DEO

I

Razvoj web aplikacija pomoću Active Server Pages

U OVOM DELU:

1. Uvod u web aplikacije
2. Vaša ASP stranica
3. Uvod u Visual InterDev
4. Rad sa ASP objektima

POGLAVLJE 1

Uvod u web aplikacije

Ova knjiga će da Vam pokaže kako da napravite web aplikacije sa SQL Serverom u pozadini. Prvo treba da vidimo i da shvatimo šta je to web aplikacija i kako možemo da koristimo Microsoft tehnologiju za njeno kreiranje. U uvodnom poglavlju ćemo da istražimo teme kao što su HTML, IIS, ASP i razvoj u više nivoa. Ovo poglavlje pretpostavlja da ste upoznati sa HTML-om. Ako to nije slučaj pogledajte prilog E.

Šta su web aplikacije?

Web aplikacije su aplikacije koje rade na Webu. Ako ovo zvuči kao da se ponavlja, razmislite o ovome: mnogi web sajtovi su samo online katalozi, koje ponekad nazivamo brošurama. To su samo strane sa slikama i opisima. Kada govorimo o kreiranju aplikacija, govorimo o kreiranju sajtova koji nešto rade. Oni Vam omogućavaju da unosite informacije i onda inteligentno odgovaraju na Vaše zahteve. Ako ne biste pravili web aplikacije, imali biste samo obične web strane koje ne bi bile više interaktivne nego strane u časopisu koji čitate.

Web aplikacije uključuju sve e-commerce sajtove koji su postali tako moderni u poslednje vreme. Ovi sajtovi Vam omogućavaju da pretražujete katalog, ali možete da smeštate neke stvari u svoju torbu za kupovinu, da izaberete način isporuke i da onda platite, ali ne preko telefona. Mnogi sajtovi omogućavaju da vidite da li onoga što Vam treba tog trenutka ima u skladištu, tako da znate kada možete da dobijete određenu stvar. Druge aplikacije sadrže i druge poslovne primene. Na primer, kompanija A može da prati demografske podatke o svojim kupcima. Druga kompanija može da se prijavi na web sajt kompanije A i da pokrene izveštaj u pokušaju da pronađe svoju ciljnu demografsku oblast. To znači da ovi sajtovi moraju da imaju pristup do baze podataka u realnom vremenu i mogućnost ne samo za čitanje podataka, već i za kreiranje strana u letu, tako da se promene u bazi podataka mogu da prikažu. Kreiranje strana u letu je srž svih web aplikacija.

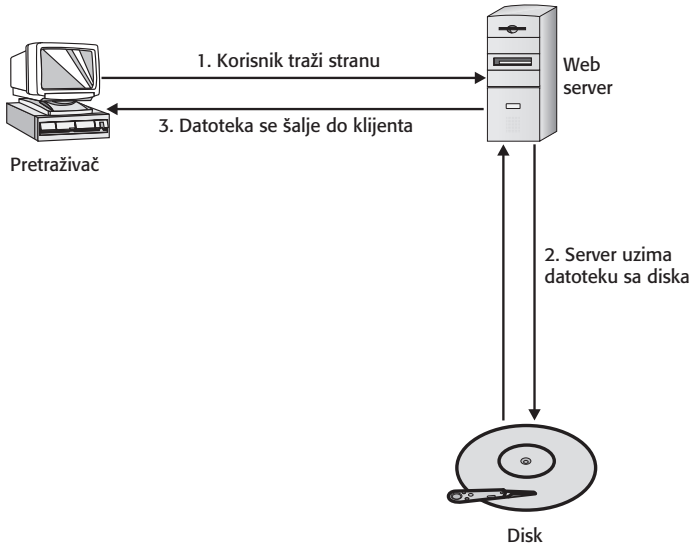
Da biste mogli da kreirate web aplikacije, morate pre toga da shvatite kako web serveri i web strane rade. Nakon toga ćemo da pokažemo kako da napravite aplikacije kojima se upravlja preko podataka koji se nalaze u SQL server bazi podataka.

Web serveri i HTML

U svom jezgru web serveri mogu da budu vrlo jednostavni programi. Računar klijent koji je povezan sa serverom zahteva konkretnu stranu od konretnog web servera. Na web serveru, strana je obična datoteka koja se čuva na fizičkom disku. Server pronalazi stranu na nekom računaru i šalje njenu kopiju (datoteke) kao odgovor na zahtev klijenta. Ovo je prikazano na slici 1.1.

Zahtev koji klijent šalje do web servera i njegov odgovor idu preko standardnog Internet protokola: HTTP (Hypertext Transfer Protocol). HTTP prima zahtev od klijenta i pakuje ga u format koji standardni web server može da prepozna. Kada server odgovara, informacije o strani se šalju nazad do klijenta preko istog protokola. HTTP se nalazi na vrhu dobro poznatog mrežnog protokola IP (Internet protocola).

Prema tome, osnovna funkcija web servera je da šalje datoteke koje pretraživač zatraži. Ovo je jedna od najvažnijih stvari koje treba da shvatite kada je reč o statičkim HTML (HyperText Markup Language) stranama. Web server ne vrši obradu ovih strana na bilo koji način. Web serveri su nemi kada se radi o statičkim web stranama. Većina web servera, međutim, ima mogućnost da obave izvesnu obradu web strana pre nego što ih pošalju do klijenta. Ova obrada strana zahteva izvestan stepen inteligencije i time se uglavnom ova knjiga i bavi.



SLIKA 1.1 *Jednostavni web server koji vraća stranu kao odgovor na zahtev klijenta*

Pre svega, međutim, treba da shvatite šta se dešava sa jednostavnim, statičkim HTML stranama. Ako se strane šalju do klijenta bez obrade, kako se onda prikazuju. Ovo je zadatak pretraživača. Dva najpopularnija programa te vrste su Microsoftov Internet Explorer i Netscape Navigator, premda postoje i drugi programi, kao što je Opera, WebTV i AOL (America Online). Pretraživači se prave sa ugrađenim mogućnostima da uzmu HTML stranu, interpetiraju njen sadržaj i prikažu je na ekranu.

HTML je prvobitno nastao sa ciljem da se sadržaj dokumenta prikaže na bilo kojoj platformi. Ovo je urađeno tako što je sadržaj označen, a onda je pretraživaču prepušteno da prikaže stvari na način koji je odgovarajući za konkretnu platformu. Ako za kreiranje dokumenta koristite aplikaciju, kao što je Microsoft Word, Word čuva dokument u binarnom obliku, da bi sačuvao i formatiranje teksta. Ovaj binarni format nije obavezno kompatibilan sa drugim platformama. Prema tome kreatori HTML-a su želeli format koji bi sačuvao formatiranje, ali koji može da se prenosi na različite platforme. Da bi to moglo da funkcioniše, HTML je morao da bude samo običan tekst, bez ikakvih dodatnih binarnih informacija.

NAPOMENA

Mnogi ljudi su uzeli slavu za osmišljavanje koncepta HTML jezika, ali su pravi stvaraoci bili Tim Berners-Lee i Robert Caillau. Njih dvojica su kreirali HTML kao način za obezbeđivanje informacionog sistema koji ne zavisi od platforme i koji može da radi na različitim računarima. Ovo su uradili u CERN (Evropska organizacija za nuklearna istraživanja) u Ženevi gde su radili. Godine 1991. CERN je kreirao Web koji je narastao do onoga što mi danas poznajemo pod tim imenom. ■

Da bi izvršio sve zahteve koji se odnose na tekst, HTML čuva informacije o formatu u obliku tzv. oznaka (tagova). Ove oznake se ne obrađuju od strane web servera, već ih interpetira pretraživač na računaru klijenta. Pretraživači su aplikacije koje se prave za određene platforme, ali su oni svi

pisani tako da mogu da na isti način interpetiraju HTML. Oni to rade na osnovu standarda koji je prihvaćen od strane World Wide web konzorcijuma u Bostonu, država Masachusets — SAD (http://www.w3.org). To znači da će svaki pretraživač oznaku <H1> da intepretira kao komandu za prikazivanje teksta u većem i masnijem fontu.

Zbog cilja koji je imao HTML da se tekst prikazuje na ekranu pod različitim platformama, pri čemu treba da se sačuva format, ja sam želeo da ukažem na nešto o čemu većina ljudi nikad nije razmišljala. HTML nije programski jezik. Programski jezici imaju izvesne elemente, među kojima su obično promenljive, petlje i strukture, a donošenje odluka (kao što su if iskazi). HTML nema nijedan od ovih elemenata. HTML je računarski jezik, ali njegova svrha je da čuva tekstualne informacije i njihov format. On nije programski jezik i prema tome ne može da obavlja funkcije koje bi se očekivale od aplikacije. Kao što ćete da vidite kasnije u ovom poglavlju, postoje izvesni načini da našoj web strani dodate neku vrstu skripta na strani klijenta koja omogućava web strani da vrši neku obradu. Realizacija međutim vodi do JavaScripta ili VBScripta, a ne do HTML jezika. Pokretanje koda na mašini klijenta zahteva mešavinu HTML i skript jezika. Da biste kod servera pokrenuli neki kod pomoću Active Server Pages, morate da imate HTML i skript jezik. Više o ovome u drugom poglavlju.

NAPOMENA

U ovoj knjizi ja ću stalno da koristim termin JavaScript. Ovo je skript jezik koji je kreiran od strane Netscapea. Microsoft je kreirao svoju verziju i nazvao je Jscript. Ovi jezici su slični, ali nisu potpuno identični. Kao pokušaj da prevaziđe ove razlike napravljen je i treći jezik ECMAScript. On je napravljen tako da radi na obema platformama.

U ovoj knjizi ja ću da koristim termin JavaScript. Molim Vas da shvatite da je ovo samo opšti termin koji se odnosi i na JavaScript i na Jscript i ECMAScript. ■

Jedno što je bitno da shvatite je to, da različiti pretraživači neće uvek isti tekst da prikažu na isti način. Pre svega, računari mogu da imaju različitu rezoluciju. Strana koja dobro izgleda sa rezolucijom od 800x600 možda neće da izgleda tako dobro sa rezolucijom 640x480. Pored toga, na različitim mašinama, dubina boja može različito da se podesi, što dovodi do toga da boje i grafika neće da se prikažu onako kako očekujete.

Ne samo da isti kod na različitim mašinama može da se različito prikaže, već i proizvođači pretraživača dodaju svoja proširenja HTML jezika. Ovo dovodi do koda koji radi samo pod jednim pretraživačom. Na primer, oznaka <BGSOUND> je ubačena od strane Microsofta da bi prikazala zvuk iz pozadine na nekoj web strani. Ova oznaka radi jedino u Internet Exploreru. Umesto da dodaju oznake, proizvođači pretraživača ponekad samo dodaju atribute za neke oznake.

Ovo nas dovodi do druge važne stvari u radu pretraživača. Oni ignorišu one oznake i atribute koje ne razumeju. Pogledajmo sledeći segment koda:

```
<H1>Welcome to Northwind</H1>
  <CRAIG>This text is formatted with the craig style</CRAIG>
```

Pogodite šta? Ne postoji oznaka <CRAIG>. Pretraživač koji intepretira ovaj kod će jednostavno da zanemari oznaku i prikaže tekst koji se nalazi između početka i kraja oznake kao običan tekst. Pretraživač očekuje da mu oznaka kaže kako da formatira tekst, i ako ne razume neku oznaku, on je ignoriše i prikazuje tekst sa podrazumevanim fontom i veličinom slova. Ono što se prikazuje na ekranu vidi se na slici 1.2.

SLIKA 1.2 *Segment koda <CRAIG> sa oznakom*

Pretraživači su namerno vrlo zaboravni. Ovo ima svoje prednosti i nedostatke. Prednost je da tekst neće videti poruke o greškama. Glavni nedostatak je, međutim, da mi kao programeri, ne vidimo greške pri pokušaju da debagujemo stranu. Pored toga, činjenica da HTML ne pravi razliku između malih i velikih slova, i da neke oznake ne moraju da budu zatvorene, vodi do vrlo loše struktuiranog jezika. Programeri vole strukture, ali HTML ne primenjuje dobro strukturno kodiranje. Postoje neki alati koji mogu da pregledaju dokument, kao što je World Wide Web Consortium's HTML Validation Server (http://validator.w3.org) i Netscapeov Web Site Garage (http://Websitegarage.netscape.com).

Slika 1.2. Deo koda sa oznakom <CRAIG>Pogledajmo na trenutak skript na strani klijenta. Drugo što je bitno kod HTML-a je da kad se jednom dokument prikaže u pretraživaču, on ne može da bude ažuriran bez osvežavanja, što znači da ponovo mora da se ide do servera i da se ponovo uzme strana. Drugim rečima tekst koji se jednom prikaže više ne može da se promeni. Ovo je očigledan dokaz za statičnost HTML-a. Postoje načini da se ovo zaobiđe. To su polja na formi koja mogu da se ažuriraju pomoću nekog koda kod klijenta, ili možete da stavite delove koda u programabilne objekte preko Dinamičkog HTML-a. I jedno i drugo, međutim, zahtevaju skript na strani klijenta, što inicijalno nije deo HTML-a i još uvek nije podržano od strane svih pretraživača.

Skript na strani klijenta

Sećate se da HTML nije programski jezik i da prema tome ne može da vrši nikakvu obradu. Mi, međutim, možemo da na strani klijenta ubacimo skript koji će da obavi izvesne akcije. Postoji nekoliko stvari koje treba imati u vidu kada se govori o skriptu na strani klijenta.

- Može li pretraživač klijenta uopšte da podrži skript
- Ako može, koje jezike podržava

Pravilo kojeg se treba držati je jednostavno. VBScript je poznat većini programera. Generalno se smatra da je programiranje sa njime lakše. On radi jedino u Internet Exploreru. JavaScript radi i u Internet Exploreru i u Netscape Navigatoru. Očigledno je da više mogućnosti ima ako koristite JavaScript.

NAPOMENA

Nemojte da pomešate JavaScript i Java Aplete. Java apletu su kompajlirane komponente koje se učitavaju i izvršavaju u Java Virtuelnoj mašini pretraživača klijenta. JavaScript i VBScript se ne kompajliraju, već ih interpetira pretraživač. ■

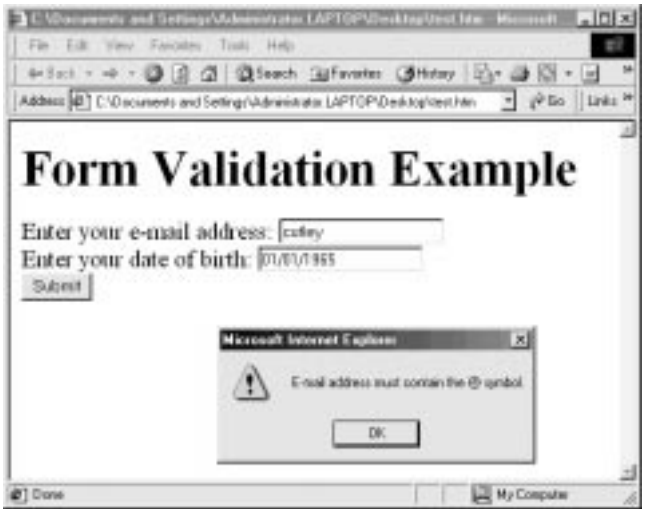
Prilikom objašnjavanja rada sa HTML jezikom ja sam ljudima pokazivao kako da kreiraju HTML forme. Forme, naravno, omogućavaju korisniku da unese izvesne informacije i da ih pošalje serveru, gde mi obično nad njima vršimo neku magiju, kao što je njihovo ubacivanje u bazu podataka. Pitanje koje se skoro uvek postavlja, ako su studenti pažljivo pratili predavanje je "kako možemo biti sigurni da su ispravni podaci koje smo uneli u formi". Odgovor je kod čistog HTML-a jednostavan. Ne možete. To znači da ako zatražite da korisnik unese svoje godine i onda unesete "abc" do servera će za to polje da bude poslat tekst "abc", tako da proveru morate da obavite na serveru. Ovo nije loše, ali pogledajmo šta korisnici rade. Oni popune formu, kliknu Submit dugme i čekaju. Podaci idu preko Interneta do servera, na kome Vi obavljate proveru podataka. Ako je vrednost u polju neispravna, Vi morate da napravite stranu koja će korisniku reći da je u pitanju greška i da tu stranu pošaljete nazad. Sada korisnik dobija poruku o grešci i on mora da u svom pretraživaču izabere dugme Back, da se vrati na početnu formu i da promeni podataka. Ceo proces može da potraje izvesno vreme.

Ako imate skript na strani klijenta, Vi biste mogli da odmah ocenite da li su neki podaci ispravni i to pre slanja na server. Na taj način korisnici ostaju na istoj strani i ne moraju da biraju dugme Back, nakon primanja poruke o grešci. Očigledno je da ne mogu svi podaci da budu ocenjeni kod klijenta. Da bismo procenili ispravnost nekih vrednosti možda moramo da pristupimo bazi podataka. Obično ne možemo direktno od klijenta da pristupimo bazi, premda je i to moguće, kao što ćemo videti u dvanaestom poglavlju.

Sledeći kod prikazuje jednu pojednostavljenu web stranu sa formom koja ima dva polja. Ovo će da nam posluži kao primer, kako se skript na strani klijenta može da upotrebi za proveru vrednosti koje se unose u formi. Ako su podaci koji se unesu u oba polja ispravni, oni će da budu poslati na server. Ako to nije slučaj, mi ćemo da obavestimo korisnika o problemu i pomoći mu da vidi u čemu je pogrešio. Provera je jednostavna. U polju email moramo da budemo sigurni da negde u tekstu postoji simbol "@". U polju date of birth (datum rođenja) proverićemo da li je u pitanju ispravan datum. Kod je dalje odgovoran za slanje forme. Da biste videli formu u akciji, pogledajte sliku 1.3. U ovom slučaju mi ne šaljemo i stvarno formu, jer još uvek nemamo server koji bi primio naše podatke. Primetićete da je kod pisan u VBScriptu, što znači da će ovaj primer da radi samo u IE-u.

```
<HTML>
   <H1>Form Validation Example</H1>
   <FORM name="frmSignUp" method="post">
      Enter your e-mail address: <INPUT name="email"><br>
      Enter your date of birth: <INPUT name="dob"><br>
```

```
      <INPUT type="button" name="cmdSubmit" value="Submit">
   </FORM>
   <SCRIPT language="VBScript">
      <!--
      Sub cmdSubmit_onClick()
         If InStr(document.frmSignUp.email.value,"@")=0 Then
            alert "E-mail address must contain the @ symbol."
         Else
            If IsDate(document.frmSignUp.dob.value) then
               alert "Data is fine, form will be submitted"
            Else
               alert "Date of birth is not a valid date format." & _
                     " Please re-enter."
            End If
         End If
      End Sub
      -->
   </SCRIPT>
</HTML>
```



SLIKA 1.3 *Skript na strani klijenta proverava email adresu pre nego što pošalje formu. Ovde je kod uhvatio adresu koja nije ispravna*

Klijent može da se proširi i primenom nekih drugih tehnika. To su na primer, ActiveX kontrole i Java apleti. Iako su ove dve tehnologije međusobno različite, njihov rezultat je sličan. Klijent može da radi ono što ne može da se radi sa običnim HTML-om. Za sada najveća razlika je u tome da ActiveX kontrole rade jedino u IE-u. Java apleti rade i Internet Exploreru i Netscape Navigatoru, iako ne rade svi apleti na isti način u oba pretraživača.

ActiveX kontrole su popularne zato što su one deo sveta Windowsa i zato što ih je lako kreirati u Visual Basicu, ili Visual C++-u između ostalog. ActiveX kontrole, međutim, zavise od osobina pretraživača. Kontrole se učitavaju i registruju na mašini klijenta, gde ostaju instalirane. Na taj način kada korisnik, ponovo poseti isti sajt, kontrola je već prisutna na lokalnoj mašini i ne mora ponovo da se prebacuje. Kontrola ima neke osobine koje biste povezali sa Windows aplikacijom. Ona je upravljana događajima, može da pristupi hardveru mašine, može da čita i piše na disku, itd. Mogućnost pristupa disku računara klijenta navodi neke ljude da brinu o sigurnosti kada rade sa ActiveX kontrolama. Tačno je da maliciozne ActiveX kontrole mogu da oštete mašinu korisnika. Postoje neke mogućnosti zaštite, kao što je na primer, tehnologija Authenticode. Mi ćemo detaljnije da objasnimo ovu tehniku u petnaestom poglavlju, kada se budemo bavili sigurnošću.

Java apleti mogu da budu upravljani događajima i mogu da korisniku ponude bogat interfejs, ali ne mogu da pristupaju hardveru kao što je slučaj sa ActiveX kontrolama. Zbog toga se ljudi prilikom rada sa Javom osećaju sigurnije. Pored toga ovo ima dodatne prednosti pošto Java apleti mogu da rade i u Internet Exploreru i u Netscape Navigatoru. Na raspolaganju Vam stoji puno Java apleta, kao što su različiti kalendari tajmeri i sl.

Skript na strani servera

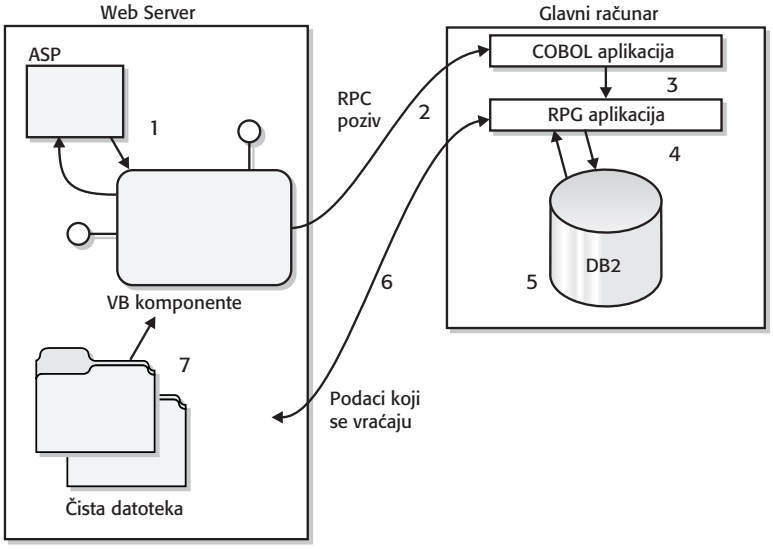
Nakon što smo rekli da web serveri jednostavno šalju nizove statičkih HTML podataka do pretraživača koji ih interpretira, da pogledamo i koncept web servera koji mogu dinamički da kreiraju web strane. Razlozi iz kojih ovo radimo su očigledni. Možete poželeti da kreirate strane koje su povezane sa podacima iz baze podataka, sa rezultatom pretraživanja ili sa nekim informacijama koje su povezane sa vremenskim trenutkom. Bez mogućnosti da web serveri prave web strane u letu. Mi ne bismo imali online liste za kupovinu, skladišta ili neke druge web aplikacije.

Prvi web serveri koji su dozvoljavali dinamičko kreiranje strana su koristili CGI (Common Gateway Interface). CGI se i danas dosta koristi. IIS može da koristi i CGI i Active Server Pages. CGI nije programski jezik. To je tehnologija koja omogućava poziv kompajliranih programa ili skripti na web serveru. Ovi programi ili skripti mogu da prihvate ulaz od korisnika i da onda dinamički kreiraju strane na bazi tog ulaza. CGI se često koristi na UNIX serverima i većina CGI programa su Perl skripta (to se vidi po tome što imaju ekstenziju .pl).

Kao što sam pomenuo IIS može da koristi CGI, ali on koristi i drugu tehnologiju, odnosno Active Server Pages (aktivne server strane). ASP Vam omogućava da ugradite skripta direktno u HTML stranu. To znači da, za razliku od CGI-a ne pokrećete poseban proces kada dinamički kreirate stranu. ASP Vam omogućava da radite sa delovima krajnje HTML strane i da ugradite skript direktno u HTML stranu. Ovaj prilaz je postao vrlo popularan, tako da Java Server Pages sada rade na skoro isti način.

Pošto se ASP izvršava na serveru izbor jezika za skript je posao programera. Sećate se da kod skripta koji je na strani klijenta, morate da vodite računa o tome da li pretraživač podržava jezik koji koristite i ako je to slučaj, o kojoj je verziji reč. Kod skripta na strani servera, ne morate da brinete o mogućnostima pretraživača. Ceo kod se obrađuje na serveru i klijentu se šalje samo HTML strana. To znači da pretraživač klijenta uopšte ne mora da podržava jezik. Kada klijent pogleda ovu stranu, ona njemu izgleda kao obična HTML strana. On ne vidi kod koji se koristio za kreiranje te strane.

ASP nam omogućava da kreiramo mnogo složenije okruženje aplikacije nego ako koristimo samo HTML i skript na strani klijenta. Na primer, možemo poželeti da pristupimo nekim podacima ili čak i programima ili glavnom računaru. Ovi programi bi mogli da obave obradu značajne količine podataka i da onda pošalju rezultat. Mi bismo onda mogli da stavimo podatke u bazu podataka i da ih tako spremimo za dalju manipulaciju pre nego što se napravi završna strana, koja se šalje do klijenta. Jedna takva aplikacija je prikazana na slici 1.4. Mogućnost da pokrenete programe na glavnom računaru, dodajete podatke u bazu ili dobijate podatke iz nje i da pravite strane na osnovu rezultata je mnogo izvan mogućnosti HTML-a i skripta na strani klijenta. Inteligentni web serveri, koji podržavaju skript na strani servera, nam omogućavaju da napravimo bogate aplikacije bez toga da brinemo o pretraživaču koji klijent koristi.



SLIKA 1.4 *Kompleksna web aplikacija*

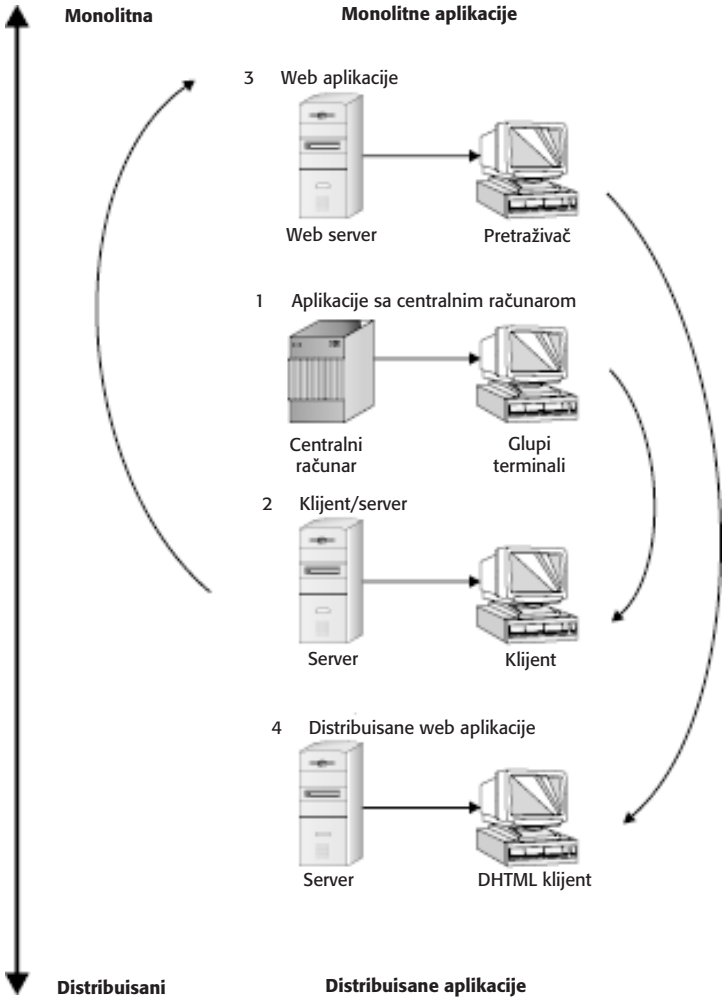
Ako ste ikad radili u okruženju centralnog računara, ovo Vam zvuči veoma poznato. Kada su računari napravljeni, oni su bili ogromne mašine koje su zauzimale veliku zapreminu. Svi su radili na jednom računaru, a ljudi su tom glavnom računaru pristupali preko glupih terminala. Svi terminali su mogli da prihvataju ulaz od strane korisnika i da na ekranu prikazuju informacije. U toku 1991. godine, pojavila se klijent/server tehnologija. Slogan "snaga ljudima" se odnosi na to da se snaga prebacuje na PC računare. Za svakim računarom je sedela po jedna osoba. Podaci su se čuvali na jednom mestu (*server*) i prebacivali su se do PC-a (*klijent*) gde su se obrađivali i prikazivali. Jedino što se delilo su bili podaci, uz nešto poslovne logike koja se obično čuvala kao *uskladištene procesure* u bazi podataka. Ovo je rođendan tzv. distribuisane obrade. Ovakva klijent/server arhitektura, koja se sada naziva *arhitekturom* u dva nivoa ili puni klijent, ima neke prednosti nad tradicionalnim monolitnim sistemima. Prikazivanje je mnogo bogatije i sadrži stvari kao što su podrška za miša, meniji, dugmad, grafici, animacije, boje i sl. Više korisnici ne moraju da kucaju dugačka imena komandi i da čekaju da bi izveštaj dobili na zelenom papiru. Umesto toga oni selektuju nekoliko dugmadi i brzo dobijaju rezultat, na ekranu.

Tu su sada grafici i boje i sva druga pomagala (kada govorimo o menadžmentu ovo nazivamo featuresima).

Klijent/server arhitektura nas je za 180 stepeni okrenula od monolitnih sistema i uvela u eru distribuisanih sistema. Iako ovo korisnicima donosi blagodet, to dovodi do izvesnih glavobolja za IT tehnologije. Svako ko je radio sa klijent/server aplikacijama u ovom periodu se seća horora koji se javljao prilikom distribucije. DLL konflikti, INI datoteke, imena ODBC izvora. Ove stvari koje se odnose na distribuciju se odnose takođe i na održavanje. Morali ste da vodite računa o mašini klijenta. Da li ona ima dovoljno RAM memorije. Da li je veza dovoljno brza. Kako ćete da se povežete sa serverom. Da li ima dovoljno prostora na disku. Sve ovo je ponekad dovodilo do toga da programeri zažale za starim danima kada se radilo sa monolitnim aplikacijama i kada ni o čemu nije trebalo brinuti.

Dolaze web aplikacije. Web aplikacije dovode IT nazad do monolitne strukture koja je poznata i voljena. Sve funkcije obavlja web server. Pretraživač nije ništa više nego glupi terminal. On omogućava korisniku da unese tekst (preko HTML formi) i da prikaže tekst (preko HTML-a). To je sve što pretraživač treba da uradi tako da on u mnogome liči na običan interfejs. Ništa ne mora da bude distribuisano do klijenta. Dokle god klijent ima pretraživač koji je u stanju da razume standardni HTML, on je idealan kandidat za rad sa web aplikacijama. Pored toga povezivanje sa udaljenim korisnicima ne zahteva više posebne linije i modeme. Udaljeni korisnici sada mogu da koriste bilo koji ISP, bilo gde u svetu i da se lako povežu sa aplikacijom. Na slici 1.5 je prikazan ciklus kroz koji treba da prođe jedna aplikacija.

Pre nego što programeri koji razmišljaju na monolitni način postanu suviše radosni, setite se da se ponovo vraćamo na distribuisane tehnike sa tehnologijama, kao što su Dinamički HTML (DHTML). DHTML u ovom trenutku u mnogome zavisi od pretraživača, što znači da ponovo morate da brinete o tome. Pored toga, neke tehnologije kao što je Microsoftovo vezivanje podataka sa klijenta, zahtevaju da izvesne komponente postoje na mašini klijenta. Ova proširenja nas vode nazad do stare dileme. Što je bogatiji korisnički interfejs to je aplikacija manje bogata. Drugim rečima, ako u aplikaciju ubacite sva zvona, pištaljke i osobine koje se mogu naći u IE-u, aplikacija više neće da radi sa Netscapeom, Operom i drugim pretraživačima. Ako ostavimo da to bude čisti HTML to će da radi u svakom pretraživaču, ali korisnički interfejs neće da bude tako bogat. U ovom razmatranju ne postoje unapred određena rešenja. Sve to zavisi od onog što je Vašem klijentu potrebno. U suštini na sva pitanja u web razvoju može da se odgovori jednom reči "zavisi".



SLIKA 1.5 *Ciklus jedne aplikacije*

Pre nego što ostavimo ovu diskusiju i pogledamo šta to Microsoft nudi u ovoj oblasti, treba da imate na umu sledeće:

- Standardni HTML može da posluži jedino za prikazivanje podataka i prihvatanje ulaza od strane korisnika.
- Za proveru podataka koji su uneti u formi, koristi se skript na strani klijenta.
- VBScript na strani klijenta je podržan jedino u Internet Exploreru.
- JavaScript je podržana i u Internet Exploreru i Netscape Navigatoru.
- Dinamički HTML je mešavina skripta i HTML i često je specifičan za određeni pretraživač.

- Skript na strani servera je kod koji se izvršava na web serveru.
- Skript na strani servera nema nikakve veze sa pretraživačom koji klijent koristi.

Microsoftov IIS i ASP

Microsoftov web server se naziva Internet Information Services (IIS). IIS je ugrađen u operativne sisteme Windows NT i Windows 2000. Na operativnim sistemima Windows 95 i Windows 98 može da se koristi podskup IIS-a po imenu Personal Web Server. IIS može da se ponaša kao jednostavan web server, odnosno da servisira statičke web strane kao i bilo koji drugi web server. Međutim, IIS ima i druge osobine koje ga čine vrlo popularnom platformom za razvoj web aplikacija. IIS nudi i takve funkcije kao što su ugrađeno upravljanje stanjima, integracija sa sigurnosnim modelom Windowsa 2000, objekte koji olakšavaju kreiranje web aplikacija, kao i skripte na strani servera preko ASP-a.

Upravljanje stanjima je važno za razvoj web aplikacija. Kada se prijavite na mreži ili na centralnom računaru, pokreće se proces koji verifikuje ko ste Vi i onda drži jednu sesiju otvorenom za Vas sve dok ste prijavljeni. Na nekim sistemima, ako duže vreme budete neaktivni, može doći do toga da Vas sistem odjavi, odnosno da se uništi Vaša sesija i uklone dozvole za rad sa mrežnim resursima. Ovo je primer veze koja ima svoje stanje, odnosno zna se kada ste i kada niste povezani i na osnovu toga može da se upravlja informacijama o Vama dok ste povezani.

HTTP je, sa druge strane, protokol koji nema svoje stanje. Kada korisnik zahteva svoju stranu od web servera, taj zahtev se šalje do servera, koji zatim šalje datoteku do klijenta. Kada se dokument pošalje do klijenta, HTTP veza se prekida. Korisnici sada imaju HTTP dokument u svom pretraživaču, tako da nisu ni svesni da je veza prekinuta. Kada korisnik zatraži sledeću stranu, zahtev ide do servera i server ga vidi kao nov zahtev. Drugim rečima server svaki zahtev smatra novim zahtevom.

Da biste bolje shvatili probleme koji se javljaju kod veze koja nema stanje, zamislite online kupovnu listu. Istraživali ste okolo i pronašli ono što želite da naručite. Izabrali ste neke stavke i sistem je odgovorio da je ta stavka u Vašoj listi. Sada birate dugme Check Out i idete u deo za odjavljivanje. Ali Vaša stavka nije u listi. U okruženju koje nema stanje, baš će se ovo desiti. Aplikacije koje nemaju stanje ne mogu da zapamte da li je nešto bilo naručeno. HTTP, je prema tome izazov kada se radi o kreiranju aplikacija. Aplikacije skoro uvek zahtevaju neku vrstu upravljanja stanjem, tako da programeri moraju da pronađu neki način da to stanje drže pod kontrolom u trenutku kada se veza prekine. Jedna od prednosti IIS-a je u tome da on može automatski da upravlja stanjem za nas, čak i kada koristimo standardni HTTP. Programer ne mora da brine o kreiranju kukija ili prosleđivanju skrivenih promenljivih. Mi ćemo ovim da se detaljnije pozabavimo u četvrtom poglavlju.

IIS je okruženje koje je domaćin za mašinu Active Server Pages. ASP su datoteke sa mešavinom HTML i skript jezika. Ova datoteka se, pre slanja kod klijenta, obrađuje na serveru. ASP su vrlo dobar način za kreiranje web aplikacija i one mogu da koriste različite skript jezike. ASP može lako da pozove COM komponente koje im daju dodatne funkcije, uključujući i komponente koje omogućavaju pristup do baze podataka.

IIS ima i izvestan broj ugrađenih objekata koji olakšavaju kreiranje web aplikacija. Ovi objekti mogu da se pozovu iz bilo kojeg skript jezika koji se koristi u ASP. Na primer, postoje objekti koji olakšavaju čitanje podataka koji se šalju od klijenta preko HTML forme. Drugi objekti olakšavaju pozivanje COM komponenti koje mogu unedogled da proširuju Vašu aplikaciju. Mi ćemo da ispitamo sve ove ugrađene objekte u četvrtom poglavlju.

ASP mapina može da koristi veliki broj različitih skript jezika. Bilo koji interpreter za bilo koji jezik može da se ugradi u ovu mašinu. ASP podržava VBScript i JavaScript. Postoje i interpreteri za PerlScript, Python i Rexx. Ova nezavisnost od jezika je moguća jer o obradi skripta brine web server. Kada ASP mašina završi obradu svih skripta na strani servera, rezultat se šalje do klijenta, kao da je u pitanju standardni HTML. Skript je ugrađen u HTML i strana je obrađena unutar IIS-a. To znači da, osim ako ne koristite CGI nema zasebnog procesa koji bi se pokretao za kreiranje dinamičke strane.

ASP se često naziva tehnologijom "bez kompajliranja". Ovo ponekad može da zbuni programere. ASP se ne kompajliraju od strane programera. Umesto toga datoteke se samo postavljaju na odgovarajući direktorijum web servera. Kada klijent zatraži ASP stranu, ona se interpretira pomoću ASP mašine i stavlja u memoriju u kompajliranom stanju. Strana se kešira radi poboljšanja performansi, pošto to znači da ne mora da se interpretira svaki put kada se zatraži.

Keširanje dovodi do poboljšanja brzine rada, ali ako nema kompajliranja, kako se onda zna kada je strana promenjena. Odgovor je jednostavan. IIS proverava svaki poziv. Recimo da imamo stranu po imenu Order.asp, koja je keširana u memoriji u kompajliranom stanju. Kada Vi, kao programer, promenite ovu stranu i zapamtite je, stara verzija će i dalje da bude u memoriji. Međutim, kada se strana Order.asp pozove sledeći put, IIS proverava da li je datoteka na disku promenjena. Ako to nije slučaj, onda se koristi verzija koja se nalazi u memoriji. U ovom slučaju, međutim, datoteka je promenjena, tako da ASP mašina ide kroz proces interpretiranja nove strane, nakon čega se kompajlirana verzija čuva u memoriji. Na taj način korisnik uvek dobija poslednju verziju, a da Vi ne morate da uradite ništa drugo osim da upamtite datoteku.

Active Server Pages su deo okruženja IIS. Postoji, međutim, i proizvod nezavisnog proizvođača, po imenu ChilliSoft (http://www.chilisoft.com) koji omogućava i drugim serverima da pokreću ASP strane. Na primer, Lotus Domino ili Netscape Enterprise Server bi mogli da se koriste kao web serveri, a da i dalje mogu da se koriste ASP strane. Postoji čak i verzija za Apache web server koji obično radi pod Unixom. Ovaj proizvod omogućava da se ASP strane koriste i izvan Microsoftove platforme, tako da je ovo potencijalno rešenje za mešovita okruženja.

Web aplikacije u n nivoa

Ako se prisetite diskusije da su web aplikacije slične sa monolitnim, mainframe aplikacijama, setićete se da sam govorio o klijent/server aplikacijama u dva nivoa, a ne o aplikacijama u n nivoa koje su snažnije i bolje. N nivoa ili tri nivoa, podrazumeva da se logika sistema (poslovna logika), odvoji od korisničkog interfejsa. Sada imate tri logička nivoa. To je nivo za prezentaciju, poslovni nivo i nivo podataka. Nivo za prezentaciju sadrži samo komponente korisničkog interfejsa, kakve su forme koje imate u svojoj aplikaciji. Na ovom nivou nema poslovne logike. Ovde želite samo onoliko koda, koliko je potrebno za prikazivanje podataka i prihvatanje ulaza od strane korisnika (zvuči poznato). Ovakav pristup se često naziva tankim klijentom.

Klijent je tanak zato što ne radi ništa drugo nego samo prikazuje podatke i prima ulaz od strane korisnika. Najveći deo posla se radi u nivou poslovne logike. Ovde ćete da kreirate niz komponenti koje mogu više puta da se koriste, a koje implementiraju pravila kojima se podvrgava aplikacija. Ove komponente mogu da se koriste u većem broju aplikacija, da bi se obavljale iste funkcije.

Nivo podataka ponekad sadrži samo bazu podataka, a ponekad i skup komponenti koje leže između poslovnih komponenti i baze. Ovo je odluka koja se donosi u procesu projektovanja aplikacije i koja zavisi od različitih faktora. Mi ćemo da pomenemo neke od tih faktora kasnije. Bez obzira da li se u ovom nivou nalaze neke komponente, baza podataka može da ima izvesnu poslovnu logiku koja bi bila implementirana u vidu uskladištenih procedura ili okidača. Ovo često donosi bolje performanse, ali ima svoju cenu u kasnijoj promeni baze ako to bude potrebno.

Kada govorimo o komponentama u poslovnom nivou ili nivou podataka, obično mislimo na COM komponente. COM je Microsoftova tehnologija, koja omogućava jednoj aplikaciji da kreira objekte nezavisno i da ih postavlja u posebne delove kompajliranog koda. Postoje neki faktori koji su potrebni za COM komponente, a i COM radi mnogo više od onoga što smo upravo pomenuli. Za sada je dovoljno da znate da COM omogućava našem prednjem delu da poziva poslovne komponente koje smo kreirali.

Pogledajmo jedan primer za aplikaciju u n nivoa. Bolnica čuva podatke o svojim pacijentima prema određenoj strukturi podataka. Pacijenti imaju neki oblik svog id broja, ime, adresu, jedno ili više osiguravajućih društava itd. Svaka bolnica može da ima posebne sisteme za upravljanje pacijentima. Svaki od tih sistema bi trebalo da ima uvid u strukturu pacijenata na potpuno isti način. Prema tome, ako je na primer ID oznaka pacijenta skup znakova od deset karaktera, svaki sistem koji upravlja pacijentima bi trebalo da u sebi ima ovu informaciju. Ona bi bila kodirana na neki način. Sada dolazi problem kod kodiranja. Šta ako treba da se promeni format ID oznake pacijenta. Kod tradicionalnih sistema, svaka aplikacija bi trebalo da bude promenjena, testirana i onda vraćena nazad na posao. I ako nema grešaka neki sistem bi mogao da bude zaboravljen, tako da dolazimo u situaciju da je neki sistem izvan upotrebe.

Rešenje za ove probleme je da napravimo komponentu Pacijent. Napravite samo jednu komponentu Pacijent i fizički je smestite na centralnu lokaciju. Svaki sistem koji radi sa pacijentima jednostavno pristupa ovoj komponenti. Aplikacije kreiraju objekte na osnovu komponente i svaki objekat ima identičnu strukturu. Ako format ID oznake pacijenta treba da se promeni, Vi možete (u idealnom svetu) da promenite komponentu i da radite dalje. Aplikacije koje koriste ovu komponentu će automatski da koriste ovu novu komponentu kada sledeći put treba da se kreira objekat na osnovu te komponente.

Ako je sve ovo novo za Vas, ne brinite. Mi ćemo detaljnije da obradimo ovakve aplikacije u n nivoa u trinaestom i četrnaestom poglavlju. Za sada treba da shvatite jedno što je važno: rad sa n nivoa Vam omogućava da kreirate komponente koje ispunjavaju naša poslovna pravila, dok aplikacija samo poziva te komponente. Promene ovih pravila mogu da se rade nezavisno od nivoa za prezentaciju, što omogućava našim aplikacijama da lakše i brže upravljaju promenama koje se u poslu dešavaju.

Aplikacije u N nivoa. To zvuči dobro i može da bude dobro i to kako za tradicionalne Windows aplikacije (Visual Basic, Visual C++ i sl.) tako i za web aplikacije. IIS omogućava našim ASP stranama da pozivaju COM komponente koje implementiraju naša poslovna pravila. Nema razloga zašto komponenta koja se odnosi na pacijenta i koja je malopre pomenuta, ne bi mogla da se korsiti i sa prednje strane Weba. Postoji puno razloga za kreiranje web aplikacija u više nivoa. Mnoge od njih mi ćemo da pomenemo u toku ove knjige. Opisaćemo kako upotreba ovih komponenti može Vašu aplikaciju da napravi prilagodljivom, da dovede do njenog boljeg rada i da omogući podršku za različite transakcije sa bazom podataka.

Kod web aplikacija u n nivoa, skoro sve je pod kontrolom IT odeljenja. ASP strane, komponente i baza podataka, sve se to nalazi u računarskom odeljenju pod kontrolom IT odeljenja. Jedini deo aplikacije koji se nalazi na klijentovoj mašini je njegov pretraživač. Ovo IT odeljenju daje istu kontrolu, koju ima i kod monolitnih sistema. To dalje znači da se stvari kao što je backup, distribucija, sigurnost i održavanje nalaze pod kontrolom odeljenja koje (nadajmo se) ima dovoljno znanja i resursa da ostvari da sve teče bez problema.

Zaključak

World Wide Web je popularna platforma za razvoj web aplikacija pošto Web nudi niz značajniih prednosti. To su mogućnost pristupa skoro celom svetu, nema potrebe za distribucijom i moguća kontrola od strane IT odeljenja. Da bismo mogli da napravimo web aplikaciju moramo da imamo na raspolaganju neki metod za dinamičko kreiranje strana. Microsoftov Internet Information Servis nam pruža tu mogućnost sa svojim Aktivnim server stranama (ASP).

Kreiranje velikiih, robustnih web aplikacija je postalo moguće zahvaljujući skirptima na strani servera. Mi ćemo da koristimo ASP za pristup COM komponentama, koje mogu da se koriste više puta. Koristićemo i SQL Server. Ispitaćemo kako da naše aplikacije učinimo prilagodljivim i kako da one bolje rade uz pomoć COM+ tehnologije. Ako se pitate zašto još uvek nismo pominjali baze podataka, ne brinite. Pre svega moramo da napravimo neke ASP strane da bismo bili sigurni da znamo kako one rade, ali će naša web aplikacija na kraju sigurno da bude povezana sa bazom podataka.