DEO

I

Osnovno Web programiranje

U OVOM DELU:

1. Iza scene: Kako rade Web aplikacije
2. HTML osnove
3. Kratak vodič za dinamičke Web aplikacije

POGLAVLJE

1

Iza scene: Kako rade Web aplikacije

PRE NEGO ŠTO SHVATITE ŠTA C# APLIKACIJE MOGU DA RADE, POTREBNO JE DA RAZUMETE ŠTA SE UOPŠTE DEŠAVA SA WEB ZAHTEVIMA. POŠTO JE WEB APLIKACIJA ČESTO KOMBINACIJA JEDNOSTAVNIH INFORMATIVNIH HTML STRANA I KOMPLEKSNIJIH DINAMIČKIH STRANA, TREBALO BI DA SHVATITE KAKO SERVER ISPUNJAVA ZAHTEVE KOJI NE TRAŽE KOD. ZNAČAJNA KOLIČINA POZADINSKOG PREGOVARANJA I PRENOSA PODATAKA NASTUPA ČAK I PRE NEGO ŠTO KORISNIČKI ZAHTEV DOSPE DO VAŠEG KODA.

WEB APLIKACIJA JE NEIZOSTAVNO PODELJENA IZMEĐU BAR DVA SLOJA - KLIJENT I SERVER. SVRHA OVOG POGLAVLJA JE DA VAM RAZJASNI KAKO KLIJENT I SERVER KOMUNICIRAJU. OSIM TOGA, NAUČIĆETE KAKO SE C# INTEGRIŠE U OVAJ PROCES KOMUNIKACIJE I ŠTA ON MOŽE URADITI DA BI VAM POMOGAO U PISANJU WEB APLIKACIJA.

U ovom poglavlju naći ćete:

- Kako rade Web zahtevi
- Kakav je sadržaj klijent zahteva
- Kako Web server odgovara - priprema
- Kako Web server odgovara - izvršenje
- Šta klijent radi s odgovorom
- Uvod u dinamičke Web strane
- Šta C# može da uradi
- Rezime

Kako rade Web zahtevi

Web zahtev traži dve komponente: Web server i klijent. Klijent je najčešće Web pretraživač, ali može biti i drugi tip programa, kao što je pauk (program koji prati Web linkove, sakupljajući informacije) ili agent (program zadužen za pronalaženje određenih informacija, često koristeći pretraživače), standardna izvršna aplikacija, bežični ručni uređaj, ili zahtev iz čipa uključenog u spravu, kao što je frižider. U ovoj knjizi, usredsredićete se uglavnom, ali ne i isključivo, na Web pretraživače kao klijente; prema tome, izraze "Web pretraživač" i "klijent" možete smatrati u suštini istom stvari u većem delu knjige. Kada se značenje ovih termina bude razlikovalo, ja ću Vas upozoriti.

Server i Web pretraživač su obično na različitim računarima, ali to nije obavezno. Možete koristiti Web pretraživač da biste zahtevali strane od Web servera koji se izvršava na istom računaru - zapravo, to je verovatno postavka koju ćete koristiti za pokretanje većine primera iz ove knjige na svom razvojnom računaru. Poenta je u ovome: bilo da su Web server i Web pretraživač na istom računaru ili na različitim stranama sveta, zahtev radi skoro na isti način.

I server i klijent koriste definisani *protokol* za međusobnu komunikaciju. Protokol je jednostavno dogovoreni način za iniciranje sesije komunikacije, prosleđivanje informacija napred i nazad i prekidanje sesije. Nekoliko protokola se koristi za Web komunikacije: najčešći je Hypertext Transfer Protocol (HTTP), koji se koristi za zahteve Web strana; tu je i File Transfer Protocol (FTP), koji se koristi za prenos binarnih fajlova podataka; a imamo i Network News Transfer Protocol (NNTP), koji se koristi za news grupe. Nezavisno od toga koji se protokol koristi, Web zahtevi se prenose pomoću pozadinskog mrežnog protokola nazvanog Transmission Control Protocol/Internet Protocol (TCP/IP). To je globalni komunikacioni standard koji određuje osnovna pravila kojih se drže dva računara da bi razmenila informacije.

Server računar strpljivo čeka ne radeći ništa, dok ne stigne zahtev za inicijalizaciju komunikacije. U Web aplikaciji, klijent uvek šalje inicijalizaciju za početak sesije - server može samo da odgovori. Otkrićete da to može da bude izvor frustracije ukoliko ste navikli na pisanje samostalnih programa. Inicijalizacija sesije se sastoji od definisane serije bajtova. Sadržaj bajtova nije važan - jedina važna stvar je da oba računara prepoznaju seriju bajtova kao inicijalizaciju. Kada server primi zahtev za inicijalizaciju, on potvrđuje transmisiju vraćajući drugu seriju bajtova klijentu.

Konverzacija između dva računara se nastavlja na ovaj način "napred - nazad." Da računari govore, komunikacija bi bila ovakva:

Klijent Halo?

Server Pozdrav. Ja govorim srpski.

Klijent I ja govorim srpski.

Server Šta želiš?

Klijent Želim fajl `/mojSajt/mojiFajlovi/fajl1.htm`.

Server Fajl je premešten na `/mojSajt/stariFajlovi/fajl1.htm`.

Klijent Izvini. Doviđenja.

Server Doviđenja.

Klijent Halo?

Server Pozdrav. Ja govorim srpski.

Klijent I ja govorim srpski.

Server Šta želiš?

Klijent Želim fajl `/mojSajt/stariFajlovi/fajl1.htm`.

Server Evo nekih informacija o tom fajlu.

Klijent Hvala; molim te da pošalješ podatke.

Server Počinjem prenos podataka, šaljem paket 1, šaljem paket 2, šaljem paket 3...

Klijent Dobio sam paket 1, paket 2 ima greške, dobio sam paket 3, dobio sam paket 4.

Server Ponovo šaljem paket 2.

Konverzacija se nastavlja dok se transmisija ne završi.

Server Svi paketi su poslati.

Klijent Svi paketi su dobijeni u dobrom stanju. Doviđenja.

Server Doviđenja.

TCP/IP je samo jedan od mnogih protokola za komunikaciju računara, ali zbog popularnosti Interneta, on je postao sveprisutan. Neće Vam biti potrebno da znate mnogo više od toga o TCP/IP da biste ga koristili - pozadinski protokol je gotovo u potpunosti transparentan. Međutim, potrebno je da znate bar nešto o tome kako jedan računar pronalazi drugi da bi inicirao sesiju komunikacije.

Kako klijent zahteva sadržaj

Kada ukucate zahtev u polje adrese Web pretraživača ili kliknete hiperlink, Web pretraživač pakuje zahtev i šalje važan deo URL, nazvan *ime domena* serveru imenovanja, normalno zvanog DNS server, koji je tipično lociran kod Vašeg Internet Service Providera (ISP). Server imenovanja čuva bazu podataka s imenima, od kojih je svako povezano sa IP adresom. Računari ne razumeju reči baš najbolje, tako da server imenovanja prevodi zahtevanu adresu u broj. Tekstualno ime koje vidite u linku ili polju adrese je zapravo ljudima naklonjena verzija IP adrese. IP adresa je set od četiri broja između 0 i 255, razdvojenih tačkama: na primer, 204.235.113.34. Svaka grupa od tri cifre se naziva "oktet."

Svaka IP adresa jedinstveno identifikuje jedan računar. Ukoliko prvi server imenovanja nema zahtevanu adresu u svojoj bazi podataka, on prosleđuje zahtev serveru imenovanja dalje u hijerarhiji. Konačno, ukoliko nijedan server imenovanja ne može da prevede zahtevano ime u IP adresu, zahtev stiže do jednog od moćnih servera imenovanja koji čuvaju glavne liste svih javno registrovanih IP adresa. Ukoliko nijedan server imenovanja ne može da prevede adresu, neuspeli odgovor putuje nazad kroz hijerarhiju servera imenovanja dok ne stigne do Vašeg Web pretraživača. U tom trenutku, Vi vidite poruku o grešci.

Ukoliko server imenovanja pronađe upis IP adrese zahteva, on kešira zahtev tako da neće morati da kontaktira servere imenovanja višeg nivoa za sledeći zahtev ka istom serveru. Keš ističe nakon perioda vremena nazvanog Time to Live (TTL), tako da ukoliko sledeći zahtev premaši TTL, server imenovanja će morati da kontaktira server višeg nivoa u svakom slučaju, u zavisnosti od toga kada sledeći zahtev pristigne. Server imenovanja vraća IP adresu Web pretraživaču, koji koristi IP adresu za kontaktiranje Web servera koji odgovara toj adresi. Mnoge Web strane sadrže reference ka drugim fajlovima koje Web server mora da stavi na raspolaganje da bi strana bila kompletna; međutim, Web pretraživač može da zatraži samo jedan fajl u isto vreme. Na primer, slike referencirane u Web strani zahtevaju posebne zahteve za svaku sliku.

Prema tome, proces prikazivanja Web strane se obično sastoji od serija kratkih konverzacija između Web pretraživača i servera. Tipično, Web pretraživač prima glavnu stranu, raščlanjuje je tražeći druge zahtevane referencirane fajlove, i potom počinje da prikazuje glavnu stranu dok zahteva referencirane fajlove. To je razlog zašto često vidite "čuvare mesta" za slike dok se strana učitava. Glavna strana sadrži reference ka drugim fajlovima koji sadrže slike, ali glavna strana ne sadrži same slike.

Kako odgovara Web server - priprema

Iz ugla Web servera, svaka konverzacija je potpuno novi kontakt. Po osnovnom podešavanju, Web server servisira zahteve na bazi "prvi stigao - prvi servisiran." Web serveri ne "pamte" pojedine Web pretraživače između dva zahteva.

Moderni Web pretraživači i serveri koriste HTTP verziju 1.1, koja održava vezu koja "ostaje u životu." Kao što očekujete, to znači da sama veza, kada je jednom uspostavljena, može da ostane aktivna kroz serije zahteva, umesto da server i klijent moraju da prolaze kroz korake IP traženja i inicijalizacije za svaki fajl. Uprkos HTTP vezama koje ostaju u životu, svako slanje fajla i dalje zahteva posebni ciklus zahteva i odgovora.

Delovi URL

Red koji ukucavate u polje adrese u Web pretraživaču da biste zahtevali fajl naziva se Uniform Resource Locator (URL). Server izvodi standardnu proceduru za servisiranje svakog zahteva. Prvo, on raščlanjuje zahtev tako što razdvaja zahtevanu URL u njene sastavne delove. Kose crte, dve tačke, tačke, znaci pitanja i simboli & - svi zvani *ograničavači* - čine lakim razdvajanje delova. Svaki deo ima određenu funkciju. Evo primera URL zahteva:

```
http://www.microsoft.com:80/CSharpASP/default.htm?Page=1&Para=2
```

Naredna lista pokazuje šemu i funkciju svakog dela primera URL zahteva.

http *Protokol.* Govori serveru koji protokol bi trebalo da koristi da bi odgovorio na zahtev.

www.microsoft.com *Ime domena.* Ovaj deo URL se prevodi u IP adresu. Sam domen se sastoji od nekoliko delova razdvojenih tačkama: host ime, `www`; ime domena kompanije, `microsoft`; i ime Internet domena vrhunskog nivoa, com. Postoji još nekoliko drugih imena Internet domena vrhunskog nivoa, uključujući `org` (organization), `gov` (government), i `net` (network).

80 *Broj porta. Web server ima mnoštvo portova.* Svaki označava mesto gde server "osluškuje" očekujući komunikaciju. Broj porta jednostavno označava jednu od ovih specifičnih lokacija (postoji 65,537 mogućih portova). Vremenom je upotreba određenih brojeva portova postala standardizovana. Na primer, ja sam upotrebio `80` za broj porta u primeru, jer je to standardni (i podrazumevani) HTTP broj porta, ali Vi možete imati servere koji osluškuju očekujući zahteve na bilo kom portu.

CSharpASP *Virtuelan direktorijum.* Server prevodi ovo ime u fizičku putanju na hard disku. Virtuelan direktorijum je skraćeno ime, "pokazivač" koji referencira fizički direktorijum. Imena virtuelnog i fizičkog direktorijuma ne moraju biti ista. Jedan način za definisanje virtuelnih direktorijuma je preko administrativnog interfejsa Web servera. Drugi način za njihovo kreiranje je putem kreiranja novog projekta Web aplikacije ili Web servisa u VS.NET. Na primer, VS.NET kreira virtuelan direktorijum umesto Vas kad god kreirate novi projekat Web aplikacije ili Web servisa.

default.htm *Ime fajla.* Server će vratiti sadržaj fajla. Ukoliko je fajl prepoznat kao izvršivi preko Web servera (ASP fajl) umesto HTML fajla, server će izvršiti program sadržan u fajlu i vratiti rezultate umesto da vrati sadržaj fajla. Ukoliko fajl nije prepoznat, server nudi mogućnost njegovog preuzimanja.

? (Znak pitanja) *Separator.* Znak pitanja odvaja zahtev fajla od dodatnih parametara koji su poslati uz njega. URL primer sadrži dva parametra: `Page=1` i `Para=2`.

Page *Ime parametra.* Programi koje pišete, kao što su ASP strane, mogu da čitaju parametre i da ih koriste za snabdevanje informacijama.

= (Znak jednakosti) *Separator.* Znak jednako odvaja ime parametra od vrednosti parametra.

1 *Vrednost parametra.* Parametar sa imenom `Page` ima vrednost `1`. Obratite pažnju da Web pretraživač šalje vrednosti svih parametara kao string podatke. String je niz

znakova: Reč je string, rečenica je string, slučajni niz brojeva i slova je string - tekst u bilo kom obliku je string. Vaši programi imaju slobodu interpretiranja stringova koji sadrže samo numeričke znakove kao brojeve, ali zbog sigurnosti, trebalo bi da im eksplicitno promenite tip u numerički oblik.

& *Separator*. Ovaj znak razdvaja parove parametar=vrednost.

Para=2 *Parametar i vrednost*. Drugi parametar i vrednost.

Server prevodi putanju

Ne pravite Web zahteve sa stvarnom ili fizičkom putanjom; umesto toga, strane zahtevate pomoću virtuelne putanje. Nakon raščlanjivanja URL, server prevodi virtuelnu putanju u fizičku putanju. Na primer, virtuelan direktorijum na URL `http://mojServer/mojaPutanja/mojFajl.asp` je `mojaPutanja`. Virtuelan direktorijum mojaPutanja je mapiran u lokalnom direktorijumu poput `c:\inetpub\wwwroot\CSharpASP\mojFajl.asp` ili u mrežno Universal Naming Convention (UNC) ime kao što je `\\nekiServer\nekaPutanja\CSharpASP\mojFajl.asp`.

Server proverava resurse

Server proverava zahtevani fajl. Ukoliko on ne postoji, server vraća poruku o grešci - obično `HTTP 404 -- File Not Found`. Verovatno ste videli ovu poruku o grešci dok ste pretraživali Web; ukoliko niste, imate više sreće nego ja.

Server proverava odobrenja

Nakon lociranja resursa, server proverava da li nalog zahteva ima dovoljna odobrenja za prisup resursu. Po osnovnom podešavanju, Internet Information Server (IIS) Web zahtevi koriste poseban nalog gosta nazvan `IUSR_ImeraCunara`, gde je `Imeracunara` ime server računara. Često ćete čuti da se on naziva "anonimnim" klijentom, jer server nema načina da sazna prave informacije naloga za korisnika koji je emitovao zahtev. Kod ASP.NET strana, IIS koristi nalog `SYSTEM` ili drugi nalog gosta sa imenom aspnet_wp_account (ASPNET) po osnovnom podešavanju.

Na primer, ukoliko je korisnik zahtevao fajl za koji taj nalog nema zahtevana odobrenja, server vraća poruku o grešci, obično HTTP `403 -- Access Denied`. Stvarni tekst greške zavisi od toga koja je tačno greška generisana. Na primer, postoje nekoliko podnivoa poruka o grešci 403. Kompletnu listu poruka o grešci možete pronaći u IIS dijalogu Default Web Site Property. Web serveri obezbeđuju osnovne poruke o grešci, ali Vam obično omogućavaju da ih posebno prilagodite. Po osnovnom podešenju, IIS čita tekst poruka o grešci iz HTML fajlova iz Vašeg direktorijuma `%SystemRoot%\help\common`, gde promenljiva `%SystemRoot% treba` da bude ime Vašeg NT direktorijuma, obično nazvanog `winnt`.

Kako odgovara Web server - ispunjenje

Grafički fajlovi, Word dokumenti, HTML fajlovi, ASP fajlovi, izvršni fajlovi, CGI skriptovi - kako server zna kako da obradi zahtevani fajl? Zapravo, server razlikuje tipove fajlova na nekoliko različitih načina.

Internet Information Server (IIS) razlikuje tipove fajlova na osnovu ekstenzija fajlova (kao što su `.asp`, `.htm`, `.exe itd`), baš kao i Windows Explorer. Kada duplo kliknete fajl ili ikonu u Windows Exploreru, on traži ekstenziju fajla u Registryu, posebnoj bazi podataka koja sadrži sistemske informacije i informacije o aplikacijama. Registry sadrži jedan upis za svaku registrovanu ekstenziju fajla. Svaka ekstenzija ima upis odgovarajućeg tipa fajla. Svaki upis tipa fajla, s druge strane, ima odgovarajući izvršni fajl ili fajl manipulator. Server otkida ekstenziju fajla od imena fajla, traži odgovarajući program i pokreće taj program da bi vratio fajl. IIS prati istu seriju koraka da bi odredio kako da odgovori na zahtev.

Drugi Web serveri takođe koriste ekstenzije fajlova da bi odredili kako da obrade zahtev fajla, ali oni ne koriste Registry asocijacije. Umesto toga, oni koriste nezavisnu listu asocijacija "ekstenzija fajla - program." Upisi iz ove liste se nazivaju MIME tipovi, što je skraćenica za Multiple Internet Mail Extension, jer e-mail programi treba da znaju tip sadržaja uključenog uz poruke. Svaki MIME tip - baš kao i Registry asocijacije - povezan je sa određenom akcijom ili programom. Web server pretražuje listu u potrazi za upisom koji odgovara ekstenziji zahtevanog fajla.

Većina Web servera obrađuje neodgovarajuće ekstenzije fajlova tako što ponudi preuzimanje fajla. Neki serveri takođe obezbeđuju osnovne akcije ukoliko zatražite URL koji ne sadrži ime fajla. U tom slučaju, većina servera pokušava da vrati jedan sa liste podrazumevanih imena fajlova - obično fajl nazvan default.htm ili index.htm. Možete biti u mogućnosti da konfigurišete podrazumevana imena fajlova za svoj Web server (IIS), bilo globalno za sve virtuelne direktorijume na tom serveru ili za svaki pojedini virtuelni direktorijum na tom serveru. Server može početi da emituje struju odgovora nazad klijentu dok generiše odgovor ili može da baferiše kompletan odgovor i da ga šalje sve odjednom kada je odgovor završen. Postoje dva dela odgovora: zaglavlje odgovora i telo odgovora. Zaglavlje odgovora sadrži informacije o tipu odgovora. Uz druge stvari, zaglavlje odgovora može da sadrži sledeće:

- Kod odgovora
- MIME tip odgovora
- Datum i vreme nakon koga odgovor više nije ispravan
- URL za preusmeravanje
- Sve vrednosti kolačića koje server želi da skladišti na klijentu

Kolačići su tekstualni stringovi koje Web pretraživač snima u memoriju ili na hard disk klijent računara. Kolačić može trajati za vreme sesije Web pretraživača ili može trajati do zadatog datuma isteka. Web pretraživač šalje kolačiće povezane sa sajtom nazad serveru sa svakim sledećim zahtevom tog sajta.

NAPOMENA

U medijima je velika larma o kolačićima. Neki ljudi su toliko zastrašeni ovom taktikom, da svoje Web pretraživače podešavaju u "turn off cookies" (isključi kolačiće). To znači da Web pretraživač neće da prihvata kolačiće, što može imati glavni uticaj na Vaš sajt jer morate imati neki način za povezivanje pojedine sesije Web pretraživača sa vrednostima skladištenim na sloju servera u Vašoj aplikaciji. Iako postoje metode za pravljenje povezivanja i bez upotrebe kolačića, ona nisu ni približno tako podesna, niti opstaju između sesija Web pretraživača. ■

Šta klijent radi sa odgovorom

Klijent, obično Web pretraživač, treba da zna tip sadržaja kojim je server odgovorio. Klijent čita MIME tip iz zaglavlja da bi odredio tip sadržaja. Za većinu zahteva, MIME tip zaglavlje je ili `text/html` ili tip slike kao što je `image/gif`, ali takođe može biti i fajl programa za obradu teksta, video ili audio fajl, animacija ili bilo koji drugi tip fajla. Web pretraživači, poput servera, koriste Registry vrednosti i liste MIME tipova za određivanje kako će fajl biti prikazan. Za standardne HTML fajlove ili fajlove slika, Web pretraživači koriste ugrađenu mašinu za prikazivanje. Za druge tipove fajlova, Web pretraživači pozivaju servise pomoćnih aplikacija ili plug-inove, kao što je RealPlayer, Microsoft Office aplikacije koje mogu da prikažu informacije. Web pretraživač dodeljuje ceo svoj prozor ili njegov deo kao "platno" na kome pomoćni program ili plug-in "iscrtava" sadržaj.

Kada se telo odgovora sastoji od HTML, Web pretraživač raščlanjuje fajl zbog odvajanja simbola od sadržaja. On potom koristi simbole da bi odredio kako da rasporedi sadržaj na ekran. Moderni HTML fajlovi mogu da sadrže nekoliko različitih tipova sadržaja osim simbola, teksta i slika; Web pretraživači svaki od njih obrađuju na drugačiji način. U najčešće dodatne tipove sadržaja spadaju:

Cascading Style Sheets To su tekst fajlovi u posebnom formatu koji sadrže direktive o tome kako treba formatirati sadržaj HTML fajla. Moderni Web pretraživači koriste kaskadne liste stilova (Cascading Style Sheets - CSS) za dodeljivanje fontova, boja, ivica, vidljivosti, pozicioniranja i drugih informacija o formatiranju elementima strane. CSS stilovi mogu biti sadržani u okviru oznake, mogu biti postavljeni u posebnu oblast u okviru HTML strane ili mogu da postoje u kompletno odvojenom fajlu koji Web pretraživač zahteva nakon što raščlani glavnu stranu, ali pre nego što prikaže sadržaj na ekran.

Skript Svi moderni računari mogu da izvršavaju JavaScript, iako ga ne izvršavaju uvek na isti način. Termin *JavaScript* se primenjuje specifično na skript pisan u Netscapeovom JavaScript skriptnom jeziku, ali dve bliske varijante - Microsfotov skriptni jezik JScript i ECMA-262 specifikacija (ECMAScript) - imaju u suštini istu sintaksu i podržavaju gotovo identični set komandi.

NAPOMENA

Obratite pažnju: skriptni jezik JScript nije JScript.NET. Ovo drugo je mnogo robusnija verzija JScripta koju je Microsoft izdao kao dodatak u Visual Studiu.NET. ■

Osim JScipta, Internet Explorer podržava VBScript, koji predstavlja podskup Visual Basica za aplikacije, koji je, s druge strane, podskup Microsoftovog jezika Visual Basic (pre VB.NET).

NAPOMENA

Kompletnu ECMA-262 specifikaciju možete pronaći na `http://www.ecma.ch/stand/ecma-262.htm`. ■

ActiveX komponente ili Java apleti Ovi mali programi se izvršavaju kod klijenta umesto na serveru. ActiveX komponente se izvršavaju samo na Internet Exploreru na Windows platformama (otprilike 60 procenata svetskog tržišta, u trenutku kada je ova knjiga pisana), dok se Java apleti izvršavaju na gotovo svim Web pretraživačima i platformama.

XML Extensible Markup Language (XML) je sličan sa HTML - oba se sastoje od oznaka i sadržaja. To nije iznenađujuće, jer su oba izvedena iz Standard Generalized Markup Language (SGML). HTML oznake opisuju kako se prikazuje sadržaj i, u ograničenom stepenu, funkciju sadržaja. XML oznake opisuju šta je sadržaj. Drugim rečima, HTML je primarno jezik formatiranja i prikazivanja, dok je XML jezik opisa sadržaja. Ova dva jezika dobro dopunjuju jedan drugog. XML je prvi put korišćen u IE 4 za *kanale*, relativno neuspešnu tehnologiju koja omogućava ljudima da se prijave za informacije na različitim sajtovima. IE 4 je imao traku sa kanalima da bi pomogao ljudima da upravljaju svojim kanalskim prijavama. Kod IE 5, Microsoft je odbacio kanale, ali je proširio razumevanje Web pretraživača za XML olakšicu, tako da ih danas možete koristiti da biste obezbedili "ostrva" podataka u HTML fajlovima. Takođe, možete da isporučujete kombinaciju XML i XSL/XSL/XSLT (jezik pravila pisan u XML koji je po svrsi sličan sa Cascading Style Sheets, ali je moćniji) da biste generisali HTML kod na klijentu. Kombinacija XML/XSL Vam omogućava da rasteretite server od obrade i na taj način poboljšate skalabilnost svog sajta. Netscape 6 nudi drugačiji i - u svrhe prikazivanja - moderniji tip podrške XML. Netscapeova mašina za raščlanjivanje može da kombinuje XML i CSS liste stilova za formatiranje XML direktno za pregled. Na nesreću, Netscape ne podržava direktno XSLT transformacije, tako da ste ograničeni na prikazivanje podataka u svojim XML dokumentima bez međuobrade.

Uvod u dinamičke Web strane

Proces "klijent - server - klijent" koji sam upravo opisao važan je jer se dešava svaki put kada klijent kontaktira server da bi dobio neke podatke. Ovo je znatno drugačije od samostalnog ili client/server modela sa kojim ste možda već bliski. Kako server i klijent zapravo ne znaju ništa jedan o drugome, za svaku interakciju morate da pošaljete, inicijalizujete ili restaurirate odgovarajuće vrednosti da biste održali kontinuitet svoje aplikacije.

Kao jednostavan primer, pretpostavimo da imate obezbeđen sajt sa formom za prijavu. U standardnoj aplikaciji, kada se korisnik uspešno prijavi, to je sve što treba da uradite radi provere autentičnosti. Činjenica da se korisnik uspešno prijavio znači da je njegova autentičnost važeća za sve vreme trajanja aplikacije. Sa druge strane, kada se prijavite na Web sajt obezbeđen samo sa prijavom i lozinkom, server mora ponovo izvršiti proveru Vaše autentičnosti pri svakom sledećem zahtevu. To može biti jednostavan zadatak, ali mora biti izvršen pri svakom zahtevu u aplikaciji.

Zapravo, to je jedan od razloga zbog kojeg su dinamičke aplikacije postale popularne. Na sajtu koji dozvoljava anonimno povezivanje (poput većine javnih Web sajtova), možete proveriti autentičnost korisnika samo ukoliko možete da proverite vrednosti prijava/lozinka koje je uneo korisinik sa "pravim" kopijama smeštenim na serveru. Iako je HTML adekvatan jezik za raspoređivanje za većinu namena, on nije programski jezik. On traži kod za proveru autentičnosti korisnika. Drugi razlog zbog koga su dinamičke strane postale popularne jeste promenljiva priroda informacija. Statičke strane su sasvim dobre za članke, školske novine, knjige i slike - uopšteno za informacije koje se retko menjaju. Ali statičke strane su jednostavno neadekvatne za izdvajanje zaposlenog i liste kontakata, informacije kalendara, obezbeđivanje novosti, sportskih rezultata - uopšteno, za tipove podataka sa kojima interagujete svakodnevno. Podaci se menjaju isuviše često da bi se uspešno održavali na statičkim stranama. Osim toga, ne želite uvek da gledate podatke na isti način. Shvatio sam da počinjem da propovedam horu - ne biste kupili ovu knjigu da niste svesni da

dinamičke strane imaju moć koju statične HTML strane ne mogu da postignu. Međutim, korisno je napomenuti da čak i dinamički podaci obično imaju predvidljivu brzinu promena - nešto o čemu ću diskutovati kasnije u kontekstu keširanja.

Kako server razdvaja kod od sadržaja?

U klasičnim Active Server Pages (ASP) stranama, mogli ste da mešate kod i sadržaj tako što biste postavili posebne oznake koda (`<% %>`) oko koda ili tako što biste napisali script blokove, u kojima se kod pojavljuje između oznaka `<script>` i `</script>`. Klasičan ASP koristi .asp ekstenzije fajlova. Kad server dobije zahtev za ASP fajlom, on prepoznaje - preko asocijacija ekstenzija - da odgovor na zahtev traži ASP procesor. Zbog toga, server prosleđuje zahtev ASP mašini, koja raščlanjuje fajl da bi odvojila sadržaj oznaka koda od sadržaja simbola. ASP mašina procesira kod, integriše rezultate sa ostalim HTML-om na strani i šalje rezultat klijentu.

ASP.NET prolazi kroz sličan proces, ali je ekstenzija ASP.NET fajlova .aspx umesto .asp. I dalje možete da mešate kod i sadržaj na potpuno isti način, iako sada možete (i obično bi trebalo) da postavite kod u odvojeni fajl, nazvan *pozadinska klasa*, jer tim postupkom obezbeđujete jasniju razliku između koda za prikazivanje i koda aplikacije, a to čini lakšom višestruku upotrebu oba. U ASP.NET, možete pisati kod na sva tri mesta - u klase pozadinskog koda i u okviru oznaka koda i script blokova u Vašim HTML fajlovima. I pored toga, ASP.NET mašina i dalje mora da raščlanjuje HTML fajl u traženju oznaka koda.

Kako i kada server obrađuje kod?

Sama ASP.NET mašina je Internet Server Application Programming Interface (ISAPI) aplikacija. ISAPI aplikacije su DLL fajlovi koji se učitavaju u adresni prostor servera, pa su vrlo brzi. Različite ISAPI aplikacije obrađuju različite vrste zahteva. Možete kreirati ISAPI aplikacije za posebne ekstenzije fajlova, kao što su `.asp` ili `.aspx`, ili da izvedu posebne operacije na standardnim tipovima fajlova kao što su HTML i XML.

Postoji dve vrste ISAPI aplikacija: proširenja i filteri. ASP.NET mašina je ISAPI proširenje. ISAPI proširenje zamenjuje ili povećava standardni IIS odgovor. Proširenja se učitavju po potrebi kada server primi zahtev sa ekstenzijom fajla povezanom sa DLL ISAPI proširenja. Za razliku od toga, ISAPI filteri se učitavaju sa IIS i obaveštavaju server o setu obaveštenja događaja filtera koje obrađuju. IIS podiže obaveštenje događaja (koji obrađuje filter) kada god se desi događaj filtera tog tipa.

NAPOMENA

Ne možete kreirati ISAPI aplikacije pomoću C# - ili zaista u upravljanom kodu - iako možete da ih kreirate u Visual Studiu.NET pomoću neupravljanog C++ i Active Template Library (ATL). Međutim, možete da redefinišete podrazumevanu `HttpApplication` implementaciju tako da obezbeđuje mnoge prednosti ISAPI aplikacija pomoću C#. ■

ASP.NET strane zaobilaze standardnu IIS proceduru odgovora ukoliko sadrže oznake koda ili su povezane sa klasom pozadinskog koda. Ukoliko Vaš ASPX fajl ne sadrži kod, ASP.NET mašina prepoznaje ovo kada završi raščlanjivanje strane. Za strane koje ne sadrže kod, ASP.NET prespaja svoj odgovor i nastavlja se standardni server proces. Kod IIS 5 (ASP verzija 3.0), klasične ASP strane su počele da prespajaju strane koje ne sadrže kod. Zbog toga su ASP i ASP.NET strane koje ne sadrže kod samo malo sporije od standardnih HTML strana.

Kako se klijenti ponašaju sa dinamičkim server stranama?

Kratak odgovor je: ne ponašaju se ništa drugačije nego sa bilo kojim drugim zahtevom. Prisetite se da klijent i server znaju vrlo malo jedan o drugome. Zapravo, klijent obično potpuno ignoriše sve u vezi sa serverom osim njegove adrese, dok server mora da zna dovoljno o klijentu da bi obezbedio odgovarajući odgovor.

Web programeri početnici su često zbunjeni u vezi sa razlikama između zahteva statičkih i dinamičkih Web strana. Stvar koju treba zapamtiti je da, što se tiče klijenta, nema *razlike* između zahteva ASPX fajla i HTML fajla. Prisetite se da klijent interpretira odgovor na osnovu vrednosti MIME tipa zaglavlja - a nema posebnih MIME tipova za dinamički generisane fajlove. MIME tip zaglavlja su identična bilo da je odgovor generisan dinamički ili da je pročitan iz statičkog fajla.

Kad HTML nije dovoljan?

Pomenuo sam nekoliko različitih tipova MIME tip odgovora ranije u ovom poglavlju. Ovi tipovi su važni jer HTML, sam za sebe, jednostavno nije dovoljno moćan. Na sreću, ulazite u Web programiranje u pravom trenutku. Web pretraživači su prošli svoje detinjstvo (verzije 2 i 3), preko mladalačkog perioda (verzija 4), i napredovali ka tome da postanu platforme za isporuku aplikacija. Iako nisu još uvek sposobni kao Windows Forms, oni su prošli dugačak put u prethodnih pet godina i sada su sposobni da manipulišu i HTML i XML informacijama na moćan način.

Sve ove promene su se desile jer je HTML jezik rasporeda. HTML nije jezik za stilove; zbog toga je CSS postao popularan. HTML nije jezik za opisivanje ili manipulaciju slikama; zbog toga je nastao Document Object Model (DOM) da bi Vam omogućio da manipulišete pojavom i pozicijom objekata na ekranu. HTML nije dobar jezik za prenos ili opisivanje opštih podataka; zbog toga je XML brzo postao integralni deo alatne opreme modernih Web pretraživača. Konačno, i za ovu knjigu najvažnije, HTML nije programski jezik. Morate imati programski jezik za izvođenje provera validnosti ili logičkih operacija. Moderni Web pretraživači su delimično tu; oni uglavnom podržavaju skript jezike. U Internet Exploreru 5x i, u manjem stepenu, Netscapeu 6x, sve ove tehnologije su postale isprepletane. Možete raditi sa XML preko CSS ili XSL/XSLT. Možete koristiti DOM da biste dinamički menjali CSS stilove i pojavu objekata. Možete reagovati na neke događaje korisnika direktno sa CSS (poput promene oblika kursora) i možete reagovati ili ignorisati gotovo sve događaje korisnika kroz skript.

Šta može da uradi C#

Pošto se nalazite na samom početku programiranja najnovije tehnologije na strani servera za kreiranje dinamičkih Web aplikacija, trebalo bi da znate šta C# može da uradi. Iznenađujuće, kada razbijete Web program na njegove sastavne delove, postoji vrlo malo razlike između Web programiranja i programiranja standardnih aplikacija.

Donošenje If/Then odluka

If/Then odluke su srž svog programiranja. C# može da donosi odluke na osnovu poznatog kriterijuma. Na primer, u zavisnosti od toga da li je korisnik prijavljen kao administrator, nadzornik ili radnik na traci, C# može da odabere odgovarajuće nivoe odobrenja i odgovore.

Pomoću koda koji donosi odluku, C# može da isporuči neke delove fajla, ali ne i druge, uključujući ili isključujući cele fajlove, ili da kreira potpuno novi sadržaj skrojen za određenu osobu u određenom trenutku.

Obrada informacija klijenta

Čim kreirate aplikaciju, potrebno je da obrađujete informacije klijenata. Na primer, kada korisnik popuni formu, potrebno je da proverite validnost informacija, možda ih uskladištite zbog budućih referenci i odgovorite korisniku. Pomoću C#, imate kompletan pristup svim informacijama koje klijenti šalju i imate kompletnu kontrolu nad sadržajem odgovora servera. Možete koristiti svoje programersko znanje da biste izvršili proveru, snimili podatke na disk i formatirali odgovor. Osim što Vam daje programski jezik za ispunjavanje ovih zadataka, C# Web aplikacije obezbeđuju mnogo pomoći.

C# Web aplikacije koriste ASP.NET kostur koji Vam pomaže da proverite validnost unosa korisnika. Na primer, možete postaviti kontrole na ekran koje obezbeđuju da zahtevano polje sadrži vrednost i automatski proveravaju da li je ta vrednost ispravna. C# Web aplikacije obezbeđuju objekte koji pojednostavljuju operacije sa diskom i bazom podataka i omogućuju Vam da lako radite sa XML, XSLT i kolekcijama vrednosti. Pomoću C#, možete pisati kod za izvršavanje na strani servera koji se ponaša kao da je skript za izvršavanje na strani klijenta. Drugim rečima, možete napisati kod koji se nalazi na serveru, ali odgovara na događaje na strani klijenta u centralizovanom kodu umesto u skriptu za izvršavanje na strani klijenta koji je manje moćan i težak za debagovanje. ASP.NET Vam pomaže da održavate podatke za pojedinačnog korisnika preko objekta Session, umanjite opterećenje svog servera pomoću keširanja i održavate konzistentno vizuelno stanje tako što ćete automatski restaurirati vrednosti ulaznih kontrola preko zaobilaznog puta do servera.

Pristup podacima i fajlovima

U većini aplikacija, potrebno je da čitate ili smeštate permanentne podatke. Za razliku od prethodnih verzija ASP, ASP.NET koristi .NET kostur za obezbeđivanje vrlo moćnog pristupa fajlovima. Na primer, mnoge poslovne aplikacije primaju podatke, često preko noći, iz velikih računara ili servera baze podataka. Tipično, programeri pišu posebne programe sa zakazivanjem za čitanje ili raščlanjivanje i slanje novih fajlova podataka u obliku odgovarajućem za aplikaciju. Kada iz nekog razloga fajlovi podataka kasne ili se nikad ne pojave, često se dese veliki poslovni poremećaji. Slično, da li ste ikada napisali program koji kreira fajl i kasnije proba da mu pristupi i otkrije da ga je korisnik u međuvremenu obrisao ili premestio? Znam - sigurno ste napisali zaštitni kod tako da Vaš program može da obnovi fajl ili bar da se završi graciozno, zar ne?

Mnoge aplikacije će biti mnogo lakše za pisanje i održavanje ukoliko sam program može da sarađuje sa fajl sistemom da bi primio obaveštenje kad god se promeni sadržaj određenog direktorijuma. Na primer, ukoliko napišete kod koji počinje proces uvoza podataka kada god pristignu podaci sa velikog računara, možete izbeći pisanje vremenskih petlji koje proveravaju da li se fajl pojavio ili aplikacije sa zakazivanjem koje se pokreću čak i ako podaci nisu dostupni. Slično, da biste primili obaveštenje pre nego što korisnik obriše taj kritični fajl, možete ne samo da izbegnete obavezu da pišete zaštitni kod već i da sprečite problem pre nego što nastane!

Otkrićete da možete da izvodite ove tipove zadataka mnogo lakše pomoću C# nego što ste to mogli u ranijim verzijama bilo kog programskog jezika. Otkrićete da su najčešće operacije sa fajlovima i bazama podataka lakše (iako razvučenije) u C#. Na primer, jedna od češćih operacija je prikazivanje rezultata upita baze podataka u HTML tabeli. Pomoću VBScript ili JScript koda u klasičnoj ASP aplikaciji, morali ste da petljom prolazite kroz set zapisa koji je vraćen iz upita i da sami formatirate vrednosti u tabelu. U C#, možete prihvatiti set podataka i upotrebiti kontrolu Repeater da biste izveli dosadnu operaciju petlje.

Formatiranje odgovora pomoću XML, CSS, XSLT i HTML

Kao što sam rekao ranije, imate kompletnu kontrolu nad odgovorom vraćenim iz Vaše aplikacije. Donedavno, programeri Web aplikacija su morali da brinu samo o Web pretraživaču i verziji korišćenoj za klijente aplikacije, ali sada je eksplozija drugih tipova klijenata iskomplikovala stvari. Priručni uređaji, hardver namenjen pristupanju Internetu, pejdžeri, telefoni sa pristupom Internetu i stalno rastući broj standardnih aplikacija povećavaju zahteve za formatiranjem toliko da ljudi ne mogu da ih prate.

U prošlosti, za većinu strana sa jednostavnim HTML i skript potrebama, mogli ste obično da se izvučete sa dve ili tri verzije strane - jednu za totalno idiotske Web pretraživače bez ikakvih DHTML ili skript mogućnosti, jednu za Netscape 4 i jednu za IE 4 i noviju. Međutim, kako se broj i tip klijenata povećava, kreiranje ručno formatiranih HTML strana za svaki novi tip klijenta postaje sve manje i manje sposobna i privlačna opcija. Na sreću, široka i rastuća dostupnost CSS i XML korak je u pravom smeru.

Koristeći CSS stilove, često možete podesiti stranu da se prilagodi različitim rezolucijama, dubini boja i dostupnosti. Ali CSS stilovi utiču samo na karakteristike prikaza sadržaja - ne možete da podesite sam sadržaj za različite uređaje koristeći samo CSS. Međutim, kroz kombinaciju XML, CSS i XSLT, možete dobiti najbolje iz oba sveta. XML fajlovi sadrže podatke, XSLT filtrira podatke u zavisnosti od tipa klijenta, a CSS stilovi kontrolišu način na koji se pojavljuju filtrirani podaci na ekranu klijenta. Visual Studi Vam pomaže da kreirate sve ove tipove fajlova, a C# Vam omogućava da njima manipulišete programski. Krajnji rezultat je HTML skrojen prema specifičnim zahtevima klijenta koji se odnose na prikazivanje.

Pokretanje i komunikacija sa .NET i COM+ objektima

U proteklih par godina, najskalabilniji model za ASP je bio upotreba ASP strana kao nešto više od HTML fajlova koji mogu da pokrenu COM komponente smeštene u Microsoft Transaction Serveru (MTS) ili COM+ aplikacijama. Microsoft je ovaj model nazvao *Windows DNA*. Ukoliko ste pravili aplikacije koristeći ovaj model, otkrićete da se malo šta promenilo osim što je sada mnogo lakše instaliranje, pomeranje, promena imena i verzije komponenti. Naravno, to nije tako mala promena.

Do .NET, morali ste da koristite C++ ili Delphi da biste kreirali slobodnonitne COM objekte odgovarajuće za upotrebu u Web aplikacijama. (Da budemo potpuno iskreni, neki ljudi jesu pisali kod koji omogućava VB upotrebu više niti, ali to nije bio prijatan prizor, niti je bio zadatak za programere sa tipičnim veštinama.) Višenitnost možda ne izgleda kao toliko važna stvar ukoliko ste pisali samostalne aplikacije. Na kraju krajeva, većina samostalnih i klijent/server aplikacija ne zahteva višenitnost. Međutim, u svetu Weba, to jeste velika stvar. Web aplikacije gotovo uvek rade sa više istovremenih korisnika, tako da je .NET, da bi bio jezik odgovarajuć za Web aplikacije kao Java, morao da pojača

mogućnost višenitnosti. Mnogi klasični ASP programeri su migrirali iz klasičnog VB, pa su prirodno težili upotrebi tog jezika za generisanje komponenti. Na nesreću, VB5/6-generisane DLL su bili jednonitni stanovi. Bez zalaženja u detalje, to znači da Web aplikacije nisu mogle da skladište objekte pisane pomoću VB5/6 kroz zahteve bez izazivanja ozbiljnih problema sa performansama.

C#-generisani objekti su nasledno slobodnonitni, tako da Vaše Web aplikacije mogu bezbedno, kroz zahteve, da skladište objekte koje kreirate pomoću C#. Naravno, i dalje morate da se bavite problemima izazvanim situacijom kada više niti istovremeno koristi Vaš objekat, ali možete označiti određene delove koda kao kritične i prema tome serijalizovati pristup tim delovima. Ali, to je druga priča. C# Vam takođe omogućava pristup zaveštanim COM DLL fajlovima, tako da možete da koristite postojeći binarni kod bez njegovog prepisivanja u .NET jezik. Postoje neke debate na temu još koliko ćete dugo moći da to radite. Lično, mislim da imate nekoliko godina milosti za ažuriranje svojih COM DLL fajlova u .NET. Da biste koristili postojeće COM DLL fajlove u .NET, "uvozite" biblioteku tipova. Jedan način da to uradite je pomoću usluge `TlbImp.exe`, koja kreira "omotač" za interfejs klase, preko koga možete da pozivate metode i svojstva klase. Naravno, postoji mala cena u performansama kada se omotač koristi za bilo šta, ali to je često prihvatljivo kada je alternativa ponovno pisanje postojećeg i isprobanog koda.

Možete baš tako lako krenuti u suprotnom smeru i izvesti .NET asembler sklopove za upotrebu sa neupravljanim C++, VB5/6, Delphi ili bilo kojim COM naklonjenim jezikom. Da biste to uradili, upotrebite uslugu `TlbExp.exe`. Ova usluga kreira biblioteku tipova, ali je ne registruje. Iako je `TlbExp` lakši za pamćenje (suprotan je od `TlbImp`), druga usluga, nazvana `RegAsm.exe`, može da uradi i kreiranje i registraciju biblioteke tipova u isto vreme. Upotrebite indikator `/tlb` uz `RegAsm.exe` da biste joj rekli da kreira fajl biblioteke tipova. `RegAsm.exe` takođe možete koristiti za kreiranje REG (registracionog) fajla, umesto da zaista registrujete klase iz svog asembler sklopa, što je korisno kada kreirate instalacione programe za instalaciju koda aplikacije na drugi računar.

Prednosti C# u Web aplikacijama

C# je ekstremno moćan alat za izradu aplikacija za Windows platformu (možda jednog dana i za druge operativne sisteme). Ali to sigurno nije jedini alat za izradu aplikacija. Malo toga C# može da uradi što stariji jezici ne mogu ukoliko ste raspoloženi da se dovoljno udubite u API ili napišete dovoljno koda. Međutim, obezbeđujući ugrađenu podršku za određene vrste aplikacija, za upravljanje memorijom i za objektno orijentisano programiranje, C# znatno reducira napore uložene u njihovu izgradnju.

Web servisi

Web servis nije ništa više nego Web interfejs za objekte koji se izvršavaju na serveru. Reći ćete: "Čekaj, zar to nije isto što i Distributed COM (DCOM)?" Ne baš, ali je slično. DCOM dozvoljava Vašim aplikacijama da pokreću i koriste udaljene aplikacije i DLL fajlove kao da se izvršavaju na lokalnom računaru. On to radi tako što kreira proxy "talone" sa obe strane transakcije. DCOM obavija poziv funkcije, potprograma, metode ili svojstva iz Vaše lokalne aplikacije, zajedno sa svim pridruženim parametrima, i šalje ih preko mreže primajućem talonu na serveru. Talon servera raspakuje vrednosti, pokreće objekat ili aplikaciju (ukoliko je potrebno) i pravi poziv, prosleđujući parametre. Do suprotne operacije dolazi uz pomoć povratnih vrednosti. DCOM koristi visokoefikasan binarni omotač za slanje podataka preko mreže.

DCOM je bio kreiran u eri kada su udaljeni pozivi dolazili sa računara koji su smešteni u čvrsto povezane vlasničke mreže. Kako su kompanije počele da koriste javni Internet za poslovne namene, mreža nije više vlasnička; umesto toga, DCOM pozivi moraju da pređu granice između javne mreže i privatne korporativne mreže. Međutim, puštanje binarnih podataka kroz tu granicu je nasledno opasno, jer ne možete znati šta će podaci da naprave. Na primer, podaci mogu da sadrže virtuelne programe. Zbog toga, kompanije stavljaju vatrene zidove da spreče binarne podatke da pređu granicu. Tekst podaci, poput HTML, mogu proći granicu bez smetnji, ali binarni podaci ne mogu. Na nesreću, to ima sporedan efekat u sprečavanju DCOM da lako operiše kroz vatreni zid, jer su vatreni zidovi generalno nesposobni da prave razliku između potencijalno nebezbednih javnih binarnih podataka i savršeno bezbednih DCOM binarnih podataka.

Web servisi su rešili taj problem. Oni izvršavaju potpuno iste zadatke kao DCOM - dopuštaju Vam da koristite udaljene objekte. Međutim, oni koriste drugačiji sistem, nazvan Simple Object Access Protocol (SOAP), za pakovanje podataka poziva i parametara. SOAP ima format tekst fajla. On koristi XML za pojednostavljivanje sintakse za identifikaciju različitih tipova vrednosti podataka potrebnih za pravljenje generičkih udaljenih poziva. Pošto je SOAP tekst fajl, on može da prelazi granice vatrenog zida. Međutim, SOAP nije obavezan za pravljenje udaljenih poziva; on je jednostavno standardizovan i prema tome pogodan metod da se to uradi. Drugim rečima, potpuno ste slobodni da pišete sopsvene daljinske omotače - ali ako uradite to, potrebno je i da kreirate sopstvene funkcije za prevođenje.

C# i Visual Studio imaju opsežnu podršku za SOAP. Zapravo, upotreba SOAP u C# je transparentna; .NET kostur vodi računa o svim zahtevima prevođenja vrednosti i prenosa, ostavljajući Vas slobodnim da se koncentrišete na samu izgradnju aplikacija. Proces izgrađivanja Web servisa je neverovatno sličan procesu izgrađivanja COM DLL - ili u tom smislu, pisanju bilo kog drugog .NET koda, jer sve što je potrebno da uradite da biste izložili metodu ili čitavu klasu kao Web servis je da dodate atribute - komadiće metapodataka koji sadrže informacije o kodu.

Najveći problem sa Web servisima i SOAP su performanse; jednostavno nije efikasno prevoditi vrednosti u tekst prezentacije i iz njih, kao što je to slučaj prilikom njihovog prevođenja u binarni format, poput onog koji koriste DCOM i CORBA, i iz njega. I pored toga, u opasnom svetu, SOAP je neophodno zlo i mislim da ćete biti prijatno iznenađeni time kako brzo rade Web servisi. Iako je stvarna razlika u performansama sigurno merljiva, opažena razlika medju njima je zanemarljiva osim ukoliko ne izvodite duge serije udaljenih poziva u petlji (a to bi trebalo da izbegavate sa svakom udaljenom tehnologijom).

Tanki-klijent aplikacije (Web forme)

C# radi u harmoniji sa ASP.NET da bi Vam omogućio da izgrađujete aplikacije na bazi Web formi. Web forma, kao što ćete videti u poglavljima 4, "Uvod u ASP.NET," i 5, "Uvod u Web forme," jeste HTML forma integrisana sa C# (ili sa bilo kojim od mnoštva .NET jezika koji će se sigurno pojaviti uskoro) kodom. Ukoliko su Vam bliski Active Server Pages (ASP), JavaServer Pages (JSP) ili PHP Hypertext Processor (PHP), brzo ćete se sprijateljiti sa C# Web aplikacijama i Web formama. Ukoliko niste pisali Web aplikacije pomoću jedne od ovih tehnologija, imate sreće što ćete biti uvedeni u polje Web aplikacija sada, a ne ranije, jer C# čini izradu Web aplikacija sličnom izradi Windows aplikacija.

Web forme izgrađujete prevlačeći i spuštajući (dragging adn droping) kontrole na površinu za dizajniranje forme. Nakon što postavite kontrolu, možete je duplo kliknuti i dodati kod koji reaguje na događaje kontrole. Web forme podržavaju Web kontrole analogne većini poznatih Windows kontrola, kao što su tekst kontrole, natpisi, panel kontrole i polja liste. One čak podržavaju i nevidljive kontrole kao što su tajmeri.

Na stranu pogodnost Web formi; Vi i dalje izgrađujete aplikacije na bazi Web pretraživača ili tanki-klijent aplikacije, tako da možete da očekujete da ćete izgubiti nešto od funkcionalnosti koju imate sa Windows klijentima. Međutim, (a ja mislim da je to najvažnija promena koju ćete videti u .NET), niste više ograničeni na tanki-klijent Web aplikacije. Kombinujući Windows klijente sa Web servisima, možete izgraditi bogat-klijent aplikacije gotovo sa istom lakoćom. Zapravo, tehnologija olakšava izgradnju oba tipa aplikacija - i opslužuje oba zajedničkom centralizovanom bazom koda.

Bogat-klijent aplikacije (Windows forme)

Može delovati čudno što sam uključio aplikacije Windows formi u knjigu o izgradnji Web aplikacija, ali Vas mogu uveriti da to neće izgledati čudno kada budete završili knjigu. Razlika između bogat klijent i tanki-klijent aplikacija rapidno iščezava. Kako Web pretraživači dodaju mogućnosti, oni postaju deblji, a kako aplikacije Windows formi stiču mrežne mogućnosti, postaju sposobnije za korišćenje servisa na bazi Weba. Zato sada, kada su u pitanju aplikacije Web formi i Windows formi, jedino treba da utvrdite da li možete lako da isporučite kod aplikacije Windows formi bazi klijenta ili se morate osloniti na funkcionalnost bilo kog Web pretraživača ili "korisničkog agenta" koji je instaliran na klijent računarima. Izgradićete oba tipa aplikacija u ovoj knjizi. Videćete razlike u projektovanju i distribuciji aplikacije, a potom možete da odlučite sami.

Rezime

Videli ste da klijenti komuniciraju sa Web serverom u kratkim transakcionim naletima. Klijent zahtevi su napravljeni kao anonimni, tako da morate da planirate i kodirate bezbednost i proveru autentičnosti ukoliko Vaša aplikacija radi sa osetljivim podacima. Između zahteva, server "zaboravi" na klijenta, pa osim ukoliko ne primorate klijenta da prosledi kolačić ili neki drugi identifikacioni simbol za svaki zahtev, server podrazumeva da je klijent potpuno nov. Web aplikacije koriste ove identifikacione simbole za povezivanje vrednosti podataka sa pojedinim Web pretraživačima ili (kod bezbednih sajtova) pojedinim korisnicima. Strategija koju odaberete za održavanje vrednosti ovih podataka kroz više zahteva naziva se "održavanje stanja" i to je jedini teži problem u izgradnji Web aplikacija.

C# pomaže u pojednostavljivanju procesa izgradnje Web aplikacija preko Web formi, Web servisa, robusnih mrežnih sposobnosti i tesne integracije sa ASP.NET, koji obezbeđuje infrastrukturu za servisiranje Web zahteva. Nezavisno od postojanja Web Form editora Visual Studia, i dalje postoji prednost učenja pozadinskog jezika koji se koristi za kreiranje Web formi - HTML. Na sreću, kao programer naviknut na pamćenje kompleksnih operacija koda, otkrićete da je HTML direktan i jednostavan. Osnove HTML možete naučiti za otprilike pola sata. U poglavlju 2, "HTML osnove," poći ćemo na polučasovnu turneju kroz HTML, koja bi trebalo da bude dovoljna da razumete HTML kod koji ćete viđati u ostatku ove knjige. Ukoliko već znate HTML, možete to poglavlje pregledati zbog podsećanja ili ga jednostavno preskočiti i skoncentrisati se na poglavlje 3, "Kratak vodič za dinamičke Web aplikacije."