DEO 1

Uvod u .NET

POGLAVLJE 1

.NET okruženje

Microsoft već duže vremena drži reputaciju kreatora novih tehnologija kojima dodeljuje takve nazive koji zbunjuju korisnike. .NET Framework je najsvežiji primer - ovaj sistem je često opisivan kao nedorasli Java klon, besmisleni marketinški termin ili pokušaj da se primenom zaštićene tehnologije preuzme čitav Internet. Nijedan od navedenih opisa, međutim, nije potpuno tačan.

.NET je, praktično, skup različitih tehnologija - neke od njih su revolucionarne, dok druge nisu - koje su razvijene kao pomoć programerima u kreiranju najrazličitijih aplikacija. Pomoću okruženja .NET Framework programeri mogu kreirati složene Windows aplikacije, servise, pa čak i alatke koje rade u režimu komandne linije. Naravno, čitaoci ove knjige su verovatno najviše zainteresovani za kreiranje web aplikacija pomoću .NET-a. U tom cilju ćemo koristiti samo jedan deo .NET okruženja - ASP.NET i jedan od osnovnih .NET jezika: C#.

U ovom poglavlju ćemo upoznati tehnologije koje sačinjavaju .NET. Na početku ćemo prikazati istoriju web programiranja i objasniti motive nastanka .NET Frameworka. Nakon toga ćemo dati opšti prikaz različitih komponenti .NET sistema i videti kako se ASP.NET 3.5 uklapa u takvu opštu sliku.

Evolucija web programiranja

Internet je nastao krajem šezdesetih godina prošlog veka kao rezultat jednog eksperimenta. Cilj eksperimenta je bio stvaranje potpuno elastične informacione mreže - mreža koja bi nastavila sa radom i u slučaju gubitka jednog broja računara, bez uticaja na komunikaciju preostalih računara. Finansijer eksperimenta je bilo američko ministarstvo odbrane, koje je razmatralo posledice eventualnog scenarija katastrofe (poput nuklearnog napada).

Prvobitni Internet je bio ograničen na obrazovne i odbrambene institucije. Mreža se razvila u akademsku alatku pomoću koje su naučnici širom sveta mogli da razmenjuju informacije. Početkom devedesetih godina napravljeni su i modemi koji su mogli da komuniciraju preko postojećih telefonskih linija, nakon čega je Internet postao dostupan i komercijalnim korisnicima. 1993. godine kreiran je i prvi HTML web čitač, sa čime je započela i prava Internet revolucija.

HTML i HTML formulari

Prvi web sajtovi bi se teško mogli nazvati web aplikacijama. Umesto toga, prvi sajtovi su više podsećali na obične brošure sa nekoliko fiksnih HTML stranica, čiji sadržaj je ažuriran ručno.

Elementarna HTML stranica donekle podseća na dokument nastao u programu za obradu teksta - na njoj se nalazi formatirani sadržaj koji se može prikazati na računaru, ali takva stranica, u suštini, ne obavlja nikakav posao. Naredni primer prikazuje najjednostavniji HTML kod koji definiše dokument sa zaglavljem i jednim redom teksta:

```
<html>
    <head>
        <title>Sample Web Page</title>
    </head>
    <body>
        <h1>Sample Web Page Heading</h1>
        <p>This is a sample web page.</p>
    </body>
</html>
```

Sadržaj HTML dokumenta sačinjavaju dve osnovne komponente: tekst i elementi (tagovi, ili markeri) koji definišu način formatiranja teksta. Elemente je lako prepoznati, zato što su smešteni između znakova "više/manje od" (< >). HTML definiše elemente za različite nivoe naslova, pasuse, hiperlinkove, italik i bold format, horizontalne linije, itd. Fragment <h1>Probni tekst</h1>, na primer, koristi element <h1>. Zahvaljujući njemu, Probni tekst će biti prikazan sa stilom Heading 1, koji koristi povećani bold font. Fragment Ovo je primer web stranice. , na sličan način, kreira pasus koji sadrži jedan red teksta. Element <head> grupiše sve podatke zaglavlja (engl. header), uključujući i naslov koji se pojavljuje u prozoru web čitača, dok element <body> grupiše pravi sadržaj dokumenta koji se prikazuje u prozoru čitača.

Slika 1.1 prikazuje HTML stranicu koja odgovara gornjem kodu. Za sada je to fiksna datoteka (njen naziv je sample_web_page_heading.html) sa HTML sadržajem. Stranica ne obezbeđuje nikakvu interaktivnost, ne zahteva web server i sigurno se ne može svrstati u web aplikaciju.

SLIKA 1.1 *Običan HTML: stil "brošura"*

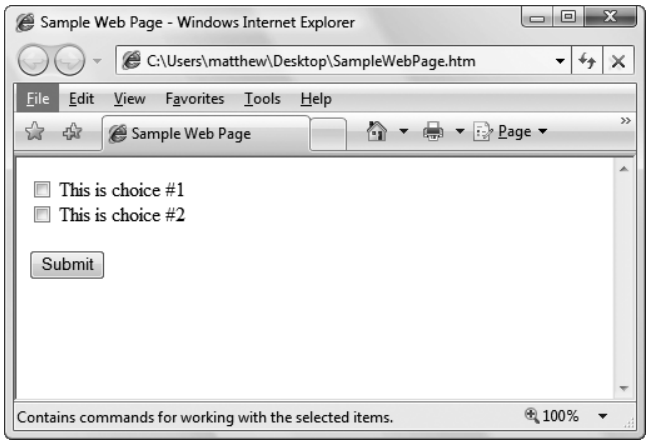
SAVET

Za kreiranje ASP.NET stranica nije neophodno poznavanje HTML jezika, mada osnovna znanja mogu biti korisna. Ako želite da brzo upoznate osnove HTML-a, pročitajte jedan od najboljih HTML priručnika na Interntu, na adresi www.w3schools.com/html. Četvrto poglavlje ove knjige takođe sadrži skraćeni uvod u HTML.

HTML 2.0 je uveo osnovne elemente web programiranja kroz tehnologiju koja se naziva HTML formulari (engl. forms). HTML formulari proširuju jezik tako što pored tagova za formatiranje uvode i tagove za grafičke dodatke, tzv. kontrole. Ove kontrole obuhvataju klasične elemente, kao što su padajuće liste, tekstualna polja i komandna dugmad. Sledeći primer opisuje web stranicu sa HTML formularom u kome se nalaze određene kontrole.

```
<html>
    <head>
        <title>Sample Web Page</title>
    </head>
    <body>
        <form>
            <input type="checkbox" />
             This is choice #1<br />
            <input type="checkbox" />
             This is choice #2<br /><br />
            <input type="submit" value="Submit" />
        </form>
    </body>
</html>
```

Sve kontrole u okviru HTML formulara smeštene su između tagova <form> i </form>. Prethodni primer sadrži dva kontrolna polja (definisani elementom <input type="checkbox" /> i komandno dugme (definisano elementom <input type="submit" />). Element
 ubacuje prelom reda. Izgled stranice u web čitaču prikazan je na slici 1-2.



SLIKA 1.2 *HTML formular*

Zahvaljujući HTML formularima web programeri mogu kreirati standardne stranice za unos podataka. Kada korisnik klikne na dugme Submit sa slike 1.2, svi podaci sa kontrola za unos podataka (dva kontrolna polja, u našem primeru) biće spakovani u jedan dugačak tekstualni string i poslati na web server. Aplikacija na strani servera prima i obrađuje takve podatke.

Zanimljivo je da kontrole koje su se koristile za HTML formulare pre deset i više godina i danas predstavljaju osnovu za kreiranje dinamičkih ASP.NET stranica! Razlika je u vrsti aplikacija koje se izvršavaju na serveru. Kada bi korisnik u prošlosti kliknuo na dugme na stranici-formularu, podaci sa stranice su slati putem e-pošte do administratora naloga ili do aplikacije na serveru koja je koristila ambiciozni CGI (Common Gateway Interface) standard. Današnji korisnici rade sa znatno moćnijim i elegantnijim ASP.NET platformama.

Serverski programski model

Da bismo potpuno shvatili motive za kreiranje ASP.NET sistema, moramo poznavati probleme sa kojima su se susretale rane tehnologije web programiranja. Web serveri koji su koristili prvobitni CGI standard morali su da pokrenu novu, zasebnu instancu aplikacije za svaki primljeni zahtev. Serveri na popularnim web sajtovima su na taj način morali da se izbore sa stotinama odvojenih kopija neke aplikacije, tako da su često postajali žrtve sopstvenog uspeha. Pored toga, CGI i slične tehnologije nude samo rudimentarno programersko okruženje. Operacije višeg nivoa, kao što su identifikacija korisnika, skladištenje ličnih podataka ili prikaz slogova izdvojenih iz baze podataka, zahtevale su pisanje obimnog koda "od nule". Takvo kreiranje web aplikacija je svakako veoma naporno i podložno greškama.

Da bi rešio navedene probleme, Microsoft je kreirao razvojne platforme višeg nivoa, kao što su ASP i ASP.NET. Ove tehnologije omogućuju kreiranje dinamičkih web stranica bez upuštanja u implementacione detalje nižeg nivoa. Zahvaljujući tome, obe platforme su doživele veliki uspeh.

Prva verzija ASP platforme privukla je veliku pažnju ogromne grupe od skoro milion programera, postižući uspeh koji je prevazišao čak i Microsoftova očekivanja. Nakon veoma kratkog vremena ova platforma je našla svoju primenu u najrazličitijim oblastima, od kritičnih poslovnih aplikacija do sajtova koji su se bavili elektronskom trgovinom. Dizajn ASP-a, međutim, nije razvijan za takve oblasti, tako da su se veoma brzo pojavili problemi u vezi performansi, zaštite i konfigurisanja sistema.

To su upravo pitanja koja su u prvi plan izbacila ASP.NET. ASP.NET je razvijan kao industrijsko aplikativno okruženje koje će prevazići ograničenja ASP-a. U poređenju sa klasičnim ASP-om, ASP.NET je stekao široku popularnost odmah nakon objavljivanja - tačnije, ovo okruženje je primenjeno na nekoliko velikih komercijalnih web sajtova još dok je bilo u beta verziji.

NAPOMENA

Bez obzira na zajedničke korene, ASP i ASP.NET su potpuno različiti sistemi. ASP je programski jezik zasnovan na skriptovima, koji zahteva dobro poznavanje HTML jezika i dosta detaljnog programiranja. ASP.NET je, sa druge strane, objektno-orijentisani programski model u kome kreiranje web stranice nije ništa teže od pisanja obične Windows aplikacije. Dodatni tekst u okviru pod naslovom "Različita lica ASP.NET-a", u nastavku ovog poglavlja, daje detaljniji opis različitih verzija ASP.NET-a.

Klijentski programski model

Dok je programiranje na strani servera krčilo svoj put kroz skup novih tehnologija, još jedan stil programiranja je sticao sve više popularnosti. Programeri su sve više eksperimentisali sa različitim tehnikama koje su poboljšavale njihove web stranice, ugrađujući na njih minijaturne aplete razvijene u JavaScritpu, ActiveX, Javi i Flashu. Navedene tehnologije namenjene klijentima nisu zahtevale nikakvo angažovanje servera. Umesto toga, korisnik bi preuzeo kompletnu aplikaciju i prebacio je u svoj web čitač, gde se ona izvršavala u lokalu.

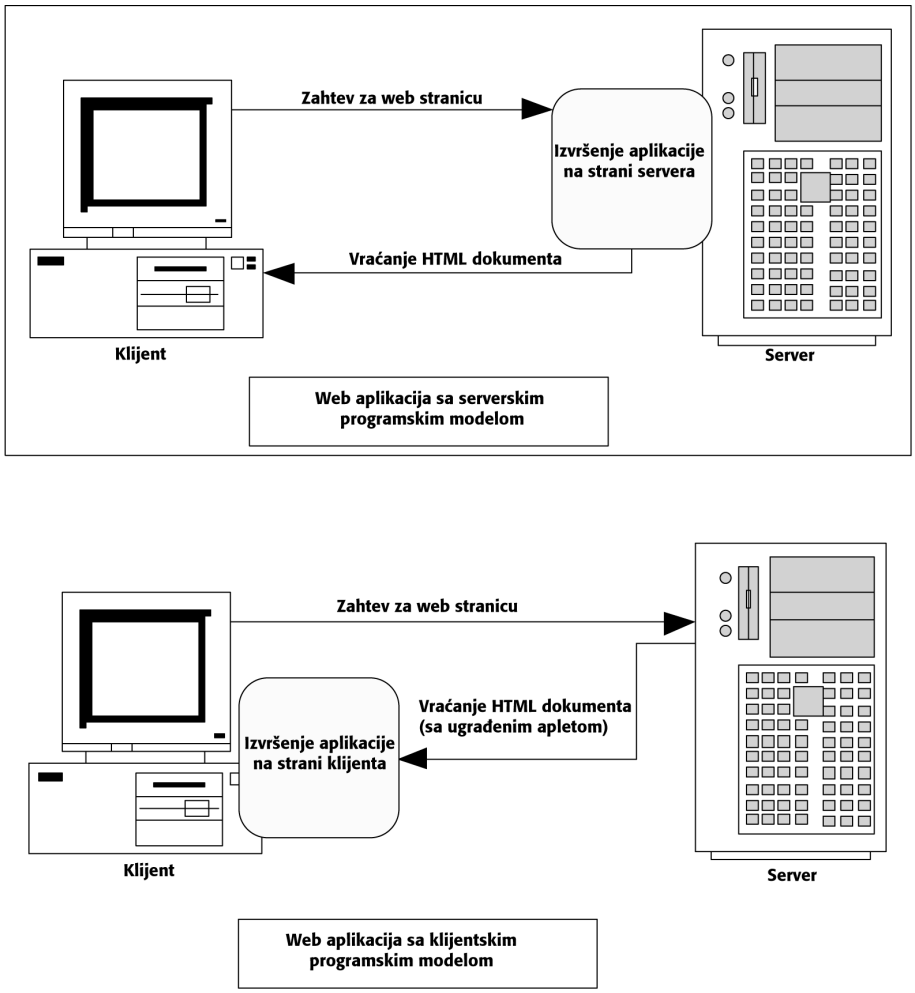
Osnovni problem sa klijentskim tehnologijama ogledao se u činjenici da one nisu bile podjednako podržane u svim web čitačima i u svim operativnim sistemima. Web programiranje svoju popularnost prvenstveno duguje činjenici da web aplikacije ne zahtevaju nikakve instalacione CD-ove, preuzimanje programa sa Interneta i druge zamorne instalacione postupke (koji su ujedno podložni i greškama). Umesto toga, web aplikacije se mogu jednostavno izvršavati na svakom računaru koji ima pristup Internetu. Primenom navedenih klijentskih tehnologija, međutim, programeri nailaze na nekoliko standardnih problema koji mogu izazvati glavobolju. Kompatibilnost web čitača se odjednom javlja kao jedan od glavnih problema. Programeri su bili prisiljeni da testiraju svoje aplikacije pod različitim operativnim sistemima i čitačima, a često su bili primorani da distribuiraju ažurne dodatke za različite web čitače svojim klijentima. Drugim rečima, klijentski model je žrtvovao neke od najvažnijih prednosti web programiranja.

Zbog toga je ASP.NET dizajniran kao serverska tehnologija. Celokupni ASP.NET kod izvršava se na serverima. Nakon izvršenja programskog koda na serveru, korisnik dobija klasičnu HTML stranicu koju može pregledati u bilo kom web čitaču. Slika 1-3 prikazuje razlike između serverskog i klijentskog modela.

Evo još nekoliko razloga za napuštanje klijentskog programskog modela:

> **Izolacija:** Kod koji se izvršava na strani klijenta ne može pristupiti resursima koji se nalaze na serveru. Klijentska aplikacija, na primer, ne može lako pročitati sadržaj datoteke ili ostvariti vezu sa bazom podataka koja je smeštena na serveru (pristup je u principu moguć, ali uz velike probleme po pitanju zaštite i kompatibilnosti web čitača).
>
> **Zaštita:** Krajnji korisnici imaju pristup kodu koji se izvršava na klijentu. Kada zlonamerni korisnik otkrije kako program radi, veoma lako može uneti neželjene izmene u njega.
>
> **"Laki" (engl. thin) klijenti.** Sa razvojem Interneta pojavljuju se novi uređaji poput mobilnih telefona, palmtop računara i PDA uređaja (lični digitalni pomoćnici), koji mogu pristupiti mreži. Ovi uređaji mogu komunicirati sa web serverima, ali oni ne podržavaju sve opcije klasičnih web čitača. "Laki" klijenti mogu koristiti serverske web aplikacije, ali ne podržavaju klijentske opcije poput JavaScripta.

Klijentsko programiranje, međutim, još uvek nije mrtvo. ASP.NET i dalje omogućava kombinovanje najkorisnijih opcija klijentskog i serverskog programskog modela. Najbolje ASP.NET kontrole, na primer, mogu same detektovati opcije koje nudi klijentski web čitač. Ukoliko čitač podržava JavaScript, takve kontrole će učitati web stranice sa bogatijim korisničkim interfejsom izgrađenim u JavaScritpu. U poglavlju 25 ćemo objasniti kako se obična ASP.NET stranica može dodatno ojačati AJAX opcijama, koje koriste obiman JavaScript kod na strani klijenta. Bez obzira na mogućnosti web čitača, vaš kod se uvek izvršava na strani servera. Klijentski kod je samo dodatni šlag na torti.



SLIKA 1.3. *Web aplikacije sa serverskim i klijentskim programskim modelom*

.NET okruženje

Kao što smo već objasnili, .NET Framework (okruženje) predstavlja skup više različitih tehnologija. Ovo okruženje obuhvata:

.NET jezike: To su Visual Basic, C#, JScript .NET (serverska verzija JavaScripta), J# (Java klon) i C++.

CLR (Common Language Runtime, izvršno okruženje zajedničko za sve jezike): To je mehanizam u kome se izvršavaju svi .NET programi i koji obezbeđuje automatske servise za sve aplikacije (zaštita, upravljanje memorijom i optimizacija).

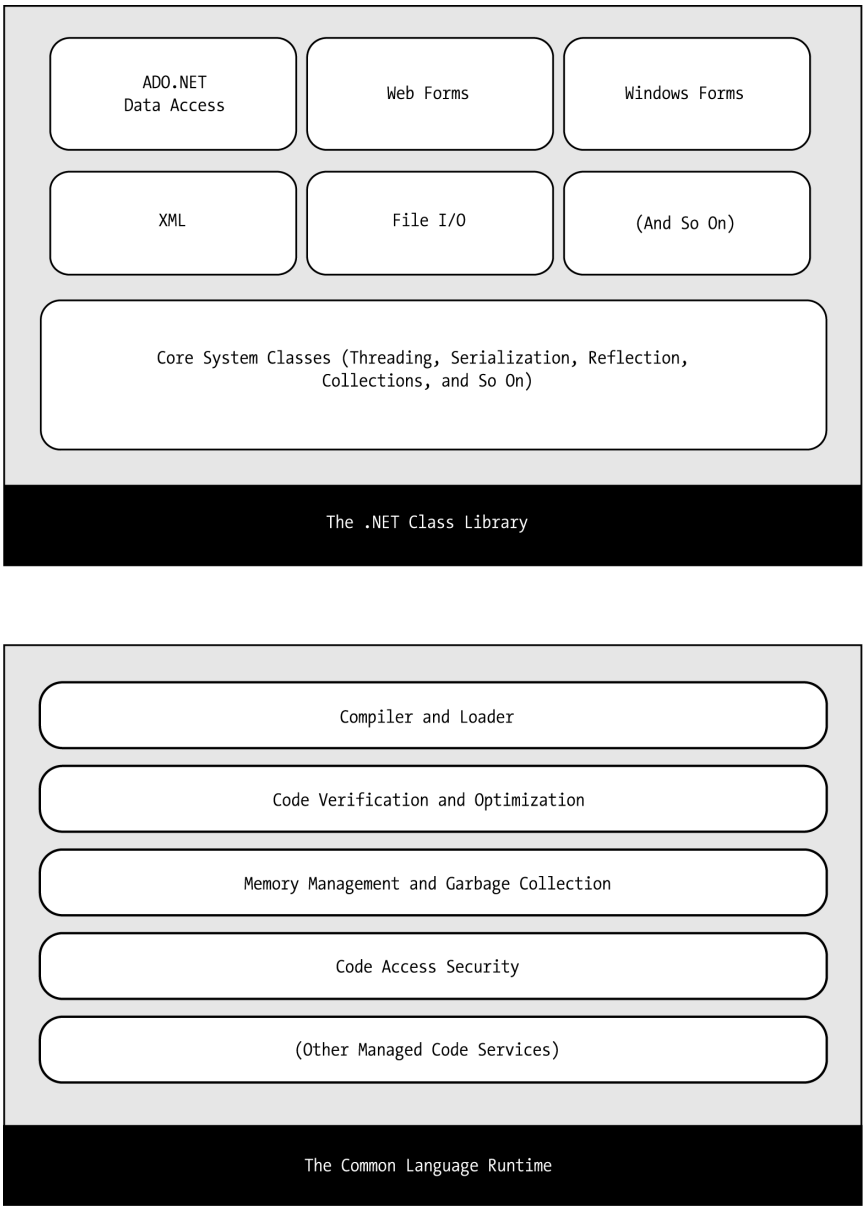
.NET Framework biblioteka klasa: Biblioteka klasa sadrži više hiljada unapred kreiranih opcija (funkcionalnosti) koje se mogu "prilepiti" u korisničke aplikacije. Te opcije su delimično grupisane u tehnološke skupove, kao što je ADO.NET (tehnologija za kreiranje aplikacija koje rade sa bazama podataka) ili Windows Forms (tehnologija za kreiranje korisničkog interfejsa, poput onoga u desktop aplikacijama).

ASP.NET: Mehanizam koji podržava web aplikacije kreirane u .NET sistemu i podržava skoro sve opcije iz .NET biblioteke klasa. Pored toga, ASP.NET obuhvata i određene web servise, poput identifikacije korisnika uz zaštitu prenosa i skladištenja podataka.

Visual Studio: Opciona razvojna alatka koja sadrži brojne opcije za poboljšanje produktivnosti u pisanju programa i za otklanjanje grešaka (debug). Instalacioni DVD Visual Studija sadrži kompletan .NET Framework, tako da ga korisnik ne mora posebno preuzimati sa Interneta.

Granice između prikazanih komponenti ponekad nisu u potpunosti jasne. Pojam ASP.NET se, na primer, ponekad koristi u užem smislu za označavanje dela .NET biblioteke klasa namenjenih dizajnu web stranica. Sa druge strane, ASP.NET se često koristi i kao zajednički naziv za kompletnu oblast .NET web aplikacija, obuhvatajući .NET jezike i deo biblioteke klasa koje se ne odnose na web. (I mi ćemo, u principu, koristiti ovaj termin pod tim značenjem u knjizi. Kroz detaljnu obradu ASP.NET-a mi ćemo izučiti .NET osnove, jezik C# i druge opcije koje .NET programeri mogu koristiti, poput programiranja zasnovanog na komponentama i pristupa bazama podataka.)

Slika 1.4 prikazuje .NET biblioteku klasa i CLR - dve osnovne komponente .NET sistema.



SLIKA 1.4 *.NET Framework*

U nastavku ovog poglavlja upoznaćemo osnovne .NET Framework komponente.

Različita lica ASP.NET-a

Sa verzijom 3.5 Microsoft je pokušao da poboljšanjima i ojačanjima nastavi uspešan razvoj svoje ASP.NET platforme. Povoljno je to što u novoj verziji Microsoft nije ukinuo ni jednu opciju iz ranijih verzija, niti je zamenio neku od funkcionalnih karakteristika ili promenio smer razvoja. Umesto toga, nova verzija je uvela nove opcije na višim nivoima programiranja, koje omogućuju znatno produktivnije pisanje programa.

ASP.NET je do sada prošao kroz četiri osnovne verzije:

- ASP.NET 1.0: prva verzija koja je donela osnovu ASP.NET platforme i širok spektar najbitnijih opcija.
- ASP.NET 1.1: druga verzija je donela nove elemente za podešavanje performansi i otklanjanje grešaka, bez uvođenja novih, suštinskih opcija.
- ASP.NET 2:0: treća verzija je donela veliki broj novih opcija, koje su zasnovane na postojećem ASP.NET sistemu. Namera je bila da se programerima na raspolaganje stave novi, unapred definisani elementi koji se mogu koristiti bez pisanja obimnog koda (ponekad čak i bez ikakvog pisanja koda). Nove opcije su uključivale podršku za navigaciju po web sajtu, teme koje su omogućavale standardizaciju izgleda web stranica, kao i pojednostavljene načine za izdvajanje informacija iz baza podataka.
- ASP.NET 3.5: četvrta verzija je zadržala osnovni mehanizam koji je postojao i u ASP.NET 2.0, ali je donela nekoliko poboljšanja i dve suštinske izmene. Najznačajnije poboljšanje ogleda se u uvođenju ASP.NET AJAX alatki, koje su omogućavale kreiranje stranica sa visokim stepenom interakcije i elementima korisničkog interfejsa karakterističnim za desktop aplikacije (poput autokompletiranja unosa teksta i tehnike prevuci-i-pusti). Druga značajna izmena ogleda se u LINQ podršci, odnosno u uvođenju jezičkih dodataka koji su omogućavali pretraživanje podataka u memoriji na isti način na koji se vrši pretraživanje u bazama podataka.

Ukoliko ste se upitali gde je u tom lancu ASP.NET 3.0 - znajte da ta verzija uopšte ne postoji! Microsoft je iskoristio naziv .NET 3.0 za objavljivanje seta novih tehnologija, među kojima su Windows Presentation Foundation (WPF), platforma za kreiranje uglađenih Windows aplikacija; Windows Workflow Foundation (WF), platforma za modelovanje aplikativne logike pomoću dijagrama tokova; kao i Windows Communication Foundation (WCF), platforma namenjena kreiranju servisa koji se mogu pozivati sa drugih računara. .NET 3.0, međutim, nije sadržao i novu verziju ASP.NET-a.

C#, VB i .NET jezici

Ova knjiga koristi C#, Microsoftov favorit iz grupe .NET jezika. C# je novi jezik koji je dizajniran za .NET 1.0. Po sintaksi je sličan jezicima Java i C++, mada ne postoji direktan način za migraciju programa napisanih u ovim jezicima u C#.

Zanimljivo je da su jezici VB i C# prilično slični. Njihova sintaksa se doduše razlikuje, ali i VB i C# koriste .NET biblioteku klasa i mogu se izvršavati u CLR okruženju. Tačnije, bilo koji blok C# koda možemo prevesti (liniju po liniju) u ekvivalentan VB blok (i obratno). Postoje izvesne razlike u detaljima između ova dva jezika (VB, na primer, podržava tzv. opcione parametre, koje C# ne podržava), mada, u principu, programer koji je naučio jedan .NET jezik lako može preći na neki drugi.

Najkraće rečeno, i VB i C# su elegantni, savremeni jezici idealni za kreiranje web aplikacija naredne generacije.

NAPOMENA

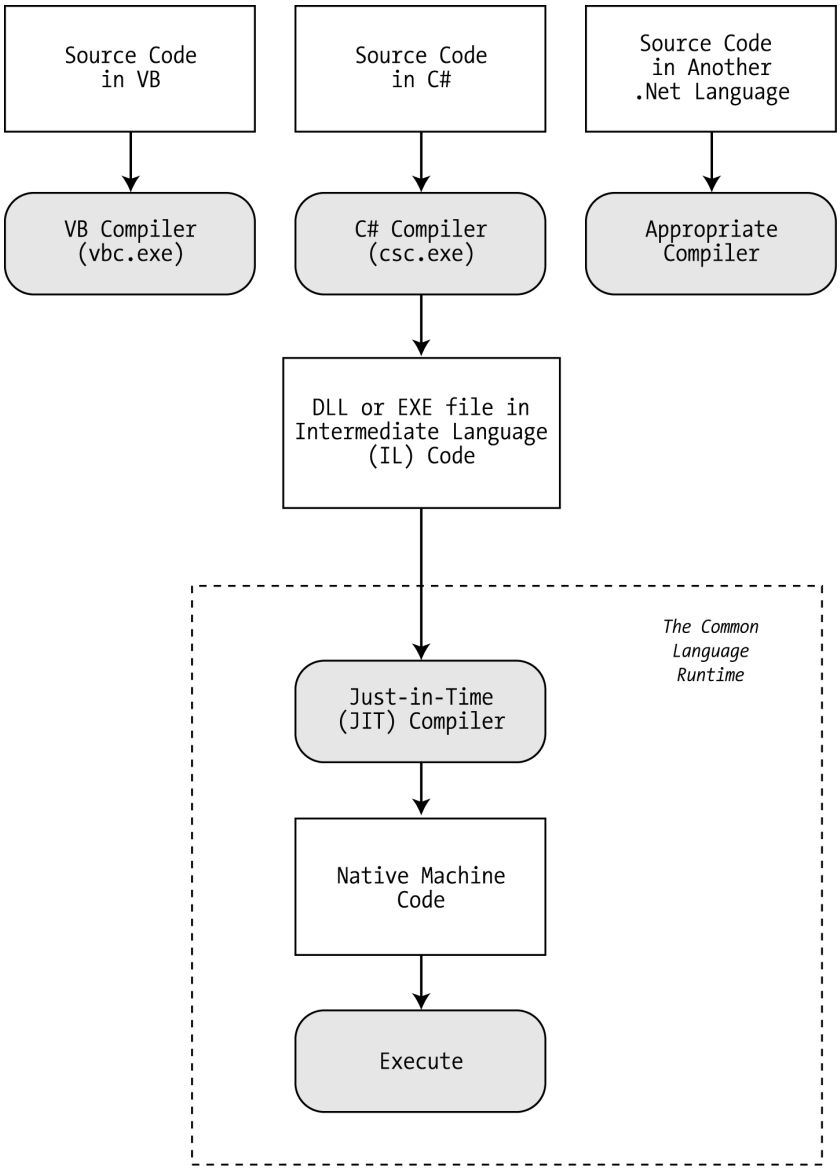
.NET 1.0 je uveo potpuno nove programske jezike, za razliku od narednih verzija, koje su donele samo manje izmene. Bez obzira što je .NET 3.5 doneo nekoliko novih opcija za VB i C#, ovi jezici su u najvećoj meri ostali i dalje neizmenjeni. U drugom i trećem poglavlju ćemo prikazati osnove C# sintakse, zajedno sa osnovama objektno-orijentisanog programiranja.

Međujezik

Svi programi napisani u nekom od .NET jezika pre izvršenja prevode se u zaseban jezik nižeg nivoa. To je CIL, odnosno Common Intermediate Language (zajednički međujezik, često se označava i kao IL). CLR sistem, izvršno okruženje .NET platforme, koristi isključivo IL programski kod. Pošto je dizajn svih .NET jezika zasnovan na IL jeziku, jasno je da između njih postoje krupne sličnosti. Odatle i potiče velika sličnost u opcijama i performansama između jezika VB i C#. Tačnije, stepen kompatibilnosti ovih jezika je takav da web stranica napisana u C# može koristiti VB komponente na isti način kao i one komponente napisane u C#, i obratno.

.NET Framework dodatno formalizuje ovu kompatibilnost putem tzv. Common Language Specifikacije (CLS). CLS je u osnovi ugovor koji, ukoliko se poštuje, garantuje da će komponenta napisana u jednom .NET jeziku raditi i sa ostalim jezicima. Deo specifikacije je i tzv. common type system (CTS, zajednički sistem tipova), koji definiše pravila za tipove podataka kao što su stringovi, brojevi i nizovi - tipovi koji su zajednički za sve .NET jezike. Pored toga, CLS definše i objektno-orijentisane elemente kao što su klase, metodi, događaji i drugi. .NET programeri u najvećem broju slučajeva uopšte ne moraju voditi računa o tome kako CLS funkcioniše, bez obzira što se u svakodnevnom radu oslanjaju na ovu specifikaciju.

Slika 1.5 prikazuje kako se kod napisan u .NET jezicima prevodi u IL. Svaka EXE ili DLL datoteka koju kreirate u nekom od .NET jezika sadrži IL kod. To su upravo datoteke koje postavljate na druge računare. Kada je reč o web aplikacijama, tako kompajliran kod postavljate na neki aktivan web server.



SLIKA 1.5 *Prevođenje programa u .NET sistemu*

CLR može izvršavati samo IL kod, što znači da ovo izvršno okruženje ne može ni da pretpostavi koji .NET jezik je korišćen za pisanje programa. Očigledno je, međutim, da CLR obavlja još jedno prevođenje-kompilaciju - IL kod se u njemu prevodi u mašinski jezik koji odgovara računarskoj platformi. Ovo prevođenje se obavlja pri pokretanju aplikacije, neposredno pre izvršenja programskog koda. Kada je reč o ASP.NET aplikacijama, datoteke sa mašinskim kodom se čuvaju u kešu tokom čitavog vremena izvršenja web aplikacije, tako da se kod može ponovo upotrebiti ukoliko je to potrebno, čime se obezbeđuju optimalne performanse izvršenja programa.

NAPOMENA

Verovatno ste se upitali zašto .NET kompajleri ne prevode program direktno u mašinski kod. Razlog leži u činjenici da mašinski kod zavisi od nekoliko faktora, uključujući i CPU računara. Ukoliko, na primer, kreirate mašinski kod za računar sa Intelovim procesorom, kompajler će verovatno iskoristiti mogućnost korišćenja hiper niti (engl. Hyper-Threading) radi postizanja boljih performansi. Takva mašinska verzija programa možda neće odgovarati drugim računarima na kojima se koristi drugi procesor.

Ostali .NET jezici

VB i C# nisu jedini izbor za pisanje ASP.NET programa. Programeri mogu upotrebiti i J# (jezik sa sintaksom sličnom Javinoj). Mogu se koristiti i .NET jezici koje isporučuju nezavisne firme, poput .NET verzija Eiffela ili čak COBOL-a. Sve je veći broj .NET jezika u ponudi, zahvaljujući CLS i CTS sistemu, koji definiše osnovne zahteve i standarde na osnovu kojih se mogu kreirati novi jezici koji se kompajliraju u IL.

Bez obzira što se za kreiranje ASP.NET web aplikacija mogu koristiti svi jezici iz .NET grupe, neki od njih nemaju isti nivo podrške u Visual Studiju, tako da praktično svi ASP.NET programeri rade u VB i C#. Dodatne podatke o .NET jezicima nezavisnih proizvođača možete naći na adresi www.dotnetlanguages.net.

Zajedničko izvršno okruženje

Zajedničko izvršno okruženje (CLR, Common Language Runtime) podržava sve .NET jezike. Veliki broj savremenih jezika koristi sistem izvršnih okruženja. Izvršno okruženje jezika VB 6 sadržano je u DLL datoteci pod nazivom msvbvm60.dll. Veliki broj aplikacija napisanih u C++ povezuje se sa datotekom pod nazivom mscrt40.dll i na taj način stiču sve prednosti zajedničke funkcionalnosti. Izvršna okruženja obezbeđuju biblioteke koje se koriste uz pojedine jezike, ili obavljaju određene poslove tokom izvršenja programskog koda (kao što je slučaj kod Jave).

Izvršna okruženja nisu novina u programiranju, ali Microsoftov CLR predstavlja najambiciozniji projekat na ovom polju do današnjeg dana. Pored toga što izvršava programski kod, CLR obezbeđuje veliki broj servisa, kao što je verifikacija koda, optimizacija i upravljanje objektima.

NAPOMENA

Pojedini programeri su upravo zbog CLR okruženja optužili .NET da nije ništa drugo do običan klon programskog jezika Java. Takva tvrdnja je prilično neinteligentna. .NET jeste sličan Javi po nekim osnovnim elementima (oba sistema koriste specijalna okruženja i nude veliki broj opcija putem bogatih biblioteka klasa), ali je tačna i činjenica da svaki novi programski jezik "krade" ili poboljšava rešenja prethodnika. To je slučaj i sa Javom, koja je prihvatila deo C/C++ jezika i sintakse već pri kreiranju. Naravno, u svim ostalim elementima .NET se razlikuje od Jave, onoliko koliko se razlikuje i od VBScripta.

Kompletan .NET programski kod izvršava se u okviru CLR okruženja. Ova tvrdnja važi i za Windows aplikacije, baš kao i za web servise. Kada neki klijent zatraži ASP.NET web stranicu, na primer, u CLR okruženju se pokreće odgovarajući ASP.NET servis koji izvršava programski kod, kreira traženu HTML stranicu i vraća je nazad klijentu.

Izvršavanje programa u CLR okruženju ima brojne implikacije:

Visok stepen integracije jezika: VB i C# se, kao i ostali .NET jezici, prevode (kompajliraju) u IL kod. Drugim rečima, CLR ne pravi razliku među programskim jezicima - tačnije, CLR ne može znati u kom jeziku je pisan izvršni program. Ova osobina prevazilazi običnu kompatibilnost jezika; ovde se radi o integraciji programskih jezika.

Paralelno izvršavanje: CLR može učitati i više različitih verzija neke komponente u jednom trenutku. Drugim rečima, neka komponenta se može nadograditi više puta, ali će svaka pojedinačna aplikacija učitati i primeniti onu verziju koja njoj odgovara. Zbog toga je moguće instalirati različite verzije .NET Frameworka, što znači da korisnik može preći na novu verziju ASP.NET-a bez brisanja stare i bez dorade postojećih aplikacija.

Smanjen broj grešaka: Veliki broj grešaka jednostavno nije moguć u CLR okruženju. Ovo okruženje, primera radi, onemogućava brojne greške u radu sa memorijom koje su inače moguće u programskim jezicima nižeg nivoa, kao što je C++.

Pored ovih zaista revolucionarnih prednosti, CLR ima i neke potencijalno slabe strane. Navešćemo nekoliko problema na koja programeri početnici često ne dobijaju odgovore:

Performanse: Tipična ASP.NET aplikacija je znatno brža od odgovarajuće ASP aplikacije, zato što se ASP.NET programski kod prevodi u mašinski pre izvršenja. Procesorski zahtevni algoritmi, međutim, još uvek ne mogu da dostignu zadivljujuću brzinu dobro napisanog C++ koda, zato što CLR uvodi dodatne režijske troškove tokom izvršenja. Navedeni problem je redak i odnosi se samo na manji broj kritičnih aplikacija sa velikim radnim opterećenjem (poput igara u realnom vremenu). Kod obimnih web aplikacija procesorsko opterećenje retko predstavlja usko grlo, ali se ono može javiti zbog manje brzine eksternih resursa kao što su baze podataka ili datotečni sistemi web servera. Odlične performanse web aplikacija se, međutim, mogu postići primenom ASP.NET sistema keširanja i pažljivim pisanjem koda za rad sa bazama podataka.

Transparentnost programskog koda: IL se znatno jednostavnije disasemblira, što znači da ostali programeri mogu lako otkriti kako funkcioniše vaš kod koji ste distribuirali. To i nije neki problem kada se radi o ASP.NET aplikacijama, koje se inače i ne distribuiraju, već se izvršavaju na serveru.

Problematična višeplatformska podrška: Niko ne može sa sigurnošću tvrditi da li će .NET biti prihvaćen i na drugim platformama i operativnim sistemima. Danas postoje neki ambiciozni projekti kao što je Mono (besplatna implementacija .NET-a na Linux, Unix i Windows platformama; videti adresu www.mono-project.com). Ipak, .NET verovatno neće nikada dostići rasprostranjenost Jave, zato što koristi brojne opcije koje su zavisne od platforme i operativnog sistema.

SAVET

Na tržištu se mogu naći razne .NET implementacije za druge platforme, ali one ne obezbeđuju potpunu funkcionalnost niti uživaju podršku Microsofta. Takve implementacije nisu najbolje rešenje za kritične poslovne sisteme.

.NET biblioteka klasa

.NET biblioteka klasa je ogroman skup klasa koje obezbeđuju punu funkcionalnost za veliki broj najrazličitijih poslova - od čitanja XML datoteke do slanja elektronske pošte. Ukoliko ste već radili sa Javom, sigurno imate predstavu o tome šta je to biblioteka klasa. .NET biblioteka je, međutim, znatno ambicioznija i sveobuhvatnija u odnosu na bilo koje drugo programersko okruženje. Bilo koji .NET jezik može koristiti usluge ove biblioteke jednostavnom interakcijom sa odgovarajućim objektom. Time se obezbeđuje konzistentnost između različitih .NET jezika i izbegava instalacija prekobrojnih komponenti na računaru korisnika ili na web serveru.

Pojedini delovi biblioteke sadrže opcije koje vam nikada neće zatrebati za kreiranje web aplikacija (poput klasa za kreiranje desktop aplikacija sa Windows interfejsom). Neki delovi su direktno usmereni ka web programiranju. Veliki broj klasa se može koristiti u različitim programerskim scenarijima i nisu direktno vezani ni za web ni za Windows programiranje. Tu spada i osnovni skup klasa koje definišu jedinstvene tipove promenljivih, kao i klase namenjene pristupu bazama podataka, između ostalih. Sadržaj .NET Frameworka ćemo detaljno upoznati u narednim poglavljima.

Biblioteku klasa možemo posmatrati i kao dobro uređen skup programerskih alata. Microsoft na taj način obezbeđuje programerima gotovu infrastrukturu, tako da se oni mogu koncentrisati na pisanje delova programa koji su zaduženi za poslovnu logiku. Na taj način, primera radi, .NET Framework preuzima sve obaveze oko rešavanja transakcija u bazama podataka i pitanja konkurentnosti, obezbeđujući istovremeni pristup više hiljada korisnika ka jednoj web stranici. Programeru preostaje samo da definiše logiku koja je potrebna njegovoj specifičnoj aplikaciji.

Visual Studio

Poslednju komponentu .NET sistema čini Visual Studio, bogato razvojno okruženje koje obezbeđuje brzo kreiranje složenih aplikacija. Teoretski, bilo koju ASP.NET aplikaciju možete kreirati i bez Visual Studija (kompletan izvorni kod možete napisati u nekom editoru teksta, a zatim ga prevesti pomoću .NET kompajlera koji se startuju iz komandne linije). Takav pristup je, međutim, naporan i podložan greškama, zbog čega profesionalni ASP.NET programeri obavezno koriste razvojne alatke kao što je Visual Studio.

Nabrojaćemo nekoliko opcija koje nudi Visual Studio:

Dizajn web stranica: Programer može kreirati atraktivne stranice jednostavnom tehnikom prevuci-i-pusti uz pomoć dizajnera web formulara koji je integrisan u Visual Studio. Poznavanje HTML jezika nije neophodno.

Automatsko otkrivanje grešaka: Visual Studio može detektovati greške i izvestiti o njima pre pokretanja aplikacije, što vam može uštedeti više sati rada. Potencijalni problemi su prikazani podvučeno, poput opcije provere pravopisa u brojnim programima za obradu teksta.

Alatke za otkrivanje grešaka: Visual Studio je zadržao svoje legendarne debug alatke, koje omogućuju praćenje izvršenja koda i sadržaja promenljivih. Web aplikacije možete testirati baš kao i sve ostale aplikacije, zato što Visual Studio ima ugrađen web server koji je namenjen isključivo proveri ispravnosti programa.

IntelliSense: Visual Studio omogućava kompletiranje naredbi na osnovu prepoznatih objekata i automatsko izlistavanje informacija kao što su parametri funkcija u praktičnom pomoćnom tekstu.

Korišćenje Visual Studija za kreiranje web aplikacija nije obavezno. Umesto toga, sa Interneta možete preuzeti besplatan .NET Framework i uz pomoć običnog editora teksta kreirati ASP.NET web stranice i web servise. Time ćete, međutim, usložiti svoj posao i provesti znatno više vremena u otkrivanju i otklanjanju grešaka, organizaciji i održavanju svog koda. U četvrtom poglavlju ćemo upoznati poslednju verziju Visual Studija.

Visual Studio se isporučuje u nekoliko različitih verzija. Standard Edition poseduje sve opcije za kreiranje svih tipova aplikacija (Windows i Web). Professional Edition i Team Edition poseduju dodatne alatke i "ukrase" o kojima neće biti reči u ovoj knjizi. Između ostalog, ove verzije sadrže alatke za upravljanje programskim kodom u timskom okruženju, kao i alatke za automatizovane testove.

Osiromašena verzija Visual Studija, pod nazivom Visual Web Developer Express Edition, je potpuno besplatna. Njene mogućnosti su iznenađujuće velike, ali ona ipak poseduje i neka bitna ograničenja. Visual Web Developer Express Edition nudi punu podršku za kreiranje web aplikacija, ali ne podržava i ostale tipove aplikacija. To znači da u njoj ne možete kreirati zasebne komponente radi upotrebe u drugim aplikacijama, niti možete razvijati Windows aplikacije. Preostale opcije, međutim, garantuju da Visual Web Developer Express Edition ipak predstavlja pravu verziju Visual Studija, sa sličnim skupom opcija i razvojnim interfejsom.

Završna reč

Prvo poglavlje daje grubi prikaz ASP.NET-a i .NET Frameworka, koji je dovoljan za osnovno razumevanje ovog okruženja. Pored toga, prikazali smo i istorijski razvoj web programiranja, od elementarnih HTML obrazaca do najnovijih dostignuća u verziji .NET 3.5.

U narednom poglavlju ćemo dati sveobuhvatan prikaz jezika C#.