DEO

I

Osnove

U OVOM DELU:

1 Deplhi 7 i njegov IDE

2 Delphijev programski jezik

3 Izvršna biblioteka

4 Klase osnovne biblioteke

5 Vizuelne kontrole

6 Pravljenje korisničkog interfejsa

7 Rad sa formularima

POGLAVLJE

1

Deplhi 7 i njegov IDE

U vizuelnom programskom alatu kao što je Delphi, uloga integrisanog razvojnom okruženja (IDE) je čak važnija od samog programskog jezika. Delphi 7 uvodi mnoge nove mogućnosti i pored ionako velikih mogućnosti IDEA-a Delphija 6. U ovom poglavlju ću objasniti nove mogućnosti kao i mogućnosti koje su ugrađene u skorije verzije Delphija. Takođe ćemo se pozabaviti nekim tradicionalnim mogućnostima Delphija koje nisu u širokoj upotrebi ili nisu očigledne novim korisnicima. Ovo poglavlje nije potpuno uputstvo već, uglavnom, kolekcija saveta i sugestija namenjena prosečnom korisniku Delphija.

Ukoliko ste programer početnik, nemojte da se plašite. Delphijevo integrisano razvojno okruženje je prilično intuitivno. Delphi sadrži uputstvo (možete ga pronaći u formatu Acrobat na Delphijevom Companion Tools CD-u) sa delom koji predstavlja razvoj Delphi aplikacija. Instrukcije korak-po-korak za Delphi i njegov IDE možete pronaći u mojoj knjizi Essential Delphi (koji ću predstaviti u Dodatku C, "Besplatne knjige za Delphi"). U ovoj knjizi ću pretpostaviti da ste već naučili kako da u IDE-u izvršite osnovne operacije. Sva poglavlja nakon ovog se bave temama i tehnikama vezanim za programiranje.

U ovom poglavlju su objašnjene sledeće teme:

- Snalaženje u IDE-u
- Editor
- Tehnologija CodeInsight
- Dizajniranje formulara
- Project Manager
- Delphijeve datoteke

Izdanja Delphija 6

Pre nego što uronimo u detalje programskog okruženja Delphija, posvetimo pažnju dvema ključnim idejama. Prvo, ne postoji jedno izdanje Delphija; postoje mnoga izdanja. Drugo, svako Delphi okruženje se može prilagoditi. Zbog toga se Delphi ekrani, koje ćete videti u ovom poglavlju, mogu razlikovati od ekrana na Vašem računaru. Evo aktuelnih izdanja Delphija:

- Verzija "Personal" je namenjena novim korisnicima Delphija i povremenim programerima pa ne sadrži podršku za programiranje baza podataka niti bilo koju napredniju mogućnost Delphija.
- Verzija "Professional Studio" je namenjena profesionalnim programerima. Sadrži sve osnovne mogućnosti i podršku za programiranje baza podataka (uključujući podršku za ADO), osnovnu podršku za web server (WebBroker) i neke dodatne alate, uključujući ModelMaker i IntraWeb. U knjizi se podrazumeva da koristite bar verziju Professional.
- Verzija "Enterprise Studio" je namenjena razvoju velikih aplikacija. Sadrži sve nove tehnilogije kakve su XML i napredne web usluge, CORBA podršku, arhitekturu tri čvora i mnoge druge alate. Neka poglavlja se odnose samo na mogućnosti koje su dostupne u izdanju Delphi Enterprise; ti odeljci su jasno označeni.
- Verzija "Architect Studio" pored mogućnosti verzije Enterprise obuhvata podršku za Bold, okruženje za pravljenje aplikacija kojima se u vreme izvršavanja upravlja pomoću UML modela i koje su u stanju da svoje objekte mapiraju u baze podataka i svoj korisnički interfejs, zahvaljujući velikom broju naprednih komponenti. U ovoj knjizi se ne obrađuje Bold.

Pored toga što su dostupna različita izdanja, postoje i brojni načini prilagođavanja Delphi okruženja. U ilustracijama ekrana kroz knjigu pokušao sam da koristim standardni korisnički interfejs (onako kako izgleda posle instalacije); ipak, ja imam neke sklonosti, naravno, i instaliram mnoge dodatke koji mogu uticati na neke prikaze ekrana.

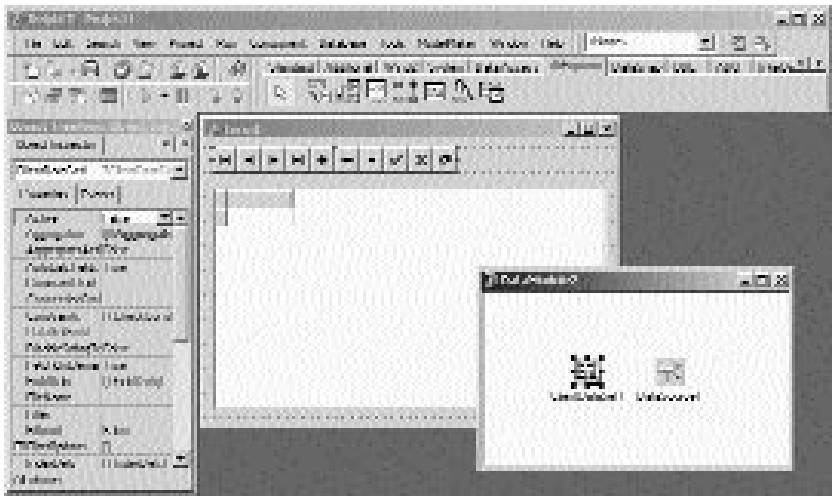
Verzija Professional i bolje verzije Delphija 7 obuhvataju kopiju Kylixa 3. U ovoj knjizi, osim objašnjenja biblioteke CLX i mogućnosti rada Delphija na različitim platformama, neće biti reči o Kylixu i programiranju pod Linuxom. Za više informacija potražite knjigu *Mastering Kylix 2* (Sybex 2002). (Između Kylixa 2 i Kylixa 3 nema mnogo razlika. Najvažnija nova karakteristika Kylixa 3 jeste podrška za programski jezik C++.)

Pregled IDE-a

Kada radite u vizuelnom razvojnom okruženju, vreme trošite u radu sa dva različita dela aplikacije: vizuelnim dizajnerima i editorom koda. Dizajneri Vam omogućavaju da radite sa komponentama na vizuelnom nivou (kao kada na formular smeštate kontrolu) ili na ne-vizuelnom nivou (kao kada komponentu DataSet smeštate u modul podataka). Formular i modul podataka vidite na slici 1.1. Dizajneri Vam u oba slučaja omogućavaju da odaberete komponente koje su Vam potrebne i da zadate početne vrednosti njihovih svojstava.

Editor koda je mesto gde pišete kod. Najočigledniji način za pisanje koda u vizuelnom okruženju jeste reagovanje na događaje, pri čemu se počinje od događaja koji su vezani za operacije koje obavljaju korisnici programa, kao što je korišćenje miša ili biranje elementa iz liste. Isti pristup možete upotrebiti kako biste obradili interne događaje, kao što su promene u bazi podataka ili obaveštenja operativnog sistema.

Pošto programeri tokom vremena sve više upoznaju Delphi, oni obično počinju pišući kod za obradu događaja pa zatim prave sopstvene klase i komponente, a na kraju često najveći deo vremena provode koristeći editor. Pošto se u ovoj knjizi obrađuje mnogo više od vizuelnog programiranja i pošto se pokušava da Vam se pomogne da savladate sve mogućnosti Delphija, kako budete čitali knjigu tako ćete nailaziti na sve više koda, a sve manje formulara.



SLIKA 1.1 *Formular i modul podataka u IDE-u Delphija 7.*

IDE za dve biblioteke

Veoma važna izmena se po prvi put pojavila u Delphiju 6. IDE Vam sada omogućava da radite sa dve različite vizuelne biblioteke: VCL (Visual Component Library) i CLX (Component Library for Cross-Platform). Kada napravite nov projekat, potrebno je samo da odaberete koju ćete od ove dve biblioteke koristiti, započinjući klasičan VCL Windows program izborom komande File➥New➥Application, dok CLX aplikaciju započinjete izborom komande File➥New➥CLX Application.

NAPOMENA

CLX je Delphijeva biblioteka za više platformi koja Vam omogućava da kod koji ste napisali možete kompajlirati pomoću Kylixa i pokrenuti pod Linuxom. Više o poređenju biblioteka VCL i CLX možete pročitati u Poglavlju 5, "Vizuelne kontrole". Korišćenje biblioteke CLX je veoma interesantno u Delphiju 7 jer se Delphijeva verzija jezika za Kylix dobija uz paket namenjen za Windows. ■

Prilikom pravljenja novog projekta ili otvaranja postojećeg, paleta komponenti se preuređuje tako da prikazuje samo kontrole koje se odnose na tekuću biblioteku (mada su mnoge od njih deljene). Kada koristite ne-vizuelne dizajnere (kakav je modul podataka), kartice palete komponenti (Component Palette) sadrže samo vizuelne komponente koje su sakrivene.

Podešavanja radne površine

Delphijev IDE omogućava programerima da ga prilagode na razne načine - tipično otvarajući veliki broj prozora, njihovim raspoređivanjem i dokiranjem jednih uz druge. Ipak, programerima je često potrebno da otvore jedan skup prozora prilikom dizajniranja, a drugi skup prozora prilikom debagovanja. Slično tome, programerima je, možda, potreban jedan izgled kada rade sa formularima, a potpuno drugačiji kada izrađuju komponente ili kod niskog nivoa kada koriste samo editor. Preuređenje IDE-a za svaki od ovih zadataka je dosadan posao.

Zbog ovakvih razloga, Delphi Vam omogućava da sačuvate raspored IDE prozora (ovaj raspored se naziva radna površina (*desktop*, odnosno Global Desktop kako bi se razlikovao od Project Desktopa)) pod nekim imenom kako biste ga kasnije lako obnovili. Takođe, neko od ovih uređenja možete načiniti osnovnim izgledom prilikom debagovanja tako da se automatski uspostavlja kada pokrenete debager. Sve ove funkcije su dostupne sa palete alata Desktops. Takođe, možete raditi sa izgledom radne površine koristeći meni View➥Desktops.

Informacije o izgledu radne površine se čuvaju u DST datotekama (koje se zapisuju u Delphijevom direktorijumu `bin`), koje su u osnovi INI datoteke. Sačuvana uređenja sadrže poziciju glavnog prozora, Project Managera, Alignment Palette, Object Inspectora (uključujući nove vrednosti svojstava kategorije), prozore editora (sa statusom Code Explorera i Message Viewa) i mnoge druge, kao i status dokiranja raznih prozora.

Evo malog dela DST datoteke koji bi trebalo da je lako razumljiv:

```
[Main Window]
Create=1
Visible=1
State=0
Left=0
Top=0
Width=1024
Height=105
ClientWidth=1016
ClientHeight=78

[ProjectManager]
Create=1
```

```
Visible=0
State=0
...
Dockable=1

[AlignmentPalette]
Create=1
Visible=0
...
```

Uređenje radne površine zanemaruje uređenje projekta, koje se zapisuje u DSK datoteku slične strukture. Ovim se prevazilazi problem prebacivanja projekta sa mašine na mašinu (ili između programera) i problem preuređenja prozora prema sklonostima. Delphi odvaja sklonosti prema korisniku i prema mašini od uređenja projekta da bi bolje podržao timski razvoj.

SAVET

Ukoliko pokrenete Delphi i ne možete videti formular ili druge prozore, savetujem Vam da proverite (ili uklonite) podešavanja radne površine. Ukoliko otvorite projekat koji ste dobili od nekog korisnika i pri tom ne možete da vidite neke prozore ili Vam se ne dopada izgled radne površine, onda ponovo učitajte Vaša globalna podešavanja radne površine ili uklonite DSK datoteku projekta. ■

Opcije za podešavanje okruženja

Nekoliko izmena se odnose na često korišćeni okvir za dijalog Environment Options. Stranice ovog okvira za dijalog su u Delphiju 6 bile preuređene pri čemu su opcije Form Designera premeštene sa stranice Preferences na novu stranicu Designer. U Delphiju 6 se pojavljuje nekoliko novih opcija i stranica:

- Stranica Preferences okvira za dijalog Environment Options sadrži polje za potvrdu pomoću koga se sprečava dokiranje Delphijevih prozora jednih uz druge.
- Stranica Environment Variables Vam omogućava da vidite promenljive sistemskog okruženja (kao što su standardne putanje i podešavanje operativnog sistema) i promenljive koje je definisao korisnik. Sjajna stvar je to što možete koristiti sistemske i sopstvene promenljive okruženja u svakom okviru za dijalog IDE-a - na primer, možete izbeći često korišćene putanje i zameniti ih promenljivom. Drugim rečima, promenljive okruženja se koriste slično promenljivoj `$DELPHI`, koja se odnosi na Delphijev osnovni direktorijum, ali ih može definisati korisnik.
- Na Internet stranici možete zadati ekstenzije datoteka koje će se koristiti za HTML i XML datoteke (uglavnom u radnom okruženju WebSnap), a svakoj ekstenziji možete pridružiti spoljašnji editor.

Par reči o menijima

Osnovna Delphijeva linija menija (koja u Delphiju 7 ima savremeniji izgled) je važan način interakcije sa IDE-om, iako ćete mnoge poslove verovatno obavljati pomoću tastaturnih prečica i kontekst menija. Linija menija se ne menja mnogo u odnosu na tekuće operacije: Potrebno je da kliknete desni taster miša kako biste dobili potpuni spisak operacija koje možete izvršiti u tekućem prozoru ili komponenti.

Linija menija se značajno može izmeniti u zavisnosti od alata i čarobnjaka koje ste instalirali. U Delphiju 7, ModelMaker ima sopstveni meni. Ostale menije ćete videti ako instalirate popularne dodatke kao što je Gexperts ili moje čarobnjake (za više detalja pogledajte Dodatak B, "Dodatni Delphijevi alati", i Dodatak A, "Dodatni Delphijevi alati autora"). Važan meni koji je u poslednjim verzijama Delphija pridodat jeste meni Window. U ovom meniju se prikazuju otvoreni prozori; u ranijim verzijama ste listu otvorenih prozora mogli da dobijete pomoću kombinacije tastera Alt+0 ili pomoću elementa menija View➥Window. Meni Window je zaista koristan jer se prozori često nalaze jedni iza drugih što otežava njihovo pronalaženje. Uređenje ovog menija možete kontrolisati pomoću podešavanja u Windows registry: potražite Delphijev podključ `Main Window` (u grani `HKEY_CURRENT_USER\Software\Borland\Delphi\7.0`). Za ključ Registrya se koristi stringovna vrednost (umesto Boolean vrednosti) gde '-1' i 'True' označavaju tačno, a '0' i 'False' netačno.

SAVET

U Delphiju 7, meni Window se završava novom komandom: Next Window. Ova komanda je naročito korisna u obliku prečice (kombinacija tastera za ovu komandu je Alt+End). Prelazak iz jednog u drugi prozor IDE nikada nije bio ovako jednostavan (bar ne bez dodatnih alata). ■

Okvir za dijalog Environment Options

Kao što sam već napomenuo, neka od podešavanja IDE-a zahtevaju direktne izmene u Registryu. Ja ću još nekoliko ovakvih podešavanja pomenuti u ovom poglavlju. Naravno, najuobičajenija podešavanja se lako mogu podesiti pomoću okvira za dijalog Environment Options do koga možete doći pomoću menija Tools kao i do okvira za dijalog Editor Options i Debugger Options. Većina podešavanja je prilično intuitivna i dobro objašnjena u datoteci Delphi Help. Na slici 1.2 vidite moja podešavanja na stranici References ovog okvira za dijalog.

SLIKA 1.2 *Stranica Preferences okvira za dijalog Environment Options.*

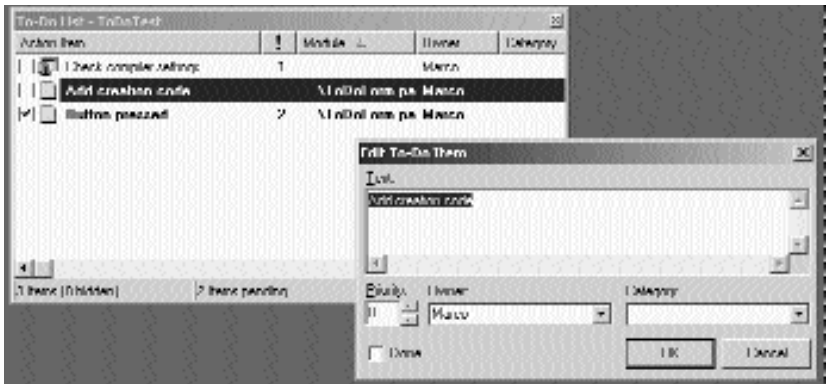
Spisak stvari koje treba uraditi

Još jedna funkcija koja je dodata u Delphi 5 IDE, a koja je još uvek veoma interesantna, jeste spisak stvari koje treba uraditi. To je spisak zadataka koje treba da uradite da biste kompletirali projekat - skup beležaka za programera (ili programere, jer ovaj alat može da bude veoma koristan u timskom radu). Mada ideja nije nova, ključni koncept spiska u Delphiju jeste to što se spisak ponaša kao dvosmerni alat.

Elemente spiska možete dodavati ili menjati dodavanjem specijalnih `TODO` komentara izvornom kodu bilo koje datoteke projekta; zatim ćete videti odgovarajuće stavke u listi. Pored toga, elemente liste možete i vizuelno menjati kako biste modifikovali odgovarajuće komentare izvornog koda. Na primer, evo kako element spiska može izgledati u izvornom kodu:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  // TODO -oMarco: Add creation code
end;
```

Isti element se može vizuelno menjati u prozoru prikazanom na slici 1.3. kao i u prozoru To-Do List.



SLIKA 1.3 *Prozor Edit To-Do Item se može koristiti za izmenu elementa liste što je operacija koju možete izvršiti direktno u izvornom kodu.*

Izuzetak od ovog dvosmernog pravila je definicija elemenata spiska koji se odnose na ceo projekat. Takve elemente morate uneti direktno u spisak. Da biste to učinili, možete upotrebiti kombinaciju tastera Ctrl+A u prozoru To-Do List ili možete kliknuti desnim tasterom miša u prozoru i odabrati Add iz kontekst menija. Ovi elementi se čuvaju u posebnoj datoteci čija je ekstenzija .TODO.

Postoji više opcija koje možete upotrebiti uz komentar `TODO`. Možete koristiti `-o` (kao u predhodnom delu koda) da biste naznačili vlasnika (programera koji je uneo komentar), opciju `-c` da biste naznačili kategoriju, ili jednostavno broj između 1 i 5 da biste naznačili prioritet (0 ili ukoliko nema broja, znači da nije određen nivo važnosti). Na primer, upotrebom komande Add To-Do Item iz kontekst menija editora (ili tastaturne prečice Ctrl+Shift+T) generiše se ovakav komentar:

```
{ TODO 2 -oMarco : Button pressed }
```

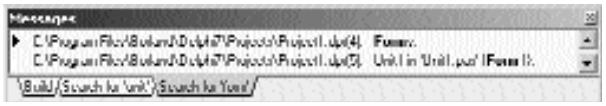
Delphi sve iza zareza, sve do kraja reda ili zatvorene zagrade, tretira prema tipu komentara kao tekst elementa spiska.

Konačno, u prozoru To-Do List možete potvrditi element da biste naznačili da je zadatak urađen. Komentar izvornog koda će se promeniti iz `TODO` u `DONE`. Takođe, možete ručno promeniti komentar u izvornom kodu da biste videli oznaku u prozoru To-Do List.

Jedan od najmoćnijih elemenata ovakve arhitekture je glavni prozor To-Do List, koji automatski može da prikupi informacije iz datoteka sa izvornim kodom dok ih unosite, da ih sortira i izveze na Cliboard kao običan tekst ili kao HTML tabelu. Sve ove opcije se mogu odabrati iz kontekst menija.

Proširene poruke kompajlera i rezultati pretraživanja u Delphiju 7

Ispod editora se, onako kako je unapred definisano, pojavljuje mali prozor Messages. U ovom prozoru se prikazuju poruke kompajlera i rezultati pretraživanja. Ovaj prozor je u Delphiju 7 značajno izmenjen. Prvo, rezultati pretraživanja se prikazuju na zasebnoj kartici tako da se ne mešaju sa porukama kompajlera kao što je to bilo ranije. Drugo, svaki put kada sprovedete novu pretragu, Vi od Delphija možete zatražiti da rezultat prikaže na novoj stranici tako da rezultati prethodne pretrage ostanu dostupni:



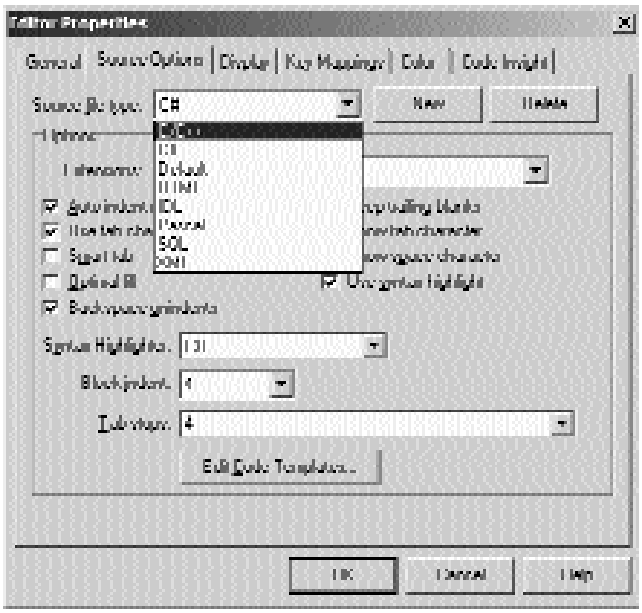
Za prelazak između kartica ovog prozora možete koristiti kombinacije tastera Alt+Page Down i Alt+Page Up. (Iste komande možete koristiti i za druge poglede u kojima postoje kartice.) Ako se prilikom kompajliranja dogodi greška, možete aktiviriati novi prozor upotrebom komande View➥Additional Message Info. Prilikom kompajliranja programa, prozor Message Hints će sadržati dodatne informacije o nekim uobičajenim greškama i ponudiće načine kako da ih ispravite:

Ovakav oblik pomoći je prvenstveno namenjen programerima početnicima, ali je korisno koristiti ovaj prozor. Veoma je važno da zapamtite da se ove informacije mogu potpuno prilagoditi: vođa projekta za uobičajene greške može napraviti opise koji će novopridošlim programerima biti od pomoći. Da biste to uradili, pratite komentare u datoteci u kojoj se nalaze podešavanja za ovu mogućnost Delphija, datoteku msginfo70.ini koja se nalazi u Delphijevom direktorijumu bin.

Delphijev editor

Na prvi pogled ne izgleda da je Delphijev editor značajno promenjen u verziji 7 IDE-a. Međutim, ako bolje pogledate, videćete da je to potpuno nov alat. Pored toga što ga možete koristiti u radu sa datotekama u programskom jeziku Object Pascal (odnosno Delphijevom jeziku kako Borland više voli da ga zove), sada ga možete koristiti u radu sa ostalim datotekama koje se koriste prilikom programiranja u Delphiju (kao što su SQL, XML, HTML i XSL datoteke), kao i pri radu sa datotekama drugih programskih jezika (uključujući C++ i C#). Edtiovanje XML i HTML datoteka je bilo moguće i u Delphiju 6, ali su izmene u ovoj verziji ipak značajne. Na primer, prilikom editovanja HTML datoteke imate podršku za isticanje sintakse i kompletiranje koda.

Podešavanja editora koja se koriste u radu sa datotekama (uključujući i ponašanje tastera kakav je taster Tab) zavise od ekstenzije datoteke koja se otvara. Ova podešavanja možete podesiti na novoj stranici Source Options u okviru za dijalog Editor Properties koji vidite na slici 1.4. Ova karakteristika je obogaćena i učinjena je prilagodljivijom tako da čak možete konfigurisati editor pravećI DTD za datoteke XML formata ili pravljenjem čarobnjaka koji ističu sintaksu za druge programske jezike. Druga karakteristika editora, šabloni koda, je sada karakteristična za svaki programski jezik (unapred napravljeni šabloni u Delphiju će imati malo smisla ako ih korstite za HTML i C#).



SLIKA 1.4 *Raznim programskim jezicima u Delphiju se mogu pridružiti različite ekstenzije datoteka na stranici Source Options okvira za dijalog Editor Properties.*

NAPOMENA

Programski jezik C# je nov programski jezik koji je Microsoft predstavio zajedno sa svojom .NET arhitekturom. Od Borlanda se očekuje da programski jezik C# podrži u sopstvenoj .NET arhitekturi koja se radno zove Galileo. ■

Ako u obzir uzmemo samo Delphijev jezik, editor koji je deo IDE se nije mnogo izmenio u odnosu na prethodne verzije. Međutim, postoje mnoge funkcije za koje mnogi Delphi programeri ne znaju i ne koriste ih, pa je mislim da ih vredi ukratko objasniti.

Delphijev editor Vam omogućava da istovremeno radite sa nekoliko datoteka pri čemu se koristi smulacija "beležnice sa više kartica". Sa jedne na drugu stranicu u editoru se može prelaziti ako upotrebite kombinaciju tastera Ctrl+Tab (ili Ctrl+Shift+Tab za pomeranje u suprotnom smeru). Kartice sa imenima jedinica možete prevlačiti u gornji deo editora kako biste promenili njihov redosled tako da u bilo kom trenutku možete samo jednom upotrebiti kombinaciju tastera Ctrl+Tab da biste prelazili iz jedne u drugu jedinicu. Kontekst meni editora sadrži komandu Pages u čijem se podmeniju prikazuju sve stranice (korisna karakteristika kada je učitan veliki broj jedinica).

Takođe, možete otvoriti više prozora editora i u svakom od njih imati više kartica. To je jedini način da kod dve jedinice prikažete jedan pored drugog. (Zapravo, kada želim da uporedim dve Delphijeve jedinice, ja obavezno koristim Beyond Compare - `www.scootersoftware.com` - odličan, jeftin program za poređenje datoteka koji je napisan u Delphiju.)

Nekoliko opcija utiču na editor što možete videti u okviru za dijalog Editor Properties na slici 1.4. Međutim, da biste podesili karakteristiku AutoSave, morate preći na stranicu Preferences okvira za dijalog Environment Options (pogledajte sliku 1.2). Ova opcija primorava editor da zapiše sve datoteke sa izvornim kodom svaki put kada pokrenete program čime se sprečava gubljenje podataka u (retkim) situacijama kada dođe do ozbiljne greške u debageru.

Delphijev editor ima veliki broj komandi, uključujući neke koje postoje još od WordStar emulacije (od vremena prvih Turbo Pascal kompajlera). Ja se neću baviti raznim podešavanjima editora jer su prilično inutitivna i opisana u ugrađenoj pomoći. Ipak, zapamtite, da stranicu na kojoj se opisuju tastaturne prečice možete dobiti u celosti samo ako u indeksu potražite shortcuts.

SAVET

Savet koji treba zapamtiti jeste da komande Cut i Paste nisu jedini način za prebacivanje izvornog koda. Reči, izraze i cele redove koda možete selektovati i prevlačiti. Pored toga, tekst možete kopirati, a ne samo prebacivati, ako prilikom prevlačenja držite pritisnut taster Ctrl. ■

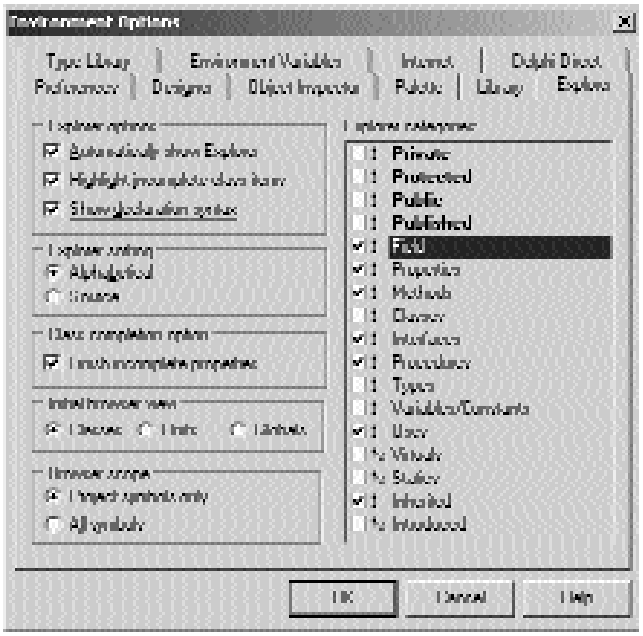
Code Explorer

Prozor Code Explorer, koji je, uopšte uzev, najkorisniji kada je priljubljen uz stranu editora, jednostavno prikazuje sve tipove, promenljive i rutine koje su definisane u okviru jedinice, plus i druge jedinice koje se prikazuju u iskazima `uses`. Za kompleksne tipove, kao što su klase, Code Explorer može prikazati spisak detaljnih informacija uključujući i spisak polja, svojstava i metoda. Sve informacije se ažuriraju čim počnete da unosite u prozor editora.

Da biste se kretali kroz editor, možete koristiti Code Explorer. Ukoliko dva puta kliknete na neku stavku u Code Exploreru, editor prelazi na odgovarajuću deklaraciju. Takođe, imena pomenljivih, svojstava i metoda možete menjati direktno u Code Exploreru. Međutim, ako prilikom rada sa sa klasama želite da koristite vizuelan alat, uvidećete da ModelMaker pruža daleko više mogućnosti.

Mada je sve ovo očigledno posle nekoliko minuta korišćenja Delphija, neke funkcije Code Explorera nisu tako intuitivne. Jedan važan aspekt jeste taj da imate potpunu kontrolu nad izgledom informacija i da možete ograničiti dubinu drveta koje se obično prikazuje u ovom prozoru

podešavanjem Code Explorera (smanjivanje drveta može pomoći bržem označavanju). Code Explorer možete konfigurisati upotrebom odgovarajuće stranice okvira za dijalog Environment Options, kao što možete videti na slici 1.5.



SLIKA 1.5 *Code Explorer možete konfigurisati u okviru za dijalog Evironment Options.*

Primetićete da, kada poništite neki od elemenata Explorer Categories na desnoj strani okvira za dijalog, Explorer ne uklanja odgovarajuće elemente iz pogleda - on jednostavno dodaje čvor na drvo. Na primer, ukoliko uklonite znak potvrde iz polja Uses, Delphi ne sakriva spisak upotrebljenih jedinica iz Code Explorera. Suprotno, upotrebljene jedinice su prikazane u spisku kao glavni čvorovi umesto da se čuvaju u direktorijumu `Uses`. Ja obično uklanjam znak potvrde iz polja Types, Classes i Variables/Constants.

Kako je svaki element drveta Code Explorer označen ikonom koja identifikuje njegov tip, uređenje po polju i metodu izgleda manje važno nego uređenje prema specifikatoru pristupa. Više volim da sve elemente prikažem u jednoj grupi jer to zahteva najmanju upotrebu miša kako bi se pristupilo nekom elementu. Označavanje elemenata u Code Exploreru obezbeđuje zgodan način kretanja kroz izvorni kod velike jedinice. Kada dva puta kliknete metod u Code Exploreru, prelazi se na definiciju u deklaraciji klase. Možete upotrebiti Module Navigation (kombinaciju tastera Ctrl+Shift i kursor tastera nagore ili nadole) kako biste prešli sa definicije metoda ili procedure iz interfejsa jedinice na potpunu definiciju u delu implementacije (ili ponovo natrag).

NAPOMENA

Neke od Explorer Categories koje možete videti na slici 1.5 koriste se u Project Browseru umesto u Code Exploreru. Ove kategorije obuhvataju, između ostalih, opcije grupisanja Virtuals, Statics, Inherited i Introduced. ■

Pretraživanje u editoru

Još jedna funkcija editora je Tooltip Symbol Insight. Pomerite pokazivač miša iznad simbola u editoru, a Tooltip će prikazati gde je deklarisan identifikator. Ova funkcija može da bude izuzetno značajna za praćenje identifikatora, klasa i funkcija u okviru aplikacije koju pišete, kao i referenca za izvorni kod biblioteke.

UPOZORENJE

Mada na prvi pogled može izgledati kao dobra ideja, Tooltip Symbol Insight ne možete koristiti da biste saznali koja jedinica deklariše identifikator koji želite da upotrebite. Ukoliko odgovarajuća jedinica nije već uključena, Tooltip se neće pojaviti. ■

Pravi bonus ove fukcije je ipak to što je možete pretvoriti u pomoć pri navigaciji koja se naziva pretraživanje koda. Kada držite pritisnut taster Ctrl i pomerite pokazivač miša iznad identifikatora, Delphi pravi aktivni link ka definiciji umesto da prikaže Tooltip. Ovi linkovi su prikazani plavom bojom i podvučeni su, što je tipično za web pretraživače, a pokazivač menja oblik u ruku kadgod se nađe iznad linka.

Na primer, možete pritisnuti taster Ctrl i kliknuti identifikator TLabel da biste otvorili definiciju u izvornom kodu VCL. Kako selektujete reference, editor pamti različite pozicije na koje ste skočili, pa se možete kretati unapred i unazad - ponovo kao u web pretraživaču - koristeći kontrole Browse Back i Browse Forward koje se nalaze u gornjem desnom uglu prozora editora ili kombinaciju tastera Alt+kursor taster u levo i Alt+kursor taster u desno. Takođe, možete kliknuti na strelice nadole pored kontrola Back i Forward da biste prikazali detaljan spisak redova izvornog koda na koje ste već skočili i da biste imali više kontrole nad kretanjem unapred i unazad.

Kako možete skočiti direktno u izvorni kod VCL ako nije deo Vašeg projekta? Editor može pronaći ne samo jedinice iz putanje Search (koje se kompajliraju kao deo Vašeg projekta), već i one koje se nalaze u putanjama Delphijevog Debug Sourcea, Browsinga i Librarya. Ovi direktorijumi se pretražuju po redosledu koji sam naveo, a možete ih odrediti na stranici Directories/Conditionals okvira za dijalog Project Options i na stranici Library okvira za dijalog Environment Options. Po definiciji, Delphi dodaje direktorijume izvornog koda VCL u Browsing putanju okruženja.

Class Completion

Delphijev editor Vam može pomoći generisanje izvornog koda kompletirajući pri tom ono što ste već napisali. Ova karakteristika se naziva *Class Completion*, a aktivira se upotrebom kombinacije tastera Ctrl+Shift+C. Dodavanje obrade događaja aplikaciji je brza operacija, jer Delphi u klasu automatski dodaje deklaraciju novog metoda za obradu događaja i obezbeđuje Vam strukturu metoda u implementacionom delu jedinice. Ovo je deo Delphijeve podrške vizuelnom programiranju.

Novije verzije Delphija pojednostavljuju život programerima koji dodaju kod u obrade događaja. Nove funkcije generisanja koda se odnose na opšte metode, metode rukovanja porukama i svojstva. Na primer, ukoliko sledeći kod unesete u deklaraciju klase:

```
public
   procedure Hello (MessageText: string);
```

i pritisnete kombinaciju tastera Ctrl+Shift+C, Delphi će Vam obezbediti definiciju metoda u implementacionom delu jedinice, generišući sledeće redove koda:

```
( Tform1 )
procedure Tform1. Hello(MessageText string);
begin
end;
```

Ovo je zaista korisno u poređenju sa tradicionalnim pristupom mnogih Delphijevih programera koji kopiraju jednu ili više deklaracija, dodaju nazive klasa i na kraju dupliraju kod `begin` … `end` za svaki kopirani metod. Class Completion može da funkcioniše i obrnuto. Možete napisati implementaciju metoda direktno njegovim kodom, a zatim pritisnuti kombinaciju tastera Ctrl+Shift+C kako biste generisali stavku u deklaraciji klase.

Najvažniji i najkorisniji primer kompletiranja klase jeste automatsko generisanje koda za svojstva koja su deklarisana u klasi. Na primer, ukoliko u klasi napišete

```
property Value: Integer;
```

i pritisnete kombinaciju tastera Ctrl+Shift+C onda će Delphi taj red pretvoriti u

property Value: Integer read fValue write SetValue;

Delphi će takođe u deklaraciju klase dodati metod `SetValue` i obezbediti njegovu osnovnu implementaciju. Više o svojstvima možete pročitati u narednom poglavlju.

Code Insight

Pored Code Explorera, Class Completiona i funkcija za kretanje, Delphi editor još uvek podržava tehnologiju Code Insight. Sve u svemu, tehnike Code Insight su zasnovane na stalnoj analizi sintakse u pozadini, kako izvornog koda koji pišete, tako i izvornog koda sistemskih jedinica na koji se refereiše Vaš izvorni kod.

Code Insight se sastoji iz pet delova: kompletiranja koda, šablona koda, parametara koda, izračunavanja izraza Tooltip i intuitivnih Tooltip simbola. Poslednja od ovih karakteristika je već obrađena u odelljku "Kretanje kroz editor"; preostale četiri karakteristike će se razmatrati u narednim pododeljcima. Možete aktivirati, deaktivirati i konfigurisati svaku od ovih karakteristika na stranici Code Insight okvira za dijalog Editor Properties.

Kompletiranje koda

Code Completion Vam omogućava da odaberete svojstvo ili metod objekta tako što ćete jednostavno potražiti u spisku ili unošenjem početnih slova. Da biste aktivirali ovu listu, treba samo da unesete naziv objekta, recimo Button1, zatim dodate tačku i sačekate. Da biste primorali program da prikaže listu, pritisnite kombinaciju tastera Ctrl+razmak; da biste uklonili prikaz kada ga ne želite, pritisnite taster Esc. Code Completion Vam takođe omogućava da pogledate odgovarajuću vrednost u iskazu dodele.

Prilikom unošenja karaktera, sadržaj liste se filtrira prema početnom delu elementa koji unosite. U listi za kompletiranje koda koriste se boje, a prikazuje se i više detalja da bi se napravila razlika između različitih elemenata. Na stranici Code Insight okvira za dijalog Editor Options možete odabrati boje koje želite da koristite. Druga karakteristika jeste da se u slučaju funkcija sa parametrima zagrade unose u kod koji se generiše, a oblačić sa parametrima se odmah prikazuje.

Kada iza promenljive ili svojstva unesete := , Delphi će prikazati spisak ostalih promenljivih ili objekata istog tipa, kao i objekte koji sadrže svojstva tog tipa. Dok je spisak prikazan, možete ga kliknuti desnim tasterom miša kako biste promenili redosled elemenata, sortirajući ih prema oblasti delovanja ili prema nazivu, a možete promeniti i veličinu prozora.

Od Delphija 6, kompletiranje koda radi i u interfejs odeljku jedinice. Ukoliko pritisnete kombinaciju tastera Ctrl+razmak kada se kursor nalazi unutar definicije klase, prikazaće se spisak virtuelnih metoda koje možete zaobići (uključujući apstraktne metode), metode implementiranih interfejsa, svojstva osnovne klase, a možda i sistemske poruke koje možete obraditi. Biranjem nekog od elemenata iz spiska dodaćete odgovarajući metod deklaraciji klase. U ovom slučaju kompletiranje koda dozvoljava da odaberete više od jednog elementa.

SAVET

Kada kod koji ste napisali nije korektan, Code Insight neće funkcionisati i možda ćete videti samo opštu poruku o grešci kojom je identifikovana takva situacija. Moguće je prikazati specifične greške Code Insight u panelu Message (koji mora biti već otvoren - ne otvara se automatski da bi prikazao greške prilikom kompajliranja). Da biste aktivirali ovu funkciju, potrebno je da podesite još jednu nedokumentovanu stavku u Registryu, podešavajući vrednost stringovnog ključa `Delphi\7.0\Compiling\ShowCodeInsiteErrors` u vrednost "1". ■

Postoje napredne mogućnosti kompletiranja koda koje nije lako uočiti. Jedna od njih, za koju nalazim da je vanredno korisna, odnosi se na otkrivanje simbola u jedinicama koje ne koristi projekat na kome radite. Kada pokrenete kompletiranje koda (to jest, kada pritisnete kombinaciju tastera Ctrl+razmak) kada se kursor nalazi u praznom redu, u spisku se prikazuju simboli koji se nalaze u standardnim jedinicama, (kakve su Math, StrUtils i DateUtils) koje nisu uključene u iskazima `uses` tekuće jedinice. Biranjem jednog od ovih spoljašnjih simbola, Delphi umesto Vas dodaje jedinicu u iskaz `uses`. Ova mogućnost (koja inače ne radi unutar izraza) je obezbeđena listom dodatnih jedinica koja se može prilagoditi, a koja se čuva u ključu Registrya `\Delphi\7.0\CodeCompletion\ExtraUnits`.

SAVAT

Delphi 7 ima mogućnost da pretražuje deklaracije elmenata u listi kompletiranja koda kada za bilo koji identifikator iz liste upotrebite taster Ctrl i kliknete na identifikator. ■

Šabloni koda

Ova karakteristika Vam omogućava da umetnete jedan od unapred definisanih šablona koda, kao što su, recimo, složeni iskazi u kojima postoji unutrašnji blok `begin…end`. Šabloni koda se moraju ručno aktivirati tako što ćete upotrebiti kombinaciju tastera Ctrl+J da bi se prikazao spisak svih šablona. Ukoliko unesete nekoliko slova (recimo ključnu reč) pre nego što pritisnete kombinaciju tastera Ctrl+J, Delphi će prikazati spisak samo onih šablona koji počinju tim slovima.

Vi možete dodati sopstvene šablone koda i možete napraviti prečice za često korišćene blokove koda. Na primer, ukoliko često koristite funkciju `MessageDlg`, možda želite da za nju dodate šablon. Da biste izmenili šablone, pređite na stranicu Source Options okvira za dijalog Editor Options, iz liste Source File Type odaberite Pascal i kliknite Edit Code Templates. Kada to uradite, otvoriće se novi Delphi 7 okvir za dijalog Code Templates. Tada kliknite Add kako biste uneli ime novog šablona (recimo, **mess**), unesite deklaraciju, a zatim unesite sledeći tekst u telo šablona za kontrolu Code Memo:

```
MessageDlg ('/', mtInformation, [mbOK], 0);
```

Sada, svaki put kada imate potrebu da napravite dijalog za poruku, jednostavno unesite **mess** i pritisnite kombinaciju tastera Ctrl+J i dobićete ceo tekst. Vertikalna linija (odnosno, pipe) označava poziciju u okviru izvornog koda gde će se nalaziti kursor posle dodavanja šablona. Potrebno je da odaberete poziciju tako da to bude pozicija na kojoj ćete početi unos, da biste kompletirali kod koji generiše šablon.

Mada na prvi pogled može izgledati da šabloni koda previše zavise od ključnih reči jezika, oni su zapravo opštiji mehanizam. Šabloni se čuvaju u datoteci `DELPHI32.DCI`, tekstualnoj datoteci jednostavnog formata koju možete direktno menjati. Delphi 7 Vam omogućava da izvezete podešavanja jezika u datoteku i da ih uvezete čime programerima olakšava razmenu prilagođenih šablona.

Parametri koda

Prilikom unošenja funkcije ili metoda, parametri koda prikazuju tip podataka parametara funkcije ili metoda u oblačiću ili prozoru Tooltip. Unesite naziv funkcije ili metoda i otvorite (levu) zagradu i odmah će se prikazati nazivi parametara i njihovi tipovi. Da biste primorali program da prikaže parametre koda, možete upotrebiti kombinaciju tastera Ctrl+Shift+razmak. Kao dodatna pomoć, tekući parametar je prikazan masnim slovima.

Tooltip Expression Evaluation

Tooltip Expression Evaluation je funkcija koja je aktivna prilikom otklanjanja grešaka. Prikazuje Vam vrednost identifikatora, svojstva ili izraza koji se nalaze ispod pokazivača miša. Ako je u pitanju izraz, onda je obično neophodno da ga u editoru selektujete pa da zatim pomerite pokazivač miša iznad označenog teksta.

Još tastaturnih prečica editora

U editoru postoji još mnogo tastaturnih prečica koje zavise od stila editora koji ste odabrali. Evo nekih manje poznatih tastaturnih prečica:

- Ctrl+Shift i neki taster sa brojem između 0 i 9 aktivira oznaku (bookmark) koja je naznačena na margini na strani editora. Da biste se vratili na oznaku, možete pritisnuti Ctrl i taster na kojem je broj. Upotrebljivost oznaka u editoru je ograničena činjenicom da nova oznaka može prepisati oznaku koja nije stalna (oznake se gube kada zatvorite datoteku).
- Ctrl+E aktivira pretraživanje uvećavanjem. Možete upotrebiti kombinaciju tastera Ctrl+E i uneti reč koju želite da pronađete, a da ne morate da koristite specijalni okvir za dijalog i kliknete Enter da biste obavili pretraživanje.

- Ctrl+Shift+I uvlači više linija koda odjednom. Broj razmaka koji se koristi jeste onaj koji je određen opcijom Block Indent na stranici Editor okvira za dijalog Editor Options. Ctrl+Shift+U je odgovarajuća kombinacija kojom se poništava uvlačenje koda.
- Ctrl+O+U menja velika u mala slova koda i obrnuto; možete, takođe, upotrebiti kombinaciju tastera Ctrl+K+E da biste velika slova promenili u mala, a kombinaciju Ctrl+K+F da biste mala slova pretvorili u velika.
- Ctrl+Shift+R započinje snimanje makroa, koji kasnije možete upotrebiti ukoliko pritisnete kombinaciju tastera Ctrl+Shift+P. Makro snima sve što otkucate, pomeranja i uklanjanja u datoteci sa izvornim kodom. Upotreba makroa samo ponavlja sekvencu - operacija koja nema nekog značaja kada pređete u drugu datoteku izvornog koda. Makroi editora su prilično korisni prilikom ponovnog obavljanja operacija koje imaju više koraka, kakve su ponovno formatiranje izvornog koda ili povećanje njegove čitljivosti.
- Dok držite pritisnut taster Alt, mišem možete označiti četvorougaone oblasti editora, ne samo uzastopne redove i reči.

Pogledi koji mogu da se učitavaju

Još jedna važna karakteristika koja je uvedena u Delphi 6 jeste podrška za više pogleda u editoru. Kada se pojedinčno učita bilo koja datoteka u IDE, editor sada može da prikaže više pogleda te datoteke, a ti pogledi se mogu programski definisati i pridružiti sistemu, pa se zatim mogu učitati da određene datoteke - odatle i ime pogledi koji se mogu učitavati.

Najčešće korišćen pogled jeste Diagram, koji je bio na raspolaganju za module podataka u Delphiju 5, iako je imao manje mogućnosti. Drugi skup pogleda je na raspolaganju za web aplikacije, uključujući HTML Script pogled, HTML Result pogled i moge druge koje ću razmatrati u Poglavljima 20 ("Web programiranje pomoću WebBrokera i WebSnapa") i 22 ("Korišćenje XML tehnologija"). Možete koristiti kombinacije tastera Alt+Page Down i Alt+Page Up kako biste prelazili sa kartice na karticu koje se nalaze u dnu ovog editora. Kombinacija tastera Ctrl+Tab menja stranice (ili datoteke) koje se prikazuju na gornjim karticama.

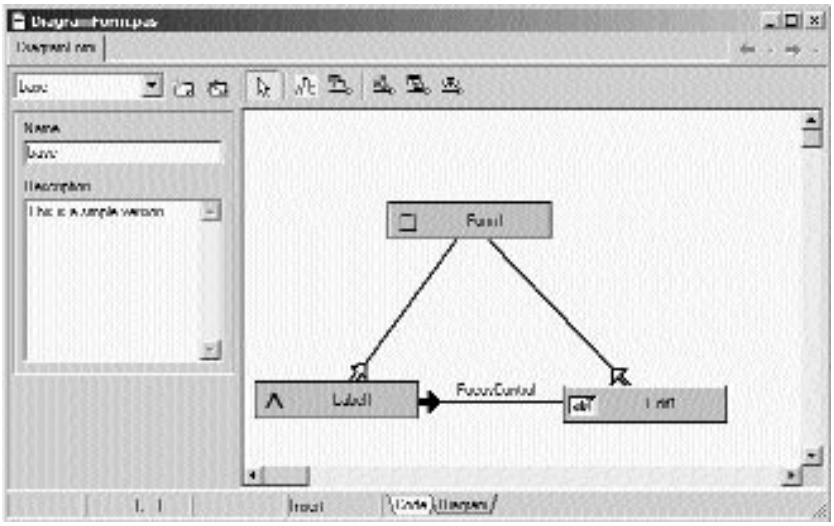
Pogled Diagram View

Pogled Diagram View prikazuje zavisnosti među komponentama, uključujući zavisnoti roditelj/dete, zavisnosti vlasništva, povezane zavisnosti i generičke zavisnosti. Za komponente skupa podataka podržava zavisnosti master/detail i uspostavljenih veza. Pomoću blokova teksta možete čak uneti sopstvene komentare koji se odnose na određene komponente.

Pogled Diagram se ne pravi automatski. Prvo u dijagram morate prevući komponente iz pogleda TreeView, koji će zatim automatski prikazati postojeće veze između komponenti koje ste prevukli. Možete odabrati više elementa u pogledu Object TreeView i prevući ih odjednom na neku od stranica pogleda Diagram.

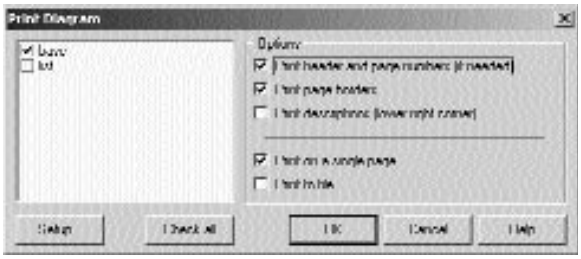
Ono što je zgodno jeste to što možete odredtiti vrednosti svojstava iscrtavanjem strelica između komponenti. Na primer, nakon što ste u pogled Diagram preneli komponente Edit i Label, možete odabrati ikonu Property Connector, kliknuti komponentu Label i prevući pokazivač miša iznad komponente Edit. Kada otpustite taster miša, pogled Diagram će uspostaviti zavisnost

između svojstava na osnovu svojstva FocusControl, jedinog svojstva komponente Label koje se odnosi na komponentu Edit. Ovakva situacija je prikazana na slici 1.6.



SLIKA 1.6 *Pogled Diagram prikazuje zavisnosti između komponenti (a omogućava Vam čak i da te zavisnosti uspostavite).*

Kao što se može videti, određivanje vrednosti svojstava ima svoj *smer*: Ukoliko prilikom određivanja zavisnosti svojstava prevučete pokazivač miša od komponente Edit do komponente Label, rezultat je pokušaj da komponentu Label upotrebite kao vrednost svojstva komponente Edit. Kako ovo nije moguće, prikazaće se poruka o grešci kojom se opsuje nastali problem i ponuda da se uspostavi veza između komponenti, ali u obrnutom smeru. Pogled diagram Vam omougćava da napravite više dijagrama za svaku od jedinica Delphija - to jest, za svaki formular ili modul podataka. Imenujte dijagram i ako želite dodajte opis, kliknite kontrolu New Diagram, pripremite novi dijagram, pa ćete moći da prelazite sa dijagrama na dijagram pomoću kontrole koja se nalazi na paleti alata pogleda Diagram. Mada pogled Diagram možete korisiti za uspostavljanje zavisnosti, njegova osnovna namena je dokumentovanje Vašeg dizajna. Zbog ovoga je veoma važno imati mogućnost štampanja sadržaja ovog pogleda. Kada upotrebite standardnu komandu File➥Print dok je pogled Diagram aktivan, Delphi će Vam ponuditi razne opcije, kao što se to može videti na slici 1.7, omogućavajući Vam da na taj način prilagodite štampanje.



SLIKA 1.7 *Dijalog Print Options koji se prikazuje za pogled Diagram.*

Informacije o pogledu Diagram View se čuvaju u zasebnoj datoteci, a ne kao deo DFM datoteke. Delphi 5 su koristili datoteke sa informacijama u vreme dizajniranja (DTI), koje su imale strukturu sličnu datotekama INI. Delphi 6 i 7 mogu da pročitaju stari .DTI format, ali koriste nov Delphi Diagram Portfolio format (`.DDP`). Izgleda da se ove datoteke zapisuju u DFM binarnom formatu (ili nekom sličnom) pa ih ne možete editovati kao tekstualne datoteke. Očigledno je da su sve ove datoteke beskorisne u vreme izvršavanja (nema nikakvog razloga da ih uključite u kompajliranje izvršne datoteke).

NAPOMENA

Ako želite da eksperimentišete sa pogledom Diagram View, možete početi tako što ćete otvoriti projekat DiagramDemo koji se nalazi u primerima ovog poglavlja. Formular programa ima dva pridružena dijagrama: jedan vidite na slici 1.6 i jedan složeniji sa menijima i elementima menija. ■

Form Designer

Još jedan prozor Delphija, u kojem ćete često raditi, jeste Form Designer, vizuelni alat koji Vam pomaže da smestite komponente na formular. U Form Designeru možete odabrati komponentu pomoću miša; takođe, možete koristiti Object Inspector ili Object TreeView što je zgodno kada se kontrola nalazi ispod neke druge kontrole ili je kontrola mala. Ukoliko jedna kontrola u potpunosti prekriva drugu, možete pritisnuti taster Esc da biste selektovali roditeljsku kontrolu koja je selektovana. Taster Esc možete pritisnuti jednom ili više puta kako biste selektovali formular ili pritisnuti i držati pritisnut taster Shift kada kliknete selektovanu komponentu. Na ovaj način ćete iz selekcije ukloniti komponentu i po definiciji selektujete formular.

Postoje dve mogućnosti pri upotrebi miša za određivanje pozicije komponente. Možete zadati vrednosti za svojstva `Left` i `Top` ili možete upotrebiti kursor-tastere dok držite pritisnut taster Ctrl. Upotreba kursor-tastera je naročito korisna za fino pozicioniranje elemenata (kada je aktivna opcija Snap to Grid) kao kada je pritisnut taaster Alt dok miša koristite za pomeranje komponente. Ukoliko pritisnete kombinaciju tastera Ctrl+Shift i neki od kursor-tastera, komponenta će se pomeriti za veličinu mreže.

Kada koristite kursor-tastere dok držite pritisnut taster Shift, možete fino podesiti veličinu komponente. Ponoviću, isto možete uraditi ukoliko držite pritisnut taster Alt i koristite miša.

Da biste poravnali više komponenti, ili da biste im dodelili jednaku veličinu, možete selektovati nekoliko komponenti i podesiti svojstva `Top`, `Left`, `Width` ili `Height` za sve komponente odjednom. Da biste selektovali nekoliko komponenti, komponente možete kliknuti mišem dok držite pritisnut taster Shift ili, ukoliko sve komponente mogu stati u četvorougaonu oblast, možete prevući mišem da biste "nacrtali" četvorougao koji ih obuhvata. Da biste selektovali dete-kontrole (recimo kontrole koje se nalaze u panelu) obuhvatite ih miše dok držite pritisnut taster Ctrl inače ćete pomeriti panel. Kada ste selektovali više komponenti, možete odrediti njihove relativne pozicije upotrebivši okvir za dijalog Alignment (upotrebom komande Align iz kontekst menija formulara) ili paletu Alignment (kojoj možete pristupiti preko komande menija View➥Alignment Pallete).

Kada završite dizajniranje formulara, možete upotrebiti komandu Lock Controls iz menija Edit da biste izbegli slučajnu promenu pozicije komponente na formularu. Ova komanda je naročito korisna jer zapravo ne postoji prava operacija Undo za formulare (samo Undelete), ali vrednosti nisu nepromenljive.

Među ostalim funkcijama, Form Designer nudi nekoliko vrsta saveta Tooltip:

- Kada pomerite pokazivač iznad komponente, savet će prikazati naziv i tip komponente. Počev od verzije 6, Delphi nudi proširene savete, sa detaljima koji se odnose na poziciju kontrole, njenu veličinu, redosled i drugo. Ovo je dodatak podešavanju okruženja Show Component Captions, koje je kod mene uvek aktivno.
- Kada promenite veličinu kontrole, pomoć prikazuje aktuelnu veličinu (svojstva `Width` i `Height`). Naravno, ova funkcija postoji samo za kontrole, ne za nevizuelne komponente (koje su u Form Designeru naznačene ikonama).
- Kada pomerite komponentu, pomoć prikazuje aktuelnu poziciju (svojstva `Left` i `Top`).

Na kraju, DFM (Delphi Form Module) datoteke možete sačuvati u starom binarnom formatu umesto da ih sačuvate kao tekstualne datoteke, što je inače unapred zadato. Ovu opciju možete uključiti ili isključiti za svaki formular ponaosob upotrebom kontekst menija Form Designera ili možete odrediti vrednost za sve novonapravljene formulare na stranici Designer okvira za dijalog Environment Options. Na istoj stranici možete, takođe, odrediti da li će se sekundarni formulari programa automatski praviti prilikom pokretanja, odluku uvek možete poništiti za pojedine formulare (koristeći stranicu Forms okvira za dijalog Project Options).

Mogućnost čuvanja DFM datoteka kao tekstualnih datoteka Vam omogućava bolje operisanje sistemima kontrole verzija. Programeri neće dobiti stvarnu prednost ovom funkcijom jer ste i ranije mogli da otvorite binarnu DFM datoteku u Delphi editoru koristeći kontekst meni dizajnera. Sistemima kontrole vezija je, s druge strane, potrebno da sačuvaju tekstualnu verziju DFM datoteka da bi mogli da ih uporede i pronađu razlike između dve verzije iste datoteke. U svakom slučaju, zapamtite da ukoliko koristite DFM datoteke kao tekstualne, Delphi će ih ipak konvertovati u binarni format resursa pre nego što ih uključi u izvršnu datoteku Vašeg programa. DFM datoteke se povezuju u izvršne programe u binarnom formatu kako bi se smanjila veličina izvršne datoteke (mada se zapravo ne kompresuju) i da bi se poboljšale performanse prilikom izvršavanja (mogu se brže učitati).

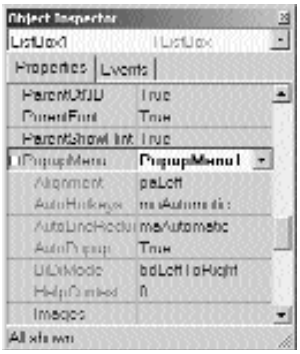
NAPOMENA

Tekstualne DFM datoteke se lakše prebacuju iz jedne verzije Delphija u drugu nego njihove binarne verzije. Mada starije verzije Delphija možda neće prihvatiti novo svojstvo kontrole koje je navedeno u DFM datoteci koja je napravljena u novijoj verziji Delphija, starija verzija će ipak moći da pročita ostatak DFM datoteke. Ukoliko pomoću novije verzije Delphija dodate nov tip podataka, starije verzije Delphija neće moći da pročitaju binarnu DFM datoteku. Mada sve ovo ne zvuči uverljivo, 64-bitni sistemi su bliska budućnost. Svaki put kada se dvoumite, DFM datoteku sačuvajte u tekstualnom formatu. Takođe, zapamtite da sve verzije Delphija podržavaju tekstualne DFM datoteke, koristeći alat Convert koji se nalazi u direktorijumu bin. Na kraju, zapamtite da biblioteka CLX koristi ekstenziju XFM umesto ekstenzije DFM, kako u Delphiju tako i u Kylixu. ■

Object Inspector

Da biste videli i menjali svojstva komponenti koje se nalaze na formularu (ili drugom dizajneru) u vreme dizajniranja, koristite Object Inspector. U poređenju sa ranim verzijama Delphija, Object Inspector ima veliki broj novih osobina. Poslednja od njih, koja je predstavljena u Delphiju 7, jeste korišćenje masnih slova za isticanje svojstava koja imaju vrednost koja se razlikuje od unapred zadate vrednosti.

Druga važna izmena (predstavljena u Delphiju 6) jeste mogućnost Object Inspectora da prikaže reference komponente. Svojstva koja se odnose na druge komponente sada se prikazuju drugom bojom i mogu se prikazati kada kliknete znak + koji se nalazi na levoj strani, kao što je to slučaj sa internim podkomponentama. Tada možete promeniti svojstva te druge komponente, a da ne morate da je označite. Na slici možete videti povezanu komponentu (kontekst meni) koji je prikazan u Object Inspectoru dok radite sa drugom komponentom (listom):



Ova osobina interfejsa se odnosi i na podkomponente što je pokazano pomoću kontrole LabelEdit. Odgovarajuća osobina Object Inspectora Vam omogućava da selektujete komponentu na koju se odnosi svojstvo. Da biste to učinili levim tasterom miša dva puta kliknite vrednost svojstva dok pri tom držite pritisnut taster Ctrl. Na primer, ako se na formularu nalazi komponenta MainMenu, a Vi u Object Inspectoru proveravate svojstva, komponentu MainMenu možete selektovati tako što ćete preći na svojstvo Menu formulara i dva puta kliknuti njegovu vrednost dok držite pritisnut taster Ctrl. Kada to uradite, Vi ćete selektovati glavni meni koji je naznačen vrednošću svojstva u Object Inspectoru.

Evo još nekih značajnih izmena Object Inspectora:

- Lista pri vrhu Object Inspectora prikazuje tip objekta i omogućava Vam da odaberete komponentu. Ovu listu možete ukloniti radi dobijanja prostora, imajući na umu da komponente možete selektovati u Object TreeViewu (koji se nalazi iznad prozora Object Inspector).
- Svojstva koja se pozivaju na objekat sada se prikazuju drugom bojom i mogu se prikazati, a da se ne mora promeniti selektovani objekat.
- Ukoliko izaberete, u Object Inspectoru možete prikazati svojstva koja se mogu samo čitati. Naravno, ova svojstva su prikazana sivom bojom.
- Object Inspector sadrži novi okvir za dijalog Properties, koji Vam omogućava da za različite tipove svojstava odaberete boju i odredite ponašanje ovog prozora.
- Od Delphija 5, lista za svojstvo može sadržati grafičke elemente. Ova mogućnost se koristi za svojstva kakva su Color i Cursor, a naročito je korisna za svojstvo `ImageIndex` komponenti koje su povezane sa komponentom ImageList.

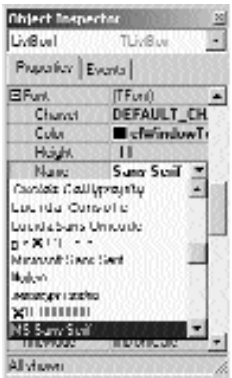
NAPOMENA

Svojstva interfejsa se sada pomoću Object Inspectora mogu konfigurisati u vreme dizajniranja. Ovo omogućava upotrebu Interfaced Component Reference modela koji je predstavljen u Kylixu/Delphi 6, gde komponente mogu implementirati i sadržati reference na interfejse ukoliko su interfejsi implementirani komponentama. Interfaced Component Reference rade kao obične stare reference, izuzev što se svojstva interfejsa mogu vezati za bilo koju komponentu koja implementira neophodni interfejs. Za razliku od svojstava komponente, svojstva interfejsa nisu ograničena na određeni tip komponente (klasu ili izvedenu klase). Kada kliknete listu u Object Inspector editoru za određeno svojstvo, sve komponente tekućeg formulara (i povezanih formulara) koje implementiraju taj interfejs se prikazuju. ■

LISTA FONTOVA U OBJECT INSPECTORU

Delphijev Object Inspector sadrži grafičku listu za nekoliko svojstava. Možda želite da dodate onu koja prikazuje aktuelnu sliku fonta koji selektujete, prema za podsvojstvu `Name` svojstva Font. Ova mogućnost je, zapravo, ugrađena u Delphi, ali je isključena jer je na većini računara instaliran veliki broj fontova i njihovo renderovanje može u mnogome usporiti Vaš računar. Ukoliko želite da uključite ovu funkciju, potrebno je da u Delphi instalirate paket koji omogućava globalnu promenljivu `FontNamePropertyDisplayFontNames` nove jedinice VCLEditors. Ja sam to učinio u paketu OiFont Pk, koji možete pronaći među primerima programa ovog poglavlja.

Kada je paket instaliran, možete preći na svojstvo `Font` bilo koje komponente i upotrebiti grafički meni Name, kao što je ovde prikazano:



Postoji i drugo, složenije prilagođavanje Object Inspectora koje se meni dopada i koje često koristim: font za ceo Object Inspector, da bi tekst bio čitljiviji. Ova funkcija je naročito korisna za javne prezentacije. Pogledajte Dodatak A da biste saznali kako da dođete do ovog paketa.

Kategorije svojstava

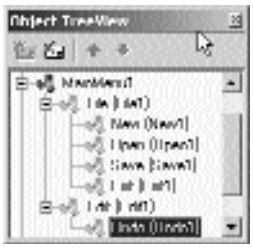
Delphi obuhvata ideju kategorija svojstava, koje se aktiviraju opcijom Arrange, a koja se nalazi u lokalnom meniju Object Inspectora. Ukoliko uključite ovu opciju, svojstva neće biti prikazana u abecednom poretku već će biti grupisana, a neka od svojstava će se verovatno nalaziti u više grupa. Kategorije imaju prednost smanjenja složenosti Object Inspectora. Možete upotrebiti podmeni View iz kontekst menija da sakrijete svojstva datih kategorija, bez obzira na način na koji su prikazana (dakle, iako više volite tradicionalni prikaz i uređenje po nazivima, još uvek možete da sakrijete svojstva nekih kategorija). Iako su kategorije svojstva mogle da se koriste još od Delphija 5, programeri su ih retko koristili.

Pogled Object TreeView

U Delphiju 5 predstavljen je pogled TreeView za module podataka u kojem ste mogli da vidite zavisnosti između ne-vizuelnih komponenti, kakvi su skupovi podataka, polja, akcije i tako dalje. Delphi 6 ovu ideju još više unapređuje obezbeđujući pogled Object TreeView za svaki od dizajnera, uključujući i formulare. Unapred je određeno da se pogled Object TreeView prikazuje iznad Object Inspectora. Paleta Object TreeView prikazuje sve komponente i objekte koji se nalaze na formularu u obliku drveta, prikazujući njihove zavisnosti. Najočiglednija je zavisnost roditelj/dete: ukoliko na formular postavite panel, komandu na panel i komandu van panela, u drvetu će se prikazati po jedna komanda za formular i panel:

Primetićete da je TreeView sinhronizovan sa Object Inspectorom i Form Designerom. Dakle, ukoliko odaberete element i promenite fokus u bilo kojm od ova tri alata, fokus se menja i u preostala dva.

Pored zavisnosti roditelj/dete, pogled Object TreeView prikazuje i druge zavisnosti, kao što su vlasnik/vlasništvo, komponenta/podobjekat, kolekcija/element, kao i različite specifične zavisnosti uključujući skup podataka/uspostavljena veza i izvor podataka/zavisnosti skupa podataka. Ovde prikazujem primer strukture menija:



Ponekad TreeView prikazuje "lažne" čvorove koji ne odgovaraju stvarnim objektima već odgovaraju objektima koji su unapred određeni. Primer ovakvog ponašanja možete videti kada prevučete komponentu Table (koja se nalazi na stranici BDE) jer će se tada prikazati dve sive ikone koje predstavljaju sesiju i alias. Dakle, pogled Object TreeView sivim ikonama prikazuje komponente koje nemaju funkciju u vreme dizajniranja. Ove komponente zaista postoje (i u vreme dizajniranja i u vreme izvršavanja), ali kako su to unapred određeni objekti, oni se prave u vreme izvršavanja, te nemaju podatke koji se mogu menjati u vreme dizajniranja, pa vam Data Module Designer ne dozvoljava da izmenite njihova svojstva. Ukoliko na formular prevučete komponentu Table, prikazaće se elementi koji su označeni crvenim upitnikom koji se nalazi unutar žutog kruga. Ovim simbolom se označavaju delimično nedefinisani elementi.

Paleta Object TreeView podržava različite tipove *prevlačenja*:

- Sa palete možete odabrati komponentu (tako što ćete je kliknuti, a ne zaista prevući), zatim pomeriti pokazivač miša iznad drveta, pa kliknuti komponentu da biste je prebacili na željeno mesto. Na ovaj način Vam je omogućeno da komponentu prebacite u odgovarajući kontejner (formular, panel ili neki drugi) bez obzira na činjenicu da je površina kontejnera pretrpana drugim komponentama, čime se sprečava da komponentu prebacite u dizajner, a da predhodno ne preuredite komponente koje se već nalaze u kontejneru.
- Komponente možete prevlačiti unutar pogleda Object TreeView - recimo, možete da premeštate komponente iz jednog kontejnera u drugi. Pomoću Form Designera to možete obaviti samo operacijama Cut i Paste. Prednost premeštanja nad isecanjem jeste u tome da ukoliko imate uspostavljene veze između komponenti te veze neće biti izgubljene, kao što je to slučaj kada komponentu uklonite tokom operacije isecanja.
- Možete prevlačiti komponente iz pogleda Object TreeView u pogled Dijagram, što ćete kasnije videti.

Kada bilo koji element pogleda Object TreeView kliknete desnim tasterom miša, prikazaće se kontekst meni sličan meniju komponente koji dobijate kada se komponenta koju ste kliknuli nalazi na formularu (u oba slučaja kontekst meni može sadržati elemente koji se odnose na editore). Možete čak ukloniti elemente iz drveta. TreeView može poslužiti kao editor kolekcija, kao što je to ovde pokazano za svojstvo `Columns` kontrole ListView. U ovom slučaju, ne samo da možete preurediti i ukloniti elemente, već možete dodati nove elemente kolekciji.

SAVET

Sadržaj pogleda Object TreeView možete odštampati ukoliko Vam je potrebna dokumentcija. Aktivirajte prozor pogleda i upotrebite komandu File➥Print (jer kontekst meni ne sadrži komandu Print). ■

Tajne palete Component Palette

Paleta Component Palette se koristi za biranje komponenti koje želite da smestite u tekući dizajner. Da biste saznali ime komponente, pomerite pokazivač miša iznad nje. U Delphiju 7 se u oblačiću prikazuje i ime jedinice u kojoj je komponenta definisana. Paleta Component Palete ima veliki broj kartica - zaista previše. Kartice na kojima se nalaze komponente koje ne planirate da koristite možete sakriti i reorganizovati paletu Component Palette tako da odgovara Vašim potrebama. U Delphiju 7, kartice možete prevlačiti kako biste im promenili redosled. Pomoću stranice Palette okvira za dijalog Environment Options možete potpuno preurediti komponente na raznim stranicama, dodajući nove elemente ili ih premeštajući sa stranice na stranicu.

Kada u paleti Component Palette ima mnogo stranica, morate proći kroz njih kako biste došli do komponente koja Vam je potrebna. U tom slučaju možete koristiti jednostavan trik: preimenujte kartice tako da imaju kraća imena pa će onda sve stati na jedan ekran. (Očigledno je, kada to jednom uradite.)

U Delphiju 7 postoji nova osobina. Kada se na jednoj stanici nalazi previše komponenti, Delphi prikazuje dvostruku strelicu na dole; kliknite je kako biste prikazali ostale komponente i kako ne biste morali da da skrolujete unutar stranice Palette.

U kontekst meniju palete Component Palette postoji podmeni Tabs u kome se prikazuju sve stranice palete u abecednom redosledu. Ovaj podmeni možete koristiti kako biste promenili aktivnu stranicu, naročito kada se stranica koja Vam je potrebna ne vidi na ekranu.

SAVET

Redosled elemenata podmenija Tabs kontekst menija palete Component Palette možete podesiti tako da odgovara redosledu u samoj paleti, a ne u abecednom redosledu. Da biste to uradili, pronađite u Registryu `Main Window` **(pod ključem za tekućeg korisnika** `\Software\Borland\Delphi\7.0`**) i ključu** `Sort Palette Tabs Menu` **zadajte vrednost 0 (False).**

Značajna nedokumentovana funkcija Component Palette je aktiviranje "hot-track". Određivanjem specijalnih tastera u Regitryu možete selektovati stranicu palete prelaskom na jezičak, a da ne morate da kliknete mišem. Ista funkcija se može dodeliti klizačima komponenata na obe strane palete, koji se prikazuju kada stranica sadrži previše komponenata. Da biste aktivirali ovu skrivenu funkciju, potrebno je da dodate ključ `Extras` pod ključem `\Software\Borland\Delphi\7.0` u odeljku `HKEY_CURRENT_USER\Software`. Pod ovim ključem potrebno je da unesete dve stringovne vrednosti, `AutoPaletteSelect` i `AutoPaletteScroll`, i da svakoj dodelite vrednost stringa "1".

Kopiranje i smeštanje komponenata

Interesantna funkcija Form Designera je mogućnost kopiranja i smeštanja komponenata sa jednog formulara na drugi ili dupliranje komponenata formulara. Tokom ove operacije Delphi duplira sva svojstva i zadržava sva povezana rukovanja događajima i, ukoliko je potrebno, menja naziv kontrole (jer naziv mora biti jedinstven u okviru formulara).

Takođe je moguće kopirati komponente iz Form Designera u editor i obrnuto. Kada komponentu kopirate na Clipboard, Delphi, takođe, smešta i tekstualni opis. Možete čak i da promenite tekst verzije komponente, kopirati tekst na Clipboard, a zatim ga smestiti natrag u formular kao novu komponentu. Na primer, ukoliko na formular smestite kontrolu, kopirate je, a zatim smestite u editor (koji može da bude Delphijev editor izvornog koda ili bilo koji tekst procesor) i dobićete sledeći opis:

```
object Button1: TButton
  Left = 152
  Top = 104
  Width = 75
  Height = 25
  Caption = 'Button1'
  TabOrder = 0
```

```
end
```

Ukoliko sada promenite naziv objekta, njegov naslov ili poziciju, na primer, ili dodate novo svojstvo, ove promene se mogu kopirati i smestiti natrag na formular. Evo primera nekoliko izmena:

```
object Button1: TButton
  Left = 152
  Top = 104
  Width = 75
  Height = 25
  Caption = 'My Button'
  TabOrder = 0
  Font.Name = 'Arial'

end
```

Kopiranje ovog opisa i njegovo smeštanje na formular napravićeće kontrolu na naznačenoj poziciji, a naslov kontrole će biti *My Button* u fontu Arial.

Da biste iskoristili ovu tehniku, potrebno je da znate kako da izmenite tekstualnu reprezentaciju komponente, koja svojstva su valjana za određenu komponentu i kako da unesete vrednosti za tekstualna svojstva, podesite svojstva i druga specijalna svojstva. Kada Delphi interpretira tekstualni opis komponente ili formulara, može promeniti vrednosti drugih svojstava koja se odnose na svojstva koja ste promenili, a može i promeniti poziciju komponente tako da ne preklapa prethodnu kopiju.

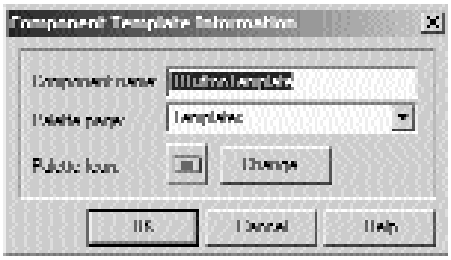
Naravno, ukoliko napišete nešto što je potpuno pogrešno i pokušate to da smestite na formular, Delphi će prikazati poruku o grešci obaveštavajući Vas o tome šta je pogrešno.

Možete selektovati nekoliko komponenata i sve ih kopirati odjednom, bilo na drugi formular bilo u editor teksta. To može da bude korisno onda kada je potrebno raditi na nizu sličih komponenti. Jednu komponentu možete kopirati u editor, replicirati je više puta, načiniti neophodne izmene, a zatim celu grupu komponenti smestiti ponovo na formular.

Od šablona komponentido okvira

Kada kopirate jednu ili više komponenti sa jednog formulara na drugi, kopirate sva njihova svojstva. Mnogo moćniji pristup jeste da napravite *šablon komponente* (component template), čime stvarate kopiju kako svojstava tako i izvornog koda obrade događaja. Kada šablon smestite u novi formular, selektovanjem pseudokomponente sa palete, Delphi će replicirati izvorni kod obrade događaja na novom formularu.

Da biste napavili šablon komponente, selektujte jednu ili više komponenti i odaberite komandu menija Component➥Create Component Template. Na ovaj način ćete otvoriti okvir za dijalog Component Template Information u koji možete uneti naziv šablona, stranicu palete Component Palette na kojoj treba da se prikaže i ikonu.



Po definiciji, naziv šablona je naziv prve komponente koju ste selektovali za kojom sledi reč *Template*. Unapred određena ikona šablona je ikona prve komponente koju ste selektovali, ali je možete promeniti izborom datoteke sa ikonom. Naziv koji dodelite šablonu komponente će se korisititi kao opis u paleti Component Palette (kada Delphi prikaže oblačić).

Sve informacije o šablonima komponenata se čuvaju u jednoj datoteci, `DELPHI23.DCT`, ali izgleda da nije moguće iz datoteke dobiti informacije i izmeniti šablon. Ono što ipak možete učiniti je da postavite šablon komponente na potpuno novi formular, izmenite ga i ponovo instalirate kao šablon komponente koristeći isti naziv. Na ovaj način možete zameniti prethodnu definiciju.

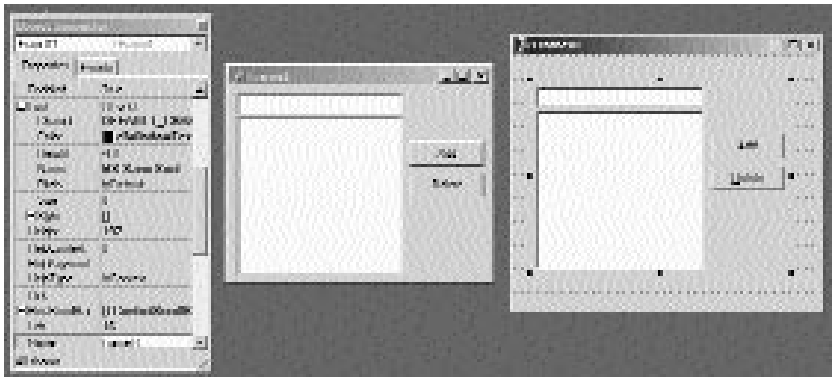
SAVET

Grupa programera Delphija može deliti šablone komponenata čuvajući ih u zajedničkom direktorijumu, dodajući u Registry stavku `CCLibDir` pod ključem `Software\Borland\Delphi\7.0\Component` Tempaltes. ■

Šabloni komponenata su zgodni kada je na različitim formularima potrebna ista grupa komponenata i odgovarajuće obrade događajima. Problem nastaje kada jednom postavite instancu šablona na formular; Delphi stvara kopiju komponenata i njihovog koda koji nije više u vezi sa šablonom. Ne postoji način da izmenite samu definiciju šablona i sasvim sigurno nije moguće da se izmene odslikaju na sve formulare koji koriste *šablon*. Da li ja tražim previše? Nikako. To je ono što u Delphiju možete postići tehnologijom okvira (frames).

Okvir je vrsta panela sa kojim možete da radite prilikom dizajniranja na način sličan radu sa formularom. Napravite novi okvir, na njega postavite nekoliko kontrola i dodate kod za obradu događaja. Kada je okvir spreman, možete otvoriti formular, odabrati pseudokomponentu Frame sa stranice Standard palete Component Palette i odabrati jedan od mogućih okvira (tekućeg projekta). Kada na formular postavite okvir, on će biti prikazan kao da su komponente kopirane. Ukoliko izmenite prvobitni okvir (u njegovom dizajneru), izmene će se odslikati na svaku instancu na formularu.

Možete pogledati jednostavan primer, nazvan Frames1, na slici 1.8. Snimak ekrana zapravo ne znači mnogo; trebalo bi da otvorite program ili ponovo izradite sličan, ukoliko želite da se poigrate okvirima.



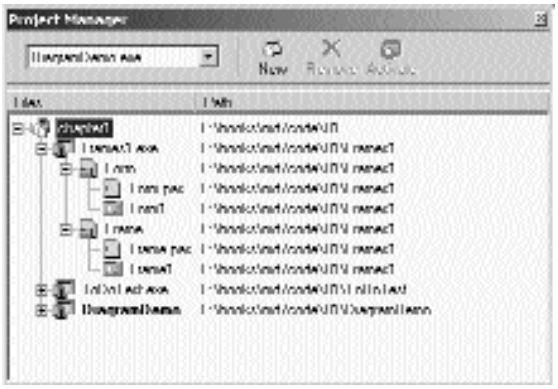
SLIKA 1.8 *Primer Frames1 demonstrira upotrebu okvira. Okvir (na levoj strani slike) i njegove instance na formularu (na desnoj strani slike) su sinhronizovani.*

Slično formularima, okviri definišu klase, tako da se mnogo lakše uklapaju u VCL objektni model nego šabloni komponenata. U Poglavlju 8, "Arhitektura Delphi Aplikacija", možete naći detaljno objašnjenje VCL-a, a sadrži i detaljniji opis okvira. Kao što možete pretpostaviti iz ovog kratkog uvoda, okviri su moćna tehnika.

Upravljanje projektima

Delphijev višeciljni Project Manager (View➥Project Manager) radi sa grupom projekta koja može imati jedan ili više projekata u sebi. Na primer, grupa projekta može sadržati DLL i izvršnu datoteku, ili više izvršnih datoteka. Svi otvoreni paketi će prikazati projekte u pogledu Project Manager čak i kada nisu pridodati grupi projekta.

Na slici 1.9 možete videti Project Manager sa prostom grupom projekta, uključujući sve primere ovog poglavlja. Kao što se vidi, Project Manager je zasnovan na prikazu drveta, kojim se prikazuje hijerarhijska struktura grupe projekta, projekata i svih formulara i jedinica koje sačinjavaju svaki od projekata. Možete koristiti jednostavnu paletu alata i složenije kontekst menije Project Managera da biste obavljali operacije sa projektima u grupi. Kontekst meniji su kontekst senzitivni; opcije kontekst menija zavise od selektovanog elementa. Postoje elementi menija za dodavanje novih ili postojećih projekata grupi projekta, za kompajliranje ili izradu određenog projekta ili za otvaranje jedinice.



SLIKA 1.9 *Delphijev višeciljni Project Manager.*

SAVET

Počev od Delphija 6, Project Manager prikazuje sve otvorene pakete, čak i kada nisu pridodati grupi projekta. Paket je kolekcija komponenata ili drugih jedinica koje su kompajlirane u posebnu datoteku, što ćete naučiti u Poglavlju 10, "Biblioteke i paketi". ■

Od svih projekata u grupi, samo jedan je aktivan; to je projekat sa kojim operišete kada odaberete komandu kao što je Project➥Compile. Meni Project glavnog menija sadrži dve komande koje možete upotrebiti za kompajliranje ili izradu svih projekata grupe. (Prilično je neobično što ove komande nisu dostupne u kontekst meniju Project Managera grupe projekta.) Kada je potrebno da izradite više projekata, možete odrediti relativni redosled upotrebom komandi Build Sooner i Build Later. Ove dve komande u osnovi iznova uređuju projekte u listi.

Među naprednim funkcijama postoji funkcija kojom možete prevući datoteke sa izvornim kodom iz Windowsovog direktorijuma ili Windows Explorera u projekat u prozoru Project Managera da biste ih dodali projektu (prevlačenje datoteka možete koristiti i za otvaranje datoteka u editoru koda). Lako možete odrediti koji je projekat selektovan, a možete promeniti tekući projekat pomoću combo polja koje se nalazi pri vrhu prozora Project Manager ili pomoću strelice na dole pored komande Run na Delphijevoj paleti alata.

Pored dodavanja Pascal datoteka i projekata, Project Manageru možete dodati Windowsove resursne datoteke; ove datoteke se kompajliraju uz projekat. Pređite na projekat, odaberite Add iz kontekst menija i odaberite Resource File (*.rc) za tip datoteke. Ovaj resurs će se automatski vezati za projekat, čak i bez odgovarajuće direktive $R.

Delphi grupu projekta čuva pod ekstenzijom .BPG, što je skraćenica za Borland Project Group. Ova funkcija dolazi iz C++ Buildera i ranijih Borlandovih kompajlera za C++, istorije koja je jasno vidljiva kada otvorite izvorni kod grupe projekta, što je u osnovi makefile C/C++ razvojnog okruženja. Evo jednostavnog primera.

```
#------------------------------
VERSION = BWS.01
#------------------------------
```

```
!ifndef ROOT
ROOT = $(MAKEDIR)\..
!endif
#----------------------------
MAKE = $(ROOT)\bin\make.exe -$(MAKEFLAGS) -f$**
DCC = $(ROOT)\bin\dcc32.exe $**
BRCC = $(ROOT)\bin\brcc32.exe $**
#----------------------------
PROJECTS = Project1.exe
#----------------------------
default: $(PROJECTS)
#----------------------------
Project1.exe: Project1.dpr
  $(DCC)
```

Opcije projekta

Project Manager ne omogućava da odredite opcije dva različita projekta odjednom. Ono što umesto toga možete učiniti jeste da pozovete okvir za dijalog Project Options iz Project Managera za svaki projekat. Prva stranica Project Options (Forms) prikazuje listu formulara koje bi trebalo automatski napraviti prilikom pokretanja programa i formulare koje pravi sam program. Naredna stranica (Application) se koristi za određivanje naziva aplikacije i naziva Help datoteke, kao i za izbor ikona. Ostale opcije okvira za dijalog Project Options se odnose na Delphi kompajler i linker, informacije o verziji i upotrebi paketa prilikom izvršavanja.

Postoje dva načina za određivanje opcija kompajlera. Jedan je upotreba stranice Compiler okvira za dijalog Project Options. Drugi je određivanje ili uklanjanje pojedinih opcija u izvornom kodu pomoću komandi `{$X+}` ili `{$X-}`, gde bi X trebalo zameniti opcijom koju želite da upotrebite. Drugi pristup je mnogo fleksibilniji jer Vam omogućava da promenite opciju za određenu datoteku izvornog koda ili za samo nekoliko redova koda. Opcije na nivou koda preinačuju opcije na nivou kompajliranja.

Sve opcije projekta automatski se čuvaju sa projektom, ali u zasebnoj datoteci sa ekstenzijom .DOF. To je tekstualna datoteka koju je lako izmeniti. Delphi, takođe, čuva opcije kompajlera u drugom CFG formatu datoteke, za kompajliranje sa komandne linije. Ove dve datoteke imaju sličan sadržaj, ali su zapisane u različitom formatu: dcc kompajler komandne linije ne može koristiti .DOF datoteke već mora da koristi .CFG format.

Još jedna alternativa za čuvanje opcija kompajlera je da pritisnete kombinaciju tastera Ctrl+O+O (pritisnite dva puta taster O dok držite pritisnut taster Ctrl). Na ovaj način umećete direktive kompajlera, koje odgovaraju aktuelnim opcijama projekta, na vrh aktuelne jedinice (uključujući nova podešavanja upozorenja kompajlera), kao u sledećem listingu.

```
{$A8,B-,C+,D+,E-,F-,G+,H+,I+,J-,K-,L+,M-,N+,O+,P+,Q-,R-,S-,T-,U-,V+,W-,X+,Y+,Z1}
{$MINSTACKSIZE $00004000}
{$MAXSTACKSIZE $00100000}
{$IMAGEBASE $00400000}
{$APPTYPE GUI}
```

```
{$WARN SYMBOL_DEPRECATED ON}
{$WARN SYMBOL_LIBRARY ON}
{$WARN SYMBOL_PLATFORM ON}
{$WARN UNIT_LIBRARY ON}
{$WARN UNIT_PLATFORM ON}
{$WARN UNIT_DEPRECATED ON}
{$WARN HRESULT_COMPAT ON}
{$WARN HIDING_MEMBER ON}
{$WARN HIDDEN_VIRTUAL ON}
{$WARN GARBAGE ON}
{$WARN BOUNDS_ERROR ON}
{$WARN ZERO_NIL_COMPAT ON}
{$WARN STRING_CONST_TRUNCED ON}
{$WARN FOR_LOOP_VAR_VARPAR ON}
{$WARN TYPED_CONST_VARPAR ON}
{$WARN ASG_TO_TYPED_CONST ON}
{$WARN CASE_LABEL_RANGE ON}
{$WARN FOR_VARIABLE ON}
{$WARN CONSTRUCTING_ABSTRACT ON}

{$WARN COMPARISON_FALSE ON}
{$WARN COMPARISON_TRUE ON}
{$WARN COMPARING_SIGNED_UNSIGNED ON}
{$WARN COMBINING_SIGNED_UNSIGNED ON}
{$WARN UNSUPPORTED_CONSTRUCT ON}
{$WARN FILE_OPEN ON}
{$WARN FILE_OPEN_UNITSRC ON}
{$WARN BAD_GLOBAL_SYMBOL ON}
{$WARN DUPLICATE_CTOR_DTOR ON}
{$WARN INVALID_DIRECTIVE ON}
{$WARN PACKAGE_NO_LINK ON}
{$WARN PACKAGED_THREADVAR ON}
{$WARN IMPLICIT_IMPORT ON}
{$WARN HPPEMIT_IGNORED ON}
{$WARN NO_RETVAL ON}
{$WARN USE_BEFORE_DEF ON}
{$WARN FOR_LOOP_VAR_UNDEF ON}
{$WARN UNIT_NAME_MISMATCH ON}
{$WARN NO_CFG_FILE_FOUND ON}
{$WARN MESSAGE_DIRECTIVE ON}
{$WARN IMPLICIT_VARIANTS ON}
{$WARN UNICODE_TO_LOCALE ON}
{$WARN LOCALE_TO_UNICODE ON}
{$WARN IMAGEBASE_MULTIPLE ON}
{$WARN SUSPICIOUS_TYPECAST ON}
```

```
{$WARN PRIVATE_PROPACCESSOR ON}
{$WARN UNSAFE_TYPE OFF}
{$WARN UNSAFE_CODE OFF}
{$WARN UNSAFE_CAST OFF}
```

Kompajliranje i izrada projekata

Postoji nekoliko načina za kompajliranje projekta. Ukoliko ga pokrenete (pritiskom tastera F9 ili ukoliko kliknete ikonu Run na paleti alata), Delphi će prvo kompajlirati projekat. Kada Delphi kompajlira projekat, kompajlira samo datoteke koje su izmenjene. Ukoliko umesto toga odaberete komandu Project➥Build All, kompajlira se svaka datoteka čak i kada nije izmenjena. Ova druga komanda Vam neće često biti potrebna, jer Delphi obično može da prepozna datoteke koje su izmenjene i kompajlira ih po potrebi. Jedini izuzetak je kada promenite neke opcije projekta. U tom slučaju, potrebno je da upotrebite komandu Build All da bi nove opcije imale efekta.

Da bi izradio projekat, Delphi prvo kompajlira svaku od datoteka sa izvornim kodom, generišući Delphi kompajliranu jedinicu (DCU). (Ovaj korak se izvršava samo ukoliko DCU datoteka nije ažurirana.) Drugi korak, koji obavlja linker, je spajanje svih DCU datoteka u izvršnu datoteku, opciono sa kompajliranim kodom iz biblioteke VCL (ukoliko niste odlučili da koristite pakete u vreme izvršavanja). Treći korak je povezivanje u izvršnu datoteku bilo koje od resursnih datoteka, kao što je RES datoteka projekta, koja sadrži glavnu ikonu i DFM datoteke formulara. Bolje ćete razumeti korake kompajliranja i lakše ćete ih pratiti ukoliko uključite opciju Show Compiler Progress (na stranici Preferences okvira za dijalog Environment Options).

UPOZORENJE

Delphi nije uvek u stanju da pravilno vodi računa kada treba ponovo izraditi jedinice koje su zasnovane na drugim jedinicama koje ste izmenili. To je svakako tačno u slučajevima (a ima ih mnogo) kada korisnikova intervencija poremeti logiku kompajlera. Na primer, promena naziva datoteka, izmena datoteka sa izvornim kodom van IDE-a, kopiranje starijih izvornih datoteka ili DCU datoteka na disk, ili ukoliko imate više kopija jedinice izvorne datoteke u putanji za pretraživanje, može dovesti do prekida kompajliranja. Svaki put kada kompajler prikaže neku čudnu grešku, prva stvar koju bi trebalo da učinite je da pokušate izvršavanje komande Build All da biste ponovo sinhronizovali funkciju "make" sa tekućim datotekama na disku. ■

Komanda Compile se može koristiti samo kada ste učitali projekat u editor. Ukoliko nema aktivnih projekata, a Vi učitate Pascal izvornu datoteku, ne možete da je kompajlirate. Ipak, ukoliko učitate *izvornu datoteku kao da je projekat*, možete prevariti računar i moći ćete da je kompajlirate. Da biste to učinili, odaberite kontrolu Open Project sa palete alata i učitajte PAS datoteku. Sada možete proveriti sintaksu ili je kompajlirati, formirajući DCU datoteku.

Ranije sam pomenuo da Delphi omogućava da koristite pakete u vreme izvršavanja, što utiče na distribuiranje programa više nego proces kompajliranja. Delphijevi paketi su dinamičke biblioteke za povezivanje (DLL) koje sadrže komponente Delphija. Korišćenjem paketa, izvršna datoteka može da bude daleko manja. Ipak, program se neće pokrenuti ukoliko odgovarajuće dinamičke biblioteke za povezivanje (kao što je paket `vcl70.bpl`, koji je prilično veliki) nisu dostupne na računaru na kome se program izvršava.

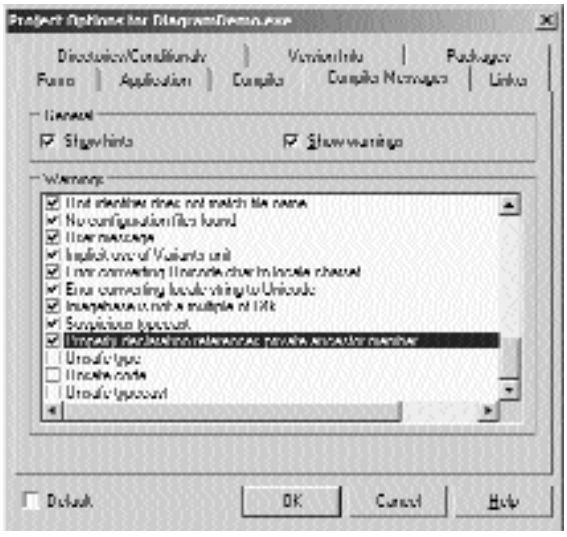
Ukoliko dodate veliku dinamičku biblioteke veličini male izvršne datoteke, ukupna količina prostora potebna na disku za očigledno manji program izrađen upotrebom paketa, je mnogo veća od količine prostora veće samostalne izvršne datoteke. Naravno, ako na jednom sistemu imate više aplikacija, ipak ćete uštedeti veliku količinu prostora i memorije. Upotreba paketa je česta, ali nije uvek preporučljiva. Razmatraću detaljno sve implikacije paketa u Poglavlju 10.

U oba slučaja, Delphijeve izvršne datoteke se veoma brzo kompajliraju, a brzina rezultujuće aplikacije se može porediti sa programima napisanim u C ili C++ jeziku. Kompajlirani kod u Delphiju se izvršava najmanje pet puta brže od ekvivalentnog koda kod alata koji interpretiraju ili polukompajliraju.

Poruke i upozorenja kompajlera

Kao što sam napomenuo na početku ovog poglavlja (u odeljku "Proširene poruke kompajlera i rezultati pretraživanja u Delphiju 7"), pored standardnih poruka kompajlera, Delphi 7 sadrži novi prozor sa dodatnim informacijama o nekim porukama o greškama. Ovaj prozor se aktivira pomoću komande View➥Additional Message Info. U prozoru se prikazuju informacije koje se čuvaju u lokalnoj datoteci koja se može ažurirati preuzimanjem novih verzija sa Borlandovog web sajta.

Druga izmena u Delphiju 7 se odnosi na poboljšanu kontrolu koju korisnik ima nad upozorenjima kompajlera. Okvir za dijalog Project Options sada sadrži stranicu Compiler Messages na kojoj možete odabrati razna upozorenja. Ova novina je uvedena verovatno zbog činjenice da u Delphiju 7 postoji novi skup upozorenja koja se odnose na kompatibilnost sa budućim Delphijevim .NET alatom. Ovih upozorenja ima zaista mnogo, a ja sam ih sve isključio onako kako je pokazano na slici 1.10.



SLIKA 1.10 *Nova stranica Compiler Messages okvira za dijalog Project Options.*

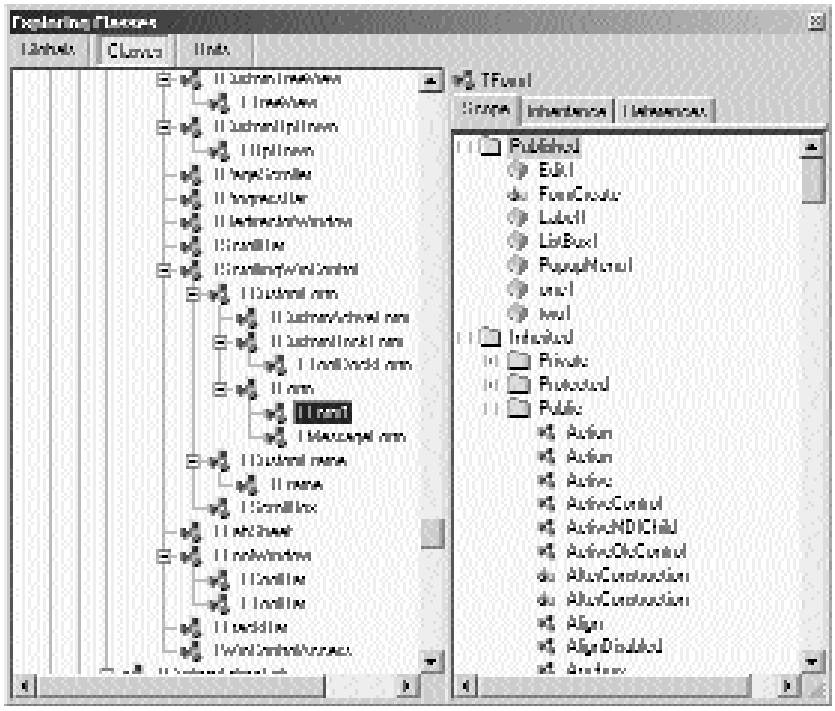
Neka od ovih upozorenja možete uključiti ili isključiti koristeći opcije kompajlera na sledeći način:

```
{$Warn UNSAFE_CODE OFF}
{$Warn UNSAFE_CAST OFF}
{$Warn UNSAFE_TYPE OFF}
```

Ova podešavanja je, u opštem slučaju, bolje držati van izvornog koda programa - to Vam sada Delphi 7 napokon omogućava.

Pretraživanje klasa projekta

Delphi je oduvek sadržao alat za pretraživanje simbola kompajliranog projekta iako se ime ovog alata mnogo puta menjalo (od Object Browser do Project Explorer, a sada se zove Project Browser). Prozor Project Browsera u Delphiju 7 aktivirate komandom View➥Browser. Pomoću ove komande će se prikazati prozor koji vidite na slici 1.11. Pretraživač će Vam omogućiti da vidite hijerarhijsku strukturu klasa projekta i da potražite simbole i redove izvornog koda u kojima se ti simboli koriste.



SLIKA 1.11 *Project Browser.*

Za razliku od Code Explorera, Project Browser se ažurira samo kada ponovo kompajlirate projekat. Ovaj pretraživač Vam omogućava da prikažete klase, jedinice i globalne promenljive i daje Vam mogućnost da prikažete samo simbole koji su definisani unutar projekta ili simbole koji su definisani u projektu i VCL-u. Podešavanja Project Browsera i Code Explorera možete promeniti na stranici Explorer okvira za dijalog Environment Options ili zadavanjem komande Properties koju možete naći u kontekst meniju Project Explorera. Neke kategorije koje vidite u ovom prozoru su specifične za Project Browser, dok se ostale odnose na oba alata.

Dodatni i spoljašnji alati Delphija

Pored IDE-a, kada instalirate Delphi, dobijate i druge, spoljašnje alate. Neki od njih, kao što su Database Desktop, Package Collection Editor (`PCE.exe`) i Image Editor (`ImagEdit.exe`) su dostupni iz menija Tools IDE-a. Pored toga, izdanje Enterprise sadrži link za SQL Monitor (`SqlMon.exe`). Drugi alati nisu direktno dostupni iz IDE-a, uključujući mnoge alate komandne linije koje možete pronaći u Delphijevom direktorijumu bin. Na pimer, postoji kompajler koji se poziva sa komandne linije (`DCC.exe`), Borlandov kompajler resursa (`BRC32.exe` i `BRCC32.exe`) i alat za prikazivanje izvršnih datoteka (`TDump.exe`).

Na kraju, neki od primera programa koje dobijate uz Delphi su zapravo korisni alati koje možete kompajlirati i imati pri ruci. Neke od ovih alata ću razmatrati u knjizi, kako bude bilo potrebno. Ovde su neki od korisnih alata višeg nivoa, od kojih se većina nalazi u direktorijumu `\Delphi7\bin` i meniju Tools:

Web App Debugger (`WebAppDbg.exe`) Web server za otkrivanje grešaka koji je predstavljen u Delphiju 6. Koristi se za praćenje zapisa koji se šalju aplikaciji i za otkrivanje grešaka u tim zahtevima. Ovaj alat je prepravljen za Delphi 7: to je sada CLX aplikacija, a njena povezivost se zasniva na prikljucima. Ovaj alat ću opisati u Poglavlju 20.

XML Mapper (`XmlMapper.exe`) Predstavlja alat za pravljenje XML tramsformacija koje se mogu primeniti na format koji pravi komponenta ClientDataSet. Više o ovoj temi u Poglavlju 22.

External Translation Manager (`etm60.exe`) Samostalna verzija Integrated Translation Managera. Ovaj spoljašnji alat se može dati spoljašnjim prevodiocima i po pri put je na raspolaganju u Delphiju 6.

Borland Registry Cleanup Utility (`D7RegClean.exe`) Pomaže Vam da iz Registrya računara uklonite sve ključeve koje je uneo Delphi 7.

TeamSource Napredan sistem za kontrolu verzije koji dobijate uz Delphi počevši od njegove verzije 5. Alat je veoma sličan predhodnoj verziji, a instalira se zasebno od Delphija 7. Uz Delphi 7 dobijate Team Source verzije 1.01, što je ista verzija koja se dobija uz zakrpu za Delphi 6.

WinSight (`Ws32.exe`) Windowsov program za presretanje poruka koji se nalazi u direktorijumu bin.

Database Explorer Alat koji se može aktivirati iz Delphijevog IDE-a ili kao samostalan alat upotrebom programa DBExplor.exe koji se nalazi u direktorijumu bin. Pošto je alat namenjen BDE-u, ovaj alat se danas malo koristi.

OpenHelp (`oh.exe`) Alat koji možete koristiti za upravljanje strukturom Delphijevih Help datoteka u koje možete integrisati datoteke nezavisnih programera.

Convert (`Convert.exe`) Alat komandne linije koji možete upotrebiti za konvertovanje DFM datoteka u ekvivalentne tekstualne opise i obrnuto.

Turbo Grep (`Grep.exe`) Pomoćni program za pretraživanje, mnogo brži od ugrađenog mehanizma Find in Files, ali nije tako lak za korišćenje.

Turbo Register Server (`TRegSvr.exe`) Alat koji možete upotrebiti za registrovanje ActiveX biblioteka i COM servera. Izvorni kod ovog alata je na raspolaganju u direktorijumu `\Demos\ActiveX\TRegSvr`.

Resource Explorer Moćan alat za pregled resursa (ali ne i potpuni resurs-editor) koji možete pronaći u direktorijumu `\Demos\ResXplor.`

Resource Workshop Stari 16-bitni resurs-editor koji se može upotrebiti za Win32 resursne datoteke. Instalacioni CD Delphija sadrži posebnu instalaciju za Resource Workshop. Ranije je bio pridružen kompajlerima Borland C++ i Pascal za Windows i bio je mnogo bolji od standardnog Microsoftovog resurs-editora koji je tada bio na raspolaganju. Mada korisnički interfejs nije izmenjen i mada ne podržava duga imena datoteka, ovaj alat još uvek može da bude koristan za izradu korisničkih ili specijalnih resursa. Alat Vam, takođe, omogućava da pretražujete resurse postojećih izvršnih datoteka.

Datoteke koje proizvodi sistem

Delphi za svaki projekat proizvodi veliki broj datoteka i trebalo bi da znate koje su to datoteke i koji su njihovi nazivi. Postoje, u osnovi, dva elementa koja imaju uticaj na imenovanje datoteka: nazivi koje dodeljujete projektima i njihovim jedinicama i unapred određene ekstenzije koje koristi Delphi. U tabeli 1.1 je spisak ekstenzija datoteka koje ćete pronaći u direktorijumu u kojem se nalazi Delphi projekat. Tabelom je prikazano kada ili pod kojim uslovima se te datoteke prave, kao i njihova važnost za buduće kompajliranje.

Tabela 1.1: Ekstenzije datoteka Delphi projekta

Ekstenzija	Tip fajla i opis	Vreme kreiranja	Potrebni za kompajliranje?
.BMP, .ICO, .CUR	Datoteke bitmapa, ikona i kursora: standardne Windows datoteke koje se koriste za čuvanje bitmapiranih slika.	Razvoj: Image Editor	Obično ne, ali su možda potrebni u vreme izvršavanja i za dalje izmene.

Ekstenzija	Tip fajla i opis	Vreme kreiranja	Potrebni za kompajliranje?
.BPG	Borland Project Group: datoteke koje koristi novi višeciljni Project Manager. To je vrsta makefilea.	Razvoj	Potrebni za ponovno kompajliranje svih projekata grupe odjednom.
.BPL	Borland Package Library: DLL koji sadrži VCL komponente koje će koristiti Delphi okruženje prilikom dizajniranja ili koje će koristiti aplikacija prilikom izvršavanja. (Ove datoteke su imale .DPL ekstenziju u Delphiju 3.)	Kompilacija: Povezivanje	Prosleđivaćete pakete drugim Delphi programerima i, opciono, krajnjim korisnicima.
.CAB	Microsoft Cabinet kompresovani format datoteke koji se koristi za Web razvoj u Delphiju. CAB datoteke mogu sadržati više kompresovanih datoteka.	Kompilacija	Prosleđuju se krajnjim korisnicima.
.CFG	Konfiguracijska datoteka sa opcijama projekta. Slična je DOF datotekama	Razvoj	Potrebne samo ukoliko se koriste specijalne opcije kompajlera.
.DCP	Delphi Component Package: datoteka sa informacijama simbola za kod koji se kompajlira u paket. Ne sadrži kompajlirani kod koji se čuva u DCU datotekama.	Kompilacija	Potrebna kada koristite pakete. Prosleđivaćete ih samo drugim programerima zajedno sa DPL fajlovima.
.DCU	Delphi Compiled Unit: rezultat kompajliranja Pascal datoteke.	Kompilacija	Samo ukoliko izvorni kod nije dostupan. DCU datoteke za jedinice koje pišete predstavljaju međukorak, te kompajliranje čine bržim.
.DDP	Novi Delphi Diagram Portfolio. Koristi ga pogled Diagram (u Delphiju 5 ekstenzija je bila .DTI)	Razvoj	Ne. Ova datoteka čuva informacije "u vreme dizajniranja", nije potrebna za rezultujući program, ali je veoma važna za svakog programera.

Ekstenzija	Tip fajla i opis	Vreme kreiranja	Potrebni za kompajliranje?
.DFM	Delphi Form File: binarna datoteka sa opisom svojstava formulara (ili modula podataka) i komponenata koje sadrži.	Razvoj	Da. Svaki formular se čuva kako u PAS tako i u DFM datoteci.
.čDF	Rezervna kopija Delphi Form File (DFM).	Razvoj	Ne. Datoteka nastaje kada sačuvate novu verziju jedinice koja ima veze sa formularom i uz nju datoteku formulara.
.DFN	Datoteka podrške za Integrated Translation Environment (postoji jedna DFN datoteka za svaki formular i svaki ciljni jezik).	Razvoj (ITE)	Da (za ITE). Ove datoteke sadrže prevedene stringove koje menjate u Translation Manageru.
.DLL	Dynamic Link Library: još jedna verzija izvrše datoteke.	Kompilacija: Linkovanje	Pogledati .EXE.
.DOF	Delphi Option File: tekstualna datoteka sa aktuelnim podešavanjima opcija projekta.	Razvoj	Potrebne samo ukoliko se koriste specijalne opcije kompajlera.
.DPK, a sada i nove ekstenzije .DPKW i .DPKL	Delphi Package: datoteka sa izvornim kodom projekta paketa (ili datoteka projekta za Windows ili Linux).	Razvoj	Da.
.DPR	Delphi Project File. (Ova datoteka zapravo sadrži Pascal izvorni kod.)	Razvoj	Da.
.čDP	Rezervna kopija Delphi Project datoteke (.DPR)	Razvoj	Ne. Ova datoteka se automatski generiše kada sačuvate novu verziju datoteke projekta.
.DSK	Desktop datoteka: sadrži informacije o poziciji Delphi prozora, datotekama otvorenim u editoru i drugim podešavanjima radne površine.	Razvoj	Ne. Zapravo bi trebalo da je uklonite kada kopirate projekat u novi direktorijum.
.DSM	Delphi Symbol Module: čuva sve informacije o simbolima pretraživača.	Kompilacija (ali samo ako se koristi opcija Symbols)	Ne. Object Browser koristi ovu datoteku, umesto podataka u memoriji, kada ne možete ponovo kompajlirati projekat.

Ekstenzija	Tip fajla i opis	Vreme kreiranja	Potrebni za kompajliranje?
.EXE	Izvršna datoteka: Windows aplikacije koju proizvodite.	Kompilacija: Linkovanje	Ne. Ovo je datoteka koju prosleđujete. Sadrži sve kompajlirane jedinice, formulare i resurse.
.HTM	Ili .HTML, za HyperText Markup Language: format datoteke koja se koristi za web strane.	Web razvoj za ActiveForm	Ne. Nije potrebna za kompajliranje projekta.
.LIC	Datoteke sa licencom koje se odnose na OCX datoteku.	ActiveX Wizard i drugi alati	Ne. Neophodna za kontrolu u drugom razvojnom okruženju.
.OBJ	Objekt (kompajlirana) datoteka, tipičan za C/C++ svet.	Međukorak prilikom kompajliranja, Object Inspector se ne koristi u Delphiju.	Možda je potrebna za spajanje Delphija sa C++ kompajliranim kodom u projektu.
.OCX	OLE Control Extension: specijalna verzija DLL-a, sadrži ActiveX kontrole ili formulare.	Kompilacija: Linkovanje	Pogledati .EXE.
.PAS	Pascal datoteka: izvorni kod Pascal jedinice, bilo da je to jedinica koja ima veze sa formularom, bilo da je samostalna jedinica.	Razvoj	Da.
.~PA	Rezervna kopija Pascal Ne. Ovu datoteku Delphi datoteke (.PAS)	Razvoj	Ne. Ovu datoteku Delphi automatski generišu kada sačuvate novu verziju izvornog koda.
.RES, .RC	Datoteka resursa: binarna datoteka pridružena projektu, a obično sadrži ikonu projekta. Projektu možete dodati druge datoteke ovog tipa. Kada pravite datoteke resursa, možete takođe koristiti tekstualni format, .RC.	Razvoj Options . dijalog ITE (Integrated Translation Environment) generiše datoteke resursa sa specijalnim komentarima.	Da. Glavna RES datoteka aplikacije Delphi ponovo generiše prema informaciji sa strane Aplication okvira za dijalog Project Options.
.RPS	Translation Repository (deo Integrated Translation Environmenta).	Razvoj (ITE)	Ne. Potrebna za upravljanje prevodima.
.TLB	Type biblioteka: datoteka koja se automatski formira ili je formira Type Library Editor za OLE server aplikacije.	Razvoj	Ovo je datoteka koja je možda potrebna za druge OLE programe.

Ekstenzija	Tip fajla i opis	Vreme kreiranja	Potrebni za kompajliranje?
.TODO	Datoteka sa spiskom zadataka, sadrži elemente koji se odnose na ceo projekat.	Razvoj	Ne. Ova datoteka sadrži beleške programera.
.UDL	Microsoft Data Link	Razvoj	Datoteku koristi ADO da bi se referisao na provajdera podataka. Slično kao alijas u BDE svetu (videti Poglavlje 12).

Pored datoteka koje se generišu prilikom razvoja projekta u Delphiju, postoje i mnogi drugi koje generiše i koristi IDE. U tabeli 1.2 dao sam kratku listu ekstenzija koje nije na odmet znati. Većina ovih datoteka je u ekskluzivnom i nedokumentovanom formatu, te je malo toga što sa njima možete učiniti.

Tabela 1.2: Odabrane Delphi IDE ekstenzije datoteka prilagođavanja

Ekstenzija	Tip datoteke
.DCI	Delphi Code Templates
.DRO	Delphi Object (Može se menjati komandom Tools➥Repository.)
.DMT	Delphi Menu Templates
.DBI	Database Explorer Information
.DEM	Delphi Edit Mask (datoteke sa specifičnim formatima za zemlje)
.DCT	Delphi Component Templates
.DST	Datoteka sa podešavanjima radne površine (po jedna za svako podešavanje koje definišete)

Prikazivanje datoteka sa izvornim kodom

Upravo sam nabrojao datoteke koje su u vezi sa razvojem aplikacije Delphi, ali želim da posvetim malo više pažnje objašnjenju njihovih formata. Osnovne Delphi datoteke su Pascal datoteke sa izvornim kodom, koje su zapravo ASCII tekstualne datoteke. Masna slova, kurziv i obojeni tekst koji vidite u editoru, zavise od označavanja sintakse, ali se ne čuvaju u okviru datoteke. Nemaju nikakvu vrednost jer postoji jedna datoteka za sav kod formulara, a ne za male fragmente koda.

SAVET

U listinzima u knjizi sam pokušao da koristim masna slova kao što to koristi editor za ključne reči i kurziv za stringove i komentare. ■

Za formular, Pascal datoteka sadrži deklaraciju klase formulara i izvorni kod obrade događaja. Vrednosti svojstava koje određujete u Object Inspectoru se čuvaju u zasebnoj datoteci sa opisom (sa ekstenzijom .DFM). Jedini izuzetak je svojstvo `Name`, koje se koristi u deklaraciji formulara da bi se referisalo na komponente formulara.

DFM datoteka je tekstualna reprezentacija formulara, ali se može sačuvati u tradicionalnom formatu Windows Resource. Na stranici Designer okvira za dijalog Environment Options možete odrediti format koji želite da koristite za nove projekte, a možete menjati format za pojedine formulare komandom Text DFM iz kontekst menija formulara. Tekstualni editor može pročitati samo tekstualne datoteke. Ipak, možete učitati DFM datoteke oba tipa u Delphi editor, koji će, ukoliko je potrebno, prvo konvertovati datoteke u tekstualni opis. Najjednostavniji način za otvaranje tekstualnog opisa formulara (u bilo kom da je formatu) jeste da odaberete komandu View As Text iz kontekst menija Form Designera. Na ovaj način ćete zatvoriti formular, čuvajući ga ukoliko je potrebno, i otvoriti DFM datoteku u editoru. Kasnije se možete vratiti na formular koristeći komandu View As Form iz kontekst menija prozora editora.

Vi, zapravo, možete editovati tekstualni opis formulara, mada to treba učiniti veoma pažljivo. Čim sačuvate datoteku, biće pretvorna u binarnu datoteku. Ukoliko ste načinili nepravilne izmene, kompajliranje će se zaustaviti uz poruku o grešci i biće potrebno da ispravite sadržaj DFM datoteke pre nego što ponovo budete mogli da otvorite formular. Zbog toga ne bi trebalo da pokušavate da ručno izmenite tekstualni opis formulara sve dok ne steknete dobro znanje o Delphi programiranju.

SAVET

U knjizi ću Vam često prikazivati delove DFM datoteka. U većini delova ću prikazivati samo najrelevantnije komponente ili svojstva; uopšte, ukloniću svojstva koja se odnose na poziciju, binarne vrednosti i druge linije koje daju malo korisnih informacija. ■

Uz dve datoteke koje opisuju formular (PAS i DFM), treća datoteka je srž za ponovnu izgradnju aplikacije. To je Delphijeva datoteka projekta (Delphi project file - DPR), koja predstavlja još jednu Pascal datoteku sa izvornim kodom. Ova datoteka se automatski formira i retko ćete imati potrebu da je ručno menjate. Ova datoteka se može prikazati komandom View➥Project Source.

Neki druge, manje važne datoteke koje proizvodi IDE, koriste strukturu Windows INI datoteka, u kojima je svaka sekcija označena nazivom koji se smešta unutar uglastih zagrada. Na primer, ovo je deo opcione datoteke (DOF):

```
[Compiler]
A=1
B=0
ShowHints=l
Showwarnings=1

[Linker]
MinStackSize=16384
MaxStackSize=1048576
ImageBase=4194304

[Parameters]
RunParams=
HostApplication=
```

Istu strukturu koriste Desktop datoteke (DSK), u kojima se čuva status Delphi IDE-a za određeni projekat, prikazujući poziciju svakog od prozora. Evo malog isečka:

```
[MainWindow]
Create=1
Visible=1
State=0
Lett=2
Top=0
Width=800
Height=97
```

Object Repository

Delphi sadrži nekoliko komandi menija koje možete upotrebiti za pravljenje novog formulara, nove aplikacije, novog modula podataka, nove komponente i tako dalje. Ove komande se nalaze u meniju File➥New i drugim menijima. Ukoliko jednostavno odabere File➥New➥Other, Delphi otvara Object Repository koji se koristi za pravljenje novih elemenata bilo koje vrste: formulara, aplikacija, modula podataka, biblioteka, komponenata, objekata automatizacije i drugih.

Okvir za dijalog New Items (prikazan na slici 1.12) sadrži više stranica na kojima se nalaze svi novi elementi koje možete napraviti, postojeći formulari i projekti koji se čuvaju u Repositoryu, Delphijevi čarobnjaci i formulari tekućeg projekta (za vizuelno nasleđivanje formulara). Stranice i stavke u ovom okviru za dijalog sa stranicama zavise od određene verzije Delphija, tako da ih ovde neću nabrojati.

SLIKA 1.12 *Prva strana okvira za dijalog New, opšte poznata kao Object Repository.*

SAVET

Object Repository sadrži kontekst meni koji omogućava da sortirate elemente na različite načine (po nazivima, autoru, datumu ili opisu) i da ih prikažete u različitim pogledima (velike ikone, male ikone, spisak ili detaljan spisak). Pogled Details Vam daje opis, autora i datum alata - informacije koje su naročito važne kada prikazujete čarobnjake, projekte ili formulare koje ste dodali u Repository. ■

Najjednostavniji način da prilagodite Object Repository jeste da dodate nove projekte, formulare i module podataka kao šablone. Takođe, možete dodati i nove stranice i urediti elemente na nekima od njih (ne uključujući stranice New i stranice tekućih projekata). Dodavanje novog šablona Delphijevom Object Repositoryu je jednostavno koliko i upotreba postojećeg šablona za izradu aplikacije. Kada imate aplikaciju koja radi, a želite da je upotrebite kao početnu tačku za buduci razvoj sličnih programa, možete sačuvati trenutni status u šablonu koji kasnije možete koristiti. Upotrebite komandu Project➥Add To Repository i popunite okvir za dijalog.

Kao što možete dodati šablone projekta u Object Repository, možete, takođe, dodati nove šablone formulara. Pronađite formular koji želite da dodate i iz kontekst menija odaberite komandu Add To Repository. Zatim naznačite naslov, opis, autora, stranu i ikonu u okviru za dijalog. Imajte na umu da kada kopirate projekat ili formular iz šablona u Repository, a zatim kopirate u drugi direktorijum, Vi izvršavate operaciju kopiranja i smeštanja. Ovo se ne razlikuje mnogo od ručnog kopiranja datoteka.

PRAZAN ŠABLON PROJEKTA

Kada započnete prazan projekat, takođe se automatski otvara prazan formular. Ukoliko želite da novi projekat bazirate na nekom od formulara ili čarobnjaka (Wizards), ovo nije ono što želite. Da biste rešili problem, možete dodati šablon Empty Project (prazan projekat) u Gallery.

Koraci koji su neophodni su sasvim jednostavni.

1. Napravite, kao i obično, novi projekat.
2. Uklonite jedini formular iz projekta.
3. Dodajte ovaj projekat šablonima, dodeljujući mu naziv Empty Project.

Kada iz okvira za dijalog Object Repository odaberte ovaj projekat, imate dve prednosti: postoji projekat bez formulara i možete odabrati direktorijum u koji će se kopirati datoteke šablona projekta. Postoji i nedostatak - morate zapamtiti da upotrebite komandu File➥Save Project As da biste projektu dodelili novi naziv jer čuvanje projekta na bilo koji drugi način automatski koristi unapred određeni naziv šablona.

Da biste dalje prilagodili Repository, možete upotrebiti komandu Tools➥Repository. Na ovaj način ćete otvoriti okvir za dijalog Object Repository koji možete upotrebiti da elemente premestite na neku drugu stranicu okvira za dijalog, da dodate nove elemente ili da uklonite postojeće. Možete čak dodati nove stranice, promeniti naziv stranici ili je ukloniti, ili promeniti njihov redosled. Važan element pri izmeni okvira za dijalog Object Repository je upotreba unapred određenih vrednosti.

- Upotrebite polje za potvrdu New Form koje se nalazi ispod liste objekata da biste naznačili formular koji treba koristiti prilikom kreiranja novih formulara (File➥New Form).
- Polje za potvrdu Main Form označava koji tip formulara se koristi prilikom pravljenja glavnog formulara aplikacije (File➥New Application) kada nije odabran nijedan novi projekat.

- Polje za potvrdu New Project, koje je dostupno kada odaberete projekat, označava unapred određeni projekat koji će Delphi koristiti kada pozovete komandu File➥New Application.

Samo jedan formular i samo jedan projekat u okviru za dijalog Object Repository može imati svako od ova tri svojstva označena specijalnim simbolom koji se pojavljuje iznad ikone. Ukoliko nijedan projekat nije označen kao New Project, Delphi pravi unapred određeni projekat na osnovu formulara označenog kao Main Form. Ukoliko nijedan formular nije označen kao glavni fomular, Delphi pravi unapred određeni projekat na osnovu praznog formulara.

Kada radite sa okvirom za dijalog Object Repository, radite sa formularima i modulima sačuvanim u poddirektorijumu OBJREPOS glavnog Delphi direktorijuma. Istovremeno, ukoliko direktno koristite formular ili bilo koji drugi objekat bez prethodnog kopiranja, tada ćete dobiti neke datoteke Vašeg projekta u ovom direktorijumu. Veoma je važno shvatiti kako Repository funkcioniše, jer ukoliko želite da izmenite projekat ili objekat sačuvan u Repositoryu, najbolji pristup je da operišete sa originalnim datotekama i da ne kopirate podatke u i iz Repositorya.

INSTALIRANJE NOVIH DLL ČAROBNJAKA

U osnovi, novi čarobnjaci dolaze u dva različita oblika: mogu da budu deo komponenata ili paketa ili se mogu distribuirati kao nezavisni DLL-ovi. U prvom slučaju, instaliraju se na isti način kao što se instaliraju komponente ili paketi. Kada dobijete samostalni DLL, potrebno je da dodate naziv DLL-a u Windows Registry pod ključem `\Software\Borland\Delphi\7.0\Experts`. Dodajte novi string ključ pod ovaj ključ, odaberite naziv koji želite (naziv nije zapravo važan) i upotrebite kao tekst putanju i naziv datoteke DLL čarobnjaka. Možete pogledati elemente koji već postoje pod ključem `Experts` da biste videli kako da unesete putanju.

Poboljšanja debagera u Delphiju 7

Kada iz Delphijevog IDE-a pokrenete program, Vi ga zapravo pokrećete u ugrađenom debageru. Možete zadati tačke prekida, izvršavati kod red po red i istraživati detalje programa, uključujući asemblerski kod koji se izvršava i upotrebu registara procesora u pogledu CPU. U ovoj knjizi nemam dovoljno prostora da bih objasnio debagiranje; pogledajte Dodatak C za više informacija o materijalu koji se odnosi na ovu temu. Ipak, želim da pomenem nekoliko novih osobina debagera.

Prvo, okvir za dijalog Run Parameters u Delphiju 7 Vam omogućava da zadate radni direktorijum u kojem će se debagirati program. To znači da će tekući direktorijum biti onaj koji ćete naznačiti, a ne onaj u koji ste kompajlirali program.

Druga važna izmena se odnosi na Watch List. Watch List sada sadrži više kartica koje Vvam omogućavaju da imate različite skupove aktivnih pogleda za razne delove programa koji debagirate tako da ne morate preklapati prozore. U Watch List možete dodati novu grupu pomoću njegovog kontekst menija, a možete i izmeniti vidljivost zaglavlja kolona i uključiti pojedine poglede pomoću odgovarajućih polja za potvrdu.

NAPOMENA

U ovoj knjizi se ne objašnjava Delphijev debager, ali je to veoma važna tema. Pogledajte reference na materijal koji možete naći na mreži u Dodatku C za više informacija o tome kako možete preuzeti besplatno poglavlje koje se bavi debagiranjem u Delphiju. ■

Šta je sledeće?

Ovo poglavlje sadrži pregled novih i naprednijih funkcija programskog okruženja Delphi 7, uključujući brojne savete i sugestije o nekim manje poznatim funkcijama koje su već bile na raspolaganju u prethodnim verzijama Delphija. Nisam naveo opise korak-po-korak za IDE, delimično zbog toga što je, uopšte uzev, lakše započeti upotrebu Delphija nego što je pročitati kako ga koristiti. Takođe, postoji detaljna Help datoteka koja opisuje okruženje i razvoj novog jednostavnog projekta, a možda već posedujete nekakvo iskustvo u korišćenju prethodnih verzija Delphija ili sličnih razvojnih okruženja.

Sada smo spremni da se u narednom poglavlju pozabavimo Delphijevim programskim jezikom, pa da zatim nastavimo sa učenjem RTL-a i biblioteke klasa koja je deo Delphija.