DEO I

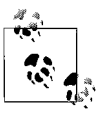
Jezik C#

POGLAVLJE 1

C# i .NET Framework

C# 2.0 je jednostavan, bezbedan, moderan, objektno orijentisan jezik visokih performansi, namenjen za pisanje .NET programa prilagođenih Internetu. C# je sada potpuno zreo jezik u koga je ugrađeno tridesetogodišnje iskustvo. Kao što se u deci prepoznaju osobine njihovih roditelja i daljih predaka, tako se jasno primećuju uticaji C++-a, Jave, Visual Basica (VB) i drugih jezika na C# kao i unapređenja u odnosu na prvu verziju jezika C#.

Centralna tema ove knjige je C# i korišćenje ovog jezika za programiranje na platformi .NET, naročito u okruženju Visual Studio .NET 2005 (potpuna ili Express verzija).



Veliki broj programa u ovoj knjizi su konzolne aplikacije (a ne aplikacije za Windows ili Web) kako bi u prvom planu bile odlike jezika a ne detalji korisničkog okruženja.

Ukoliko koristite Mono verziju C#-a ili verziju iza koje ne stoji Microsoft, ne brinite: programi iz knjige radiće bez problema, premda nismo ispitali nijednu verziju jezika sem zvanične, Microsoftove.

U ovom poglavlju predstavljamo jezik C# i platformu .NET, uključujući okruženje .NET Framework.

Platforma .NET

Kada je Microsoft u julu 2000. godine najavio C#, predstavljanje ovog jezika bilo je deo mnogo većeg događaja: najave platforme .NET. C# 2.0 je zrela verzija ovog jezika čije je pojavljivanje na programerskoj sceni usklađeno s pojavom naredne generacije alatki za .NET.

.NET je razvojna platforma s novim *interfejsom za programiranje aplikacija* (engl. *application programming interface*, API); nasledio je funkcionalnost i mogućnosti okruženja za programiranje klasičnih operativnih sistema Windows, ali i usvojio brojne, različite tehnologije koje Microsoft razvija od kraja devedesetih godina prošlog veka. To obuhvata rad s komponentama COM+ i XML-om, objektno orijentisan dizajn,

podršku za nove protokole za Web servise kakvi su SOAP, WSDL i UDDI, i orijentisanost ka Internetu; sve je to integrisano u okviru DNA arhitekture (Distrubuted interNet Applications).

Microsoft je mnogo resursa stavio u službu razvoja platforme .NET i srodnih tehnologija. Rezultati takve posvećenosti su zadivljujući. Kao prvo – opseg tehnologija koje obuhvata .NET je ogroman. Platformu čine sledeće tri grupe tehnologija:

- Skup jezika – uključujući C# i VB, skup alatki za razvoj programa (između ostalog, Visual Studio .NET), opsežna biblioteka klasa za pravljenje Web servisa i aplikacija za Web i Windows, te *zajedničko izvršno okruženje* (engl. *Common Language Runtime*, CLR) za izvršavanje koda objekata napravljenih u .NET Frameworku
- Dve generacije servera .NET Enterprise Server: postojeći, i oni koji će postati dostupni u naredne dve do tri godine
- .NET podrška koja nije samo za PC računare, već i za nove uređaje – od mobilnih telefona, do platformi za kompjuterske igrice

.NET Framework

Microsoft .NET podržava ne samo nezavisnost jezika, već i njihovu integraciju. To znači da je omogućeno nasleđivanje klasa, otkrivanje i obrada izuzetaka i polimorfizam između različitih jezika. .NET Framework to postiže pomoću specifikacije zvane sistem zajedničkih tipova (engl. *Common Type System*, CTS), koju moraju da poštuju sve komponente platforme .NET. Na primer, u okruženju .NET, sve je objekat neke klase koja se izvodi iz korene klase (engl. *root class*) zvane `System.Object`. CTS podržava opšti koncept klasa, interfejsa i delegata (koji podržavaju povratne pozive).

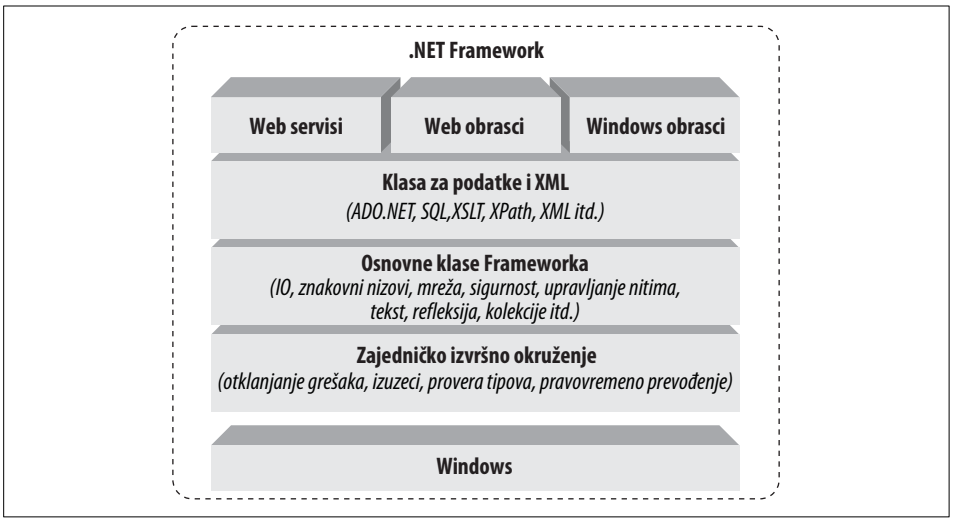
Pored toga, .NET obuhvata *zajedničku jezičku specifikaciju* (engl. *Common Language Specification*, CLS), koja sadrži niz osnovnih pravila neophodnih za integraciju jezika. CLS definiše minimalne zahteve koje jezik mora ispunjavati da bi pripadao porodici .NET jezika. Prevodioci (engl. *compilers*) usklađeni sa specifikacijom CLS prave objekte koji mogu međusobno da sarađuju. Svaki jezik koji poštuje CLS može da koristi čitavu biblioteku klasa Frameworka (engl. *Framework Class Library*, FCL).

.NET Framework je postavljen preko operativnog sistema (bilo kog Windowsa),* i sastoji se od većeg broja komponenata u koje trenutno spadaju:

- Pet zvaničnih jezika: C#, VB, Visual C++, Visual Java# i JScript.NET
- CLR, objektno orijentisana plarforma za razvoj Windows i Web aplikacija, zajednička za sve pomenute jezike
- Brojne srodne biblioteke klasa, objedinjene pod imenom biblioteka klasa Frameworka (FCL)

* Zahvaljujući arhitekturi CLR-a, to može da bude i bilo koji operativni sistem zasnovan na Unixu, ili neki potpuno drugačiji operativni sistem.

Na slici 1-1 prikazan je dijagram arhitektonskih komponenata .NET Frameworka.



Slika 1-1. Arhitektura .NET Frameworka

Najvažnija komponenta .NET Frameworka je CLR – okruženje u kome se programi izvršavaju. CLR obuhvata virtuelnu mašinu, na mnogo načina sličnu Javinoj virtuelnoj mašini. Na visokom nivou, CLR aktivira objekte, proverava njihovu bezbednost, upisuje ih u memoriju, izvršava ih i briše iz memorije kada više nisu potrebni. (Deo izvršnog okruženja CLR je i sistem zajedničkih tipova, CTS.)

Na slici 1-1, sloj na vrhu CLR-a je skup klasa Frameworka, ispod koga je sloj klasa za podatke i XML, a još dublje i sloj klasa za Web servise, Web obrasce i Windowsove obrasce. Sve ove klase sačinjavaju Frameworkovu biblioteku klasa (engl. *Framework Class Library*, FCL), jednu od najvećih biblioteka klasa ikada napravljenih, koja stavlja na raspolaganje objektno orijentisan API za sve funkcije i usluge koje .NET pruža. Sa više od 4.000 klasa, FCL olakšava brz razvoj aplikacija i usluga (servisa) za stone računare, sisteme klijent/server i Web.

Skup Frameworkovih osnovnih klasa, najniži nivo biblioteke FCL, sličan je skupu klasa u Javi. Ove klase podržavaju ulazno-izlazne operacije, rad sa znakovnim nizovima, upravljanje bezbednošću, mrežnu komunikaciju, upravljanje nitima, rad s tekstom, korišćenje refleksije i kolekcija itd.

Iznad ovog nivoa je sloj dodatnih klasa za upravljanje podacima i jezikom XML. Klase za podatke podržavaju sve postupke za upravljanje podacima iz pozadinskih baza podataka. One obuhvataju SQL klase pomoću kojih možete upravljati trajnim skladištima podataka preko standardnog SQL interfejsa. Platforma .NET podržava brojne klase za rad s podacima na XML-u, kao i za njihovo pretraživanje i prevođenje.

Povrh slojeva sa osnovnim klasama i klasama za podatke i XML, nalazi se sloj klasa za pravljenje aplikacija pomoću tri tehnologije: Web servisa, Web obrazaca i Windowsovih obrazaca. Web servisi koriste brojne klase za pravljenje jednostavnih, distribuiranih komponenata koje će raditi čak i tamo gde postoje mrežne barijere i softver za prevođenje mrežnih adresa (engl. *Network Address Translation*, NAT). Pošto Web servisi koriste komunikacione protokole HTTP i SOAP, pomenute komponente podržavaju sistem „utakni i koristi“ (engl. *Plug and Play*, PnP) u čitavom informatičkom prostoru.

Web obrasci (engl. *Web forms*) i Windowsovi obrasci (engl. *Windows forms*) omogućavaju primenu tehnologije RAD (Rapid Application Development) za pravljenje programa za Web i za Windows. Pisanje programa sada je krajnje jednostavno – prevucite kontrole u obrazac, dvaput pritisnite mišem kontrolu i napišite kôd koji odgovara pridruženom događaju.

Detaljniji opis .NET Frameworka možete naći u knjizi *.NET Framework Essentials* (O'Reilly).

Prevođenje i MSIL

U okruženju .NET, programi se ne prevode u izvršne datoteke, već u programske sklopove koji sadrže instrukcije na *Microsoftovom međujeziku* (engl. *Microsoft Intermediate Language*, MSIL), koje CLR prevodi na mašinski kôd i izvršava. MSIL (skraćeno IL) datoteke koje pravi C# skoro su identične IL datotekama nastalim u drugim .NET jezicima; za platformu je nevažno na kom je jeziku program napisan. Ključna odlika okruženja CLR jeste to da je zajedničko svim .NET jezicima – na isti način podržava pisanje programa na C#-u kao i na jeziku VB.NET.

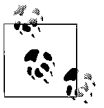
Kôd napisan na C#-u prevodi se na IL u fazi prevođenja (engl. *build*) projekta. Međujezički kôd se čuva u datoteci na disku. Kada pokrenete program, međujezički kôd se ponovo prevodi pomoću pravovremenog prevodioca (engl. *Just In Time compiler*, *JIT compiler*). (Taj proces se na engleskom često naziva *JITing*.) Kao rezultat, dobija se mašinski kôd koji izvršava procesor računara.

Standardni JIT prevodioci rade *po zahtevu* (engl. *on demand*). Kada se pozove metoda, JIT prevodilac analizira međujezički kôd i pravi vrlo efikasan mašinski kôd koji se veoma brzo izvršava. Dok se izvršava program, JIT prevodilac prevodi samo po potrebi, a prevedeni kôd se čuva u memoriji da bi se mogao ponovo iskoristiti. Tokom izvršavanja, .NET programi postaju sve brži, jer se koristi već preveden kôd.

Suština CLS-a je da svi .NET jezici proizvode vrlo sličan međujezički kôd. Zato svaki jezik može pristupati objektima napisanim na drugom jeziku i izvoditi nove objekte od njih. Dakle, moguće je napraviti osnovnu klasu na jeziku VB.NET i na C#-u iz nje izvesti novu klasu.

Jezik C#

C# je razoružavajuće jednostavan jezik, sa samo 80 rezervisanih reči i desetak ugrađenih tipova podataka, ali je veoma izražajan kada je potrebno implementirati moderne programske ideje. Podržava strukturirano, objektno orijentisano programiranje zasnovano na komponentama, što biste i očekivali od modernog jezika nastalog na osnovu Jave i C++-a. Verzija C# 2.0 sadrži mnoge od najvažnijih elemenata koji su nedostajali – na primer, generičke šablone (engl. *generics*) i anonimne metode.



Napomena programerima na C++-u: generički šabloni identični su šablonima u C++-u, ali su nešto jednostavniji i efikasniji; umanjuju gomilanje koda tako što se prilikom izvršavanja ponovo upotrebljava zajednički kôd, ali nauštrb fleksibilnosti karakteristične za šablone u C++-u.

Jezik C# stvorio je mali tim predvođen dvojicom cenjenih Microsoftovih inženjera, Andersom Hejlsbergom i Skotom Viltamutom. Hejlsberg je i tvorac Turbo Pascala, popularnog jezika za programiranje na ličnim računarima, i vođa tima koji je osmislio Borlandov Delphi – jedno od prvih uspešnih integrisanih razvojnih okruženja za programiranje klijentsko-serverskih aplikacija.

Srž svakog objektno orijentisanog jezika jeste podrška za definisanje klasa i rad s njima. Pomoću klasa definišete nove tipove, što omogućava da jezik proširite kako biste napravili što bolji model problema koji pokušavate da rešite. C# sadrži *rezervisane reči* (engl. *keywords*) za deklarisanje novih klasa i njihovih metoda i svojstava, kao i za *kapsuliranje* (engl. *encapsulation*), *nasleđivanje* (engl. *inheritance*) i *polimorfizam* (engl. *polymorphism*) – tri kamena temeljca objektno orijentisanog programiranja.

U jeziku C#, sve što se odnosi na deklaraciju klase nalazi se u samoj deklaraciji. Za definicije klasa nisu potrebne zasebne datoteke zaglavlja (engl. *header files*) niti datoteke na jeziku za definisanje interfejsa (engl. *Interface Definition Language*, IDL). Uz to, C# podržava novi XML stil umetanja dokumentacije što pojednostavljuje izradu referentne dokumentacije o programu prilagođene Webu i one spremne za štampanje.

C# podržava i *interfejse* – sredstvo za sklapanje ugovora s klasom o pružanju usluga predviđenih tim interfejsom. U jeziku C#, klasa sme da nasleđuje samo od jednog roditelja, ali može da implementira više interfejsa. Kada implementira interfejs, C# klasa se obavezuje da će izvršavati ono što interfejs predviđa.

C# podržava i *strukture*, koje su se značajno izmenile od C-a i C++-a. Strukture na C#-u su ograničeni, ne previše zahtevni tipovi čije instance manje opterećuju operativni sistem i memoriju od klasa. Struktura se ne može izvesti iz klase i obrnuto, ali se pomoću strukture može implementirati interfejs.

C# u potpunosti podržava *delegate* koji omogućavaju indirektno pozivanje metoda. U nekim drugim jezicima, to je drugačije rešeno (na primer, pomoću pokazivača na funkcije u C++-u), ali delegati su provereni referentni tipovi koji kapsuliraju metode sa određenim potpisima i tipovima rezultata.

C# podržava komponentno orijentisane elemente, poput svojstava, događaja i sastavnih delova deklaracija (na primer, *atributa*). Komponentno orijentisano programiranje podrazumeva čuvanje metapodataka u okviru koda klase. Metapodaci opisuju klasu, uključujući njene metode i svojstva, bezbednosne oznake i druge atribute (koji, na primer, određuju da li se klasa može serijalizovati); kôd sadrži logiku neophodnu za izvršavanje funkcija klase. To znači da prevedena klasa sadrži sve informacije o sebi. Zato u izvršnom okruženju programa koje zna kako da čita metapodatke klase i kôd, nisu potrebne druge informacije da bi se koristila ta klasa. C# i CLR omogućavaju da korisnik sam klasi doda metapodatke pomoću namenskih atributa. Korisnik može i da čita metapodatke o klasi koristeći CLR tipove koji podržavaju refeksiju.

Kada prevodite kôd, pravite *programski sklop* (engl. *assembly*). To je skup datoteka koje programer vidi kao jednu datoteku s dinamičkim povezivanjem (DLL), ili kao izvršnu datoteku (EXE). U okruženju .NET, programski sklop je osnovna jedinica za ponovno korišćenje koda, pravljenje više verzija, bezbednost i primenu. CLR podržava brojne klase za upravljanje programskim sklopovima.

Na kraju, recimo da C# podržava i sledeće:

- Direktan pristup memoriji pomoću pokazivača u stilu C++-a
- Rezervisane reči za izdvajanje nesigurnih operacija
- Upozoravanje skupljača smeća okruženja CLR da ne uništava objekte na koje pokazuju pokazivači dok se ti objekti ne oslobode