uključivanje glavnog programa podešavanja vaše aplikacije (setup1.exe).** Čarobnjak dodaje aplikaciju Setup Toolkit, Setup1.exe, paketu. Ta datoteka je glavni instalacijski program za vašu aplikaciju.
- **Automatsko stvaranje .cab datoteka vaše aplikacije.** Čarobnjak za pakiranje i raspoređivanje može stvoriti jednu ili više .cab datoteka za vašu aplikaciju.
- **Rad temeljen na skriptima.** Možete odabrati skripta stvorena tijekom drugog zapakiravanja s istim projektom ako želite upotrijebiti iste ili vrlo slične postavke dok se krećete kroz rad s čarobnjakom. To vam može uštedjeti značajno vrijeme. Osim toga, možete upotrijebiti prethodno snimljena skripta za pakiranje projekta u tihom modu. To je posebno korisno kao dio postupka skupnog prevođenja.
- **Neobavezno stvaranje datoteka ovisnosti.** Datoteke ovisnosti označavaju datoteke tijeka izvođenja koje moraju biti uključene s vašom aplikacijom kad se ona distribuira.
- **Automatska podrška pristupu podacima, udaljenoj automatizaciji i DCOM osobinama.** Čarobnjak automatski ustanovljava sadržava li vaš projekt djelotvornost koja mijenja postupak podešavanja. Na primjer, ako uključite određene tipove pristupa podacima, udaljenoj automatizaciji ili DCOM osobinama, možda ćete u svoj paket trebati uključiti pogonitelje ili druge datoteke. Čarobnjak provjerava vaše projekte i prikazuje ekrane koji vam omogućuju da odredite prikladne opcije u takvim slučajevima.
- **Sposobnost dijeljenja datoteka.** Čarobnjak vam omogućuje da instalirate neke datoteke kao djeljive datoteke. To znači da datoteke neće biti uklonjene s sustava tijekom deinstaliranja ako ih upotrebljavaju druge aplikacije.
- **Alternativni položaji datoteka za Internet pakete.** Kod Internet paketa, čarobnjak vam omogućuje da odredite hoće li datoteke ovisnosti biti uključene u program podešavanja ili će biti preuzete s alternativne Web stranice.
- **Zaštitne postavke za Internet pakete.** Ako u vašem projektu ne koristite sučelje IObjectSafety, čarobnjak za pakiranje i raspoređivanje vam dopušta označavanje dijelova vaše aplikacije kao sigurnih.
- **Korisnički određeni položaji za svaku datoteku u projektu.** Većina datoteka ima podrazumijevane položaje na koje se instaliraju, ovisno o tome jesu li projektne datoteke ili sistemske datoteke. Te položaje možete promijeniti ako želite instalirati datoteke na drugačiji položaj.

Za više informacija Pogledajte odlomak “Osobine raspoređivanja”, kasnije u ovom poglavlju, za više informacija o osobinama čarobnjaka za pakiranje i raspoređivanje.

Standardni paketi

Standardan paket je paket koji je oblikovan kako bi bio instaliran programom setup.exe, umjesto preuzimanja .cab datoteka s Web stranice. Stvarate standardne pakete za aplikacije temeljene na Windowsima koje će biti distribuirane disketama, CD-ima, ili mrežom.

Kad stvarate standardni paket, morate pažljivo razmisliti o postupku distribucije kojeg namjeravate upotrijebiti prije nego što stvorite svoj paket. Ako namjeravate upotrijebiti diskete, obično morate stvoriti više .cab datoteka koje mogu biti postavljene na više disketa, umjesto da stvorite jednu veliku .cab datoteku. Opcija čarobnjaka za pakiranje i raspoređivanje omogućuje vam da odredite želite li jednu ili više .cab datoteka te veličinu .cab datoteke (1.44 MB, 1.2 MB, itd.). Ako odaberete više .cab datoteka, čarobnjak razdjeljuje datoteke vaše aplikacije u nekoliko skupova koji ne prelaze naznačenu veličinu.

> Važno Čak i ako je aplikacija koju namjeravate distribuirati na disketama dovoljno mala da stane na jednu disketu kad je zapakirana u jednu veliku .cab datoteku, trebali bi i dalje odabrati opciju s više .cab datoteka tako da kasnije imate pristup postupku raspoređivanja s disketama u čarobnjaku. U ovom slučaju, bit će stvorena samo jedna .cab datoteka.

Ako namjeravate raspoređivati aplikaciju putem mreže ili lokalnog dijela mreže, CD-ima, ili preko Web stranice, možete stvoriti jednu veliku .cab datoteku ili više manjih .cab datoteka.

Dijelovi standardnog paketa

Postoji nekoliko datoteka koje su uvijek dio vaših standardnih paketa. Tu su uključeni:

- **Datoteka setup.exe.** Ova datoteka djeluje kao izvršna prije instalacije. Datoteka setup.exe je prva stvar koja se izvodi na uređaju korisnika u postupku instaliranja, i izvodi potrebne obrade koje se moraju pojaviti prije početka glavne instalacije.
- **Datoteka setup1.exe.** Ova datoteka djeluje kao glavni program podešavanja za vašu aplikaciju.
- **Sve potrebne datoteke za podršku.** Datoteke za podršku su spremljene u direktoriju \Support, unutar direktorija u kojem je stvoren paket. Uz datoteke setup.exe i setup1.exe, ovaj direktorij sadrži datoteke potrebne za prilagođavanje .cab datoteka aplikaciji ako to korisnik želi.
- **Datoteke tipa .cab za vašu aplikaciju.** Internet aplikacije i aplikacije temeljene na Windowsima se pakiraju u .cab datoteke prije distribucije. Datoteka tipa .cab zamjenjuje dugi popis sažetih datoteka aplikacije u prethodnim verzijama Visual Basica. Sve te datoteke se sad nalaze u .cab datoteci. Možete imati samo jednu .cab datoteku za vašu aplikaciju, ili možete stvoriti više .cab datoteka za isporuku na disketama.

Napomena Ako će vaša aplikacija biti izvođena na dvosmjernom (BiDi) operativnom sustavu, trebate ručno dodati datoteku vbame.dll u datoteku Setup.lst koju stvara čarobnjak za pakiranje i raspoređivanje. To možete napraviti dodavanjem datoteke na ekranu Included Files kad radite s čarobnjakom za pakiranje i raspoređivanje, izravnim editiranjem datoteke Setup.lst, ili dodavanjem ulaza za datoteku vbame.dll u datoteku VB6dep.ini tako da se automatski doda datoteci Setup.lst svaki put kad pokrenete čarobnjaka za pakiranje i raspoređivanje.

Osobine pristupa podacima

Ako vaša aplikacija koristi neku od tehnologija Visual Basica za pristup podacima, kao što su objekti za pristup podacima (Data Access Objects, DAO), ActiveX objekti podataka (ActiveX Data Objects, ADO), ili objekti udaljenih podataka (Remote Data Objects, RDO), čarobnjak za pakiranje i raspoređivanje izvodi dva dodatna koraka tijekom postupka pakiranja:

- Ako vaša aplikacija koristi sastavne dijelove tipa ADO, OLEDB ili ODBC, čarobnjak automatski dodaje datoteku imena mdac_typ.exe popisu datoteka koje treba uključiti u vaš paket. Datoteka mdac_typ.exe je samoraspakiravajuća izvršna datoteka koja instalira sve sastavne dijelove potrebne za vašu tehnologiju pristupanja podacima.
- Čarobnjak od vas traži da odaberete odgovarajuću opciju pristupa podacima kad vaša aplikacija sadržava DAO osobine. Možete odabrati odgovarajući postupak – ISAM, ODBCDirect, ODBC kroz Jet itd.

Udaljena automatizacija i DCOM osobine

Ako vaša aplikacija koristi sastavne dijelove udaljenog koda (prije znane kao OLE poslužitelji), trebate stvoriti dva paketa za aplikaciju: jedan program podešavanja za klijenta, i jedan za poslužitelja. Možete upotrijebiti čarobnjaka za pakiranje i raspoređivanje za pakiranje aplikacije, tako da ga jednostavno pokrenete dva puta u istoj projektnoj grupi – jednom za projekt klijenta i drugi put za projekt poslužitelja.

Prije nego što zapakirate klijentski ili poslužiteljski paket, obavezno morate stvoriti potrebne datoteke za udaljenu podršku (.vbr) za projekt i postaviti ih u isti direktorij gdje se nalazi .vbp datoteka za projekt.

Kako stvoriti podršku za udaljenu automatizaciju ili DCOM

1. Otvorite projektnu grupu u Visual Basicu i odaberite projekt koji će djelovati kao poslužitelj.
2. Odaberite **Project Properties** iz izbornika **Project**. Odaberite karticu **Components** dijaloškog okvira **Project Properties** te potvrdite opciju **Remote server Files**.

Kad prevedete projekt s tom opcijom potvrđenom, automatski će biti stvorena .vbr datoteka.

Datoteke registara

Ako vaš projekt upućuje na bilo koju datoteku tipa .reg ili .vbl, vidjet ćete dodatan ekran u čarobnjaku gdje možete odabrati kako bi trebala biti obrađena ta informacija registara. Vaš izbor može biti jednostavno kopiranje datoteka registara na računalo krajnjeg korisnika, ili sistem spremanja informacija u registre i automatsko registriranje na računalu krajnjeg korisnika.

Datoteke koje nedostaju ili imaju pogrešne datume

Dok se pomičete kroz čarobnjaka, može se pojaviti niz dijaloga ako nedostaje neka od datoteka potrebnih za vašu aplikaciju ili ako nekoj datoteci nedostaju ili su pogrešni podaci o datumu. Vaš izbor može biti nastavak bez informacije o ovisnosti za sastavni dio, traženje nedostajućih datoteka, ili stalno označavanje datoteke kao da ne treba oznaku ovisnosti.

Internet paketi

Internet paketi su programi podešavanja temeljeni na .cab datotekama koji su oblikovani za preuzimanje s Web stranice. Internet Explorer upotrebljava postupak poznat kao preuzimanje sastavnog dijela s Interneta (Internet Component Download) za instaliranje vaše Internet aplikacije. Čarobnjak za pakiranje i raspoređivanje automatski uključuje informacije potrebne za taj postupak u paket kojeg stvara.

Postoji nekoliko tipova Visual Basic aplikacija ili sastavnih dijelova koji mogu biti zapakirani za Internet raspoređivanje, uključujući:

- ActiveX kontrole (.ocx datoteke) koje se prikazuju na Web stranici.
- ActiveX .exe ili .dll datoteke, oblikovane za izvođenje na klijentu ili Web poslužitelju.
- ActiveX dokumenti, koji se prikazuju na Web stranici.
- Aplikacije tipa DHTML, klijentske aplikacije koje povezuju HTML stranice s kodom Visual Basica kroz upotrebu dinamičkog HTML jezika.
- Aplikacije tipa IIS, poslužiteljske aplikacije koje povezuju HTML stranice s objektom nazvanim Web klasa. Taj objekt presreće zahtjeve poslužitelja iz pretraživača i odgovara na njih kodom Visual Basica.

Za više informacija Opsežne informacije o postupku preuzimanja sastavnog dijela s Interneta mogu se pronaći u 4. poglavlju “Preuzimanje ActiveX sastavnih dijelova”, u 5. dijelu “Izgradnja Internet aplikacija” u vodiču *Microsoft Visual Basic 6.0 Component Tools Guide* biblioteke *Microsoft Visual Basic 6.0 Language Reference Library*.

Dijelovi Internet paketa

Postoji nekoliko datoteka koje su uvijek dio vaših Internet paketa. Tu su uključeni:

- **Temeljna .cab datoteka za vašu aplikaciju.** Ova datoteka za Internet pakete se koristi kao program podešavanja za vašu aplikaciju. Temeljna .cab datoteka sadrži sastavne dijelove projekta, kao što su izvršne datoteke ili dinamičke biblioteke za vašu aplikaciju ili .ocx datoteke za kontrole, .inf datoteku koja pokazuje na dodatne .cab datoteke i sadrži informacije zaštite i registara, te sve potrebne datoteke ovisnosti koje nisu u dodatnim .cab datotekama.
- **Sve potrebne datoteke podrške.** Datoteke podrške za Internet aplikaciju mogu uključivati HTML datoteke, datoteke aktivnih stranica poslužitelja (Active Server Pages, .asp), grafičke datoteke u raznim oblicima, ili druge datoteke kojima vaša aplikacija mora pristupiti kako bi se izvodila.
- **Sve dodatne .cab datoteke za vašu aplikaciju.** Uz projektne datoteke, aplikacije često upućuju na nekoliko sastavnih dijelova izvođenja, kao što su dinamičke biblioteke izvođenja Visual Basica, zasebne ActiveX kontrole i objekti pristupanja podacima. Ako su ti sastavni dijelovi dostupni unutar zapakiranih .cab datoteka, možete uputiti na te .cab datoteke iz vaše temeljne .cab datoteke, umjesto da te datoteke isporučujete zasebno.

Dodatne .cab datoteke pružaju učinkovit način kojim osiguravate da korisnik ima najažurniju trenutnu verziju sastavnih dijelova. Ako novija verzija sastavnog dijela u dodatnoj .cab datoteci postane dostupna na vanjskoj Web stranici, korisnici koji preuzimaju vašu aplikaciju će automatski dobiti ažuriranu verziju.

Napomena Ako ne možete ili ne želite da podešavanje vaše aplikacije zahtijeva povezivanje s Internetom, možete postaviti dodatne .cab datoteke na poslužitelja unutar svog intraneta. Poslužitelj intraneta često pruža brže preuzimanje i omogućuje korisnicima preuzimanje s sigurne mreže.

Kako radi preuzimanje Internet sastavnog dijela

Nakon što zapakirate svoju Internet aplikaciju ili sastavni dio za preuzimanje, trebate ga rasporediti na određeno mjesto na Web poslužitelju, gdje mu korisnici mogu pristupiti. Obično je vaš paket dio postojeće Web stranice – znači, Web stranica je domaćin vašoj kontroli ili drugom sastavnom dijelu.

Kad korisnik pristupi Web stranici koja je domaćin vašem paketu, sustav šalje vaš paket računalu korisnika. Provjerava se neoštećenost paketa, paket se raspakirava, registrira, instalira i *zatim* aktivira. Sve se to pojavljuje u pozadini i nadzirano je od pretraživača.

Čarobnjak za pakiranje i raspoređivanje igra dvije uloge u prethodno opisanom postupku:

1. Pakira vaše sastavne dijelove i pridružene datoteke u saženu (.cab) datoteku koju pretraživač upotrebljava za preuzimanje vašeg sastavnog dijela. Čarobnjak za paki-

ranje i raspoređivanje određuje koje se datoteke u vašem projektu trebaju izvesti, sakuplja te datoteke, sažima ih u .cab datoteku, i stvara HTML oznaku koja pokazuje na vaš sastavni dio.

2. Raspoređuje vaše zapakirane datoteke na mjesto Web poslužitelja po vašem izboru. Za više informacija o raspoređivanju vašeg Internet paketa za preuzimanje sastavnog dijela, pogledajte "Raspoređivanje aplikacije s čarobnjakom", kasnije u ovom poglavlju.

Pitanja sigurnosti

Kad pripremate Internet aplikacije i sastavne dijelove za preuzimanje, morate ih pakirati u datoteku koja može biti isporučena korisniku kroz pretraživač. Osim toga, morate izvesti nekoliko mjera predostrožnosti kako bi osigurali da vaša aplikacija neće naštetiti računalima korisnika. Te mjere mogu uključivati:

- Digitalno označavanje vaših sastavnih dijelova tako da korisnici mogu provjeriti sadržaj sastavnog dijela i prepoznati vas kao izvor softvera.
- Određivanje razine sigurnosti koje jamči da vaši sastavni dijelovi neće oštetiti računala korisnika niti pokvariti njihove podatke.
- Sređivanje odobrenja za sve sastavne dijelove koji to zahtijevaju. Kad dodate ActiveX kontrolu na Web stranicu, distribuirate ju svim korisnicima koji preuzmu kontrolu s stranice. Osim ako ne odobrite stranicu, postoji malo toga što može spriječiti krajnjeg korisnika da uzme vašu kontrolu i upotrijebi ju u svojim vlastitim aplikacijama. Odobrenje djeluje kao vrsta autorskog prava za vašu kontrolu, sprječavajući nedopuštenu upotrebu.

Pitanja sigurnosti mogu biti riješena unutar čarobnjaka za pakiranje i raspoređivanje. Kad radite s Internet paketom, ekran u čarobnjaku vas pita želite li provjeriti postavke sigurnosti. Taj ekran ispisuje samo objekte u vašem projektu koji ne ostvaruju sigurnosno sučelje nazvano IObjectSafety.

Napomena Označavanje i odobrenje moraju biti napravljeni izvan postupka pakiranja. Trebate urediti odobrenje za sve sastavne dijelove prije nego što zapakirate sastavni dio. Digitalno označavanje može biti napravljeno nakon što zapakirate aplikaciju – čarobnjak za pakiranje i raspoređivanje rezervira prostor unutar .cab datoteke za informacije o digitalnom potpisu.

Za više informacija Pogledajte odlomak "Koraci za pripremanje vašeg sastavnog dijela za preuzimanje" u 4. poglavlju "Preuzimanje ActiveX sastavnih dijelova", u 5. dijelu "Izgradnja Internet aplikacija" u vodiču *Microsoft Visual Basic 6.0 Component Tools Guide* za detaljna objašnjenja o sigurnosti sastavnog dijela, odobravanju i označavanju. Pogledajte "Postavljanje razine sigurnosti za ActiveX sastavne dijelove" u 4. poglavlju "Preuzimanje ActiveX sastavnih dijelova", u 5. dijelu "Izgradnja Internet aplikacija" u vodiču *Microsoft Visual Basic 6.0 Component Tools Guide*, za više informacija o sučelju IObjectSafety.

Datoteke ovisnosti

Datoteka ovisnosti (.dep) sadrži informacije o zahtjevima tijeka izvođenja aplikacije ili sastavnog dijela – na primjer, koje su datoteke potrebne, kako će biti registrirane, i gdje trebaju biti instalirane na korisnikovom uređaju. Možete stvoriti .dep datoteke za standardne projekte u svim verzijama Visual Basica. Ako imate verzije Professional ili Enterprise Visual Basica, možete stvoriti .dep datoteke za ActiveX kontrole, ActiveX dokumente, i ostale ActiveX sastavne dijelove.

Čarobnjak za pakiranje i raspoređivanje koristi .dep datoteke kad pakira vaše aplikacije. On pretražuje sve dostupne informacije ovisnosti za aplikaciju kako bi izgradio opsežni popis informacija o datotekama izvođenja koje aplikacija treba, zatim gradi informacije instaliranja iz te popis. Za standardan paket, informacije iz .dep datoteka se zapisuju u datoteku Setup.lst koja se sprema izvan zapakirane .cab datoteke. Za Internet paket, informacije .dep datoteke se zapisuju u .inf datoteku koja je spremljena izvan pakirane .cab datoteke.

Kad pakirate sastavni dio, imate mogućnost stvaranja .dep datoteke koja će mu se pridružiti kad se raspoređuje. Trebate to napraviti ako ste stvorili sastavni dio kojeg želite distribuirati s informacijom ovisnosti. Preporučljivo je da pakirate i rasporedite vaš sastavni dio prije nego što zapakirate i rasporedite vašu datoteku ovisnosti, tako da pakirajući dio čarobnjaka zna mjesto izvora sastavnog dijela na koje upućuje datoteka ovisnosti.

Tipovi datoteka ovisnosti

U Visual Basica informacije ovisnosti se spremaju u datoteke stvorene čarobnjakom za pakiranje i raspoređivanje ili ih vi stvarate ručno. Postoje dva tipa datoteka koje mogu sadržavati informacije ovisnosti:

- Datoteke tipa .dep sastavnih dijelova – Datoteka ovisnosti sadržava popis datoteka potrebnih za određenu kontrolu ili sastavni dio. Čarobnjak za pakiranje i raspoređivanje koristi tu datoteku kad stvara program podešavanja. Osim toga, čarobnjak može stvoriti ovaj tip .dep datoteke za vas.
- Datoteka VB6dep.ini – Popis datoteka ovisnosti za cijelo razvojno okruženje Visual Basica.

Kad pokrenete čarobnjaka za pakiranje i raspoređivanje, on traži informacije ovisnosti u .dep datotekama i datoteci VB6dep.ini. Ako informacija ovisnosti za sastavni dio ne može biti pronađena ni na jednom mjestu, čarobnjak vas obavješćuje o nedostajućoj informaciji ovisnosti. Možete zanemariti taj propust ili ispraviti problem stvaranjem odgovarajućih datoteka ovisnosti.

> Napomena Ako zanemarite propust, vaša aplikacija možda neće djelovati ispravno nakon instalacije. Međutim, ako ste sigurni da se datoteka ovisnosti već nalazi na uređaju korisnika, možete zanemariti upozorenje i nastaviti rad.

Datoteke ovisnosti sastavnih dijelova

Datoteka tipa .dep sadržava sve datoteke koje su potrebne za određen sastavni dio. Kad kupite ili upotrijebite sastavni dio od proizvođača, dobit ćete od njega .dep datoteku. Na primjer, sve ActiveX kontrole otpremljene s Visual Basicom imaju prateću .dep datoteku. Te .dep datoteke sadrže sve ovisne datoteke koje upotrebljava kontrola, uz informacije o verziji i registrima.

Trebali bi stvoriti .dep datoteku za svaki sastavni dio koji stvorite u Visual Basicu ako taj sastavni dio može biti upotrijebljen u drugom projektu. Informacije iz .dep datoteke za svaki sastavni dio u projektu spajaju se za oblikovanje informacije ovisnosti projekta. Ako ne stvorite .dep datoteku za vaš sastavni dio, informacije ovisnosti za sve projekte u kojima se on upotrebljava mogu biti neispravne.

Datoteka VB6dep.ini

Datoteka VB6dep.ini pruža čarobnjaku za pakiranje i raspoređivanje popisa ovisnosti i upućivanja korištenih od Visual Basica za sve namjene. Ta popis se stvara kad instalirate Visual Basic i nalazi se u poddirektoriju \Wizards\PDWizard glavnog direktorija Visual Basica.

Informacije ovisnosti koje nedostaju

Čarobnjak za pakiranje i raspoređivanje će vas obavijestiti ako nedostaje informacija ovisnosti za sastavni dio u vašem projektu. Postoje tri načina na koja možete dodati potrebne informacije ovisnosti:

- Editirajte datoteku VB6dep.ini i ručno dodajte unos za određen sastavni dio.
- Stvorite .dep datoteku za sastavni dio s čarobnjakom za pakiranje i raspoređivanje.
- Kontaktirajte proizvođača sastavnog dijela i zatražite .dep datoteku.

Raspoređivanje aplikacije s čarobnjakom

Raspoređivanje aplikacije je postupak premještanja vaše zapakirane aplikacije na medije distribucije koje ste odabrali ili na Web stranicu s koje paket može biti preuzet. Postoje dva načina na koje možete rasporediti vašu Visual Basic aplikaciju:

- Možete upotrijebiti dio Deployment čarobnjaka za pakiranje i raspoređivanje kako bi rasporedili vašu aplikaciju na diskete, lokalni ili mrežni pogon, ili na Web stranicu.
- Možete ručno kopirati datoteke na diskove ili dio mreže, ili možete ručno postaviti datoteke na odgovarajuće mjesto na Webu.

Čarobnjak za pakiranje i raspoređivanje pruža prečice i automatski izvodi neke od istih zadataka koje trebate obaviti sami ako ručno raspoređujete svoju aplikaciju.

Opći koraci postupka raspoređivanja

Bez obzira raspoređujete li svoje pakete čarobnjakom za pakiranje i raspoređivanje ili ručno, postoje određeni koraci koji moraju biti poduzeti.

1. **Stvorite paket za raspoređivanje.** To može biti jedna .cab datoteka ili niz .cab datoteka, ovisno o tome kako namjeravate distribuirati svoju aplikaciju.
2. **Odaberite paket kojeg želite rasporediti.** Možete odabrati svaki valjan paket za odabrani projekt.
3. **Odaberite postupak raspoređivanja.** Svoju aplikaciju možete rasporediti na Internet, diskete, ili u direktorij lokalnog ili mrežnog pogona.
4. **Odaberite datoteke za raspoređivanje.** Ako raspoređujete na Internet, možete dodati ili maknuti datoteke s popisa datoteka koje će biti raspoređene.
5. **Ustanovite odredište za datoteke koje će biti raspoređene.** Za raspoređivanje na Internet, to uključuje određivanje Web mjesta na koje paket treba biti raspoređen. Za raspoređivanje u direktorij, to znači naznačivanje položaja pogona na kojeg paket treba biti raspoređen. Za raspoređivanje na diskete, to znači odabir odgovarajućeg disketnog pogona.
6. **Rasporedite svoj paket.** Ako upotrebljavate čarobnjaka za pakiranje i raspoređivanje, on će rukovati tim postupkom umjesto vas. Ako ga ne upotrebljavate, morate kopirati datoteke na odgovarajuća mjesta na zajedničkom ili lokalnom pogonu, ili postaviti vaše datoteke na Web.

Osobine raspoređivanja

Upotrebom čarobnjaka za pakiranje i raspoređivanje, možete lako kopirati svoje zapakirane aplikacije na odgovarajuće mjesto. Čarobnjak za pakiranje i raspoređivanje izvodi sljedeće korake, s vašim ulaznim podacima, tijekom postupka raspoređivanja:

- **Odabir postupaka raspoređivanja.** Možete odabrati raspoređivanje na diskete, lokalni ili mrežni pogon, ili mjesto na intranetu ili Internetu.
- **Rad temeljen na skriptima.** Možete odabrati skripta iz drugog rada s raspoređivanjem za isti projekt ako želite upotrijebiti iste ili vrlo slične postavke dok se krećete kroz čarobnjaka. To vam može uštedjeti značajno vrijeme.
- **Automatski pristup tehnologiji objavljivanja na Webu.** Tehnologija Web Publishing pojednostavljuje objavljivanje datoteka na intranetu ili stranicama Interneta.

Za više informacija Za više informacija o objavljivanju na Webu pogledajte “Alati i tehnologije Interneta” u dijelu *Internet Client SDK*, stalne pomoći. Za ostale osobine čarobnjaka pogledajte “Osobine pakiranja”, ranije u ovom poglavlju.

Raspoređivanje vaše aplikacije

Čarobnjak vam nudi izbor između raspoređivanja na Web korištenjem tehnologije Web Publishing, ili raspoređivanja na diskete ili u direktorij na lokalnom ili mrežnom pogonu.

Raspoređivanje na diskete, direktorije ili CD-e

Raspoređivanje možete napraviti na diskete korištenjem čarobnjaka za pakiranje i raspoređivanje samo ako ste stvorili standardan paket korištenjem opcije Multiple Cabs. Ta opcija osigurava da se vaš paket sastoji od više .cab datoteka ili jedne .cab datoteke koja je manja od veličine diskete. Sustav vam daje mogućnost formatiranja svake diskete prije nego što na nju kopirate svoje .cab datoteke. Ne trebate formatirati disketu, ali morate upotrijebiti prazne diskete za ovaj postupak raspoređivanja.

Ako odaberete raspoređivanje u direktorij, sustav od vas traži da odaberete lokalni ili mrežni direktorij u kojeg će vaše datoteke biti kopirane. Nakon toga možete uputiti svoje korisnike da pristupe programu podešavanja za vašu aplikaciju s tog mjesta, ili možete premjestiti svoje datoteke na CD-ROM-ove.

> Napomena Ako imate pogon snimača CD-a, možete kopirati svoje datoteke izravno na taj pogon korištenjem dijela za raspoređivanje u čarobnjaku, umjesto da raspoređujete datoteke u direktorij i zatim ih kopirate na svoje CD-e.

Raspoređivanje na Web

Možete rasporediti svaki paket, bez obzira je li standardni li Internet paket, na Web. Kad odaberete postupak Web Publishing kao vaš postupak raspoređivanja, sustav smatra projektnu mapu kao *lokalnu temeljnu mapu* za svoje raspoređivanje. Lokalna temeljna mapa se koristi za određivanje kako se datoteke i direktoriji trebaju kopirati na Web stranicu koju odaberete. Datoteke i direktoriji koji su unutar lokalnog temeljnog direktorija će biti raspoređeni na Web poslužitelja s istim ustrojem direktorija kao i u temeljnom direktoriju.

> Napomena U pravilu, čarobnjak ne raspoređuje izvorne datoteke iz projektnog direktorija ili poddirektorija \Support. Pakirajući dio čarobnjaka stvara direktorij \Support i postavlja u njega datoteke koje mogu biti upotrijebljene za ponovno stvaranje vaših .cab datoteka.

Upravljanje skriptima čarobnjaka

Kad radite s čarobnjakom za pakiranje i raspoređivanje, možete stvarati i spremati skripta. *Skripta* su zapis odabira koja ste napravili tijekom pakiranja ili raspoređivanja. Stvaranje skripata čuva te odabire tako da ih možete primijeniti u idućim radovima s čarobnjakom za isti projekt. Korištenje skripata može vam uštedjeti značajno vrijeme tijekom vaših pakiranja i raspoređivanja. Osim toga, možete upotrijebiti skripta za pakiranje i raspoređivanje vaše aplikacije u tihom modu.

Svaki put kad pakirate ili raspoređujete projekt, Visual Basic sprema informacije o tom radu kao skripta. Sva skripta za projekt su spremljene u posebnoj datoteci unutar direktorija s projektom aplikacije. Možete vidjeti potpuni popis skripata za trenutan projekt korištenjem opcije Manage Scripts u čarobnjaku za pakiranje i raspoređivanje. Ta opcija vam omogućuje:

- Pregled popisa svih skripata pakiranja ili raspoređivanja.
- Promjenu imena skripata.
- Stvaranje kopije skripata s novim imenom.
- Brisanje skripata koje više ne trebate.

Oprez Ako uklonite skripta pakiranja, dio za raspoređivanje u čarobnjaku za pakiranje i raspoređivanje više neće prepoznati paket stvoren tim skriptima kao paket kojeg može rasporediti. Nakon toga ćete trebati ponovno zapakirati datoteke kako bi ih mogli rasporediti. Obrišite samo ona skripta za koja ste sigurni da ih više nećete trebati.

Kako vidjeti popis skripata

1. Pokrenite čarobnjaka i odaberite opciju **Manage Scripts** s glavnog ekrana.

   Važno Ako ste pokrenuli čarobnjaka kao samostalnu aplikaciju, morate odabrati projekt Visual Basica kojeg želite prije odabira opcije Managing Scripts.

2. Odaberite odgovarajući panel za skripta koja želite vidjeti.

Alat za podešavanje

Alat za podešavanje (Setup Toolkit) je projekt instaliran s Visual Basicom kojeg upotrebljava čarobnjak za pakiranje i raspoređivanje kad stvara program podešavanja. Projekt Setup Toolkit sadrži forme i programski kod koje program podešavanja aplikacije koristi za instaliranje datoteka na računalo korisnika. Kad upotrebljavate čarobnjaka za pakiranje i raspoređivanje, čarobnjak uključuje datoteku setup1.exe koju stvara projekt Setup Toolkit. Ta datoteka se koristi kao glavna instalacijska datoteka aplikacije.

Napomena Postoje dva programa podešavanja upletena u postupak instalacije – setup.exe i setup1.exe. Program setup.exe izvodi obradu koja prethodi instalaciji na računalo korisnika, uključujući instaliranje programa setup1.exe i svih drugih datoteka potrebnih za izvođenje glavnog instalacijskog programa. Samo program setup1.exe se može prilagoditi kroz alat za podešavanje.

Osim što ima sporednu ulogu u postupku stvaranja programa za podešavanje, alat za podešavanje može biti upotrijebljen za mijenjanje ekrana koji se vide u postupku instalacije, ili za izravno stvaranje programa podešavanja. Možete stvoriti korisnički program podešavanja ako svom tijeku instalacije trebate dodati dodatnu djelotvornost koja nije podržana čarobnjakom.

Projekt Setup Toolkit nalazi se u poddirektoriju \Wizards\PDWizard\Setup1 glavnog direktorija Visual Basica.

Oprez Datoteke u tom projektu su iste datoteke koje koriste izlazni rezultati čarobnjaka za pakiranje i raspoređivanje. Nemojte ih mijenjati ako prethodno niste napravili rezervnu kopiju u drugom direktoriju. Ako promijenite program setup1.exe, svi sljedeći programi stvoreni čarobnjakom za pakiranje i raspoređivanje će upotrebljavati promijenjenu verziju.

Alat za podešavanje koristite učitavanjem datoteke Setup1.vbp u Visual Basic i mijenjanjem izgleda i djelotvornosti tog projekta. Dok to radite, možda ćete trebati ručno proći kroz korake koje bi inače napravio čarobnjak za pakiranje i raspoređivanje umjesto vas. Sljedeći odlomci opisuju korake u tom postupku i objašnjavaju kako ustanoviti koje datoteke trebate uključiti u vaše podešavanje, kako stvoriti datoteku Setup.lst, kako stvoriti medij distribucije, te kako ispitati vaše podešavanje.

Općeniti koraci za mijenjanje čarobnjaka za pakiranje i raspoređivanje

Kad mijenjate alat za podešavanje s namjerom mijenjanja izlaznog rezultata stvorenog čarobnjakom za pakiranje i raspoređivanje, trebate slijediti ove korake:

1. Promijenite projekt Setup Toolkit tako da sadrži sve nove upite, ekrane, funkcije, programski kod, ili druge informacije koje želite uključiti. Kad ste gotovi, prevedite projekt kako bi stvorili datoteku setup1.exe.
2. Pokrenite čarobnjaka za pakiranje i raspoređivanje, slijedeći upite na svakom ekranu, kako bi stvorili svoj medij distribucije.

Općeniti koraci stvaranja korisničkog programa podešavanja

Kad stvarate program podešavanja ručno korištenjem alata za podešavanje umjesto čarobnjaka za pakiranje i raspoređivanje, morate slijediti ove korake:

1. Ako je potrebno, promijenite projekt Setup Toolkit tako da sadrži sve nove upite, ekrane, funkcije, programski kod ili druge informacije koje želite uključiti.
2. Odredite datoteke koje želite distribuirati, uključujući sve datoteke izvođenja, podešavanja i ovisnosti.
3. Odredite gdje na korisnikovom računalu treba instalirati datoteke.
4. Ručno stvorite svoju datoteku Setup.lst koja će odražavati imena i mjesta instalacije svih datoteka koje moraju biti uključene u vaš projekt.
5. Odredite kako ćete distribuirati datoteke.
6. Stvorite .cab datoteke za svoj projekt koristeći uslužnu aplikaciju Makecab.

   Savjet Možete upotrijebiti čarobnjaka za pakiranje i raspoređivanje kako bi stvorili svoje .cab datoteke, te zatim ručno promijeniti .cab datoteke. Kad čarobnjak stvori vaše .cab datoteke, stvara .ddf datoteku i skupnu datoteku u poddirektoriju \Support direktorija vašeg projekta. Kako bi promijenili .cab datoteke, editirajte .ddf datoteku, te pokrenite pruženu skupnu datoteku. Skupna datoteka će zauzvrat pokrenuti aplikaciju Makecab.exe za ponovno stvaranje vaših .cab datoteka.

7. Stvorite program setup1.exe za vaš projekt prevođenjem projekta Setup Toolkit s vašim promjenama.
8. Kopirajte vaše datoteke na medij distribucije, ili ručno postavite vaše datoteke na Web stranicu korištenjem čarobnjaka za objavljivanje na Webu (Web Publishing Wizard), dostupnog u ActiveX SDK-u.

Za više informacija Za više informacija o korištenju čarobnjaka za objavljivanje na Webu, pogledajte "Alati i tehnologije Interneta" u dijelu *Internet Client SDK*, stalne pomoći. Pogledajte sljedeći odlomak "Mijenjanje projekta za podešavanje", za više informacija o mijenjanju projekta Setup Toolkit. Pogledajte odlomke "Datoteke koje trebate distribuirati" i "Gdje na korisnikovom uređaju instalirati datoteke" za više informacija o tome kako postaviti datoteke na računalo korisnika, te odlomak "Ručno stvaranje medija distribucije" za više informacija o kopiranju vaših datoteka na prikladan medij, sve u nastavku ovog poglavlja.

Mijenjanje projekta za podešavanje

Možete promijeniti projekt Setup1.vbp ako želite dodati nove ekrane, upite ili događaje slijedu instalacije stvorenom čarobnjakom za pakiranje i raspoređivanje. Programski kod za program podešavanja pišete kao što bi ga pisali u svakoj aplikaciji Visual Basica. Dostupan je niz funkcija koje su posebno korisne u rutinama podešavanja.

Neki primjeri situacija u kojima možete promijeniti projekt Setup Toolkit uključuju:

- Trebate dodati posebne korisničke upite tijekom instalacije.
- Želite stvoriti prilagođen izgled i osjećaj vašeg programa podešavanja.
- Želite prikazati panele s obavijestima tijekom instalacije. Paneli s obavijestima predstavljaju informacije o osobinama, servisu i podršci, registraciji, te ostalim srodnim informacijama o vašem proizvodu.
- Želite upotrijebiti svoju vlastitu uslužnu aplikaciju za sažimanje kako bi kopirali datoteke vaše aplikacije na medij distribucije.

  Važno Budući da čarobnjak za pakiranje i raspoređivanje koristi datoteke u projektu Setup Toolkit, uvijek bi trebali napraviti rezervnu kopiju projekta prije bilo kakvih promjena. Osim toga, trebali bi napraviti i rezervnu kopiju cijelog sadržaja direktorija Setup1.

Kako promijeniti projekt Setup Toolkit

1. Napravite rezervnu kopiju datoteke \Wizards\PDWizard\setup1.exe te cijelog sadržaja direktorija \Wizards\PDWizard\Setup1 prije bilo kakvih promjena.
2. Otvorite projekt setup1.vbp iz direktorija \Wizards\PDWizard\Setup1.
3. Napravite sve promjene u programskom kodu, formama ili modulima ovog projekta.
4. Snimite projekt i prevedite ga kako bi stvorili datoteku setup1.exe.

5. Ako koristite čarobnjaka za pakiranje i raspoređivanje za pakiranje vaše aplikacije, pokrenite čarobnjaka za pakiranje i raspoređivanje i stvorite paket za voju aplikaciju.
6. Ako stvarate svoj vlastiti korisnički paket podešavanja, nastavite kroz korake istaknute u odlomku “Alat za podešavanje”.

   Važno Svaki put kad stvorite paket, korištenjem čarobnjaka za pakiranje i raspoređivanje ili projekta Setup Toolkit, provjerite jesu li brojevi verzije za vaš projekt postavljeni na kartici Make dijaloškog okvira Project Properties u Visual Basicu. To je posebno važno ako distribuirate novu verziju postojeće aplikacije – bez prikladne promjene brojeva verzije, računalo krajnjeg korisnika može ustanoviti da kritične datoteke ne trebaju biti ažurirane.

Datoteke koje trebate distribuirati

Prvi korak stvaranja korisničkog programa podešavanja je određivanje koje datoteke distribuirati. Sve Visual Basic aplikacije trebaju minimalan skup datoteka, označen kao *samopodizajuće datoteke* (*bootstrap files*), koje su potrebne prije nego što vaša aplikacija može biti instalirana. Osim toga, sve Visual Basic aplikacije zahtijevaju datoteke specifične za samu aplikaciju, kao što su izvršne datoteke (.exe), datoteke s podacima, ActiveX kontrole, ili .dll datoteke.

Postoje tri glavne kategorije datoteka potrebnih za izvođenje i distribuciju vaše aplikacije:

- Datoteke izvođenja
- Datoteke podešavanja
- Datoteke specifične za aplikaciju

Datoteke izvođenja

Datoteke izvođenja su datoteke koje vaša aplikacija mora imati kako bi ispravno radila nakon instalacije. Ove datoteke su potrebne za sve Visual Basic aplikacije. Slijedi spisak datoteka izvođenja za Visual Basic projekte:

- Msvbvm60.dll
- Stdole2.tlb
- Oleaut32.dll
- Olepro32.dll
- Comcat.dll
- Asycfilt.dll
- Ctl3d32.dll

Iako su te datoteke potrebne za sve Visual Basic aplikacije, možda neće biti neophodne za svaki tip instalacijskog paketa. Na primjer, kad stvarate Internet paket, čarobnjak za pakiranje i raspoređivanje pretpostavlja da svako računalo sposobno za preuzimanje s Interneta već ima sve te datoteke osim datoteke Msvbvm60.dll. Zbog toga je to jedina datoteka izvođenja koju čarobnjak uključuje u Internet paket.

Napomena Datoteke izvođenja mogu nadalje biti razvrstane prema njihovom mjestu instaliranja. Pogledajte odlomak "Gdje na korisnikovom uređaju instalirati datoteke" za više informacija.

Datoteke podešavanja za standardne pakete

Datoteke podešavanja su jedine datoteke koje su potrebne za podešavanje vaše standardne aplikacije na uređaju korisnika. One uključuju izvršne datoteke podešavanja (setup.exe i setup1.exe), popis datoteka podešavanja (Setup.lst), te aplikaciju deinstaliranja (st6unst.exe).

Aplikacije Visual Basica koje su oblikovane za distribuciju na disketama, CD-ima, ili s mjesta na mreži, koriste iste datoteke podešavanja, neovisno o tome upotrebljavate li čarobnjaka za pakiranje i raspoređivanje ili alat za podešavanje kako bi stvorili svoje programe podešavanja. Te datoteke su ispisane u sljedećoj tablici.

ime datoteke	opis
setup.exe	Program kojeg pokreće korisnik kako bi na njegov uređaj bile instalirane datoteke potrebne za vašu aplikaciju. Na primjer, datoteka setup.exe instalira datoteku setup1.exe, dinamičku biblioteku izvođenja Visual Basica, te ostale datoteke bez kojih se ne može izvoditi ostatak postupka podešavanja.
setup1.exe	Program podešavanja za vašu Visual Basic aplikaciju. Ova izvršna datoteka se stvara alatom za podešavanje i uključena je u paket čarobnjakom za pakiranje i raspoređivanje. Možete promijeniti ime ovoj datoteci ako je novo ime sadržano u datoteci Setup.lst.
Setup.lst	Tekstualna datoteka koja sadrži upute instalacije i popis sve datoteke koje će biti instalirane na korisnikov uređaj.
Vb6stkit.dll	Biblioteka koja sadrži razne funkcije korištene u datoteci Setup1.exe.
St6unst.exe	Uslužni program za uklanjanje aplikacije.

Napomena Aplikacija oblikovana za isporuku putem Interneta općenito ne koristi ni jednu od ovih datoteka. Pogledajte odlomak "Internet paketi" ranije u ovom poglavlju za više informacija o datotekama sadržanim u isporuci putem Interneta.

Ovisnosti aplikacije

Kako bi izvodili vašu aplikaciju, krajnji korisnici će trebati određene datoteke osim uobičajenih datoteka izvođenja i posebnih datoteka podešavanja. Većina tih datoteka biti će vam očigledna: izvršna datoteka, sve datoteke s podacima, te sve ActiveX kontrole koje ste upotrebljavali. Manje očigledne datoteke su ostale datoteke ovisnosti vašeg projekta. Na primjer, neke od ActiveX kontrola koje upotrebljava vaš projekt mogu zauzvrat zahtijevati druge datoteke. Jedan od zadataka čarobnjaka za pakiranje i raspoređivanje je da ustanovi potpuni popis takvih potrebnih datoteka.

Za više informacija Pogledajte odlomak "Datoteke ovisnosti", ranije u ovom poglavlju, za informacije o korištenju čarobnjaka za pakiranje i raspoređivanje kod stvaranja datoteka ovisnosti za vašu aplikaciju.

Gdje na korisnikovom uređaju instalirati datoteke

Prije zapisivanja vašeg programa podešavanja, morate ustanoviti gdje na uređaju korisnika treba instalirati sve potrebne datoteke. Te informacije zapisujete u datoteku Setup.lst. Pogledajte odlomak "Ručno editiranje datoteke Setup.lst" u ovom poglavlju za više informacija o tome kako zapisati te informacije u datoteku.

Datoteke potrebne za vašu aplikaciju mogu se podijeliti u nekoliko kategorija.

- Datoteke specifične za aplikaciju – datoteke koje zahtijeva vaša aplikacija kako bi se izvodila i koje ne koriste druge aplikacije.
- Dijeljene datoteke – datoteke koje koristi vaša aplikacija, ali im također pristupaju i druge aplikacije na uređaju korisnika.
- Sastavni dijelovi poslužitelja udaljene automatizacije – datoteke potrebne za udaljenu automatizaciju ili DCOM djelotvornost.

Svaki tip datoteke se najbolje instalira na različitom mjestu.

Programske datoteke

Programske datoteke su datoteke koje vaša aplikacija mora imati kako bi se izvodila i koje su korisne samo u sklopu vaše aplikacije – na primjer, izvršna datoteka aplikacije i njezine potrebne datoteke s podacima.

Programske datoteke trebaju biti instalirane u direktorij aplikacije kojeg određuje korisnik tijekom instalacije. Programski kod u projektu Setup1.vbp pokazuje kako zapisati datoteke na to mjesto. U pravilu, alat za podešavanje koristi direktorij \Program Files kao korijensko mjesto za instaliranje aplikacija na sustave s Windowsima 95 ili kasnijim te Windowsima NT. Na primjer, projekt Setup1 savjetuje instaliranje aplikacije Project1 u direktorij \Program Files\Project1.

Oprez Kad instalirate datoteku na uređaj korisnika, ne bi trebali kopirati stariju verziju datoteke preko nove verzije. Funkcija CopyFile u modulu Setup1.bas koristi

API funkciju VerInstallFile za kopiranje datoteka na uređaj korisnika. Funkcija VerInstallFile *neće* zapisati stariju verziju preko postojeće datoteke.

Djeljive datoteke aplikacije

Djeljive datoteke aplikacije su datoteke koje može koristiti više od jedne aplikacije na sustavu. Na primjer, nekoliko različitih proizvođača može isporučiti aplikacije koje upotrebljavaju istu ActiveX kontrolu. Ako stvorite aplikaciju koja upotrebljava kontrolu, u vašem instalacijskom programu bi trebali naznačiti da je .ocx datoteka kontrole oblikovana za dijeljenje.

Dijeljene datoteke se moraju instalirati na mjesto koje omogućuje drugim aplikacijama da im pristupe. U većini slučajeva, to je direktorij \Program Files\Common Files za Windowse 95/98 i Windowse NT 4.0 ili kasnije.

Kad krajnji korisnik deinstalira vašu aplikaciju, sustav uklanja djeljivu datoteku samo ako nema drugih aplikacija koje bi mogle upotrebljavati tu datoteku.

Sastavni dijelovi udaljene automatizacije

Instalirajte sastavne dijelove poslužitelja udaljene automatizacije u direktorije \Windows\System ili \Winnt\System32. Na taj način vaša aplikacija će sigurno upotrebljavati najažurnije sastavne dijelove poslužitelja udaljene automatizacije.

Savjet Možete upotrijebiti makro $(WinSysPath) instalacije kako bi osigurali instaliranje tih datoteka u ispravan direktorij.

Ručno editiranje datoteke Setup.lst

Ako upotrebljavate čarobnjaka za pakiranje i raspoređivanje, on automatski stvara datoteku Setup.lst. Možete ručno editirati tu datoteku nakon stvaranja ako ju trebate prilagoditi.

Datoteka Setup.lst opisuje sve datoteke koje moraju biti instalirane na uređaju korisnika za vašu aplikaciju i sadrži bitne informacije o postupku podešavanja. Na primjer, datoteka Setup.lst kazuje sustavu ime svake datoteke, gdje ju treba instalirati, te kako treba biti registrirana. Postoji pet odjeljaka datoteke Setup.lst:

- Odjeljak BootStrap – sadržava popis središnjih informacija o aplikaciji, kao što je ime glavnog programa podešavanja aplikacije, privremeni direktorij za korištenje tijekom postupka instalacije, te tekst u početnom prozoru koji se pojavljuje tijekom instalacije.
- Odjeljak BootStrap Files – sadržava popis datoteka potrebnih za glavnu instalacijsku datoteku. Obično su tu uključene samo datoteke izvođenja Visual Basica.
- Odjeljak Setup1 Files – sadržava popis svih ostalih datoteka potrebnih za vašu aplikaciju, kao što su izvršne datoteke, podaci i tekst.

- Odjeljak Setup – sadržava informacije potrebne za ostale datoteke u aplikaciji.
- Odjeljak Icon Groups – sadržava informacije o grupama koje će stvoriti vaš postupak instalacije. Svaki član ovog odjeljka ima pripadajući odjeljak s ikonama koje će biti stvorene u toj grupi.

Za više informacija Pogledajte odlomak "Oblik odjeljaka BootStrap i Setup1", kasnije u ovom poglavlju, za informacije o sintaksi za te odjeljke.

Odjeljak BootStrap

Odjeljak BootStrap sadrži sve informacije koje treba datoteka setup.exe za podešavanje i pokretanje glavnog instalacijskog programa vaše aplikacije.

Napomena Zapamtite da za vašu instalaciju postoje dva programa podešavanja: setup.exe, koji je program uvodne instalacije, i setup1.exe, koji je preveden iz alata za podešavanje. Odjeljak BootStrap pruža informacije za datoteku setup.exe.

Odjeljak BootStrap sadrži sljedeće dijelove:

dio	opis
SetupTitle	Prikazani naslov dijaloškog okvira koji se pojavljuje dok datoteka setup.exe kopira datoteke na sustav korisnika.
SetupText	Tekst prikazan u dijaloškom okviru koji se pojavljuje dok datoteka setup.exe kopira datoteke na sustav korisnika.
CabFile	Ime .cab datoteke za vašu aplikaciju, ili ime prve .cab datoteke za vašu aplikaciju ako vaš paket sadrži više .cab datoteka.
Spawn	Ime aplikacije koja se pokreće kad datoteka setup.exe završi obradu. U većini primjera, to će biti datoteka setup1.exe.
TmpDir	Mjesto koje želite upotrijebiti za privremene datoteke stvorene tijekom postupka instalacije.
Uninstall	Ime aplikacije koja se upotrebljava za deinstaliranje. Općenito, to je datoteka st6unst.exe, koja se automatski pakira u sve pakete stvorene čarobnjakom.

Odjeljak BootStrap Files

Odjeljak BootStrap Files sadrži popis svih datoteka koje moraju biti učitane na korisnikov uređaj prije nego što mogu biti učitane aplikacija i datoteke ovisnosti. Te datoteke koje prethode instaliranju, ili *samopodizajuće* datoteke, uključuju središnje datoteke potrebne za izvođenje svake Visual Basic aplikacije, kao što je dinamička biblioteka izvođenja Visual Basica (Msvbvm60.dll). Program podešavanja instalira te datoteke prije instaliranja i pokretanja glavnog programa instaliranja.

Sljedeći primjer pokazuje unose u tipičnom odjeljku BootStrap Files:

```
[Bootstrap Files]
File1=@Msvbvm60.dll,$(WinSysPathSysFile),$(DLLSelfRegister),
å1/23/98 9:43:25 AM,1457936,6.0.80.23

File2=@OleAut32.dll,$(WinSysPath),$(DLLSelfRegister),
å1/21/98 11:08:26 PM,571152,2.30.4248.1

File3=@OlePro32.dll,$(WinSysPathSysFile),$(DLLSelfRegister),
å1/21/98 11:08:27 PM,152336,5.0.4248.1
```

Odjeljak Setup1 Files

Odjeljak Setup1 Files sadržava sve ostale datoteke koje su potrebne za vašu aplikaciju, kao što vaša izvršna datoteka, podaci, tekst, i ovisnosti. Program podešavanja instalira te datoteke nakon što instalira središnje datoteke izlistane u odjeljku BootStrap Files.

Sljedeći primjer pokazuje unose u tipičnom odjeljku Setup1 Files:

```
[Setup1 Files]
File1=@PunoKontrola.exe,$(AppPath),$(EXESelfRegister,
‘’1/26/98 3:43:48 PM,7168,1.0.0.0

File2=@mscomctl.ocx,$(AppPath),$(DLLSelfRegister),
‘’1/23/98 9:43:40 AM,1011472,6.0.80.23
```

Odjeljak Setup

Odjeljak Setup datoteke Setup.lst je jednostavno popis informacija koje koriste ostali dijelovi postupka instaliranja. Sljedeća tablica ispisuje informacije sadržane u odjeljku Setup.

dio	opis
Title	Ime aplikacije koje će se pojaviti na uvodnom ekranu tijekom instalacije, u programskim grupama izbornika Start, te u stavci imena aplikacije.
DefaultDir	Podrazumijevan direktorij instalacije. Korisnik može odabrati drugi direktorij tijekom postupka instalacije.
ForceUseDefDir	Ako je ostavljen prazan, korisnik se pita za direktorij instalacije. Ako je postavljen na 1, aplikacija se automatski instalira u direktorij određen dijelom “DefaultDir” datoteke Setup.lst.
AppToUninstall	Ime koje želite vidjeti za vašu aplikaciju u uslužnoj aplikaciji Add/Remove Programs kontrolnog panela.
AppExe	Ime izvršne datoteke vaše aplikacije, kao što je Mojaapli.exe.

Odjeljak IconGroups

Odjeljak IconGroups sadrži informacije o programskim grupama izbornika Start koje stvara postupak instalacije. Svaka programska grupa koja će biti stvorena prvo je ispisana u odjeljku IconGroups, zatim dodijeljena zasebnom odjeljku (Group0, Group1, Group2, itd.) koji sadržava informacije o ikonama i naslovima za tu grupu. Grupe su obrojčene redom, počevši od nule.

Sljedeći primjer pokazuje unose u tipičnom odjeljku IconGroups i povezane pododjeljke:

```
[IconGroups]
Group0=MojaTestEXE
Group1=Grupa1

[MojaTestEXE]
Icon1=moja.exe
Title1=MojaTestExe

[Grupa1]
Icon1=Čitajme.txt
Title1=ČitajMe
Icon2=moj.hlp
Title2=Pomoć
```

Oblik odjeljaka BootStrap i Setup1

Odjeljci BootStrap Files i Setup1 datoteke Setup.lst sadrže potpuni popis datoteka koje su potrebne programima podešavanja (setup.exe i setup1.exe) za instaliranje na računalo korisnika. Svaka datoteka je zasebno ispisana, u svojoj vlastitoj liniji, i mora koristiti sljedeći oblik:

File*x*=*datoteka*,*instalirano*,*staza*,*registar*,*dijeljeno*,*datum*,*veličina*[,*verzija*]

dio	značenje
File*x*	Ključna riječ koja se mora pojaviti na početku svake linije. *X* je broj niza, počinje na 1 u svakom odjeljku i raste. Ne možete preskakati vrijednosti.
datoteka	Ime datoteke kako će se pojaviti nakon instalacije na računalo korisnika. Obično je isto kao i argument Instalirano. Ako želite da ta datoteka bude izvučena iz .cab datoteke, postavite znak @ ispred imena (na primjer, @moja.exe).
instalirano	Ime datoteke kako se pojavljuje na svakom mediju distribucije.

dio	značenje
staza	Direktorij u koji datoteka treba biti instalirana. To može biti stvarna staza direktorija, makro koji ukazuje na korisnički određenu stazu, ili kombinacija te dvije opcije. Pogledajte odlomak "Makroi za argument staze u datotekama Setup.lst", kasnije u ovom poglavlju, za više informacija o dostupnim makroima.
registar	Ključ koji ukazuje kako će datoteka biti uključena u registre sustava korisnika. Pogledajte odlomak "Ključevi registara u datoteci Setup.lst", kasnije u ovom poglavlju, za više informacija.
dijeljeno	Određuje da datoteka treba biti instalirana kao dijeljena.
datum	Posljednji datum kad je datoteka mijenjana, koji će se pojaviti u Windows Exploreru. Ova informacija vam pomaže da provjerite imate li ispravne verzije datoteka na diskovima podešavanja.
veličina	Veličina datoteke koja će se ispisati u Windows Exploreru. Program podešavanja koristi ovu informaciju za proračunavanje koliko prostora na disku korisnikovog uređaja zahtijeva vaša aplikacija.
verzija	Neobavezni ugrađeni broj verzije datoteke. Zapamtite da to nije nužno isti broj kao broj verzije kojeg vidite prikazanog provjerom svojstava datoteke.

Makroi za argument staze u datotekama Setup.lst

Argument *staza* u datoteci Setup.lst predstavlja mjesto na koje datoteka treba biti instalirana. Vrijednost upotrijebljena za ovaj argument je ili stvarna staza ili makro koji ukazuje na stazu koju je odredio korisnik. Sljedeća tablica sadržava makroe koji mogu biti upotrijebljeni u instalaciji.

makro	instalira datoteku u ovaj direktorij	valjan je za ovaj odjeljak
$(WinSysPath)	\Windows\System (Windowsi 95 ili kasniji) \Winnt\System32 (Windowsi NT)	Setup1 Files i BootStrap Files
$(WinSysPathSysFile)	\Windows\System (Windowsi 95 ili kasniji) \Winnt\System32 (Windowsi NT) Datoteka se instalira kao sistemska i ne uklanja se kad se uklanja aplikacija.	samo Setup1 Files
$(WinPath)	• \Windows (Windowsi 95 ili kasniji) • \Winnt (Windowsi NT)	Setup1 Files i BootStrap Files
$(AppPath)	Direktorij aplikacije kojeg odredi korisnik, ili vrijednost *DefaultDir* određena u odjeljku Setup.	samo Setup1 Files
$(AppPath)\Samples	\Samples, unutar direktorija aplikacije.	samo Setup1 Files

makro	instalira datoteku u ovaj direktorij	valjan je za ovaj odjeljak
staza (na primjer, c:\)	Direktorij određen argumentom *staza* (nije preporučeno).	samo Setup1 Files
$(CommonFiles)	\Program Files\Common Files Često kombiniran s poddirektorijem, kao $(CommonFiles)\Moja tvrtka\Moja aplikacija.	samo Setup1 Files
$(CommonFilesSys)	$(Common Files)\System	samo Setup1 Files
$(ProgramFiles)	\Program Files	samo Setup1 Files

Ključevi registara u datoteci Setup.lst

Ključ *registar* u datoteci Setup.lst ukazuje kako datoteka treba biti registrirana na računalu korisnika. Možete naznačiti da datoteka ne treba biti registrirana, ili naznačiti jednu od nekoliko opcija registriranja za datoteke koje trebaju registraciju.

Sljedeća tablica ispisuje moguće ključeve.

ključ registra	značenje
(bez ključa)	Datoteka ne sadrži povezane ili umetnute objekte i ne treba biti registrirana.
$(DLLSelfRegister)	Datoteka je samoregistrirajuća .dll, .ocx ili svaka druga .dll datoteka s samoregistrirajućom informacijom.
$(EXESelfRegister)	Datoteka je ActiveX .exe sastavni dio stvoren u Visual Basicu, ili svaka druga .exe datoteka koja podržava prekidače /RegServer i /UnRegServer naredbene linije.
$(TLBRegister)	Datoteka je tipska biblioteka i treba biti registrirana sukladno tome.
$(Remote)	Datoteka je datoteka udaljene podrške (.vbr) i treba biti registrirana sukladno tome.
Imedatoteke.reg	Datoteka je sastavni dio distribucije koji treba biti registriran, ali se sam ne registrira. Ovaj ključ naznačuje .reg datoteku koja sadrži informacije koje trebaju biti ažurirane u sistemskim registrima. Datoteka tipa .reg mora također biti dodana u vašu datoteku Setup.lst i instalirana.

Ključ *imedatoteke*.reg nije preporučen postupak postavljanja informacija registriranja u registre. Unosi registara dodani na ovaj način ne mogu biti automatski deinstalirani aplikacijom Application Removal.

Određivanje sastavnih dijelova udaljenog poslužitelja u datoteci Setup.lst

Ako u vašoj aplikaciji upotrebljavate sastavne dijelove poslužitelja, morate u datoteku Setup.lst postaviti unos označavajući datoteku kao sastavni dio udaljenog poslužitelja i određujući informaciju povezivanja. Tu informaciju upotrebljava uslužna aplikacija Client Registration za registriranje poslužitelja udaljene automatizacije.

Sljedeći primjer pokazuje kako određujete sastavne dijelove udaljenog poslužitelja:

```
File1=@PunoKontrola.vbr,$(WinSysPath),$(Remote), ,,1/26/98 3:43:48 PM,1024,1.0.0.0
File2=@Server2.vbr,$(WinSysPath),$(Remote), ,,1/23/98 9:43:40 AM,1024,6.0.80.23
Remote1=”Schweizer”,”ncacn_ip_tcp”,1,RA
Remote2=,,1,DCOM
```

Unos Remote*x* sastoji se od adrese poslužitelja, protokola mreže, i informacije za provjeru valjanosti, razdvojenih zarezima. Morate također odrediti hoće li sastavni dio biti upotrijebljen u okruženju udaljene automatizacije ili DCOM okruženju, korištenjem argumenata *RA* ili *DCOM*.

Ručno stvaranje medija distribucije

Postoji nekoliko načina na koje možete distribuirati vašu aplikaciju – na disketama ili kompaktnim diskovima, putem mreže ili na Web stranici. Kao dio postupka raspoređivanja, morate kopirati vaše pakirane datoteke na medij distribucije kojeg namjeravate koristiti.

> Napomena Ako pakirate i raspoređujete vaše datoteke korištenjem čarobnjaka za pakiranje i raspoređivanje, ovi koraci nisu potrebni jer će čarobnjak umjesto vas automatski stvoriti medij distribucije za diskove, mjesto na mreži, ili Internet sastavni dio za preuzimanje.

Možete upotrijebiti bilo koji od sljedećih postupaka kad ručno stvarate medij distribucije:

- Ako namjeravate distribuirati na disketama, morate kopirati datoteke na diskete po određenom redoslijedu. Morate postaviti datoteke setup.exe i Setup.lst na prvu disketu, te sve datoteke iz odjeljka BootStrap Files datoteke Setup.lst vašeg projekta, pa zatim ostale .cab datoteke na preostale diskete.
- Za distribuciju na dio mreže, jednostavno kopirajte vaše zapakirane datoteke na odgovarajuće mjesto, koristeći Windows Explorer, naredbeni upit, ili nekim drugim postupkom.
- Za distribuciju na CD-ima, jednostavno kopirajte svoje datoteke na CD-e.
- Za distribuciju Internetom, kopirajte svoj Internet paket na odgovarajuću Web stranicu, koristeći čarobnjaka za objavljivanje na Webu (Web Publishing Wizard), dostupnog u *ActiveX SDK*-u.

Za više informacija Za više informacija o stvaranju mjesta za objavljivanje na Webu korištenjem čarobnjaka za objavljivanje na Webu, pogledajte "Alati i tehnologije Interneta" u dijelu *Internet Client SDK*, stalne pomoći.

Stvaranje medija distribucije

Svoj medij distribucije možete stvoriti nakon što odredite datoteke koje ćete uključiti u svoj program podešavanja, stvorite datoteku Setup.lst, sažmete sve potrebne datoteke i odlučite kako ćete postaviti datoteke na medij:

- Ako upotrebljavate diskete za svoju distribuciju, trebate kopirati datoteke na jednu ili više disketa.
- Ako upotrebljavate drugi mehanizam, kao CD-e ili mjesto na mreži, trebate kopirati svoje datoteke na odgovarajuće mjesto ili područje zbivanja za proizvodnju CD-a.
- Za isporuku putem Interneta ili intraneta, trebate objaviti svoje datoteke na odgovarajućoj Web stranici, nakon što ih oblikujete za preuzimanje sastavnih dijelova s Interneta.

Stvaranje diskova distribucije

Dobra je praks označiti svoje diskete distribucije brojem i imenom diskete te dodati upute za podešavanje svoje aplikacije. Ponovite takvu poruku o instalaciji na svakoj od disketa distribucije. Nakon što su svoje diskete označene, spremni ste za kopiranje svojih datoteka na diskete distribucije.

Kako stvoriti diskete distribucije

1. Kopirajte sljedeće datoteke potrebne za instalaciju na prvu disketu:
   - Setup.exe
   - Setup.lst
2. Kopirajte preostale samopodizajuće .cab datoteke na ostatak prve diskete te na sve sljedeće diskete koje su potrebne.
3. Nakon kopiranja svih samopodizajućih .cab datoteka, kopirajte ostatak svojih .cab datoteka na potreban broj disketa.

Ručno raspoređivanje na Web stranicu

Ako želite ručno rasporediti svoju aplikaciju ili sastavni dio na stranicu Weba, možete upotrijebiti čarobnjaka za objavljivanje na Webu, za raspoređivanje svojih pakiranih datoteka na odgovarajuće mjesto. Čarobnjak za objavljivanje na Webu je dostupan na Microsoftovoj Web stranici s adresom www.microsoft.com, a također se instalira s Internet Explorerom 4.*x*.

Za više informacija Pogledajte odlomak "Alati i tehnologije Interneta" u dijelu *Internet Client SDK* za više informacija o čarobnjaku za objavljivanje na Webu. Pogledajte odlomak "Ručno stvaranje medija distribucije" ranije u ovom poglavlju za informacije o odlučivanju kako postaviti svoje datoteke na diskete.

Korištenje čarobnjaka za pakiranje i raspoređivanje s alatom za podešavanje

Osim korištenja projekta Setup Toolkit za stvaranje svojeg vlastitog korisničkog projekta podešavanja, možete upotrijebiti projekt Setup Toolkit zajedno s čarobnjakom za pakiranje i raspoređivanje. U tom slučaju, projekt Setup Toolkit upotrijebite za prilagođavanje ekrana ili ostalih dijelova slijeda instalacije, pa zatim upotrijebite čarobnjaka kako bi stvorili i rasporedili paket za aplikaciju. Čarobnjak prevodi projekt Setup Toolkit i stvara program setup1.exe za aplikaciju.

Na primjer, možete upotrijebiti alat za podešavanje i čarobnjaka za pakiranje i raspoređivanje zajedno kako bi dodali dijaloške okvire programu podešavanja, tražeći od korisnika da odredi želi li instalirati dodatne osobine u svojoj aplikaciji. Na primjer, možete imati datoteku stalne pomoći koju neki korisnici radije neće instalirati. Možete dodati željen broj opcija instalacije.

Kako dodati opciju instalacije svojem programu podešavanja

1. U projektu Setup1.vbp, editirajte programski kod događaja Form_Load u formi Setup1.frm. Kako bi dodali djelotvornost, dodajte programski kod nakon bloka koda koji poziva funkciju ShowBeginForm (Sub ShowBeginForm).

   Sljedeći programski kod pokazuje primjer kako ćete dodati dijaloški okvir koji pita želi li korisnik instalirati dodatne datoteke:

   ```
   Dim UčitajPomoć As Integer
   UčitajPomoć = MsgBox(“Želite li instalirati Pomoć? “, vbYesNo)
   If UčitajPomoć = vbYes Then
     CalcDiskSpace “Pomoć”
   End If
   ‘ Blok koda koji sadrži cIcons = CountIcons(strINI_FILES)
   If UčitajPomoć = vbYes Then
     cIcons = CountIcons(“Pomoć”)
   End If
   ‘ Blok koda koji sadrži CopySection strINI_FILES)
   If UčitajPomoć = vbYes Then
     CopySection “Pomoć”
   End If
   ‘ Blok koda koji sadrži CreateIcons, strINI_FILES, strImeGrupe
   ```

2. Zatvorite formu Setup1.frm, snimite formu i projekt Setup Toolkit, te ga prevedite kako bi stvorili datoteku Setup1.exe.

3. Pokrenite **čarobnjaka za pakiranje i raspoređivanje**, i odaberite **Package** s glavnog ekrana.

4. Nastavite kroz čarobnjaka, odabirući prikladne izbore. Za gore prikazan primjer, osigurat ćete da sve neobavezne datoteke koje bi korisnik mogao odabrati za instaliranje u svojem korisničkom dijaloškom okviru budu prikazane u ekranu Add and Remove.

5. Kad ste gotovi s čarobnjakom za pakiranje i raspoređivanje, stvorite medij distribucije.
6. Napravite sve potrebne promjene u datoteci Setup.lst. U prethodnom primjeru, trebate dodati novi odjeljak s odjeljkom kojeg ste koristili u odjeljku CopySection svojeg koda. U ovom slučaju, svoj odjeljak će izgledati slično ovome:

```
[Pomoć]
File1=Mojaap.HL1,Mojaap.HLP,$(AppPath),,10/12/96,2946967,0.0.0
```

7. Rasporedite i ispitajte svoj paket.

Kad korisnik pokrene program podešavanja za primjer prikazan u ovom potprogramu, program podešavanja kopira sve samopodizajuće datoteke na uređaj korisnika te zatim pita korisnika da naznači želi li instalirati datoteke pomoći. Ako korisnik odabere Yes, naredba CalcDiskSpace ustanovljava postoji li dovoljno mjesta na disku korisnikovog uređaja za datoteke pomoći. Program zatim instalira sve datoteke ispisane u odjeljku Setup1 Files datoteke Setup.lst.

Kao sljedeće, program ponovno ispituje zastavicu LoadHelp. Ako korisnik odabere instaliranje datoteka pomoći, program Setup1.exe zatim izvodi naredbu CopySection za datoteke pomoći, i instalira datoteke ispisane u odjeljku [Pomoć] datoteke Setup.lst.

Za više informacija Pogledajte odlomak “Čarobnjak za pakiranje i raspoređivanje”, ranije u ovom poglavlju, za više informacija o osobinama čarobnjaka.

Ispitivanje svojeg programa podešavanja

Nakon što ste završili postupak pakiranja i proizveli medij distribucije za svoju aplikaciju, morate ispitati svoj program podešavanja. Obavezno ispitajte svoj program podešavanja na računalu koje nema instaliran Visual Basic niti nijednu od ActiveX kontrola koje zahtijeva svoja aplikacija. svoje podešavanje bi također trebali ispitati na svim prikladnim operativnim sustavima.

Kako ispitati svoj program podešavanja temeljen na disketama ili CD-ima

1. Ubacite prvu disketu ili CD u odgovarajući pogon.
2. U Windowsima 95 i Windowsima NT 4.0 ili kasnijim, u izborniku **Start**, odaberite naredbu **Run**, i upišite:

   ***pogon*:\setup**

   - ili -

   Dva puta kliknite na datoteku **Setup.exe** na pogonu.
3. Kad se instalacija završi, pokrenite instaliranu aplikaciju kako bi bili sigurni da se ponaša prema očekivanjima.

Kako ispitati vaš program podešavanja temeljen na mrežnom pogonu

1. S drugog računala na istoj mreži na kojoj je i poslužitelj distribucije, povežite se s poslužiteljem i direktorijem koji sadrži vaše datoteke distribucije.
2. U direktoriju distribucije, dva puta kliknite na datoteku **Setup.exe**.
3. Kad se instalacija završi, pokrenite instaliranu aplikaciju kako bi bili sigurni da se ponaša prema očekivanjima.

Kako ispitati svoj program podešavanja temeljen na Webu

1. Rasporedite svoj paket na poslužitelja Weba.
2. Pristupite stranici Weba s koje se mogu pozvati .cab datoteke svoje aplikacije. Automatski će započeti preuzimanje, i bit ćete upitani želite li nastaviti.
3. Kad se instalacija završi, pokrenite instaliranu aplikaciju kako bi bili sigurni da se ponaša prema očekivanjima.

Omogućavanje korisniku da ukloni vašu aplikaciju

Kad korisnik instalira vašu aplikaciju, program podešavanja kopira uslužni dodatak St6unst.exe za uklanjanje aplikacije u direktorije \Windows ili \Winnt. Svaki put kad upotrijebite Visual Basic program podešavanja za instaliranje aplikacije, stvara se evidencijska datoteka za uklanjanje aplikacije (St6unst.log) u direktoriju u kojem je aplikacija instalirana. Datoteka tipa .log sadrži unose koji ukazuju na:

- Direktorije koji su stvoreni tijekom instalacije.
- Instalirane datoteke i njihove položaje. Ovaj popis sadrži sve datoteke u programu podešavanja, čak i ako neke datoteke nisu instalirane na korisnikovo računalo jer je već postojala novija verzija iste datoteke. Evidencijska datoteka naznačuje je li datoteka djeljiva te ako jest, je li zamijenila postojeću datoteku.
- Stvorene ili mijenjane unose registara.
- Prečice i unose izbornika Start stvorene s Windowsima 95 i Windowsima NT 4.0 ili kasnijim.
- Samoregistrirajuće .dll, .exe ili .ocx datoteke.

U Windowsima 95 i Windowsima NT 4.0 ili kasnijim, program podešavanja dodaje uslužni dodatak za uklanjanje aplikacije popisu registriranih aplikacija prikazanih u odjeljku Add/Remove Programs kontrolnog panela. Krajnji korisnici koriste odjeljak Add/Remove Programs za deinstaliranje aplikacije.

> Oprez Važno je da obavezno ispravno podesite opcije za sve datoteke koje trebaju biti dijeljene, dodavanjem datoteka na ekran Shared Files u čarobnjaku za pakiranje i raspoređivanje, ili naznačivanjem mjesta za datoteku kao direktorija s djeljivim

datotekama. Ako slučajno instalirate datoteku koja treba biti dijeljena bez ispravnih postavki, korisnici će biti u mogućnosti ukloniti ih kad deinstaliraju svoju aplikaciju, što može uzrokovati probleme drugim aplikacijama na njihovom sustavu.

U slučaju neuspješne ili poništene instalacije, uslužni dodatak za uklanjanje aplikacije automatski uklanja sve direktorije, datoteke i unose registara koje je stvorio program podešavanja tijekom pokušaja instalacije.

s Windowsima 95 ili kasnijim i Windowsima NT, dijeljene datoteke imaju brojač pokazivača u registrima. Na primjer, dijeljena datoteka koju koriste tri aplikacije imat će brojača pokazivača od tri. Kad uklonite aplikaciju koja koristi dijeljenu datoteku, brojač pokazivača za dijeljenu datoteku se smanjuje za jedan. Kad brojač za datoteku dosegne nulu, korisnik će biti upitan za konačno uklanjanje te stavke.

Situacije u kojima uslužni dodatak za uklanjanje aplikacije može zatajiti

Kako bi uslužni dodatak za uklanjanje aplikacije ispravno deinstalirao svoju aplikaciju, evidencijska datoteka stvorena programom podešavanja mora biti točna i nepromijenjena od vremena instalacije.

Uslužna aplikacija Application Removal može zatajiti ili raditi neispravno ako postoji bilo koja od sljedećih situacija:

- Krajnji korisnik ručno je kopirao dijeljene datoteke. U tom slučaju, brojač pokazivača nije ažuriran u registrima, pa sustav ne može točno presuditi kad ukloniti datoteku.
- Krajnji korisnik obrisao je instalirane datoteke ili direktorij aplikacije umjesto korištenja uslužnog dodatka za uklanjanje aplikacije. To će obrisati evidencijsku datoteku, tako da je uklanjanje aplikacije nemoguće. Osim toga, to također onemogućuje uklanjanje unos sistemskih registara za programske datoteke, dinamičke biblioteke, ili .ocx datoteke u direktoriju aplikacije jer te datoteke moraju biti pokrenute kako bi mogle biti uklonjene.
- Program podešavanja za aplikaciju koji nije sukladan s Windowsima 95/98 instalirao je iste djeljive datoteke kao i svoja aplikacija.
- Dijeljena datoteka je instalirana u direktorij različit od onog u kojem već postoji na tvrdom disku.
- Krajnji korisnik je instalirao istu Visual Basic aplikaciju u dva različita direktorija. Osim što prva instalacija više neće raditi, sukobljavat će se i scenariji uklanjanja aplikacije. Krajnji korisnik bi uvijek trebao ukloniti prvu instalaciju prije instaliranja aplikacije u drugačiji direktorij.
- Krajnji korisnik je obrisao evidencijsku datoteku podešavanja aplikacije (St6unst.log). Bez evidencijske datoteke podešavanja aplikacije, uslužni dodatak za uklanjanje aplikacije nema informacija o instalaciji i zatajit će.

Neki od ovih primjera mogli bi pogoršati vezu registara instaliranih datoteka, te uzrokovati da uslužna aplikacija Application Removal prerano dosegne nulu u brojaču pokazivača za određenu datoteku, te da zatim pita smije li ta datoteka biti obrisana. Ako je datoteka prerano obrisana, može uzrokovati prestanak djelovanja ili neispravno djelovanje drugih aplikacija zbog nedostajućih datoteka ovisnosti, sastavnih dijelova i tako dalje.

Raspoređivanje lokaliziranih ActiveX kontrola

U većini slučajeva, distribuiranje lokaliziranih ActiveX kontrola se ne razlikuje od distribuiranja originalne verzije. Međutim, može se pojaviti problem ako se nova verzija kontrole izda i distribuira na Webu s drugom aplikacijom.

ActiveX kontrole isporučene s Visual Basicom su automatski prevedene, ili lokalizirane, u ispravan jezik za verziju Visual Basica koju imate. Na primjer, u japanskoj verziji Visual Basica, kontrole su dostupne na japanskom i spremne su za distribuciju na tom jeziku. Kad distribuirate te kontrole, čarobnjak za pakiranje i raspoređivanje mora ustanoviti potrebne datoteke koje će s njima zapakirati.

> **Napomena** Ova tema vrijedi samo za ActiveX kontrole distribuirane s Visual Basicom. Ona ne vrijedi za korisničke kontrole stvorene s Visual Basicom osim ako one ne sadrže ActiveX kontrolu Visual Basica.

Lokalizirane ActiveX kontrole Visual Basica imaju dva dijela:

- Datoteku tipa .ocx za kontrolu, koja djeluje kao blok koda kontrole. Ista .ocx datoteka se koristi za svaku jezičnu verziju Visual Basica.
- Prateću DLL datoteku, koja sadrži lokalizirane tekstove za kontrolu. Ova datoteka djeluje kao blok podataka kontrole. Prateća DLL datoteka različita je među verzijama, ovisno o jezičnoj verziji Visual Basic koju ste nabavili.

Čarobnjak za pakiranje i raspoređivanje automatski uključuje ispravnu prateću DLL datoteku kad pakirate aplikaciju koja sadrži lokalizirane ActiveX kontrole Visual Basica. Kad krajnji korisnik preuzme aplikaciju, na njegovo računalo se instalira ispravna prateća DLL datoteka.

Ako se Internetom distribuira nova verzija kontrole, datoteka tipa .ocx i prateća DLL datoteka neće biti usklađene osim ako se nova, lokalizirana prateća DLL datoteka također ne preuzme s drugom aplikacijom. Kad se to dogodi, krajnji korisnik može iznenada otkriti da su tekstovi u novoj kontroli neispravni ili su na engleskom, budući da .ocx datoteka koristi tekstove na engleskom jeziku kao podrazumijevane (koji su uvijek dostupni u .ocx datoteci) ako nije pronađena sukladna prateća DLL datoteka. Kako bi umanjili taj problem, trebali bi uputiti krajnje korisnike da preuzmu posljednje verzije pratećih DLL datoteka s stranice www.microsoft.com/vstudio/.

Za više informacija Pogledajte odlomak “Oblikovanje međunarodnog softvera” u 16. poglavlju “Međunarodna izdanja”, za objašnjenje blokova koda i blokova podataka.

DODATAK A

Specifikacije, ograničenja i vrste datoteka Visual Basica

Ovaj dodatak opisuje sistemske zahtjeve, ograničenja projekata Visual Basica, tipove datoteka koje mogu biti uključene u vaš projekt Visual Basica, te opise datoteka forme (.frm) i projekta (.vbp).

> Napomena Iako je većina ovih ograničenja opisana u granicama određenog broja, imajte na umu da ostali uvjeti (kao što su raspoloživa memorija i sistemski izvori) mogu nametnuti ograničenje prije nego što je dostignuta određena granica.

Sadržaj

Sistemski zahtjevi za aplikacije Visual Basica

Za aplikacije Visual Basica je potreban sljedeći hardver i softver:

- Pentium 90MHz ili jači mikroprocesor.
- VGA 640x480 ili ekran veće razlučivosti kojeg podržavaju Microsoft Windowsi.
- 24MB RAM memorije za Windowse 95, 32MB za Windowse NT.
- Microsoft Windows NT 3.51 ili kasniji, ili Microsoft Windows 95 ili kasniji.
- Microsoft Internet Explorer verzija 4.01 ili kasnija (verzija 4.01 Service Pack 1 ili kasniji za programere DHTML aplikacija, i 4.x za krajnje korisnike tih aplikacija).

- Zahtjevi prostora na disku:
  - Verzija Standard: tipična instalacija 48MB, puna instalacija 80MB.
  - Verzija Professional: tipična instalacija 48MB, puna instalacija 80MB.
  - Verzija Enterprise: tipična instalacija 128MB, puna instalacija 147MB.
  - Dodatni sastavni dijelovi (ako su potrebni): MSDN (za dokumentaciju): 67MB, Internet Explorer 4.x: otprilike 66MB.
- CD-ROM (nije potrebna MS-DOS podrška).

Ograničenja projekta

Pojedini projekt može sadržavati do 32 000 "identifikatora" (svih nerezerviranih ključnih riječi), koji uključuju, ali nisu ograničeni na, forme, kontrole, module, varijable, konstante, potprograme, funkcije i objekte. Zapamtite da je stvaran broj identifikatora ograničen raspoloživom memorijom.

Imena varijabli u Visual Basicu ne mogu biti duža od 255 znakova, a imena formi, kontrola, modula i klasa ne mogu biti duža od 40 znakova. Visual Basic ne postavlja ograničenja na stvaran broj različitih objekata u projektu.

Ograničenja kontrola

Svaka negrafička kontrola (sve kontrole osim lika, linije, slike i natpisa) upotrebljava prozor. Svaki prozor koristi sistemske izvore, što ograničuje ukupni broj prozora koji u isto vrijeme mogu postojati. Točno ograničenje ovisi o raspoloživim sistemskim izvorima i tipu upotrijebljenih kontrola.

Kako bi smanjili potrošnju sistemskih izvora, upotrebljavajte kontrole lika, linije, natpisa i slike umjesto kontrola okvira za sliku kod stvaranja ili prikazivanja grafike.

Ukupan broj kontrola

Najveći broj kontrola koji je dopušten na pojedinoj formi ovisi o tipu upotrijebljenih kontrola i raspoloživim sistemskim izvorima. Međutim, postoji nepromjenjivo ograničenje od 254 imena kontrole po formi. Matrica kontrola broji se samo jednom u ovom ograničenju jer sve kontrole u matrici dijele isto ime kontrole.

Ograničenje indeksa matrice kontrola je od 0 do 32767 u svim verzijama.

Ako postavite kontrole jednu iznad druge, kao što je korištenje nekoliko kontrola okvira unutar drugih kontrola, Visual Basic će općenito prihvatiti najviše 25 razina ugniježđenih kontrola.

Ograničenja određenih kontrola

Sljedeća tabela ispisuje ograničenja svojstava koja vrijede za određene kontrole u Visual Basicu.

svojstvo	primjena	ograničenje
List i ListCount	kontrole okvira s popisom i kombiniranog okvira	Najveći broj stavki je 32K; ograničenje veličine svake stavke je 1K (1024 bajta).
Text	kontrola okvira s tekstom	Ograničeno na 64K.
Caption	kontrola natpisa	Ograničeno na 1024 bajta.
	kontrole naredbenog gumba, kontrolne kućice, okvira i gumba izbora	Ograničeno na 255 znakova. Svaki naslov dulji od te granice se odrezuje. Naslovi svojstava korisničkih kontrola su ograničeni na 32K.
	kontrola izbornika	Ograničeno na 235 znakova.
Tag	sve kontrole	Ograničeno raspoloživom memorijom.
Name	sve kontrole	Ograničeno na 40 znakova.

Napomena U Visual Basicu imena svojstava kontrole su ograničena na 30 znakova.

Ograničenja koda

Količina koda koji može biti učitan u formu, klasu ili standardni modul je ograničena na 65 534 linije. Pojedina linija koda može se sastojati od najviše 1023 bajta. Najviše 256 praznih mjesta može prethoditi stvarnom tekstu u pojedinoj liniji, a u pojedinu logičku liniju smije biti uključeno najviše dvadeset pet oznaka nastavka linije (_).

Potprogrami, tipovi i varijable

Nema ograničenja broja potprograma u modulu. Svaki potprogram može sadržavati do 64K koda. Ako potprogram prekorači to ograničenje, Visual Basic stvara pogrešku tijekom prevođenja. Ako otkrijete tu pogrešku, možete ju izbjeći rastavljanjem osobito velikih potprograma u nekoliko manjih potprograma, ili prebacivanjem određivanja na razini modula u drugi modul.

Visual Basic upotrebljava tabele za spremanje imena identifikatora (varijabli, potprograma, konstanti i tako dalje) u vašem kodu. Svaka tabela je ograničena na 64K.

Tabela određivanja DLL-a

Svaki modul forme i koda upotrebljava tabelu koja sadrži strukturu opisujući točku unosa DLL-a. Svaka struktura upotrebljava otprilike 40 bajtova, s ukupnom veličinom ograničenom na 64K, rezultat čega je ugrubo 1500 dopuštenih određivanja po modulu.

Tabela imena u projektu

Cijela aplikacija upotrebljava jednu tabelu koja sadrži sva imena. Tu su uključena:

- Imena konstanti
- Imena varijabli
- Korisnički određena imena i imena određivanja tipova
- Imena modula
- Imena određivanja DLL potprograma

Tabela imena u projektu nije ograničena ukupnom veličinom, ali je ograničena na ukupno 32K jedinstvenih unosa osjetljivih na vrstu slova. Ako je dostignuto to ograničenje, ponovno upotrijebite privatne identifikatore u različitim modulima kako bi ograničili broj jedinstvenih unosa na 32K.

Tabela uvoza

Svako upućivanje na identifikatora u drugom modulu stvara unos u tabeli uvoza. Svaki takav unos je velik najmanje 24 bajta i ograničen je na 64K, rezultat čega je ugrubo 2000 upućivanja po modulu.

Tabela modula

Ova tabela prihvaća do 125 bajtova po modulu, s ukupnim ograničenjem od 64K, rezultat čega je otprilike 400 modula po objektu.

Ograničenja podataka

Sljedeća ograničenja primjenjuju se za varijable u jeziku Visual Basica.

Podaci forme, standardnog modula i modula klase

Dio podataka (znači, podaci određeni u odjeljku Declarations) VBA modula svake forme ili modula u Visual Basicu može biti velik do 64K. Taj dio podataka sadrži sljedeće podatke:

- Lokalne varijable određene ključnom riječi Static.
- Varijable na razini modula osim matrica i stringova promjenjive duljine.
- 4 bajta za svaku matricu na razini modula i string promjenjive duljine.

Potprogrami, tipovi i varijable

Ako potprogram ili modul prekorači ograničenje koda od 64K, Visual Basic stvara pogrešku tijekom prevođenja.

Ako odredite potprogram koji ima više od 64K lokalno određenih varijabli, dobit ćete pogrešku "Previše lokalnih nestatičkih varijabli" (Too many local nonstatic variables).

Ako odredite modul koji ima više od 64K varijabli određenih na razini modula, ili ako odredite korisnički određen tip veći od 64K, dobit ćete pogrešku "Nepromjenjivi ili statični podaci ne mogu biti veći od 64K" (Fixed or static data can't be larger than 64K).

Ako otkrijete tu pogrešku, možete ju izbjeći rastavljanjem izuzetno velikih potprograma u nekoliko manjih potprograma, ili prebacivanjem određivanja na razini modula u drugi modul.

Matrica određena kao varijabla ne pridonosi ukupnoj veličini matrice; u ograničenje od 64K ubraja se samo opisnik matrice. Prema tome je prihvatljivo, na primjer, imati određivanje kao što je `Dim x(1000000) As Byte` na razini potprograma ili modula. Problemi sa nedostatkom memorije se pojavljuju, međutim, ako odredite veliku matricu nepromjenjive veličine u zapisu, i zatim odredite primjere tih zapisa kao varijable.

Korisnički određeni tipovi

Ni jedna varijabla korisnički određenog tipa ne može prekoračiti ograničenje od 64K, iako zbroj stringova promjenjive duljine u korisnički određenom tipu može prekoračiti 64K (stringovi promjenjive duljine zauzimaju samo 4 bajta svaki u korisnički određenom tipu; stvarni sadržaji stringa su spremljeni odvojeno). Korisnički određeni tipovi mogu biti određeni u granicama ostalih korisnički određenih tipova, ali ukupna veličina tipova ne može prekoračiti 64K.

Prostor stoga

Argumenti i lokalne varijable u potprogramima zauzimaju prostor stoga tijekom izvođenja. Varijable na razini modula i statičke varijable ne zauzimaju prostor stoga jer su smještene u dijelu podataka za forme ili module. Svi DLL potprogrami koje pozivate koriste taj stog dok se izvode.

Sam Visual Basic koristi dio stoga za vlastite svrhe, kao što je spremanje prijelaznih vrijednosti kod proračunavanja izraza.

Ukupni raspoloživi prostor stoga za Visual Basic je jedan megabajt (1MB) po niti. Unatoč tome, stog može rasti i iznad toga, ako postoji susjedna slobodna memorija.

Za više informacija Za savjete o sačuvanju prostora stoga pogledajte 15. poglavlje, "Oblikovanje u korist izvođenja i sukladnosti".

Ograničenja sistemskih izvora

Neka ograničenja Visual Basica, i aplikacija koje stvorite s njim, nametnuta su od Microsoft Windowsa. Ta ograničenja se mogu promijeniti kad instalirate drugačiju verziju Microsoft Windowsa.

Izvori Windowsa

Svaki otvoren prozor koristi neke sistemske izvore (područja podataka koja koriste Microsoft Windowsi). Ako ponestane sistemskih izvora, pojavit će se pogreška tijeka izvođenja "Opadaju dostupni izvori tijekom rada Windowsa" (Windows is running low on available resoruces). Postotak preostalih sistemskih izvora možete provjeriti odabirom stavke About iz izbornika Help aplikacija Program Manager ili File Manager u Windowsima NT 3.51, ili, u Windowsima 95/98 i Windowsima 4.0, odabirom stavke About u izborniku Help aplikacije Windows Explorer. Aplikacije mogu također pozvati naredbu Windows API-ja GetFreeSystemResources za vraćanje sistemskih izvora, zatvaranje prozora (kao što su otvorene forme i kodni prozori, kao i prozori u drugim aplikacijama), te izlaz iz aplikacija koje se izvode.

Oblici datoteka projekta

Microsoft Visual Basic iskorištava i stvara niz datoteka tijekom izrade i izvođenja. Koje će datoteke zahtijevati vaš projekt ili aplikacija ovisi o njihovom području i djelotvornosti.

Sufiksi projektnih datoteka

Visual Basic proizvodi niz datoteka kad stvarate i prevodite projekt. Te datoteke mogu biti razvrstane na ovaj način: datoteke tijeka izrade, raznovrsne datoteke razvoja i datoteke tijeka izvođenja.

Datoteke tijeka izrade su blokovi izgradnje vašeg projekta: temeljni moduli (.bas) i moduli forme (.frm), na primjer.

Raznovrsne datoteke su proizvedene raznim postupcima i funkcijama razvojnog okruženja Visual Basica: datoteke ovisnosti čarobnjaka za pakiranje i raspoređivanje (.dep), na primjer.

Datoteke tijeka izrade i raznovrsne datoteke

Sljedeća tabela ispisuje sve datoteke tijeka izrade i druge raznovrsne datoteke koje mogu biti proizvedene kad razvijate aplikaciju:

sufiks	opis
.bas	Temeljni modul
.cls	Modul klase
.ctl	Datoteka korisničke kontrole
.ctx	Binarna datoteka korisničke kontrole
.dca	Spremište aplikacije Active Designer
.ddf	Datoteka CAB informacija čarobnjaka za pakiranje i raspoređivanje

sufiks	opis
.dep	Datoteka ovisnosti čarobnjaka za pakiranje i raspoređivanje
.dob	Datoteka oblika ActiveX dokumenta
.dox	Binarna datoteka oblika ActiveX dokumenta
.dsr	Datoteka aplikacije Active Designer
.dsx	Binarna datoteka aplikacije Active Designer
.dws	Datoteka skripata čarobnjaka za raspoređivanje
.frm	Datoteka forme
.frx	Binarna datoteka forme
.log	Evidencijska datoteka za učitavanje pogreški
.oca	Datoteka spremnika tipske biblioteke kontrole
.pag	Datoteka stranice svojstava
.pgx	Binarna datoteka stranice svojstava
.res	Datoteka izvora
.tlb	Datoteka tipske biblioteke udaljene automatizacije
.vbg	Datoteka projektne grupe Visual Basica
.vbl	Datoteka odobrenja za kontrolu
.vbp	Datoteka projekta Visual Basica
.vbr	Datoteka registracije udaljene automatizacije
.vbw	Datoteka radnog prostora projekta Visual Basica
.vbz	Datoteka pokretanja čarobnjaka
.wct	HTML predložak za klasu Web

Datoteke tijeka izvođenja

Kad prevedete vašu aplikaciju, sve potrebne datoteke tijeka izrade su uključene u izvršne datoteke izvođenja. Datoteke tijeka izvođenja su ispisane u sljedećoj tabeli:

sufiks	opis
.dll	ActiveX sastavni dio u-procesu
.exe	Izvršna datoteka ili ActiveX sastavni dio
.ocx	ActiveX kontrola
.vbd	Datoteka stanja ActiveX dokumenta
.wct	HTML predložak za klasu Web

Građa forme

Iako je većina datoteka tipičnog projekta Visual Basica u binarnom obliku i čitljiva je samo određenim postupcima i funkcijama Visual Basica ili vaše aplikacije, datoteke forme (.frm) i projekta (.vbp) se snimaju kao ASCII tekst. One su čitljive u pregledniku teksta (na primjer, aplikaciji Notepad).

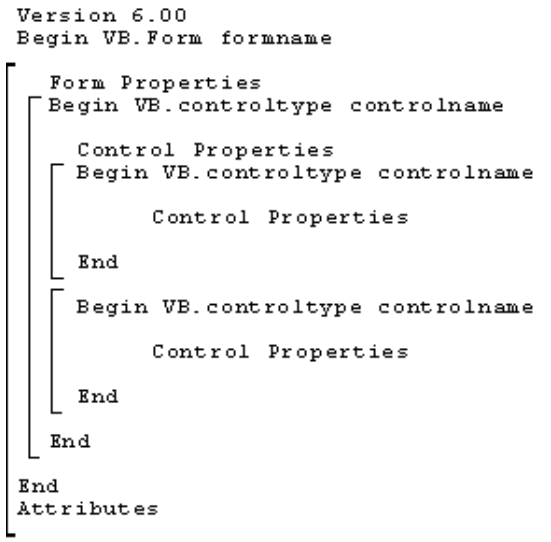
Sljedeći odlomci opisuju datoteke izrade i datoteke izvođenja u tipičnom projektu Visual Basica, te oblik datoteka forme (.frm) i projekta (.vbp).

Datoteke forme (.frm) Visual Basica se stvaraju i snimaju u ASCII obliku. Struktura forme sastoji se od:

- broja verzije oblika datoteke.
- bloka teksta koji sadržava opis forme.
- niza atributa forme.
- koda Basica za formu.

Opis forme sadrži postavke svojstava forme. Blokovi teksta koji određuju svojstva kontrola na formi su ugniježđeni unutar forme. Kontrole sadržane unutar drugih kontrola imaju svoja svojstva ugniježđena unutar teksta spremnika. Slika A.1 prikazuje strukturu opisa forme.

Slika A.1 Struktura opisa forme



Ispis opisa forme

Ako želite ispisati opis forme, možete to napraviti bez potrebe da snimite formu.

Kako ispisati opis forme

1. U izborniku **File** odaberite naredbu **Print**.
2. Potvrdite kontrolnu kućicu **Form As Text** i odaberite **OK** za tiskanje opisa forme.

Broj verzije

Broj verzije formi stvorenih Visual Basicom je 6.00. Ako je u formi izostavljen broj verzije, stvara se pogreška. Kad učitate Visual Basic aplikaciju koja ima broj verzije manji od 6.00, pojaviti će se upozoravajući dijaloški okvir koji vas obavješćuje da će datoteka biti snimljena u novom obliku.

Opis forme

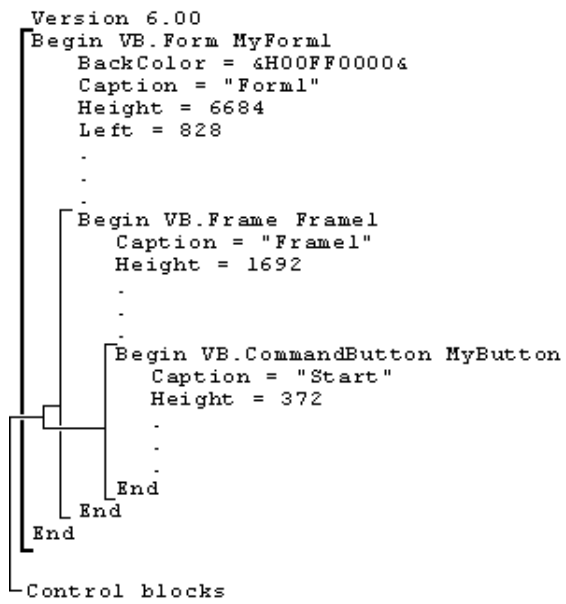
Opis forme počinje naredbom Begin i završava naredbom End. Sintaksa naredbe Begin je:

Begin VB.{**Form**|**MDIForm**} *imeforme*

Naredba End određuje gdje završava opis forme i započinje niz atributa forme. Bez naredbe End, Visual Basic pokušava čitati atribute kao da su to opisi kontrola i svojstava forme, što prouzrokuje pogreške.

Između naredbi Begin Form i End nalaze se svojstva same forme, iza kojih slijede opisi svake kontrole na formi. Slika A.2 pokazuje ugniježđenu strukturu opisa forme u detaljnijem prikazu.

Slika A.2 Ugniježđena struktura opisa forme



Blokovi kontrola

Blok kontrole se sastoji od teksta u opisu forme koji određuje svojstva pojedine kontrole. Slično samom opisu forme, blokovi kontrola počinju naredbom Begin i završavaju naredbom End. Sintaksa naredbe Begin bloka kontrole je sljedeća:

Begin *klasakontrole.tipkontrole imekontrole*

Svojstva za kontrolu se pojavljuju između naredbe Begin i naredbe End.

Redoslijed blokova kontrola

Redoslijed blokova kontrola određen je z-redoslijedom kontrola. *Z-redoslijed* je relativan redoslijed koji određuje kako se kontrole međusobno preklapaju na formi. Prva kontrola u opisu forme uspostavlja dno z-redoslijeda. Kontrole koje se pojavljuju kasnije u opisu forme više su u z-redoslijedu i zbog toga preklapaju kontrole koje su niže u z-redoslijedu.

Ugrađeni blokovi kontrola

Neke kontrole mogu sadržavati druge kontrole. Kad je kontrola sadržana unutar druge kontrole, njezin blok kontrole je ugrađen u blok kontrole spremnika. Blokove kontrola možete ugraditi unutar:

- kontrola okvira
- okvira za sliku
- izbornika
- korisničkih kontrola, ovisno o njihovoj namjeni

Ugrađene kontrole se obično koriste za postavljanje gumbi izbora unutar okvira. Visual Basic mora imati sve informacije potrebne za spremnik prije dodavanja bilo koje sadržane kontrole, tako da svojstva kontrole moraju doći prije bilo kojeg bloka ugrađene kontrole. Visual Basic zanemaruje sva svojstva unutar bloka kontrole koja se pojavljuju nakon blokova ugrađenih kontrola.

Kontrole izbornika

Kontrole izbornika se moraju pojaviti zajedno na kraju opisa forme, upravo prije početka odjeljka Attributes. Kad Visual Basic otkrije kontrolu izbornika tijekom učitavanja ASCII forme, očekuje da će naći zajedno sve kontrole izbornika. Kad jednom otkrije kontrolu koja nije izbornik iza jedne ili više kontrola izbornika, Visual Basic pretpostavlja da na formi više nema kontrola izbornika i zanemaruje sve ostale kontrole izbornika koje otkrije tijekom učitavanja te forme.

Prečice kombinacijom tipki

Prečice kombinacijom tipki su tipke koje upotrebljavate za aktiviranje kontrole izbornika. ASCII forma upotrebljava istu sintaksu kao i naredba SendKeys za određivanje kombinacija tipki: "+" = SHIFT, "^" = CTRL, i "{F*n*}" = funkcijska tipka gdje je *n* broj tipke. Abecedni znakovi predstavljaju sami sebe. Sintaksa prečice kombinacijom tipki je:

```
Prečica = ^{F4}                    ' <CTRL><F4>
```

Napomena Izbornici najviše razine ne mogu imati prečicu kombinacijom tipki.

Za više informacija Pogledajte odlomak "Naredba SendKeys", u stalnoj pomoći.

Komentari u opisu forme

Možete dodati komentare u opis forme. Jednostruki znak navoda (‘) je odjelitelj komentara.

Oprez Komentari i oblikovanje u opisu forme ne zadržavaju se kad snimite formu u Visual Basicu. Međutim, komentari i uvlačenja odjeljaka programskog koda u datoteci forme su sačuvani.

Svojstva opisa forme

Kad Visual Basic snima formu, uređuje svojstva po podrazumijevanom redoslijedu. Međutim, možete poredati svojstva po bilo kojem redoslijedu kad stvarate formu.

Svako svojstvo kojeg ne ispišete se postavlja na podrazumijevanu vrijednost kad se učita. Kad Visual Basic snimi formu, uključuje samo ona svojstva koja ne upotrebljavaju podrazumijevane vrijednosti u svojim postavkama. Svaka kontrola određuje hoće li sva njezina svojstva biti snimljena ili neće, ili će biti snimljena samo ona svojstva čije su vrijednosti različite od podrazumijevanih postavki.

Sintaksa

Upotrijebite ovu sintaksu za određivanje svojstava u opisu forme:

svojstvo = vrijednost

Tekstualne vrijednosti svojstava moraju biti postavljene u dvostruke znakove navodnika. Svojstva s vrijednostima tipa Boolean imaju vrijednost od -1 za True i 0 za False. Visual Basic tumači svaku vrijednost različitu od -1 ili 0 kao True. Svojstva sa ispisanim vrijednostima sadrže svoju brojčanu vrijednost s opisom vrijednosti uključenim kao komentarom. Na primjer, svojstvo BorderStyle pojavljuje se ovako:

```
BorderStyle = 0           ‘ None
```

Vrijednosti binarnih svojstava

Neke kontrole imaju svojstva koja imaju binarne podatke kao vrijednosti, kao što je svojstvo Picture kontrola okvira za sliku i kontrola slike ili određena svojstva korisničkih kontrola. Visual Basic snima sve binarne podatke za formu u datoteku binarnih podataka koja je odvojena od forme.

Visual Basic snima datoteku binarnih podataka u isti direktorij kao i formu. Datoteka binarnih podataka ima isto ime datoteke kao i forma, ali ima nastavak imena .frx. Visual Basic čita datoteku binarnih podataka kad učitava formu. Datoteka binarnih podataka (.frx) mora biti dostupna formi kad ju Visual Basic učitava. Ako dijelite forme s drugima koji koriste datoteku binarnih podataka, obavezno pružite datoteku binarnih podataka (.frx) kao i formu (.frm).

Svojstva koja imaju binarne podatke kao svoje vrijednosti pojavljuju se u formi kao pokazivači na bajtni pomak u datoteci binarnih podataka. Na primjer, vrijednost svojstva Picture pojavljuje se na ovaj način u opisu forme:

```
Begin VB.Image imgDemo
   Picture = "Mojaform.frx":02EB
End
```

Ispis svojstva znači da binarni podatak koji određuje svojstvo Picture ove kontrole počinje na bajtu 2EB (hex) u datoteci Mojaform.frx.

Svojstvo Icon

Vrijednost svojstva Icon forme ovisi o tome koja se ikona upotrebljava za formu. Sljedeća tabela ispisuje vrijednosti svojstva Icon te način na koji se pojavljuju u formi.

postavka svojstva Icon	Sadržaj ASCII forme
Podrazumijevana ikona	Nema pokazivača na svojstvo Icon
(None)	Icon = 0
Svaka ikona osim podrazumijevane ikone	Bajtni pomak upućuje na datoteku binarnih podataka. `Na primjer: Icon = "Mojaform.frx":0000`

Svojstvo TabIndex

Ako svojstvo TabIndex nije određeno, Visual Basic dodjeljuje kontroli najraniji mogući položaj u tabulatornom redoslijedu jednom kad su sve ostale kontrole učitane.

Jedinice mjere

Veličine kontrola, koordinate *x* i *y*, i ostale vrijednosti svojstava koje upotrebljavaju jedinice mjere, izražene su u twipovima. Kad kontrola upotrebljava mod mjerila različit od twipova, Visual Basic pretvara vrijednosti twipova iz ASCII forme u jedinice mjere određene svojstvom ScaleMode kontrole kad učitava formu.

Vrijednosti boja

Vrijednosti boja se pojavljuju kao RGB vrijednosti. Na primjer, svojstvo ForeColor pojavljuje se ovako:

```
ForeColor = &H00FF0000&
```

Visual Basic može također čitati vrijednosti tipa QBColor, pretvarajući ih u tip RGB kad učitava formu. ASCII forme koje upotrebljavaju QBColor vrijednosti moraju imati ovu sintaksu:

ForeColor = QBColor(*qbboja*)

gdje je *qbboja* vrijednost od 0 do 15.

Zapamtite da se argument *qbboja* podudara s vrijednostima boja koje upotrebljavaju grafički izrazi u drugim verzijama Basica, kao što su Visual Basic za MS-DOS, Microsoft QuickBasic i Microsoft Basic Professional Development System.

Objekti svojstava

Neki objekti svojstava, kao što je objekt Font, pojavljuju se kao odvojeni blok, koji pokazuje sve postavke za razna svojstva objekta. Ti blokovi su zatvoreni u izraze BeginProperty i EndProperty u sljedećem obliku:

BeginProperty *imesvojstva*

svojstvo1 = vrijednost1

svojstvo2 = vrijednost2

EndProperty

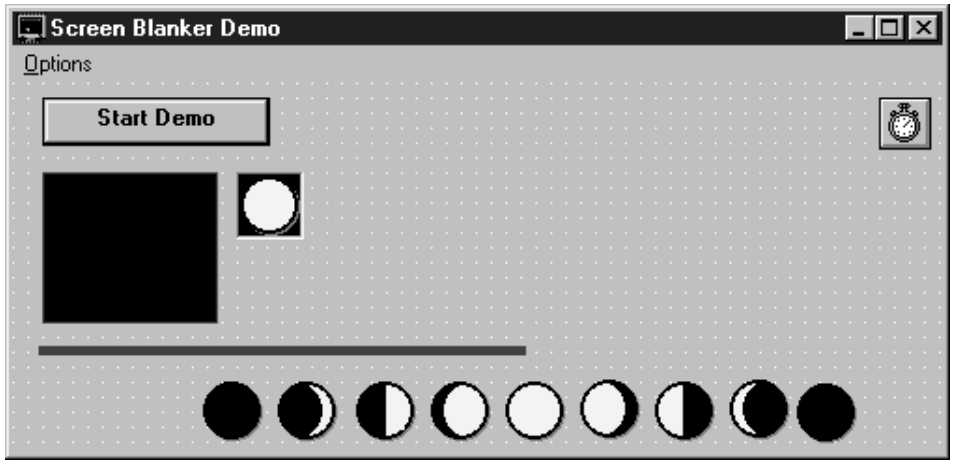
Programski kod Basica

Programski kod Basica se pojavljuje u formi odmah iza odjeljka Attributes i završava posljednjom naredbom End u opisu forme. Izrazi u odjeljku Declarations forme se pojavljuju prvi, a iza njih slijede potprogrami događaja, opći potprogrami i funkcije.

Primjer ASCII forme

Slika A.3 prikazuje formu Blanker.frm iz primjera aplikacije Blanker.

Slika A.3 Primjer forme iz primjera aplikacije Blanker.vbp



Ovdje je dio forme Blanker snimljene u Visual Basicu. Dijelovi forme koji su uklonjeni radi sačuvanja prostora označeni su točkicama.

```
VERSION 6.00
Begin VB.Form DemoForm
   BackColor = &H00000000&
   Caption = “Screen Blanker Demo”
   ClientHeight = 960
   ClientLeft = 1965
   ClientTop = 1965
   ClientWidth = 7470
   ForeColor = &H00000000&
   BeginProperty Font
      Name = “MS Sans Serif”
      Charset = 0
      .
      .
      .
   End Property
   Height = 5115
   Icon = “Blanker.frx”:0018
   Left = 900
   LinkMode = 1                    ‘ Source
   LinkTopic = “Form1”
   ScaleHeight = 4425
   ScaleWidth = 7470
   Top = 1335
   Width = 7590
   Begin VB.Timer Timer1
      Interval = 1
      Left = 6960
      Top = 120
   End
   Begin VB.CommandButton cmdStartStop
      BackColor = &H00000000&
      Caption = “Start Demo”
      Default = -1                         ‘ True
      Height = 390
      Left = 240
      TabIndex = 0
      Top = 120
      Width = 1830
   End
```

```
    Begin VB.PictureBox picBall
      AutoSize = -1                         ‘ True
      BackColor = &H00000000&
      BorderStyle = 0                       ‘ None
      ForeColor = &H00FFFFFF&
      Height = 465
      Left = 1800
      Picture = “Blanker.frx”:0788
      ScaleHeight = 465
      ScaleWidth = 465
      TabIndex = 1
      Top = 720
      Visible = 0                           ‘ False
      Width = 465
    End
    .
    .
    .
    Begin VB.Menu mnuOption
      Caption = “&Options”
      Begin VB.Menu mnuLineCtlDemo
            Caption = “&Jumpy Line”
            Checked = -1                    ‘ True
      End
      Begin VB.Menu mnuCtlMoveDemo
            Caption = “Re&bound”
      End
      .
      .
      .
      Begin VB.Menu mnuExit
            Caption = “E&xit”
      End
    End
End
.
.
.
Attribute VB_Name = “DemoForm”
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Dim Shared FrameNum
Dim Shared XPos
Dim Shared YPos
Dim Shared DoFlag
Dim Shared Motion
.
.
.
```

```
Sub CircleDemo()
  Dim Radius
  R = 255 * Rnd
  G = 255 * Rnd
  B = 255 * Rnd
  XPos = ScaleWidth / 2
  YPos = ScaleHeight / 2
  Radius = ((YPos * 0.9) + 1) * Rnd
  Circle (XPos, YPos), Radius, RGB(R, G, B)
End Sub
.
.
.
Private Sub Timer1_Timer()
.
.
.
End Sub
```

Pogreške učitavanja datoteke forme

Kad Visual Basic učitava formu u memoriju, najprije pretvara formu u binarni oblik. Kad napravite promjene na formi i snimite promjene, Visual Basic ponovno zapisuje datoteku u ASCII obliku.

Kad Visual Basic otkrije pogrešku dok učitava formu, stvara evidencijsku datoteku i izvješćuje tu pogrešku u evidencijskoj datoteci. Visual Basic dodaje poruke pogrešaka u evidencijsku datoteku svaki put kad otkrije pogrešku u formi. Kad je učitavanje forme završeno, Visual Basic prikazuje poruku koja vam kazuje da je stvorena evidencijska datoteka pogrešaka.

Evidencijska datoteka ima isto ime kao datoteka forme, ali sa nastavkom .log imena datoteke. Na primjer, ako se pojave pogreške kad se učitava forma Mojaform.frm, Visual Basic će stvoriti evidencijsku datoteku imena Mojaform.log. Ako kasnije ponovno učitate formu Mojaform.frm i pogreške se nastave pojavljivati kod učitavanja forme, Visual Basic zamjenjuje prethodnu datoteku Mojaform.log.

Poruke u evidencijskoj datoteci pogrešaka učitavanja forme

Sljedeće poruke pogrešaka se mogu pojaviti u evidencijskoj datoteci pogrešaka. Zapamtite da te poruke pogrešaka djeluju samo s problemima koji se mogu pojaviti kad Visual Basic učitava opis forme. One ne naznačuju probleme koji mogu postojati u potprogramima događaja, općim potprogramima, ili bilo kojem drugom dijelu programskog koda Basica.

Ne može se učitati izbornik *imeizbornika* (Cannot load Menu *imeizbornika*).

Ova se poruka pojavljuje ako Visual Basic pronađe kontrolu izbornika čiji je roditeljski izbornik određen kao razdvojnik izbornika. Kontrole izbornika koje djeluju kao roditelji drugih kontrola izbornika u podizborniku ne mogu biti razdvojnici izbornika. Visual Basic ne učitava takvu kontrolu izbornika.

Ova se poruka također pojavljuje ako Visual Basic pronađe kontrolu izbornika čiji roditeljski izbornik ima svojstvo Checked postavljeno na True. Kontrole izbornika koje djeluju kao roditelji drugih kontrola izbornika u podizborniku ne mogu biti potvrđene. Visual Basic ne učitava takvu kontrolu izbornika.

Ne može se postaviti svojstvo Checked za izbornik *imeizbornika* (Cannot set Checked property in Menu *imeizbornika*).

Ova se poruka pojavljuje ako Visual Basic pronađe kontrolu izbornika najviše razine sa svojstvom Checked postavljenim na True. Izbornici najviše razine ne mogu imati oznaku potvrde. Visual Basic učitava tu kontrolu izbornika, ali ne postavlja njezino svojstvo Checked.

Ne može se postaviti svojstvo Shortcut u *imeizbornika* (Cannot set Shortcut property in *imeizbornika*).

Ova se poruka pojavljuje ako Visual Basic pronađe kontrolu izbornika najviše razine s određenom prečicom kombinacijom tipki. Izbornici najviše razine ne mogu imati prečicu kombinacijom tipki. Visual Basic učitava tu kontrolu izbornika, ali ne postavlja njezino svojstvo Shortcut.

Klasa *imeklase* u kontroli *imekontrole* nije učitana klasa kontrole (Class *imeklase* in control *imekontrole* is not, a loaded control class).

Ova se poruka pojavljuje ako Visual Basic pronađe ime klase koje ne prepoznaje.

Kontrola *imekontrole* nije mogla biti učitana (Control *imekontrole* could not be loaded).

Ova se poruka pojavljuje ako Visual Basic otkrije nepoznat tip kontrole u opisu forme. Visual Basic stvara okvir za sliku kako bi predstavio nepoznatu kontrolu, dajući tom okviru za sliku sva valjana svojstva iz opisa nepoznate kontrole. Kad se pojavi ova pogreška, vjerojatno će uslijediti niz pogrešaka nepoznatog svojstva.

Kontrola *imekontrole* ima string u navodnicima na mjestu imena svojstva (Control *imekontrole* has, a quoted string where the property name should be).

Ova se poruka pojavljuje ako Visual Basic otkrije tekst unutar znakova navodnika umjesto imena svojstva, kojeg ne postavljate u znakove navodnika. Na primjer:

```
"Caption" = "Početak Demo-a"
```

U tom slučaju, ime svojstva Caption ne bi trebalo biti zatvoreno u znakove navodnika. Visual Basic zanemaruje liniju u opisu forme koja je izazvala ovu pogrešku.

Ime kontrole *imekontrole* je neispravno (The control name *imekontrole* is invalid).

Ova se poruka pojavljuje ako ime kontrole nije valjan tekst u Visual Basicu. Visual Basic neće učitati tu kontrolu.

Ime kontrole je predugo; skraćeno je na *imekontrole* (Control name too long; truncated to *imekontrole*).

Ova se poruka pojavljuje ako Visual Basic pronađe ime kontrole dulje od 40 znakova. Visual Basic učitava kontrolu, skraćujući njezino ime.

Nije pronađeno svojstvo indeksa i kontrola *imekontrole* već postoji. Ta kontrola se ne može stvoriti (Did not find an index property and control *imekontrole* already exists. Cannot create this control).

Ova se poruka pojavljuje ako Visual Basic pronađe kontrolu bez indeksa koja ima ime kao i prethodno učitana kontrola. Visual Basic ne učitava takvu kontrolu.

Forma *imeforme* nije mogla biti učitana (Form *imeforme* could not be loaded).

Ova se poruka pojavljuje ako Visual Basic neočekivano otkrije kraj datoteke ili ako nedostaje prva naredba Begin.

Ime forme ili MDI forme *imeforme* nije valjano; ta forma se ne može učitati (The Form od MDIForm name *imeforme* is not valid; cannot load this form).

Ova se poruka pojavljuje ako ime forme nije valjan string u Visual Basicu. Visual Basic neće učitati takvu formu.

Valjana imena formi moraju počinjati slovom; smiju sadržavati samo slova, brojeve i podvlake; moraju imati 40 znakova ili manje.

Ime svojstva *imesvojstva* u kontroli *imekontrole* je neispravno (The property name *imesvojstva* in control *controlname* is invalid).

Ova se poruka pojavljuje ako ime svojstva nije valjan string u Visual Basicu ili ako je dulje od 30 znakova. Visual Basic neće postaviti takvo svojstvo.

Svojstvo *imesvojstva* u kontroli *imekontrole* nije moglo biti učitano (Property *imesvojstva* in control *imekontrole* could not be loaded).

Ova se poruka pojavljuje ako Visual Basic otkrije nepoznato svojstvo. Visual Basic preskače to svojstvo kad učitava formu.

Svojstvo *imesvojstva* u kontroli *imekontrole* nije moglo biti postavljeno (Property *imesvojstva* in control *imekontrole* could not be set).

Ova se poruka pojavljuje ako Visual Basic ne može postaviti svojstvo određene kontrole onako kako je naznačeno opisom forme.

Svojstvo *imesvojstva* u kontroli *imekontrole* ima neispravnu vrijednost (Property *imesvojstva* in control *imekontrole* had an invalid value).

Ova se poruka pojavljuje ako Visual Basic otkrije neispravnu vrijednost za svojstvo. Visual Basic mijenja vrijednost svojstva u podrazumijevanu vrijednost za to svojstvo.

Svojstvo *imesvojstva* u kontroli *imekontrole* ima neispravno upućivanje na datoteku (Property *imesvojstva* in control *imekontrole* has an invalid file reference).

Ova se poruka pojavljuje ako Visual Basic ne može upotrijebiti upućivanje na ime datoteke. To će se dogoditi ako datoteka na koju se upućuje (vjerojatno datoteka binarnih podataka za formu) nije pronađena u određenom direktoriju.

Svojstvo *imesvojstva* u kontroli *imekontrole* ima neispravan indeks svojstva (Property *imesvojstva* in control *imekontrole* has an invalid property index).

Ova se poruka pojavljuje ako Visual Basic pronađe ime svojstva s indeksom svojstva većim od 255. Na primjer:

```
Svoj300 = 5436
```

Visual Basic zanemaruje liniju u opisu forme koja je proizvela pogrešku.

Svojstvo *imesvojstva* u kontroli *imekontrole* ima neispravnu vrijednost (Property *imesvojstva* in control *imekontrole* has an invalid value).

Ova se poruka pojavljuje ako Visual Basic pronađe svojstvo s vrijednošću koje nije ispravno za tu kontrolu. Na primjer:

```
Top = Cahr(22)                          ‘ Trebalo bi biti Char(22).
```

Visual Basic postavlja svojstvo s njegovom podrazumijevanom vrijednošću.

Svojstvo *imesvojstva* u kontroli *imekontrole* mora biti string u navodnicima (Property *imesvojstva* in control *imekontrole* must be, a quoted string).

Ova se poruka pojavljuje ako Visual Basic pronađe vrijednost svojstva bez znakova navodnika koje bi se moralo pojaviti unutar znakova navodnika. Na primjer:

```
Caption = Početak Demo-a
```

Visual Basic zanemaruje liniju u opisu forme koja je proizvela pogrešku.

Sintaksna pogreška: svojstvu imesvojstva u kontroli imekontrole nedostaje ‘=’ (Syntax error: property *imesvojstva* in control *imekontrole* is missing an ‘=’).

Ova se poruka pojavljuje ako Visual Basic pronađe ime i vrijednost svojstva bez znaka jednakosti među njima. Na primjer:

```
Text     “Početak Demo-a”
```

Visual Basic ne učitava to svojstvo.

Za više informacija Dodatne informacije o pogreškama otkrivenim tijekom učitavanja forme su dostupne klikom na gumb Help u dijaloškom okviru pogreške ili pritiskom na tipku F1.

Oblik datoteke projekta (.vbp)

Visual Basic uvijek snima datoteke projekta (.vbp) u ASCII obliku. Datoteka projekta sadrži unose koji odražavaju postavke za vaš projekt. Tu su uključene forme i moduli u vašem projektu, upućivanja, raznovrsne opcije koje ste odabrali za nadzor kompilacije itd.

Slijedi primjer kako bi .vbp datoteka mogla izgledati. Ovaj projekt uključuje module s imenima snimljenih klasa i datoteka prikazanim u sljedećoj tabeli.

tip modula	ime klase	ime datoteke
MDI forma	Aforma	A_Forma.frm
Forma	Bforma	B_Forma.frm
Standardni modul	Cmodul	C_Modul.bas
Modul klase	Dklasa	D_Klasa.cls

```
Type=Exe
Form=B_Forma.frm
Reference=*\G{00020430-0000-0000-C000-000000000046}#2.0#0#..
å\..\..\WINDOWS\SYSTEM\STDOLE2.TLB#OLE Automation
Form=A_Forma.frm
Module=Cmodul; C_Modul.bas
Class=Dklasa; D_Klasa.cls
Startup="Bforma"
Command32=""
Name="Projekt1"
HelpContextID="0"
CompatibleMode="0"
MajorVer=1
MinorVer=0
RevisionVer=0
AutoIncrementVer=0
ServerSupportFiles=0
VersionCompanyName="Microsoft"
CompilationType=0
OptimizationType=0
FavorPentiumPro(tm)=0
CodeViewDebugInfo=0
NoAliasing=0
BoundsCheck=0
OverflowCheck=0
FlPointCheck=0
FDIVCheck=0
UnroundedFP=0
StartMode=0
Unattended=0
ThreadPerObject=0
MaxNumberOfThreads=1
```

Unosi se dodaju u datoteku .vbp kad dodajete forme, module, sastavne dijelove itd. vašem projektu. Unosi se također dodaju kad određujete opcije za vaš projekt. Većina tih opcija je postavljena korištenjem dijaloškog okvira Project Properties.

Za više informacija Pritisnite F1 u dijaloški okviru Project Properties za dobivanje objašnjenja o opcijama koje mogu biti odabrane.

Pravila programiranja Visual Basica

Ovaj dodatak predstavlja skup predloženih pravila programiranja za aplikacije Visual Basica.

Pravila programiranja su programske smjernice koje se ne usredotočuju na logiku aplikacije nego na njezinu fizičku strukturu i izgled. Ona čine programski kod lakšim za čitanje, razumijevanje i održavanje. Pravila programiranja mogu sadržavati:

- Pravila za imenovanje objekata, varijabli i potprograma.
- Standardizirane oblike označavanja i komentiranja programskog koda.
- Smjernice za rastavljanje, oblikovanje i uvlačenje.

U odlomcima koji slijede, raspravljeno je svako od tih područja, zajedno s primjerima dobre upotrebe.

Sadržaj

Zašto pravila programiranja?

Glavni razlog upotrebljavanja nepromjenjivog skupa pravila programiranja je standardiziranje strukture i stila programiranja aplikacije tako da vi i drugi možete lako čitati i razumjeti programski kod.

Rezultat dobrih pravila programiranja je precizan, čitljiv i nedvosmislen izvorni programski kod koji je dosljedan s drugim pravilima programskog jezika i intuitivan koliko je moguće.

Minimalna pravila programiranja

Skup pravila programiranja opće namjene trebao bi odrediti minimalne zahtjeve potrebne za ostvarivanje gore navedenih namjena, ostavljajući programeru slobodu stvaranja programske logike i tijeka upotrebljivosti.

Cilj je napraviti aplikaciju lakom za čitanje i razumijevanje bez sprečavanja programerovog prirodnog stvaralaštva pretjeranim prinudama i nerazumnim ograničenjima.

Do te granice, pravila predložena u ovom dodatku su kratka i savjetodavna. Ona ne ispisuju svaki mogući objekt ili kontrolu, niti ne određuju svaki tip poučne napomene koja bi mogla biti vrijedna. Ovisno o vašem projektu i specifičnim potrebama vaše organizacije, možete poželjeti proširiti ove smjernice kako bi uključili dodatne elemente, kao što su:

- Pravila za određene objekte i sastavne dijelove razvijene u kućnoj radinosti ili nabavljene od drugih proizvođača.
- Varijable koje opisuju djelatnosti ili uređaje aktivnosti vaše organizacije.
- Sve druge elemente koje vaš projekt ili tvrtka smatraju važnima za jasnoću i čitljivost.

Za više informacija Za informacije o ograničenjima u imenovanju potprograma, varijabli i konstanti, pogledajte “Temelji kodiranja” u 5. poglavlju “Osnove programiranja”.

Pravila imenovanja objekata

Objekti trebaju biti imenovani s nepromjenjivim prefiksom koji olakšava prepoznavanje tipa objekta. Preporučena pravila za neke od objekata koje podržava Visual Basic dana su u sljedećoj tabeli.

Predloženi prefiksi kontrola

tip kontrole	prefiks	primjer
3D panel (3D Panel)	pnl	pnlGrupa
Kontrola ADO podataka (ADO Data)	ado	adoBiblio
Animirani gumb (Animated button)	ani	aniSandučić
Kontrolna kućica (Check box)	chk	chkSamoZaČitanje
Kombinirani okvir (Combo box), okvir s padajućim popisom (Drop-down list box)	cbo	cboEngleski
Naredbeni gumb (Command button)	cmd	cmdIzlaz
Opći dijalog (Common dialog)	dlg	dlgOtvoriDato
Komunikacije (Communications)	com	comFax
Kontrola (koristi se unutar potprograma kad je nepoznat točan tip)	ctr	ctrTrenutna
Kontrola podataka (Data)	dat	datBiblio
Kombinirani okvir s podacima (Data-bound combo box)	dbcbo	dbcboJezik

tip kontrole	prefiks	primjer
Mreža s podacima (Data-bound grid)	dbgrd	dbgrdRezultatUpita
Okvir s popisom podataka (Data-bound list box)	dblst	dblstTipPosla
Kombinirani okvir podataka (Data combo)	dbc	dbcAutor
Mreža podataka (Data grid)	dgd	dgdNaslovi
Popis podataka (Data list)	dbl	dblIzdavač
Ponavljanje podataka (Data repeater)	drp	drpPoložaj
Odabir datuma (Date picker)	dtp	dtpIzdano
Okvir s popisom direktorija (Directory list box)	dir	dirIzvor
Okvir s popisom pogona (Drive list box)	drv	drvCilj
Okvir s popisom datoteka (File list box)	fil	filIzvor
Plosnata klizna traka (Flat scroll bar)	fsb	fsbPomak
Forma (Form)	frm	frmUnos
Kontrola okvira (Frame)	fra	fraJezik
Mjerilo (Gauge)	gau	gauStatus
Grafikon (Graph)	gra	graPrihod
Mreža (Grid)	grd	grdCijene
Hijerarhijska fleksibilna mreža (Hierarchical flexgrid)	flex	flexNarudžbe
Vodoravna klizna traka (Horizontal scroll bar)	hsb	hsbGlasnoća
Kontrola slike (Image)	img	imgIkona
Kombinirani okvir sa slikama (Image combo)	imgcbo	imgcboProizvod
Popis slika (ImageList)	ils	ilsSveIkone
Kontrola natpisa (Label)	lbl	lblPorukaPomoći
Kontrolna kućica lake kategorije (Lightweight check box)	lwchk	lwchkArhiva
Kombinirani okvir lake kategorije (Lightweight combo box)	lwcbo	lwcboNjemački
Naredbeni gumb lake kategorije (Lightweight command button)	lwcmd	lwcmdUkloni
Okvir lake kategorije (Lightweight frame)	lwfra	lwfraOpcijeSnimanja
Vodoravna klizna traka lake kategorije (Lightweight horizontal scroll bar)	lwhsb	lwhsbGlasnoća

tip kontrole	prefiks	primjer
Okvir s popisom lake kategorije (Lightweight list box)	lwlst	lwlstCentriTroška
Gumb izbora lake kategorije (Lightweight option button)	lwopt	lwoptRazinaPrimanja
Okvir s tekstom lake kategorije (Lightweight text box)	lwtxt	lwtxtUlica
Okomita klizna traka lake kategorije (Lightweight vertical scroll bar)	lwvsb	lwvsbGodina
Linija (Line)	lin	linOkomita
Okvir s popisom (List box)	lst	lstKodoviPolice
Pregled popisa (ListView)	lvw	lvwZaglavlja
MAPI poruka (MAPI message)	mpm	mpmPoslanaPoruka
MAPI zasjedanje (MAPI session)	mps	mpsZasjedanje
Kontrola MCI (MCI)	mci	mciVideo
Izbornik (Menu)	mnu	mnuOtvoriDato
Mjesečni preglednik (Month view)	mvw	mvwRazdoblje
MS grafikon (MS Chart)	ch	chProdajaPodručja
MS fleksibilna mreža (MS Flex grid)	msg	msgKupci
MS kartica (MS Tab)	mst	mstPrva
OLE spremnik (OLE container)	ole	oleRadniList
Gumb izbora (Option button)	opt	optSpol
Okvir za sliku (Picture box)	pic	picVGA
Isječak slike (Picture clip)	clp	clpAlatnaTraka
Traka napretka (ProgressBar)	prg	prgUčitavanjeDato
Udaljeni podaci (Remote Data)	rd	rdNaslovi
Okvir za proširen tekst (RichTextBox)	rtf	rtfIzvješće
Kontrola lika (Shape)	shp	shpKrug
Klizač (Slider)	sld	sldMjerilo
Kontrola vrtnje (Spin)	spn	spnStranice
Statusna traka (StatusBar)	sta	staDatumVrijeme
Sistemske informacije (SysInfo)	sys	sysNadglednik
Traka (TabStrip)	tab	tabOpcije
Okvir s tekstom (Text box)	txt	txtPrezime

tip kontrole	prefiks	primjer
Mjerač vremena (Timer)	tmr	tmrAlarm
Alatna traka (Toolbar)	tlb	tlbAkcije
Pregled stabla (TreeView)	tre	treOrganizacija
Kontrola gore-dolje (UpDown)	upd	updSmjer
Okomita klizna traka (Vertical scroll bar)	vsb	vsbOdnos

Predloženi prefiksi objekata za pristup podacima (data access objects, DAO)

Upotrijebite sljedeće prefikse za naznačivanje objekata za pristup podacima.

objekt baze podataka	prefiks	primjer
Spremnik (Container)	con	conIzvješća
Baza podataka (Database)	db	dbRačuni
Mehanizam baze podataka (DBEngine)	dbe	dbeJet
Dokument (Document)	doc	docIzvješćeProdaje
Polje (Field)	fld	fldAdresa
Grupa (Group)	grp	grpFinancije
Indeks (Index)	idx	idxStarost
Parametar (Parameter)	prm	prmKodPosla
Određivanje upita (QueryDef)	qry	qryPodručnaProdaja
Skup slogova (Recordset)	rec	recPrognoza
Odnos (Relation)	rel	relOdjelDjelatnika
Određivanje tabele (TableDef)	tbd	tbdKupci
Korisnik (User)	usr	usrNovi
Radni prostor (Workspace)	wsp	wspMoj

Neki primjeri:

```
Dim dbBiblio As Database
Dim recIzdavačiUHrv As Recordset, strSQLIzraz As String
Const DB_READONLY = 4                    ‘ Postavljanje konstante.
‘ Otvaranje baze podataka.
Set dbBiblio = OpenDatabase(“BIBLIO.MDB”)
‘ Postavljanje teksta za SQL izraz.
  strSQLIzraz = “SELECT * FROM Publishers WHERE _
  State = ‘HR’”
‘ Stvaranje novog objekta tipa Recordset.
  Set recIzdavačiUHrv = db.OpenRecordset(strSQLIzraz, _
  dbReadOnly)
```

Predloženi prefiksi za izbornike

Aplikacije često upotrebljavaju puno kontrola izbornika, čime postaje korisno imati jedinstveni skup pravila imenovanja za te kontrole. Prefiksi kontrola izbornika trebali bi biti širi od početne oznake "mnu" dodavanjem dodatnog prefiksa za svaku razinu gniježđenja, s naslovom izbornika na kraju stringa imena. Sljedeća tabela ispisuje neke primjere.

niz naslova izbornika	ime rukovatelja izbornikom
Datoteka Otvori	mnuDatotekaOtvori
Datoteka Pošalji E-pošta	mnuDatotekaPošaljiEpošta
Datoteka Pošalji Faks	mnuDatotekaPošaljiFaks
Oblikovanje Znak	mnuOblikovanjeZnak
Pomoć Sadržaj	mnuPomućSadržaj

Korištenjem ovakvog pravila imenovanja, svi elementi određene grupe izbornika su ispisani jedan pored drugog u prozoru sa svojstvima Visual Basica. Osim toga, imena kontrola izbornika jasno dokumentiraju stavke izbornika kojima su pridružene.

Odabir prefiksa za druge kontrole

Za kontrole koje dosad nisu ispisane, trebali bi pokušati standardizirati jedinstven prefiks od dva ili tri znaka zbog dosljednosti. Upotrijebite više od tri znaka samo kad je to potrebno zbog jasnoće.

Za izvedene ili promijenjene kontrole, na primjer, proširite gornje prefikse tako da ne bude zabune koja se kontrola stvarno upotrebljava. Za kontrole drugih proizvođača, kratica s malim slovima za proizvođača može biti dodana prefiksu. Na primjer, primjer kontrole stvoren iz kontrole Professional 3D frame Visual Basica može upotrijebiti prefiks fra3d kako bi se izbjegla zabuna koja se kontrola zapravo upotrijebila.

Pravila imenovanja konstanti i varijabli

Osim objekata, konstante i varijable također zahtijevaju dobro oblikovana pravila imenovanja. Ovaj odlomak ispisuje preporučena pravila za konstante i varijable koje podržava Visual Basic. On također raspravlja pitanja prepoznavanja tipa podatka i područja.

Varijable bi uvijek trebale biti određene s najmanjim mogućim područjem. Opće (Public) varijable mogu stvoriti silno složene mehanizme stanja i učiniti logiku aplikacije osobito teškom za razumijevanje. Opće varijable također značajno otežavaju ponovno korištenje i održavanje vašeg koda.

Varijable u Visual Basicu mogu imati sljedeće područje:

područje	određivanje	vidljiva u
razina potprograma	'Private' u potprogramu tipa Sub ili Function	potprogramu u kojem je određena
razina modula	'Private' u odjeljku Declarations modula forme ili koda (.frm, .bas)	svakom potprogramu u modulu forme ili koda
opća	'Public' u odjeljku Declarations modula koda (.bas)	svugdje u aplikaciji

U aplikaciji Visual Basica, opće varijable trebale bi biti upotrijebljene samo kad nema drugačijeg prikladnog načina za dijeljenje podataka među formama. Kad opće varijable moraju biti upotrijebljene, dobra je praksa odrediti ih u jednom modulu, grupirane po funkciji. Dajte modulu smisaono ime koje naznačuje njegovu namjenu, kao Javni.bas.

Dobra je praksa programiranja napisati modularni kod kad god je to moguće. Na primjer, ako vaša aplikacija prikazuje dijaloški okvir, postavite sve kontrole i kod potreban za izvođenje zadatka dijaloga u jednu formu. To pomaže pri očuvanju koda aplikacije organiziranog u korisne sastavne dijelove i smanjuje nadgradnju tijekom izvođenja.

Sa iznimkom općih varijabli (koje ne bi trebale biti proslijeđene), potprogrami i funkcije bi trebali raditi samo na objektima koji su im proslijeđeni. Opće varijable koje se upotrebljavaju u potprogramima trebale bi biti prepoznate u odjeljcima određivanja na početku potprograma. Osim toga, trebali bi prosljeđivati argumente potprogramima tipa Sub i Function korištenjem izraza ByVal, osim ako izričito ne trebate promijeniti vrijednost proslijeđenog argumenta.

Prefiksi područja varijabli

Kako se povećava veličina projekta, tako raste i vrijednost brzog prepoznavanja područja varijable. To omogućuje jednoslovčani prefiks područja ispred prefiksa tipa, bez većeg povećavanja veličine imena varijabli.

područje	prefiks	primjer
Opće (globalne)	g	gstrImeKorisnika
Razina modula	m	mblnRačunUTijeku
Lokalno za potprogram	nikakav	dblBrzina

Varijabla ima opće područje ako je određena sa Public u standardnom modulu ili modulu forme. Varijabla ima područje *na razini modula* ako je određena sa Private u standardnom modulu ili modulu forme.

> **Napomena** Dosljednost je presudna za uspješno korištenje ove tehnike; provjeravanje sintakse u Visual Basicu neće uhvatiti varijable na razini modula koje počinju sa "p".

Konstante

Tijelo imena konstanti trebalo bi biti sastavljeno od malih i velikih slova sa velikim slovima na početku svake riječi. Iako standardne konstante Visual Basica ne sadrže informacije o tipu podatka i području, prefiksi poput i, s, g, i m mogu biti vrlo korisni u razumijevanju vrijednosti i područja konstante. Za imena konstanti, slijedite ista pravila kao i za varijable. Na primjer:

```
mintKorListaMax          ‘ Najveća granica unosa za korisnički popis
                         ‘ (cjelobrojna vrijednost, lokalna za modul)
gstrNovaLinija           ‘ Karakter nove linije
                         ‘ (string, opći za aplikaciju)
```

Varijable

Određivanje svih varijabli štedi vrijeme programiranja smanjujući broj pogrešaka uzrokovanih pogrešnim pisanjem (na primjer, aKorImePri protiv sKorImePri protiv sKorImePriv). Na kartici Editor dijaloga Options, potvrdite opciju Require Variable Declaration (potrebno određivanje varijabli). Izraz Option Explicit traži da odredite sve varijable u vašoj aplikaciji Visual Basica.

Varijable bi trebale imati prefiks koji naznačuje njihov tip podatka. Osim toga, posebno za velike aplikacije, prefiks bi trebao biti proširen tako da naznačuje područje varijable.

Tipovi podataka varijabli

Upotrijebite sljedeće prefikse za naznačivanje tipa podatka varijable.

tip podatka	prefiks	primjer
Boolean	bln	blnPronađeno
Byte	byt	bytPodaciRastera
Objekt zbirke (Collection)	col	colSpravice
Valuta (Currency)	cur	curPrihod
Datum i vrijeme (Date, Time)	dtm	dtmPočetak
Double	dbl	dblTolerancija
Pogreška (Error)	err	errBrojNarudžbe
Cijeli broj (Integer)	int	intKoličina
Long	lng	lngUdaljenost
Objekt (Object)	obj	objTrenutan
Single	sng	sngProsjek
String	str	strIme

tip podatka	prefiks	primjer
Korisnički određen tip	udt	udtDjelatnik
Variant	vnt	vntKontrolniZbroj

Opisna imena varijabli i potprograma

Tijelo imena varijable ili potprograma trebalo bi biti sastavljeno od malih i velikih slova te dugačko koliko je potrebno za opis njihove namjene. Osim toga, imena funkcija bi trebala započinjati glagolom, kao PokreniMatricuImena ili ZatvoriDijalog.

Za često korištena ili duga imena, preporučene su standardne kratice kako bi pomogle u očuvanju razumne duljine imena. Općenito, imena varijabli duža od 32 znaka mogu biti teško čitljiva na VGA prikazima.

Kad upotrebljavate kratice, osigurajte dosljednost kroz cijelu aplikaciju. Nasumično prebacivanje između imena Broj i Brojiti unutar projekta će dovesti do nepotrebne zbrke.

Korisnički određeni tipovi

U velikom projektu s puno korisnički određenih tipova, često je korisno dati svakom takvom tipu vlastiti troslovčan prefiks. Ako takvi prefiksi započinju sa “u” (user-defined type), i dalje će biti laki za brzo prepoznavanje kad radite s korisnički određenim tipovima. Na primjer, “ukup” bi mogao biti upotrijebljen kao prefiks za varijable korisnički određenog tipa Kupac.

Pravila strukturiranog programiranja

Osim pravila imenovanja, pravila strukturiranog programiranja, kao što je komentiranje koda i dosljedno uvlačenje linija, mogu značajno poboljšati čitljivost programskog koda.

Pravila komentiranja koda

Svi potprogrami i funkcije bi trebali započeti kratkim komentarom koji opisuje funkcionalne značajke potprograma (što on radi). Taj opis ne bi trebao opisivati detalje ostvarivanja (kako to radi) jer se oni često mijenjaju vremenom, rezultat čega može biti nepotreban posao održavanja komentara, ili još gore, pogrešni komentari. Sam programski kod i svi potrebni komentari u linijama će opisati djelovanje.

Argumenti koji su proslijeđeni potprogramu trebali bi biti opisani kad njihova funkcija nije očigledna i kad potprogram očekuje da argumenti budu u određenom opsegu. Povratne vrijednosti funkcija i opće varijable koje se mijenjaju u potprogramu, posebno kroz argumente upućivanja, trebaju također biti opisane na početku svakog potprograma.

Blokovi komentara u zaglavlju potprograma trebali bi sadržavati sljedeće naslove odjeljaka. Za primjere, pogledajte idući odlomak “Oblikovanje vašeg koda”.

naslov odjeljka	opis komentara
Namjena	Što potprogram čini (ne kako).
Pretpostavke	Popis svake vanjske varijable, kontrole, otvorene datoteke ili drugog elementa koji nije očigledan.
Učinci	Popis svake vanjske varijable, kontrole ili datoteke na koju se utječe, te učinak kojeg ima (samo ako to nije očigledno).
Ulazi	Svaki argument koji možda nije očigledan. Argumenti su u odvojenoj liniji s komentarima u liniji.
Povrati	Objašnjenje vrijednosti koje vraća funkcija.

Zapamtite sljedeće točke:

- Određivanje svake važne varijable treba sadržavati komentar u liniji koji opisuje upotrebu određene varijable.
- Varijable, kontrole i potprogrami trebali bi biti imenovani dovoljno jasno tako da je komentiranje u linijama potrebno samo za složene detalje ostvarivanja.
- Na početku modula tipa .bas koji sadrži određivanja općih konstanti projekta Visual Basica, trebali bi uključiti pregled koji opisuje aplikaciju, nabraja temeljne objekte podataka, potprograme, algoritme, dijaloge, baze podataka i ovisnosti sustava. Ponekad može biti koristan dio pseudokoda koji opisuje algoritam.

Oblikovanje vašeg koda

Budući da puno programera još uvijek upotrebljava VGA prikaze, prostor na ekranu trebao bi biti sačuvan koliko je moguće dok se i dalje omogućuje oblikovanje koda koje odražava logičku strukturu i gniježđenje. Slijedi nekoliko pokazivača:

- Standardni, ugniježđeni blokovi uređeni tabulatorima trebaju biti uvučeni za četiri razmaka (podrazumijevano).
- Komentar pregleda djelovanja potprograma treba biti uvučen za jedan razmak. Izrazi najviše razine koji slijede komentar pregleda trebaju biti uvučeni za jedno tabulatorsko mjesto, sa svakim ugniježđenim blokom uvučenim za dodatno tabulatorsko mjesto. Na primjer:

```
‘*****************************************************
‘ Namjena: Pronalaženje prvog pojavljivanja
‘          određenog korisnika u matrici ListaKor.
‘ Ulazi:
‘    strListaKor(): popis korisnika koja se pretražuje.
‘    strCiljniKor: ime korisnika koji se traži.
‘ Povrati: Indeks prvog pojavljivanja
‘          strCiljniKor u matrici strListaKor. Ako
‘          ciljni korisnik nije pronađen, vraća -1.
‘*****************************************************
Function intProna|iKorisnika(strListaKor() As String, _
strCiljniKor As String) As Integer
  Dim i As Integer               ‘ Brojač petlje.
  Dim blnNa|en As Boolean        ‘ Zastavica pronađenog cilja.
  intProna|iKor = -1
  i = 0
  While i <= UBound(strListaKor) And Not blnNa|en
    If strListaKor(i) = strCiljniKor Then
          blnNa|en = True
          intPronađiKor = i
    End If
  Wend
End Function
```

Grupiranje konstanti

Varijable i određene konstante trebaju biti grupirane po funkciji umjesto da budu razdvojene u odvojena područja posebnih datoteka. Opće konstante Visual Basica trebaju biti grupirane u jedan modul kako bi se odvojile od određivanja specifičnih za aplikaciju.

Operatori & i +

Uvijek upotrebljavajte operator & kad povezujete stringove, i operator + kad radite s brojčanim vrijednostima. Korištenje operatora + za povezivanje može prouzročiti probleme kod rada s dvije varijable tipa Variant. Na primjer:

```
vntVar1 = “10.01”
vntVar2 = 11
vntRezultat = vntVar1 + vntVar2          ‘ vntRezultat = 21.01
vntRezultat = vntVar1 & vntVar2          ‘ vntRezultat = 10.0111
```

Stvaranje tekstova za okvire s porukom, okvire za upis podataka i SQL upite

Kad stvarate dugačak tekst, upotrijebite podvlaku kao oznaku nastavka linije za stvaranje više linija koda tako da možete lako pročitati ili ispraviti tekst. Ova tehnika je posebno korisna kad prikazujete okvir s porukom (MsgBox) ili okvir za upis podataka (InputBox), ili kad stvarate SQL tekst. Na primjer:

```
Dim Por As String
Por = “Ovo je odlomak koji će biti “ _
& “u okviru s porukom. Tekst je “ _
& “razlomljen u nekoliko linija koda “ _
& “u izvornom kodu, kako bi ga programer “ _
& “lakše mogao pročitati i ispraviti.”
MsgBox Por
```

```
Dim QRY As String
QRY = “SELECT * “ _
& “FROM Titles “ _
& “WHERE [Year Published] > 1988”
TitlesQry.SQL = QRY
```

DODATAK C

Opcije prevoditelja u strojni kod

Microsoft Visual Basic vam omogućuje prevođenje vaših aplikacija u brz, učinkovit strojni kod, korištenjem iste tehnologije optimizirajućeg pozadinskog prevoditelja kao što je Microsoft C++. Prevođenje u strojni kod pruža nekoliko opcija za optimiziranje i ispravljanje pogrešaka koje nisu dostupne sa p-kodom. Ove opcije se uobičajeno nazivaju "prekidači", jer svaka opcija može biti uključena ili isključena.

Ovaj dodatak dokumentira opcije prevoditelja u strojni kod, koje se pojavljuju na kartici Compile dijaloškog okvira Project Properties, dostupnog iz izbornika Project. Za više informacija o strojnom kodu, pogledajte odlomak "Prevođenje vašeg projekta u strojni kod" u 8. poglavlju, "Više o programiranju".

Sadržaj

- Optimiziranje za brzi kod
- Optimiziranje za mali kod
- Bez optimizacije
- Podrška Pentiumu Pro
- Stvaranje simboličkih informacija za otkrivanje pogrešaka
- Prihvaćanje bez pseudonima
- Uklanjanje provjere granica matrica
- Uklanjanje provjere prekoračenja cijelih brojeva
- Uklanjanje provjere pogreške s pomičnim zarezom
- Uklanjanje provjere sigurnog dijeljenja Pentiumom
- Omogućavanje nezaokruženih operacija pomičnim zarezom

Optimiziranje za brzi kod

(Optimize for Fast Code)

Povećava brzinu prevedene izvršne datoteke kazujući prevoditelju da podrži brzinu prije veličine.

Kad prevoditelj prevede izraze Visual Basica u strojni kod, često postoji puno različitih nizova strojnog koda koji ispravno predstavljaju dani izraz ili konstrukciju. Ponekad te razlike nude zamjenu veličine za brzinu. Odabir ove opcije osigurava da kad prevoditelj prepozna takav izbor, uvijek stvara što je moguće brži slijed koda, čak i ako to može povećati veličinu prevedene aplikacije.

Optimiziranje za mali kod

(Optimize for Small Code)

Smanjuje veličinu prevedene izvršne datoteke kazujući prevoditelju da podrži veličinu prije brzine.

Kad prevoditelj prevede izraze Visual Basica u strojni kod, često postoji puno različitih nizova strojnog koda koji ispravno predstavljaju dani izraz ili konstrukciju. Ponekad te razlike nude zamjenu veličine za brzinu. Odabir ove opcije osigurava da kad prevoditelj prepozna takav izbor, uvijek stvara što je moguće manji slijed koda, čak i ako to može smanjiti brzinu izvođenja prevedene aplikacije.

Bez optimizacije

(No Optimizations)

Isključuje sve optimizacije.

Sa ovom opcijom potvrđenom, prevoditelj stvara kod koji je znatno sporiji i veći nego kad je odabrana optimizacija za brzi ili mali kod.

Podrška Pentiumu Pro

(Favor Pentium Pro)

Optimizira stvaranje koda za podršku procesoru Pentium Pro (P6). Kod stvoren ovom opcijom će se i dalje izvoditi na ranijim procesorima, ali s manjom učinkovitošću.

Arhitektura mikroprocesora Pentium Pro omogućuje određene strategije stvaranja koda koji može stvarno poboljšati učinkovitost. Međutim, kod stvoren korištenjem tih strategija ne izvodi se jednako dobro na računalima s procesorima tipa 80386, 80486 ili Pentium. Zbog toga, trebate upotrijebiti ovu opciju samo ako će svi ili većina uređaja na kojima će se izvoditi vaša aplikacija upotrebljavati procesor tipa Pentium Pro.

Stvaranje simboličkih informacija za otkrivanje pogrešaka

(Create Symbolic Debug Info)

Stvara simboličke informacije za otkrivanje pogrešaka u prevedenoj izvršnoj datoteci.

Aplikacije prevedene u strojni kod korištenjem ove opcije mogu biti ispravljane korištenjem aplikacije Visual C++ (verzija 5.0 ili kasnija) ili druge sukladne aplikacije za pronalaženje pogrešaka. Potvrđivanje ove opcije će stvoriti .pdb datoteku s potrebnim simboličkim informacijama za upotrebu sa sukladnim simboličkim aplikacijama za pronalaženje pogrešaka.

Prihvaćanje bez pseudonima

(Assume No Aliasing)

Kazuje prevoditelju da vaša aplikacija ne upotrebljava pseudonime.

Pseudonim je ime koje upućuje na položaj u memoriji na koji se već upućuje drugim imenom. To se pojavljuje kod korištenja argumenata tipa ByRef koji upućuju na istu varijablu na dva načina. Na primjer:

```
Sub Foo(x As Integer, y As Integer)
  x = 5           ‘ Kod upućuje na istu varijablu
                  ‘ (lokalna varijabla z u potprogramu Glavno)
  y = 6           ‘ kroz dva različita imena, x i y
End Sub
Sub Glavno
  Dim z As Integer
  Foo z, z
End Sub
```

Korištenje ove opcije dopušta prevoditelju da primijeni optimizacije koje drugačije ne bi mogao upotrijebiti, kao što je spremanje varijabli u registre i izvođenje optimizacija petlji. Međutim, trebate pripaziti da ne potvrdite ovu opciju ako vaša aplikacija prosljeđuje argumente tipa ByRef, jer optimizacija može uzrokovati neispravno izvođenje aplikacije.

Uklanjanje provjere granica matrica

(Remove Array Bounds Checks)

Isključuje provjeru pogreške za valjane indekse matrica i ispravne brojeve dimenzija matrice.

Visual Basic u pravilu provjerava svaki pristup matrici kako bi ustanovio je li indeks unutar opsega matrice. Ako je indeks izvan granica matrice, vraća se pogreška. Odabir ove opcije će isključiti provjeru ove pogreške, što značajno može ubrzati upravljanje

matricama. Međutim, ako vaša aplikacija pristupa matrici s indeksom koji je izvan granica, može pristupiti pogrešnim mjestima u memoriji bez upozorenja. To može izazvati neočekivano ponašanje ili rušenje aplikacije.

Uklanjanje provjere prekoračenja cijelih brojeva

(Remove Integer-Overflow Checks)

Isključuje provjeravanje pogreške koje osigurava da brojčane vrijednosti dodijeljene cjelobrojnim varijablama budu unutar ispravnog opsega za taj tip podatka.

U Visual Basicu se, u pravilu, provjerava svako proračunavanje sa varijablom koja je tipa cjelobrojnog podatka (Byte, Integer, Long i Currency) kako bi se osiguralo da je vrijednost rezultata unutar opsega tog tipa podatka. Ako je vrijednost krive veličine, pojavit će se pogreška. Odabir ove opcije će isključiti provjeru takve pogreške, što može ubrzati proračunavanja s cijelim brojevima. Ako se prekorači kapacitet tipa podatka, međutim, neće biti vraćena pogreška i mogu se pojaviti neispravni rezultati.

Uklanjanje provjere pogreške s pomičnim zarezom

(Remove Floating-Point Error Checks)

Isključuje provjeru pogreške koja osigurava da brojčane vrijednosti koje se dodjeljuju varijablama s pomičnim zarezom budu unutar ispravnog opsega za tipove podataka, te da se ne pojavi dijeljenje s nulom ili druge neispravne operacije.

U Visual Basicu se, u pravilu, provjerava svako proračunavanje s varijablom koja je tipa podatka s pomičnim zarezom (Single i Double) kako bi se osiguralo da je vrijednost rezultata unutar opsega tog tipa podatka. Ako je vrijednost krive veličine, pojavit će se pogreška. Provjera pogreške se također izvodi za utvrđivanje je li pokušano dijeljenje sa nulom ili druge neispravne operacije. Odabir ove opcije isključuje provjeravanje takve pogreške što može ubrzati operacije s pomičnim zarezom. Ako se prekorači kapacitet tipa podatka, međutim, neće biti vraćena pogreška i mogu se pojaviti neispravni rezultati.

Uklanjanje provjere sigurnog dijeljenja sa Pentiumom

(Remove Safe Pentium FDIV Checks)

Isključuje stvaranje posebnog koda koji osigurava dijeljenje s pomičnim zarezom na procesorima tipa Pentium s pogreškom pri dijeljenju s pomičnim zarezom (floating-point division, FDIV).

Prevoditelj u strojni kod automatski dodaje poseban kod za operacije sa pomičnim zarezom kako bi te operacije bile pouzdane kad se izvode na procesorima tipa Pentium koji imaju FDIV pogrešku. Odabir ove opcije proizvodi kod koji je manji i brži, ali

koji u rijetkim slučajevima može proizvesti neznatno neispravne rezultate na procesorima tipa Pentium sa FDIV pogreškom.

Omogućavanje nezaokruženih operacija sa pomičnim zarezom

(Allow Unrounded Floating-Point Operations)

Omogućuje prevoditelju da uspoređuje rezultate izraza s pomičnim zarezom bez prethodnog zaokruživanja tih rezultata na ispravnu preciznost.

Proračuni s pomičnim zarezom se normalno zaokružuju na ispravan stupanj preciznosti (Single ili Double) prije nego što se naprave uspoređivanja. Odabir ove opcije omogućuje prevoditelju da uspoređuje vrijednosti s pomičnim zarezom prije zaokruživanja, kad to može napraviti učinkovitije. To poboljšava brzinu nekih operacija s pomičnim zarezom; međutim, rezultat mogu biti proračuni koji održavaju veću preciznost od očekivane, i dvije vrijednosti s pomičnim zarezom koje bi u uspoređivanju trebale biti jednake, to neće biti.

Ova opcija općenito ne bi trebala biti upotrijebljena ako izvodite uspoređivanja jednakosti izravno na rezultatima računanja s pomičnim zarezom. Na primjer:

```
Dim Q As Single

Q = <računanje sa pomičnim zarezom>
...
If Q = <računanje sa pomičnim zarezom> Then
...
End If
```

Ako je ova opcija potvrđena, uspoređivanje varijable Q će biti napravljeno s rezultatom izraza s pomičnim zarezom, što će vjerojatno imati veću preciznost od tipa Single, tako da uspoređivanje može pogriješiti. Ako ova opcija nije potvrđena, rezultat izraza s pomičnim zarezom će biti zaokružen na odgovarajuću preciznost (Single) prije uspoređivanja, pa će uspoređivanje biti uspješno.

Dodavanje pomoći vašoj aplikaciji

Bez obzira na to kako je umješna aplikacija, u nekom trenutku većina korisnika počinje postavljati pitanja kako ju koristiti. Osim ako niste tamo kako bi osobno odgovorili na pitanja, najbolji način rukovanja time je pružanje datoteke pomoći za aplikaciju.

Visual Basic pruža podršku za dva različita sustava pomoći: tradicionalni sustav Windows Help (WinHelp), i noviji sustav HTML Help. Ovaj dodatak pokriva korake potrebne za dodavanje pomoći tipa WinHelp ili HTML Help vašoj aplikaciji, ističući nekoliko razlika između ta dva sustava gdje je to primjenjivo. U ovom dodatku nećete pronaći kako napisati pomoć – postoje brojni dostupni alati takve namjene koji vam u tome mogu pomoći.

Sadržaj

Dodavanje podrške za pomoć

Dodavanje podrške za pomoć vašoj aplikaciji Visual Basica je stvarno posve jednostavno. Sve što trebate je postavljanje jednog svojstva, HelpFile (i, naravno, pisanje i prevođenje datoteke pomoći), za prikaz pomoći kad korisnik pritisne F1 ili zatraži pomoć iz izbornika. Dodatno svojstvo, HelpContextID, može biti postavljeno tako da pruži sadržajnu temu pomoći za svaki element korisničkog sučelja u vašoj aplikaciji. Postupak hvatanja pomoći je u biti isti za sustave WinHelp i HTML Help.

Svojstvo HelpFile

Svojstvo HelpFile objekta App se koristi za određivanje imena datoteke pomoći za vašu aplikaciju. Ovo svojstvo zahtijeva valjanu datoteku tipa WinHelp (.hlp) ili HTML Help (.chm). Ako datoteka ne postoji, pojavit će se pogreška.

Kako postaviti svojstvo HelpFile

1. Odaberite **Project Properties** iz izbornika **Project** kako bi otvorili dijaloški okvir **Project Properties**.
2. U polju **Help File Name** na kartici **General**, upišite stazu i ime datoteke pomoći za vašu aplikaciju (.hlp ili .chm).

Svojstvo HelpFile možete postaviti i programski. Sljedeći programski kod će odrediti HTML Help datoteku koja se nalazi u istom direktoriju kao i izvršna datoteka aplikacije:

```
Private Sub Form_Load()
  App.HelpFile = App.Path & "\foo.chm"
End Sub
```

Objekt ErrObject također ima svojstvo HelpFile, omogućujući vam da odredite drugačiju datoteku pomoći za poruke pogrešaka. Na primjer, ako imate nekoliko aplikacija koje dijele iste poruke pogrešaka, možete postaviti pomoć za poruke grešaka u jednu datoteku pomoći koja može biti pozvana svojstvom Err.HelpFile u svakoj aplikaciji.

Svojstvo HelpContextID

Svojstvo HelpContextID se koristi za povezivanje elementa korisničkog sučelja (kao što je kontrola, forma ili izbornik) sa srodnom temom u datoteci pomoći. Svojstvo HelpContextID mora biti tipa Long koji se podudara s brojem Context ID u temi datoteke pomoći tipa WinHelp (.hlp) ili tipa HTML Help (.chm).

Na primjer, možete unijeti broj 10 000 u svojstvo HelpContextID okvira s tekstom. Kad korisnik odabere taj okvir s tekstom i pritisne F1, Visual Basic traži temu s brojem Context ID od 10 000 u datoteci pomoći određenoj u svojstvu HelpFile aplikacije. Ako je tema pronađena, otvorit će se prozor Help i prikazati temu; ako nije, pojavit će se pogreška i bit će prikazana podrazumijevana tema datoteke pomoći.

Trebate upotrebljavati jedinstvene vrijednosti svojstva HelpContextID za podudaranje sa svakom temom pomoći u vašoj datoteci pomoći. U nekim slučajevima, možete trebati dodijeliti istu vrijednost svojstvima HelpContextID više objekata ako objekti dijele zajedničku temu pomoći.

Ne trebate obavezno odrediti svojstvo HelpContextID za svaku kontrolu na formi. Ako korisnik pritisne F1 na kontroli čije je svojstvo HelpContextID postavljeno na 0 (standardno), Visual Basic će potražiti valjan broj pomoći za spremnik te kontrole.

Kako dodijeliti svojstvo HelpContextID kontroli ili formi

1. Odaberite kontrolu ili formu za koju želite odrediti svojstvo HelpContextID.
2. Dva puta kliknite svojstvo HelpContextID u prozoru sa svojstvima i upišite valjan cijeli broj tipa Long.

   Pratite vrijednosti koje unosite tako da možete upotrijebiti iste vrijednosti za broj pomoći u pridruženoj temi pomoći.

Napomena Za kontrolu CommonDialog te za neke druge kontrole, ime ovog svojstva je HelpContext umjesto HelpContextID.

Kako dodijeliti svojstvo HelpContextID izborniku

1. Odaberite **Menu Editor** iz izbornika **Tools**.
2. Odaberite stavku izbornika kojoj želite odrediti svojstvo HelpContextID.
3. Upišite valjan broj tipa Long u okvir **Select the HelpContextID**.

   Pratite vrijednosti koje unosite tako da možete upotrijebiti iste vrijednosti za broj pomoći u pridruženoj temi pomoći.

Svojstvo HelpContextID može također biti određeno programski na sljedeći način:

```
Private Sub Form_Load()
  Command1.HelpContextID = 12345
  MenuHelp.HelpContextID = 23456
  Err.HelpContext = 34567
End Sub
```

Savjet Ako imate više od nekoliko tema pomoći, moglo bi pomoći uspostavljanje sheme obrojčavanja prije nego što počnete upisivati vrijednosti svojstava HelpContextID. Dodijelite drugačije opsege brojeva svakoj formi ili glavnom elementu u vašoj aplikaciji, na primjer, 1000 – 1999 za prvu formu, 2000 – 2999 za drugu i tako dalje.

Dodavanje podrške za pomoć What's This

Visual Basic vam omogućuje da lako dodate pomoć What's This svojoj aplikaciji. Pomoć What's This pruža brz pristup tekstu pomoći u pomoćnom prozoru bez potrebe za otvaranjem preglednika pomoći. Ova pomoć se tipično koristi za pružanje jednostavne podrške elementima korisničkog sučelja kao što su polja za unos podataka. Visual Basic podržava teme pomoći What's This u datotekama tipa WinHelp (.hlp) i tipa HTML Help (.chm).

Postavljanje svojstva WhatsThisHelp forme na True omogućuje pomoć What's This. Kad je omogućena pomoć What's This, pomoć forme osjetljiva na sadržaj je onemogućena.

Kako omogućiti pomoć What's This za formu

1. S odabranom formom, dva puta kliknite na svojstvo **WhatsThisHelp** u prozoru sa svojstvima kako bi to svojstvo postavili na **True**.
2. Postavite sljedeća svojstva kako bi dodali gumb What's This u naslovnu traku forme:

svojstvo	postavka
BorderStyle	1 – Fixed Single ili 2 – Sizable
MaxButton	False
MinButton	False
WhatsThisButton	True

Napomena Forma ne može imati gumb What's This ako ima gumbe Minimize i Maximize. Kao alternativu prije prikazanim postavkama, možete također postaviti svojstvo BorderStyle na 3 – Fixed Dialog, budući da nepromjenjivi dijalozi nemaju gumbe Minimize i Maximize.

3. Odaberite svaku kontrolu kojoj želite pružiti pomoć What's This i dodijelite jedinstvenu vrijednost svojstvu WhatsThisHelpID te kontrole.

Pratite vrijednosti koje unosite tako da možete upotrijebiti iste vrijednosti za broj pomoći u pridruženoj temi pomoći.

Važno Kako bi ostvarili pomoć What's This u pomoći tipa HTML Help, sve teme pomoći What's This moraju biti sadržane u datoteci Cshelp.txt koja se prevodi u .chm datoteku. Za više informacija, pogledajte dokumentaciju za vaš alat stvaranja pomoći tipa HTML Help.

Pomoć What's This možete također omogućiti bez korištenja gumba What's This tako da postavite svojstvo WhatsThisHelp forme na True i pozovete postupak WhatsThisMode forme ili postupak ShowWhatsThis kontrole.

Distribuiranje pomoći s vašom aplikacijom

Posljednji korak dodavanja pomoći vašoj aplikaciji je osiguravanje da dođe u ruke krajnjem korisniku. Zahtjevi za distribuiranje pomoći s vašom aplikacijom se neznatno razlikuju između pomoći tipa WinHelp i tipa HTML Help.

Distribuiranje pomoći tipa WinHelp

Budući da svaki sustav Windowsa već ima instaliranog preglednika pomoći tipa WinHelp, jedina stvar koju trebate distribuirati je sama datoteka pomoći (.hlp). Čarobnjak za pakiranje i raspoređivanje će automatski dodati ovisnost za datoteku pomoći na koju upućuje vaša aplikacija. Ako stvarate podešavanje drugim sredstvima, morate osigurati da .hlp datoteka bude uključena i instalirana na pravo mjesto (obično u isti direktorij gdje je aplikacija ili u direktorij \Windows\Help).

Distribuiranje pomoći tipa HTML Help

Pomoć tipa HTML Help je relativno nova tehnologija, i zbog toga ne možete pretpostaviti da će svaki korisnik imati datoteke potrebne za pregled pomoći tipa HTML Help. Čarobnjak za pakiranje i raspoređivanje će dodati ovisnost za datoteku pomoći tipa HTML Help (.chm) na koju upućuje vaša aplikacija; međutim, možda neće dodati sve ovisnosti za datoteke pregledavanja pomoći tipa HTML Help. Trebate promijeniti svoje podešavanje tako da uključuje te datoteke. Potražite upute u dokumentaciji vašeg alata za stvaranje pomoći tipa HTML Help za više informacija o tome koje su datoteke potrebne u danoj situaciji.