

4. POGLAVLJE

Razvojna okolina

U ovom poglavlju:

- Što je Visual Studio .NET
- Kako se snaći u sučelju
- Stvaranje projekata
- Ključna olakšanja u radu
- Korištenje dokumentacije (MSDN)

Pisanje iole veće aplikacije onako kako smo to radili u prošlom poglavlju bilo bi, pogađate, srednjovekovno mučenje. Čak su i u davna vremena tekstualnih operativnih sustava postojali programi kojima je bio zadatak olakšati pisanje kôda i njegovo kompajliranje. Zahvaljujući marketingu, danas se takvi programi nazivaju “integrirana razvojna okolina” (engl. *Integrated Development Environment*), no njihova bit i dalje ostaje ista – pojednostaviti pisanje kôda.

Razvojni alati postoje za sve operativne sustave i platforme, a vrijednost pojedinog alata očituje se u njegovim mogućnostima. Upravo prema tim mogućnostima programeri (odnosno, ako ćemo koristiti marketinške pojmove, *developeri*) biraju razvojnu okolinu u kojoj će raditi.

Ponajbolji izbor za razvoj aplikacija u okruženju .NET-a jest Microsoftov Visual Studio .NET iako postoje i drugi kvalitetni razvojni alati, poput Borlandovog C# Buildera.

I. DIO: .NET IZNUTRA

Visual Studio .NET

Već niz godina Microsoftova razvojna okolina zove se Visual Studio. Prije prelaska u vode .NET-a zadnja verzija označena je brojkom 6. Nakon što je predstavljen .NET, na tržište je izašla nova verzija pod nazivom Visual Studio .NET, a u trenutku pisanja ove knjige aktualna je ona punog imena Visual Studio .NET 2003. Svaka od inačica Visual Studija .NET prati određeno izdanje .NET Frameworka. Tako je Visual Studio .NET (ponekad referenciran kao VS 7.0) bio pratitelj inačice 1.0, a Visual Studio .NET 2003 (poznat i kao VS 7.1) .NET Frameworka verzije 1.1. U razvoju su i sljedeće inačice .NET Frameworka 2.0 i pratećeg mu Visual Studija .NET kodnog imena Whidbey koje bi trebale donijeti niz poboljšanja i prednosti. Zanimljivo je da postoje planovi i za inačicu nakon Whidbeja, nazvanu Visual Studio Orcas, u kojoj će biti porađeno na tješnjoj integraciji s novom inačicom Windowsa kodnog imena Longhorn.

Da smo se ranije sreli...

Microsoft je oduvijek svoje platforme volio obogaćivati razvojnim alatima. Što više, dio zasluga za popularizaciju Microsoftovih proizvoda pripisuje se izrazitoj brizi za programere. Tako smo još od davne 1981. u MS-DOS-u mogli raditi u programskom jeziku Basic, a s pojavom Windowsa 1990. godine izdana je i prva inačica vizualnog razvojnog alata nazvanog Visual Basic. Sljedeći veliki korak napravio je Visual Studio izdan 1995. godine, u kojem

je omogućen razvoj internetskih rješenja, no nisu zapostavljene niti klasične aplikacije za Windowse.

Visual Studio .NET odnosno cijela .NET inicijativa zapravo je unifikacija programskih modela – povezivanje dotada nepovezanih metoda i principa u konzistentno programsko sučelje neovisno o jeziku ili tipu aplikacije.

Ukoliko ste se izgubili u netom pročitanoj salvi podataka, ne brinite – sve što trebate znati jest da ćemo se u pregledu primarno orijentirati na posljednju finalnu inačicu – Visual Studio .NET 2003.

Visual Studio .NET 2003 dostupan je u četiri izdanja, namijenjena različitim profilima korisnika. Osnovna inačica naziva se Professional i sadrži sve mogućnosti namijenjene individualnom razvoju aplikacija. Nedostaju tek neke napredne mogućnosti modeliranja i korištenja baza podataka, upravljanja kôdom na kojem istovremeno radi više korisnika te ne sadrži razvojne licence za korištenje Microsoftovih serverskih proizvoda.

4. POGLAVLJE: RAZVOJNA OKOLINA

Izdanje Academic namijenjeno je isključivo obrazovnim institucijama, a osim svih karakteristika izdanja Professional može se pohvaliti posebnim edukativnim alatima za korištenje u nastavi, kao i puno većom bazom primjera i dokumentacije.

Naprednijim programerima trebat će izdanja Enterprise Developer odnosno Enterprise Architect koja donose nekoliko ključnih mogućnosti za razvoj velikih rješenja. Kako njihove karakteristike prelaze gabarite ove knjige, nećemo ih eksplicitno navoditi, no sve zainteresirane upućujemo na adresu <http://msdn.microsoft.com/vstudio/howtobuy/choosing.aspx> gdje će naći detaljnu usporednu tablicu.

Procesor	Pentium II 450 MHz (preporučeno Pentium III 600 MHz)
Operativni sustav	Windows Server 2003
	Windows XP Professional
	Windows XP Home Edition (ne podržava ASP.NET)
	Windows 2000 Professional SP3
	Windows 2000 Server SP3
Radna memorija	160 MB (Windows Server 2003, Windows XP Professional)
	96 MB (Windows XP Home Edition, Windows 2000 Professional)
	192 MB (Windows 2000 Server)
Tvrdi disk	900 MB na sistemskom disku
	3,3 GB na instalacijskom disku
	1,9 GB za dokumentaciju (MSDN Library)

Tablica 4-1:
Minimalni sistemski zahtjevi
za pokretanje Visual Studija
.NET 2003

Instalacija Visual Studija

Četiri su koraka instalacije Visual Studija .NET. Prvi ili – bolje rečeno – nulti korak je instalacija za-krpa i dodataka nužnih da bi Visual Studio uopće mogao raditi. Svi se potencijalno potrebni dodaci nalaze na instalacijskom DVD-u (odnosno CD-u “Microsoft Visual Studio .NET 2003 Prerequisites”), pa nema potrebe za instalacijom preko Interneta. Oko detekcije potrebe za obaveznim dodacima pobrinut će se sam instalacijski program, tako da ćete nakon ovog koraka biti bogatiji za nekoliko Service Packova, Microsoft FrontPage 2000 Web Extensions Client i sam Microsoft .NET Framework 1.1. Ukoliko to odaberete, instalirat će se i Microsoft Visual J# .NET Redistributable

I. DIO: .NET IZNUTRA

Package 1.1. Međutim, taj potonji “paket” vam treba samo ako planirate razvijati u jeziku J#. Ako vam to nije namjera, pravovremeno ga isključite kako ne bi zauzimao dragocjen diskovni prostor.



Slika 4-1:
Instalacija Visual Studija sastoji se od četiri koraka.



Na računalu može istovremeno biti instalirano više različitih inačica Visual Studija.

Sljedeći korak je instalacija samog Visual Studija. Vjerujemo da sa snalaženjem u sučelju nećete imati problema, no savjetujemo vam da malo više vremena potrošite na pregledavanje komponenti koje dolaze u paketu. Svaka od komponenti sadrži detaljan opis, pa prema njemu možete barem otprilike zaključiti što nećete koristiti, te na temelju toga izdvojiti iz instalacije. Taj savjet posebno vrijedi za odabir programskih jezika – gotovo je sigurno da nećete koristiti sva četiri ponuđena.

Ukoliko se prvi put susrećete s programiranjem ili nemate iskustva u jezicima Java i C++, najsigurniji odabir su Visual C# .NET i Visual Basic .NET (kao što je prikazano na slici 4-2).

4. POGLAVLJE: RAZVOJNA OKOLINA



Slika 4-2:
Probajte isključiti mogućnosti koje sigurno ne namjeravate koristiti – Visual Studio je glomazan komad softvera, pa nema smisla da neke njegove komponente samo troše diskovni prostor.

Ne strahujte da ćete krivo odabrati odnosno isključiti nešto što vam može kasnije zatrebati. Sve je opcije moguće naknadno dodati, ali i oduzeti.

Slijedi instalacija dokumentacije – ogromne baze znanja poznate pod imenom Microsoft Developer Network (skraćeno MSDN) Library. Bez dokumentacije život programera je osjetno teži, pa svakako preporučujemo da je instalirate. Ipak, ukoliko su vam gotovo dva gigabajta diskovnog prostora prevelik danak, imajte na umu da je kompletan MSDN Library javno objavljen na Internetu, na adresi <http://msdn.microsoft.com/library/>. Čak štoviše, ta je inačica stopostotno ažurna i sadrži sve najnovije tekstove i specifikacije, za razliku od one na instalacijskim medijima koja je najvjerojatnije stara barem nekoliko mjeseci, ako ne i više. S druge strane, i instalirana inačica ima nekoliko prednosti, od kojih je najvažnija mogućnost zvana *dynamic help*. To u praksi znači da za vrijeme rada možete u bilo kojem trenutku stisnuti tipku F1 i dobiti informacije o dijelu sučelja ili naredbi na kojoj radite. Detaljnije informacije o MSDN Libraryju pronaći ćete na kraju ovog poglavlja.

Odmah prelazimo na zadnji, četvrti korak instalacije. Radi se o podsjetniku da je dobro s Interneta preuzeti i instalirati eventualne zakrpe, nadogradnje i *service packove* za programe koje ste upravo instalirali: imat ćete sve najnovije mogućnosti i svoje ćete računalo učiniti sigurnijim. Ne treba posebno naglašavati da ovaj korak nije nužan, no svakako je preporučljiv.

Stvaranje novog projekta

Nakon pokretanja Visual Studija dočekat će vas početna stranica (engl. *Start Page*). Na njoj ćete pronaći listu zadnje otvaranih projekata, a ispod te liste nalaze se dva ključna gumba za kreiranje

I. DIO: .NET IZNUTRA

novih i otvaranje postojećih projekata koji nisu u popisu. Ukoliko vam je ovo prvo pokretanje Visual Studija, stvaranje novog projekta jedina vam je mogućnost.

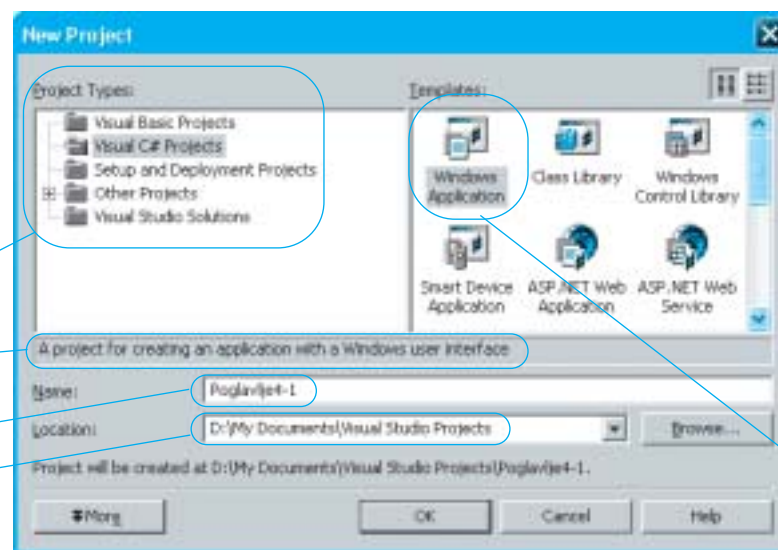
Slika 4-3:
Stvaranje novog projekta u Visual Studiju .NET 2003

Tip projekta

Opis odabranog predloška

Ime projekta

Lokacija projekta na disku



Odabrani predložak

Pri kreiranju novog projekta morate paziti na dvije ključne stvari. Prva i najvažnija je tip projekta koji želite stvoriti. Radite li u C#-u, izabrat ćete stavku “Visual C# Projects”, a na desnoj strani će se pojaviti nekoliko predložaka za projekte u tom programskom jeziku. Među ostalim, možete odabrati predložak za pisanje prozorske, konzolske ili web-aplikacije. Primijetite da koristimo pojam “predložak” – svaki tip aplikacije moguće je pisati od nule (izborom predloška “Empty Project”), ne koristeći kôd koji Visual Studio automatski generira izborom nekog predloška. Međutim, takav je pristup u većini slučajeva gubljenje vremena.

Dalje trebate paziti na ime projekta. Ime koje ovdje unesete koristit će se na raznim mjestima – od imena mape u kojoj će biti smještene datoteke koje pripadaju projektu do imena *assemblyja* i *namespacea*. Dakako, na mnogim ćete mjestima naziv moći promijeniti naknadno, no za to potreban trud i vrijeme sugeriraju da već na ovom koraku razmislite i upišete nešto pametno i logično.



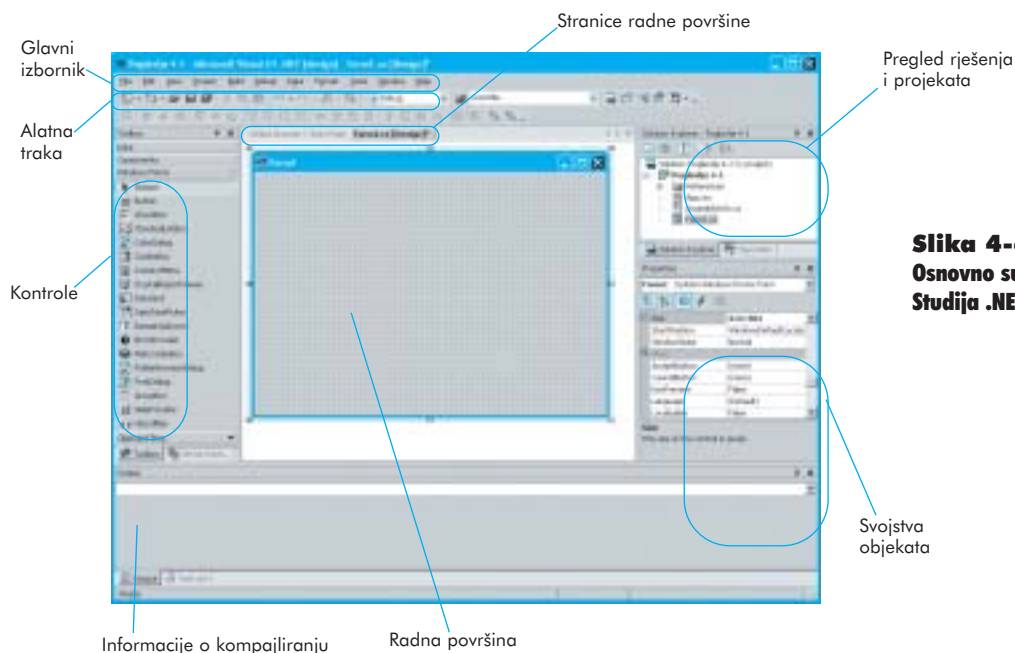
Za imenovanje projekata odnosno aplikacija ne postoje stroga pravila, no praksa je da projekt dobije ime koje opisuje njegovu funkcionalnost. Naravno, ne treba pretjerivati u dužini i preporučljivo je izbjegavati znakove poput našeg jeziku specifičnih palatala te razmake.

4. POGLAVLJE: RAZVOJNA OKOLINA

Treba spomenuti da je pojam projekta adekvatan pojmu *assemblyja* o kojem smo pričali u drugom poglavlju. Spajanjem dvaju ili više projekata (*assemblyja*) stvaramo rješenje. Prema inicijalnim postavkama, prilikom stvaranja novog projekta otvara se i novo rješenje (engl. *solution*), no zahvaljujući opcijama pri kreiranju projekta (djelomično skrivenih iza klika na “More”), takvom je ponašanju moguće doskočiti. Sasvim je izvjesno da vam za jednostavne aplikacije igranje s rješenjima neće trebati, no s obzirom na to da se taj pojam često pojavljuje, red ga je spomenuti kako vas ne bi zbunjivao.

Sučelje razvojne okoline

Mi smo za početak izabrali predložak za prozorsku aplikaciju, no prije nego što krenemo na prve redove kôda, zadržat ćemo se koji trenutak na sučelju koje nas je dočekalo nakon potvrde izbora.



Slika 4-4:
Osnovno sučelje Visual
Studijs .NET 2003

Kažu da slika vrijedi tisuću riječi, a kako imate iskustva s radom u prozorskim aplikacijama, vjerujemo da ćete brzo prepoznati dijelove koji se i ovdje pojavljuju. Ipak, ovo je sučelje znatno bogatije, zbog čega ćete vrlo brzo poželjeti veliki monitor. Naime, radnu površinu koja dominira sredinom ekrana okružuju pomoćni prozori koji su, za razliku od nekih drugih aplikacija, ne samo korisni, nego i nužni.

I. DIO: .NET IZNUTRA

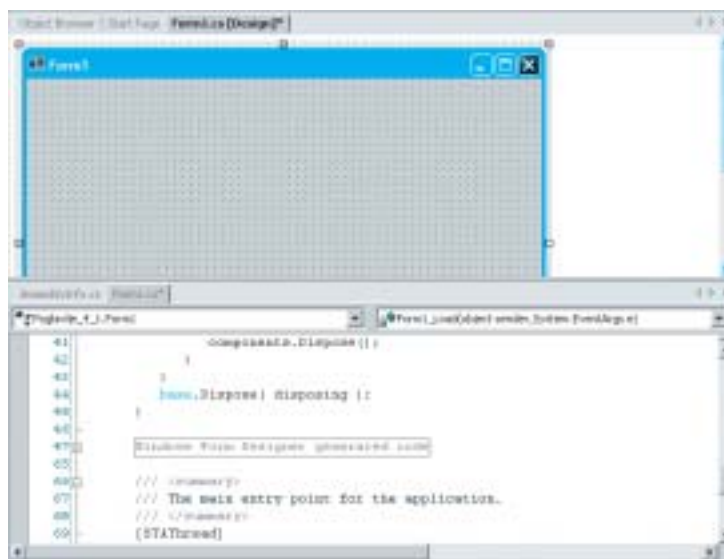
Štoviše, svaki od prozora krije više funkcionalnosti. Tako prozor Toolbox može postati Server Explorer, Output može poslužiti za popisivanje zadataka koji vas čekaju (Task List), Solution Explorer se zna pretvoriti u preglednik klasa (Class View), a prozor sa svojstvima brzo se transformira u dinamički sustav pomoći (Dynamic Help). No, krenimo redom – na stranicama što slijede pročit ćemo najvažnije dijelove Visual Studijeva sučelja, dok ćemo ostale upoznavati u hodu, kako nam u kojem primjeru zatreba.

Radna površina

U središnjem dijelu sučelja možemo otvoriti zaista sve i svašta. Sjetite se da se na tom mjestu otvorila početna stranica s popisom nedavno korištenih projekata, a nakon kreiranja projekta tu se otvorila prazna forma Windowsa (zato što smo izabrali prozorsku aplikaciju). Tu se još nalazi i Object Browser koji vam omogućava “šetanje” po *assemblyjima* i pripadajućim im *namespaceovima* i klasama, a s vremenom ćete otkriti da se u tom dijelu sučelja pojavljuju i mnoge druge funkcionalnosti.

Najčešći stanovnik ovog dijela sučelja ipak će biti datoteke s kôdom. Svaku datoteku možete otvoriti na dva načina, čak i istovremeno. Prvi način naziva se “dizajnerski način”. U njemu vizualno (zato se i zove Visual Studio) kreirate formu aplikacije. Taj način prepoznat ćete po oznaci “[Design]” u imenu kartice. Kliknete li desnom tipkom miša na njega i iz padajućeg izbornika odaberete “View Code”, otvorit će se nova kartica i radnu površinu zauzet će kôd koji se krije iza te forme. U kojem god načinu radili, rezultat vašeg rada bit će kôd. Jedina razlika je u tome što dok radite u dizajnerskom načinu, Visual Studio kôd generira automatski, dok ga u drugom načinu pišete sami.

Slika 4-5:
Radna površina
podijeljena na dva dijela,
tako da istovremeno
možete vidjeti i kôd
i formu.



4. POGLAVLJE: RAZVOJNA OKOLINA

Prema *defaultnim* postavkama kôd koji se generira u dizajnerskom načinu neće biti prikazan zajedno s ostatkom koda. To ne znači da ne postoji ili da je u nekoj drugoj datoteci, nego da je skriven. Na slici 4-5 možete vidjeti pravokutnik s natpisom “Windows Form Designer generated code”. Spomenuti kôd se krije iza njega, a vidjeti ga možete ako kliknete na znak plusa koji se nalazi u istom redu, lijevo od pravokutnika.

Među otvorenim karticama možete se kretati klikanjem na njihova imena, no pravo bogatstvo (posebno onima s većim monitorima) krije se iza klika desnom tipkom miša. Zahvaljujući tamo smještenim opcijama, radnu ćete površinu moći podijeliti na više dijelova i tako imati više istovremeno otvorenih kartica.

Klikanje desnom tipkom miša otkrit će i razliku između otvorenih datoteka i pomoćnih prozora kao što su Object Browser ili Start Page. Potonje, naime, možete sakriti, pretvoriti u plutajući prozor ili smjestiti na rub Visual Studijeva sučelja i tako u potpunosti prilagoditi razvojnu okolinu svojim željama i potrebama.

Ime izbornika	Opis
File	otvaranje projekata, rješenja i datoteka, zatvaranje istih, ispis na štampač, izlaz iz programa
Edit	radnje s međuspemnikom (<i>clipboard</i>) i pretraživanje
View	prikazivanje i uklanjanje prozora i alatnih traka te upravljanje istima
Project	upravljanje projektom, dodavanje formi, kontrola, komponenti, klasa
Build	kompajliranje programa
Debug	naredbe vezane uz <i>debugiranje</i> programa i njegovo pokretanje
Data	rad s bazama podataka
Format	naredbe za uređivanje i mijenjanje parametara kontrola na formi
Tools	razni dodatni alati i mogućnosti za prilagođavanje sučelja (sve za što nije bilo mjesta drugdje)
Windows	slaganje i upravljanje otvorenim prozorima
Help	pristup sustavu pomoći

Tablica 4-2:
Kratki vodič kroz organizaciju glavnog izbornika

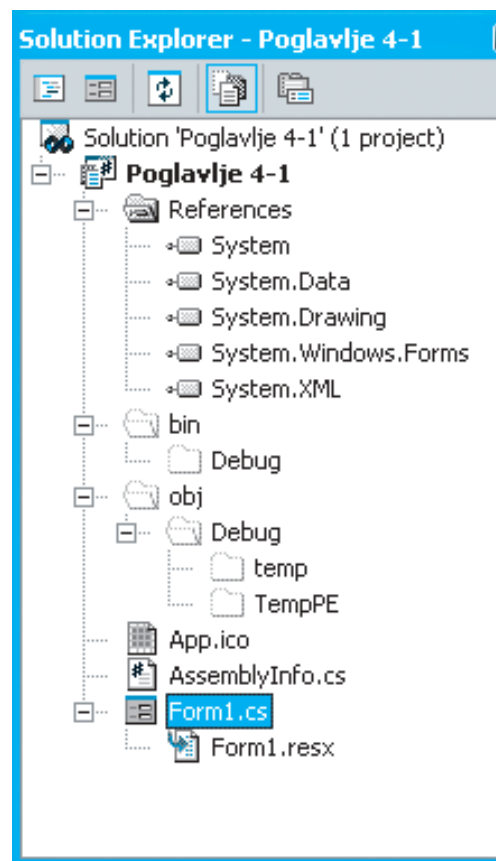
I. DIO: .NET IZNUTRA

Solution Explorer

Jedno rješenje sastoji se od velikog broja datoteka i drugih resursa, čak i kada se radi o najjednostavnijim aplikacijama. Da vam snalaženje među njima, dodavanje novih, uklanjanje postojećih i navigacija općenito budu što jednostavniji brine se Solution Explorer.



Slika 4-6:
: *Solution Explorer – tek smo stvorili rješenje, a već ima toliko resursa*



Ovisno o tipu datoteke odnosno resursa koji je označen, traka s alatima na vrhu prozora mijenjat će dostupne mogućnosti. Primjerice, ukoliko označite projekt, klikom na ikonu "Properties", dobit ćete prozor sa svojstvima projekta, a ukoliko označite neku datoteku s kôdom, moći ćete je otvoriti u kodnom ili dizajnerskom načinu. Uostalom – isprobavajte, jedino što se može dogoditi jest da si zatrpate radnu površinu prozorima, no to je lako ugasi.

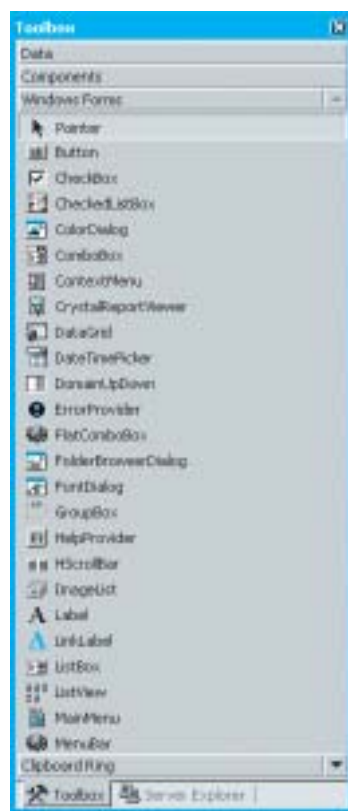
4. POGLAVLJE: RAZVOJNA OKOLINA

Kao što smo već spomenuli, sve pomoćne prozore, pa tako i ovaj, možete smjestiti negdje drugdje, ako vam tako bolje paše. Nismo spomenuli mogućnost "Auto Hide", koja će se posebno svidjeti vlasnicima manjih monitora. Ona pomoćne prozore svodi na usku traku uz rub, koja se, ukoliko vam funkcija iz prozora zatreba, otvara klikom miša, a nakon korištenja automatski zatvara. Mogućnost "Auto Hide" uključujete klikom na ikonu pribadače ili izborom adekvatne stavke iz padajućeg izbornika, koji se pojavljuje nakon što desnom tipkom miša kliknete na ime prozora.



Toolbox

Pomoćni prozor Toolbox svoje najkorisnije lice pokazuje u dizajnerskom načinu rada. U njemu se, naime, nalaze kontrole koje možete uzimati i odvlačiti na formu s njihove desne strane. Na taj način možete dodati tekst, kućicu za upis, padajući izbornik i još mnogo toga. Kako smo se prilikom kreiranja projekta odlučili za stvaranje prozorske aplikacije, kontrole će biti prilagođene našem izboru. Da smo, primjerice, izabrali predložak za web-aplikaciju, izbor kontrola bio bi sasvim drugačiji.



Slika 4-7:
Pomoćni prozor Toolbox predstavlja bit vizualnog programiranja.

I. DIO: .NET IZNUTRA

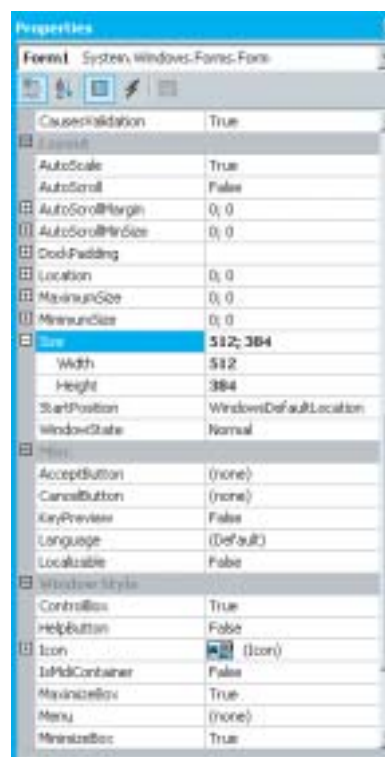
Ovakvo vizualno programiranje izuzetno olakšava stvaranje aplikacija. Svako odvlačenje kontrole na formu rezultirat će prilično velikim komadom kôda koji, kao što smo već spomenuli, ne samo da ne morate znati napisati nego ne morate niti vidjeti.

Dostupnih kontrola ima puno, pa su zbog jednostavnijeg snalaženja poslagane u nekoliko grupa (Data, Components, Windows Forms, General...). I tu do izražaja dolazi prilagodljivost sučelja – grupe možete kreirati, brisati, sortirati i preslagivati: ako vam se ne sviđa postojeći raspored, slobodno napravite organizaciju koja vam se čini efikasnijom. Osim grupama, iste stvari možete raditi i kontrolama, no o tome neki drugi put. Ukoliko ste željni samostalnog isprobavanja, ključni prvi korak je klik desnom tipkom miša...

Svojstva (*Properties*)

Kad postavite neku kontrolu na stranicu, poželjet ćete promijeniti neka njezina svojstva. Formi ćete najvjerojatnije mijenjati veličinu, kućici za upis poziciju unutar forme, padajućem izborniku ponuđene vrijednosti... Sve je to, dakako, moguće napraviti izravnom intervencijom u kôdu, no čemu se patiti kad postoji pomoćni prozor *Properties*.

Slika 4-8:
Svojstva kontrola jednostavno se podešavaju u pomoćnom prozoru *Properties*.



4. POGLAVLJE: RAZVOJNA OKOLINA

Kao što i samo ime govori, on sadrži listu svih svojstava kontrole koja se mogu mijenjati (postoje, naime, svojstva čije vrijednosti nije moguće mijenjati). Želite li promijeniti svojstvo neke kontrole, prvo je trebate označiti, bilo mišem u dizajnerskom načinu, bilo izborom iz liste kontrola u samom prozoru Properties (vidi sliku 4-8).

Nakon toga u listi se pojavljuju svojstva specifična za odabranu kontrolu. Svojstva možemo pregledavati po abecedi (zgodno uvijek kada točno znate naziv svojstva koje tražite) ili po grupama (kada ne znate točan naziv svojstva, no znate čemu služi).

Primijetit ćete da su neki nazivi svojstava napisani u zagradi. To znači da to nisu svojstva kontrole, nego neke druge vrijednosti vezane uz nju. Najpoznatiji primjer takvog "svojstva" je identifikacijsko ime kontrole – označeno nazivom Name.



Organizacija liste svojstava je jednostavna. Na lijevoj strani su nazivi svojstava, a na desnoj pripadajuće vrijednosti. Želite li promijeniti neku vrijednost, možete je ručno upisati ili odabrati iz padajućeg izbornika, ukoliko isti postoji. Neka složenija svojstva umjesto padajućeg izbornika imaju zasebne pomoćne prozore koje otvaramo klikom na gumbić označen točkicama (s konkretnim primjerima upoznat ćemo se kasnije u poglavlju i knjizi). Vrijednosti koje promijenite bit će deblje ispisane nego vrijednosti koje niste dirali, što u praksi značajno doprinosi snalaženju među svojstvima.

Događaji (Events)

Osim svojstava, svaka kontrola ima i događaje. Događaji su trenuci u kojima se dogodi nekakva promjena vezana uz kontrolu. Primjerice, kada u kućicu za upis upišemo neki tekst, nastaje događaj TextChanged. Ukoliko tom događaju pridružimo neku funkciju, ta će se funkcija izvršiti svaki put kada nastane taj događaj. Prenesimo to na naš primjer – svaki put kada korisnik programa izmijeni tekst u kućici za upis, nastat će događaj TextChanged. Ako za njega vežemo funkciju, onda će ona biti izvršena svaki put kada korisnik izmijeni sadržaj kućice za upis.

Listu događaja neke kontrole možemo pronaći u istom prozoru u kojem i svojstva, samo je potrebno kliknuti na ikonu munje. Vezanje funkcije uz neki događaj je jednostavno – u popisu pronađete događaj koji želite vezati, dvokliknete na stupac lijevo od naziva događaja i Visual Studio će vam u središnjem dijelu sučelja otvoriti kôd programa i kreirati kostur funkcije. Vama jedino preostaje napisati sadržaj funkcije, no o tome nešto kasnije...

I. DIO: .NET IZNUTRA

Vaš drugi program

Dosta razgledavanja, krenimo na konkretan primjer! Vjerujemo da ste korak stvaranja novog projekta već odradili te vas čeka prazna forma. Kako se inicijalno radi o formi skromnih dimenzija, prvi će nam zadatak biti njeno povećanje. To možete napraviti na dva načina. Vizualno – jednostavno razvući formu do veličine koju želite ili mijenjanjem svojstava `Size` u pomoćnom prozoru `Properties`.

Kad smo već kod svojstava, promijenit ćemo još jedno. To će biti naslov forme (ne brkajte ga s imenom kontrole!) – svojstvu `Text` upišite vrijednost “Naš drugi program” (bez navodnika). Primijetit ćete da se naslov prozora odmah ispravio i u dizajnerskom načinu.

Nakon toga dolazi vrijeme da u prozor smjestimo i određene kontrole iz pomoćnog prozora `Toolbox`. Otvucite s njega kontrole `Label`, `PictureBox` i `Button` kako biste kreirali inačice tih kontrola na formi. Uočite da su se kreirane inačice kontrola automatski nazvale *label1*, *pictureBox1* i *button1*.



Imena kreiranih kontrola možete promijeniti, no bitno je da to učinite (po mogućnosti) odmah nakon kreiranja. Naime, ime kontrole koristi se u kôdu svaki put kada želite nešto s njom napraviti (a u praksi je to vrlo često), pa ako ime promijenite naknadno, sve će reference na kontrolu biti pogrešne.

Kako bi netom postavljene kontrole imale neku ulogu, potrebno im je promijeniti svojstva. Tako ćemo prvoj kontroli (*label1*) promijeniti tekst koji prikazuje na ekranu. Umjesto postojećeg “label1” svojstvu `Text` upisat ćemo “Dobro došli u naš drugi program!”. Kako je tekst predugačak da bi stao u jedan redak, prelomit će se u drugi, a da bismo to spriječili povećat ćemo širinu kontrole razvlačenjem. Također, pošto se radi o naslovu, promijenit ćemo veličinu slova kojom je tekst ispisan (unutar svojstva `Font`, redak `Size`).

Drugoj, slikovnoj kontroli dodat ćemo sliku. Označite je i među svojstvima potražite `Image`. Klikom na taj redak pojavit će se sitni gumb s tri točkice, na koji valja stisnuti da bi se otvorio prozor pomoću kojeg ćemo izabrati sliku s diska. Kako odabrana slika vrlo vjerojatno ne stane u kontrolu koju smo joj namijenili, potražite svojstvo `SizeMode` i postavite ga na vrijednost `StretchImage`. To će “natjerati” sliku da se prilagodi veličini kontrole.

Ostao nam je gumb koji ćemo zadužiti da, kada bude kliknut, promijeni natpis u prvoj kontroli. Prvo ćemo mu promijeniti natpis (svojstvo `Text`) u “Promijeni naslov” te ga proširiti kako bi novi natpis bio vidljiv. Nakon toga treba događaju koji nastupa kad je gumb pritisnut pridružiti funkciju koja će izmijeniti natpis prvoj kontroli. Kako se radi o *defaultnom* događaju, nije potrebno

4. POGLAVLJE: RAZVOJNA OKOLINA

ulaziti na listu događaja za taj objekt, već u dizajnerskom načinu treba dva puta kliknuti na gumb. Kao što smo već rekli, automatski će se generirati kostur funkcije unutar kojega treba dodati naredbu za promjenu natpisa:

```
private void button1_Click(object sender, System.EventArgs e)
{
    label1.Text = "Klik je promijenio tekst!";
}
```



Slika 4-9:
Ovako izgleda naš primjer u
dizajnerskom načinu sučelja.

Treći redak našeg primjera nalaže da se svojstvu Tekst kontrole *label1* pridruži izraz “Klik je promijenio tekst!”. Nemojte nam vjerovati na riječ – isprobajmo zajedno!

Sjetite se komplikacija oko kompajliranja iz prošlog poglavlja – ovdje se sve rješava pritiskom na tipku F5 ili izborom opcije Start iz izbornika Debug. Nekoliko trenutaka kasnije pokrenut će se aplikacija koju smo upravo napisali.

Kliknite na gumb “Promijeni naslov” i uočite što će se dogoditi. Kako se radi o sasvim jednostavnom programu, to je sve što ćete u njemu moći napraviti. Međutim, svrha ovog primjera nije bila velika funkcionalnost, nego da se upoznate s osnovnim principima i mogućnostima. Morate priznati, zaista je jednostavno!

I. DIO: .NET IZNUTRA

Slika 4-10:
Naš primjer – pokrenut i
funkcionalan



Imate li volje, možete se samostalno igrati sa svojstvima – većina njih je intuitivna i brzo ćete poloviti kako koje svojstvo utječe na kontrolu kojoj pripada. Naravno, možete i pričekati osmo poglavlje, koje je posvećeno prozorskim aplikacijama i usto puno detaljnije obrađuje kontrole i njihova svojstva.

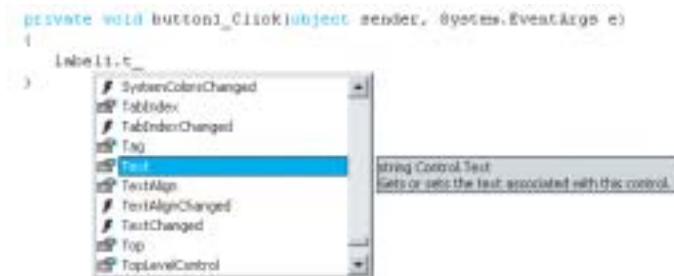
Snalaženje u kodu

Pod ovim naslovom cilj nam je ukratko izdvojiti neke zanimljive i korisne mogućnosti Visual Studio-jeva sučelja vezane uz snalaženje u kôdu. Radi se o sitnicama koje se u praksi pokazuju vrlo praktičnima i uvelike olakšavaju pisanje programa.

Najprije svakako valja spomenuti bojanje kôda – tako će, primjerice, u crnom tekstu ključne riječi biti ispisane plavom bojom, a komentari zelenom. Tu je i nezaobilazno poravnavanje kôda, koje će se u većini slučajeva aplicirati automatski, čak i prilikom *copy-paste*anja. Dakako, svu je automatiku moguće isključiti, a postoji mogućnost da uvlačenje umjesto znaka *tab* ubacuje razmake (Tools > Options > Text Editor > All languages > Tabs).

Mogućnost koju ste sigurno već sreli u marketinškim materijalima pod imenom IntelliSense omogućava automatsko dopunjavanje započetih izraza. Vjerojatno ste i sami za vrijeme pisanja

4. POGLAVLJE: RAZVOJNA OKOLINA



Slika 4-11:
IntelliSense – vjerojatno
najkorisnija mogućnost
Visual Studijeva sučelja

funkcije u prošlom primjeru primijetili da se, nakon što ste otipkali ime kontrole i stavili točku, otvorila padajuća lista sa svim svojstvima, metodama i inim stvarima vezanim uz tu kontrolu. Tada ste, umjesto tipkanja, jednostavno mogli izabrati željeni pojam s liste i tako brže (i, što je još važnije, točnije) napisati naredbu. Kao što možete vidjeti na slici 4-11, osim dopunjavanja započete riječi, IntelliSense nudi i kratki opis svakog pojma.

```
private void button1_Click(object sender, System.EventArgs e)
{
    label1.Text = "Klik je promijenio tekst!";
}

private void button1_Click(object sender, System.EventArgs e)
{
    label1.Text = "Klik je promijenio tekst!";
}
```

Slika 4-12:
Brza identifikacija poj-
mova u kodu

Kratki opis pojmova na stranici nije specifičan samo za IntelliSense. Ukoliko tijekom pregledavanja kôda naiđete na nepoznat pojam, vjerojatno će pomoći ako iznad njega zaustavite pokazivač miša na nekoliko trenutaka. Na slici 4-12 možete vidjeti dvije takve situacije. U prvoj nastojimo identificirati ime kontrole i Visual Studio nam javlja da se radi o kontroli tipa System.Windows.Forms.Label te da se ona nalazi na formi Form1. U drugoj situaciji pokazivač miša držimo nad svojstvom Text i saznajemo koji tip vrijednosti mu valja pridružiti (string), na koji tip kontrole se odnosi (riječ Control znači da se odnosi na sve kontrole), te čak pruža kratak opis svojstva.

```
private void button1_Click(object sender, System.EventArgs e)
{
    label1.Text = "Klik je promijenio tekst!";
}
```

Slika 4-13:
Ukazivanje na greške u
sintaksi

I. DIO: .NET IZNUTRA

Na slici 4-13 prikazan je jednostavan primjer kako Visual Studio ukazuje na greške u sintaksi. Crvenom valovitom crtom označeno je mjesto na kojem sasvim izvjesno nedostaje znak točka-zarez. Dakako, ovakva pomoć uskočit će i u nešto složenijim primjerima.

Slika 4-14:
Pomoćni prozor
Task List

```
private void button1_Click(object sender, System.EventArgs e)
{
    label1.Text = "Klik je promijenio tekst!";
    // TODO: Napisati primjer za knjigu
}
```



Pozabavimo se na kraju malo i pomoćnim prozorom Task List. On je zamišljen kao mjesto gdje programer može zapisati stvari koje još mora napraviti, ali i gdje će kompajler zapisati greške i upozorenja na koje naiđe. Međutim, ovdje ga spominjemo zbog zanimljiva načina pomoću kojega možete dodavati zadatke u prozor. Dovoljno je napisati komentar u ovom formatu:

```
// TODO: Tekst koji želimo da se pojavi u pomoćnom prozoru
```

...i on će se pojaviti u pomoćnom prozoru. (Ukoliko se ne pojavi, desnom tipkom miša kliknite na listu zadataka i postavite opciju Show Tasks na vrijednost All.) Osim riječi TODO, u konfiguraciji programa (Tools > Options > Environment > Task List) možete dodati i druge ključne riječi tog tipa, tzv. tokene.

Već smo spomenuli da se automatski generiran kôd skriva iza pravokutnika s natpisom "Windows Form Designer generated code". Želite li i vi sakriti dio kôda na isti način, potrebne su vam ključne riječi "#region" i "#endregion". Evo primjera:

```
#region Neke funkcije
private void button1_Click(object sender, System.EventArgs e)
{
    // neki kôd
}
#endregion
```


I. DIO: .NET IZNUTRA

Tekstovi dostupni u MSDN Libraryju ne svode se samo na specifikacije već i na velik broj primjera, stručnih članaka, vodiča i tekstova u raznim drugim formama. Sve je tekstove moguće pretraživati, no kako se radi o izuzetno velikoj količini informacija, puno se bolji rezultati postižu navigacijom kroz dostupne kategorije.

Kategorija u kojoj ćete kao programer u .NET-u najviše prebivati jest “.NET Development” (vidi sliku 4-15), pa preporučujemo da je razgledate i steknete dojam kako je organizirana. Takva “ogledna šetnja” olakšat će vam kasnije snalaženje, kada budete rješenja rješenja za konkretne probleme.

Tablica 4-3:
Putanje do nekih
važnijih tema u
MSDN Libraryju

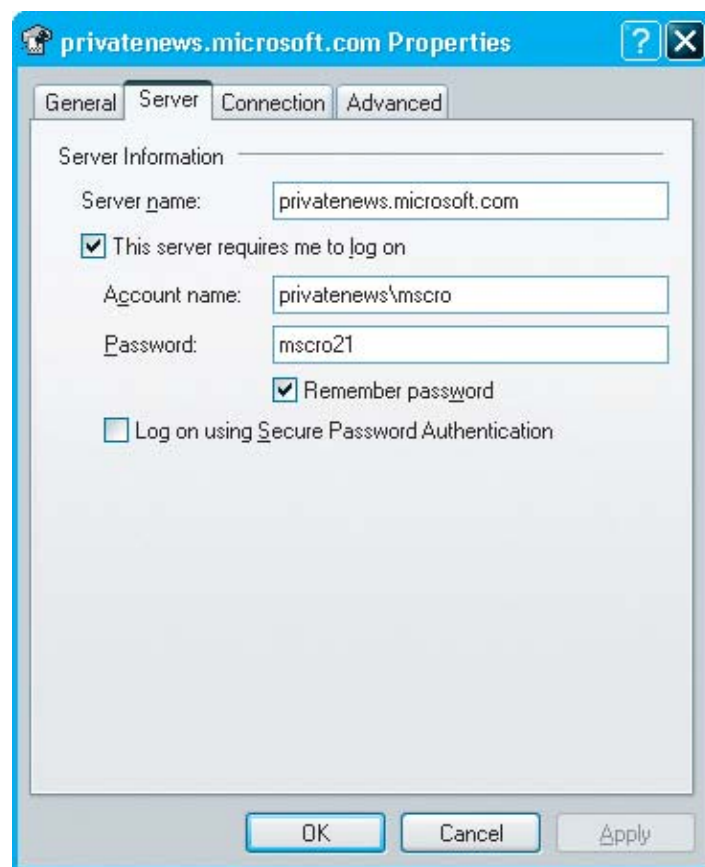
Tema	Putanja do teme
Uvod u .NET Framework	.NET Development > .NET Framework SDK > .NET Framework
Referenca jezika C#	.NET Development > Visual Studio > Product Documentation > Visual Basic and Visual C# > Reference > Visual C# Language > C# Programmer's Reference
Bazna biblioteka klasa	.NET Development > .NET Framework SDK > .NET Framework > Reference > Class Library
Prozorske aplikacije	.NET Development > Windows Forms
Rad s bazama podataka (ADO.NET)	.NET Development > ADO.NET
Web-aplikacije (ASP.NET)	.NET Development > ASP.NET

Kad se udomaćite u MSDN Libraryju, zavirite i u ostatak stranica MSDN-a koje se nalaze na adresi <http://msdn.microsoft.com/>. Tamo ćete naći još hrpu tekstova, opisa, primjera, ideja, linkova i objašnjenja. Posebno valja izdvojiti dio nazvan Downloads, u kojem možete naći zanimljive i korisne dodatke za Visual Studio te veliku bazu primjera pod nazivom MSDN Code Center.

Osim u MSDN-u, dokumentaciju, primjere i pomoć možete tražiti na brojnim web-stranicama koje se bave tematikom vezanom uz .NET platformu. Ne valja podcijeniti niti potencijal grupa na Usenetu ([nntp://microsoft.public.dotnet.*](http://microsoft.public.dotnet.*)), a u praksi se posebno korisnim pokazalo pretraživanje arhiva tamo objavljenih članaka pomoću tražilice na adresi <http://groups.google.com/>.

Osim javnih Usenetskih grupa, postoje i “privatne”, i to na hrvatskom jeziku. Riječ “privatne” u ovom slučaju ne znači da njima ne možete pristupati, nego da im se mora pristupati preko posebnog poslužitelja. Detaljne upute i parametre za spajanje potražite na adresi <http://www.microsoft.com/croatia/support/newsgroups/>.

4. POGLAVLJE: RAZVOJNA OKOLINA



Slika 4-16:
Konfiguracija Outlook
Expressa za praćenje privatnih
hrvatskih Usenet grupa (para-
metri se povremeno mijenjaju,
za aktualno važeće posjetite
web-adresu u tekstu)

