POGLAVLJE 1
Baze podataka na Webu

Razvoj Weba tokom protekle decenije doveo je do odgovarajućeg razvoja usluga dostupnih na Webu. Mnoge od tih usluga su Web lokacije koje funkcionišu pomoću podataka smeštenih u bazama podataka. Primeri baza podataka na Webu obuhvataju usluge novinskih agencija koje pružaju pristup do velike količine uskladištenih podataka, zatim programe za elektronsko poslovanje (engl. *e-commerce*) kao što su prodavnice na Webu i proizvode za podršku direktnoj komunikaciji između dva poslovna subjekta (engl. *business-to-business,* B2B).

Aplikacije koje rade s bazama podataka postoje preko 30 godina i mnoge su napravljene primenom mrežnih tehnologija poznatih davno pre nego što je Web uopšte postojao. Takozvani *point-of-service* sistemi koji omogućavaju podizanje novca iz banke očigledan su primer prvobitnih umreženih aplikacija za rad s bazama podataka. U ekspoziturama su instalirani terminali banke, a centralnoj bazi podataka pristupa se preko mreže za široko područje. Te prvobitne aplikacije bile su pogodne samo za organizacije koje su mogle da plate specijalizovanu terminalsku opremu i, u nekim slučajevima, da izgrade i koriste sopstvenu mrežnu infrastrukturu.

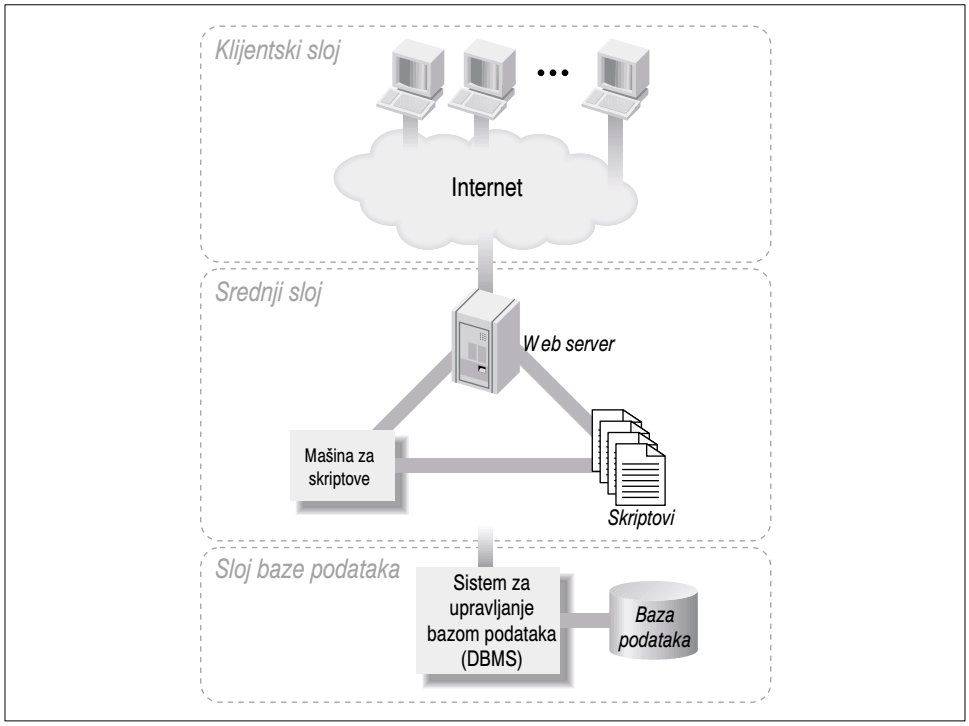
Web pruža jeftin, lako dostupan način umrežavanja. Postoji izuzetno veliki broj njegovih korisnika sa standardizovanim Web čitačima koji rade na raznim modelima običnih računara. Projektantima je besplatno dostupan softver za Web servere koji može da ispuni i zahteve za izvršavanjem programa i za učitavanjem dokumenata. Više programskih jezika za pisanje skriptova prilagođeno je ili projektovano za razvoj programa koji koriste Web servere i Web protokole.

Tema ove knjige je povezivanje baza podataka i Weba. Većina aplikacija koje rade s bazama podataka povezuje se na Web preko tri sloja aplikacione logike. U osnovi svega nalazi se sistem za upravljanje bazom podataka – DBMS (engl. *database management system*) i sama baza podataka. Na vrhu se nalazi korisnikov (klijentski) Web čitač koji služi kao interfejs aplikacije. Između ova dva sloja nalazi se najveći deo logike aplikacije, koja se obično realizuje u obliku skriptova koji komuniciraju sa DBMS-om na strani Web servera i koji mogu da dekodiraju i generišu HTML kôd potreban za prezentaciju podataka u korisnikovom čitaču Weba.

Započećemo razmatranjem arhitekture troslojnog modela koji se koristi u velikom broju Web aplikacija. Nakon toga sledi upoznavanje sa prirodom Weba i njegovim osnovnim protokolima, a zatim ćemo detaljno razmatrati sva tri sloja i njihove komponente. Na kraju ovog poglavlja upoznaćemo se i sa aplikacijom koju ćemo detaljno proučavati – Hugh and Dave's Online Wines. Ova aplikacija se često pominje u knjizi i služi kao ilustrativan primer korišćenja baze podataka na Webu.

Troslojna arhitektura

Ova knjiga opisuje Web aplikacije napravljene po modelu *troslojne arhitekture* (slika 1-1). Na najnižem nivou aplikacije nalazi se *sloj baze podataka*. On se sastoji od *sistema za upravljanje bazom podataka*, a njegova uloga je upravljanje bazom čije podatke korisnici unose, brišu, menjaju ili pretražuju. Iznad sloja baze podataka nalazi se složeni *srednji sloj* koji sadrži najveći deo logike aplikacije i prenosi podatke između preostala dva sloja. Na vrhu se nalazi *klijentski sloj*, obično Web čitač koji komunicira sa aplikacijom.



Slika 1-1. Arhitektura troslojnog modela Web aplikacije koja radi s bazom podataka.

Formalan način opisivanja većine Web aplikacija kao troslojne arhitekture skriva činjenicu da takve aplikacije moraju da povežu različite protokole i softver. Najveći deo

ove knjige bavi se razmatranjem srednjeg sloja i aplikacione logike koja povezuje suštinski različite slojeve baze podataka i klijenta.

Kada koristimo reč Web prvenstveno mislimo na tri glavna, zasebna standarda i na njima zasnovane alate: jezik za označavanje hiperteksta (engl. *Hypertext Markup Language,* HTML), protokol za prenos hiperteksta (engl. *Hypertext Transfer Protocol,* HTTP) i paket mrežnih protokola TCP/IP. HTML je pogodan za strukturisanje i prikazivanje podataka u Web čitačima. TCP/IP je efikasan mrežni protokol koji prenosi podatke između aplikacija na Internetu i ima vrlo malo značaja za projektante Web aplikacija. Problem u izradi dinamičkih Web aplikacija predstavlja komuniciranje pomoću HTTP protokola s klasičnim bazama podataka koje rade na Webu. Tu je neophodna složena aplikaciona logika.

Protokol za prenos hiperteksta (HTTP)

Troslojna arhitektura obezbeđuje konceptualni temelj za izradu Web aplikacija koje pristupaju bazama podataka. Sam Web obezbeđuje protokole i mrežu koji povezuju klijentski i srednji sloj aplikacije, tj. obezbeđuje vezu između čitača Weba i Web servera. HTTP je komponenta koja povezuje sve slojeve u troslojnoj arhitekturi. Da biste razumeli ovu knjigu, nije potrebno da detaljno poznajete HTTP, ali je vrlo važno da razumete kakve probleme HTTP protokol može da izazove u Web aplikacijama koje rade s bazama podataka. Web čitači šalju zahteve za resursima pomoću HTTP protokola, dok Web serveri šalju odgovore na te zahteve. (Duži uvod u osnovne Web protokole, uključujući i više primera HTTP zahteva i odgovora, pronaći ćete u dodatku B.)

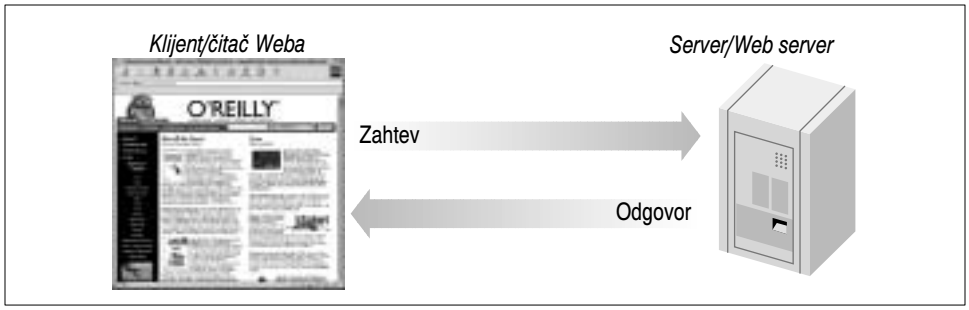
HTTP omogućava prosleđivanje i deljenje resursa na Webu. Posmatrajući iz perspektive mreže, HTTP predstavlja *protokol aplikacionog sloja* koji se nalazi neposredno iznad paketa TCP/IP mrežnih protokola. Većina Web servera i Web čitača komunicira pomoću tekuće verzije protokola, HTTP/1.1. Neki čitači i serveri koriste prethodnu verziju, HTTP/1.0, ali većina HTTP/1.1 softvera kompatibilna je s prethodnom verzijom HTTP/1.0.

Komunikacije pomoću HTTP-a dominiraju mrežnim saobraćajem na Internetu. Tokom 1997. godine, pomoću HTTP-a se odvijalo oko 75% celokupnog saobraćaja.* Smatra se da je ovaj procenat sada još veći s obzirom na porast broja i popularnosti aplikacija zasnovanih na HTTP-u (npr. usluge besplatne e-pošte).

Osnove HTTP-a

HTTP je konceptualno jednostavan: klijentski čitač Weba šalje Web serveru *zahtev* za određenim resursom, a Web server potom šalje čitaču *odgovor*. HTTP odgovor prenosi zahtevani resurs – HTML dokument, sliku ili izlazne podatke nekog programa – natrag do čitača Weba kao „koristan tovar". Jednostavan model zahtev–odgovor prikazan je na slici 1-2.

* Prema K.Thompsonu, G.J. Milleru i R. Wilderu „Wide-area internet traffic patterns and characteristics", *IEEE Network*, 11(6):10-23, novembar/decembar 1997.



Slika 1-2. Čitač Weba šalje zahtev, a Web server šalje odgovor u kome je sadržan zahtevani resurs.

HTTP zahtev se sastoji od tekstualnog opisa resursa i dodatnih informacija koje se zovu zaglavlja (engl. *headers*). Pogledajmo sledeći primer HTTP zahteva:

```
GET /index.html HTTP/1.0
From: hugh@computer.org (Hugh Williams)
User-agent: Hugh-fake-browser/version-1.0
Accept: text/plain, text/html
```

U ovom primeru se metodom `GET` zahteva HTML stranica `index.html` pomoću HTTP/1.0 verzije protokola. Tri dodatna reda zaglavlja identifikuju korisnika, čitač Weba, i definišu tipove podataka koje čitač može da prihvati. Zahtev obično šalje čitač Weba i on može da sadrži i druga zaglavlja; prethodni primer je napravljen ručno, upisivanjem zahteva u softver za Telnet.

HTTP odgovor sadrži kôd odgovora i poruku, dodatna zaglavlja, a najčešće i zahtevani resurs. Sledi primer odgovora na zahtev za Web dokumentom `index.html`:

```
HTTP/1.0 200 OK
Date: Sat, 21 Jul 2002 03:44:25 GMT
Server: Apache/1.3.20
Content-type: text/html
Content-lenght: 88
Last-modified: Fri, 1 Feb 2002 03:40:03 GMT

<html><head>
<title>Test stranica</title></head>
<body>
<h1>Ovo radi!</h1>
</body></html>
```

U prvom redu odgovora server obaveštava da prihvata upotrebu HTTP/1.0 protokola i potvrđuje da je uspešno primio zahtev ispisivanjem koda odgovora 200 i poruke OK; drugi čest odgovor je `404 Not Found` (zahtevani resurs nije nađen). U pet redova dodatnih zaglavlja ovog primera prikazani su tekući datum i vreme, softver Web servera, tip poslatih podataka, dužina odgovora (u bajtovima), kao i podaci kada je zahtevani resurs poslednji put izmenjen. Nakon praznog reda sledi sam resurs. U ovom primeru resurs je zahtevani HTML dokument, `index.html`.

Čuvanje stanja

Klasične aplikacije koje rade s bazama podataka *čuvaju stanje*: korisnici se prvo prijavljuju, zatim izvršavaju odgovarajuće transakcije i na kraju, kad obave posao, odjavljuju se. Na primer, aplikacija za banku zahteva da se operater na šalteru prvo prijavi, zatim da koristi aplikaciju pomoću niza menija dok opslužuje zahteve tekućeg komitenta, da bi se na kraju dana odjavio. Aplikacija za banku ima više stanja: kada se blagajnik prijavi, on može da koristi aplikaciju na strukturiran način, pomoću menija. Kada se blagajnik odjavi, on ne može više da koristi aplikaciju.

HTTP *ne čuva stanje*. Kad kažemo da se stanje ne čuva, to znači da je svaka interakcija između čitača Weba i Web servera nezavisna od bilo koje druge interakcije. Svaki HTTP zahtev čitača Weba sadrži identične informacije u zaglavlju kao što su identifikacioni podaci korisnika, tipovi stranica koje čitač može da prihvati i instrukcije za formatiranje odgovora. Činjenica da se stanje ne čuva ima i svojih prednosti: najvažnija je ušteda resursa pošto nema potrebe da se na Web serveru održavaju informacije pomoću kojih bi se pratile aktivnosti korisnika, kao i fleksibilnost koja korisnicima omogućava da se kreću između nepovezanih stranica ili resursa.

Pošto HTTP ne čuva tekuće stanje, vrlo je teško projektovati Web aplikacije koje bi čuvale tekuće stanje. Neophodan je način za održavanje stanja u HTTP-u tako da podaci protiču i može da se nametne određena struktura pri upotrebi aplikacije. Često rešenje je razmena žetona (engl. *token*) između čitača Weba i Web servera pomoću kojih se na jedinstven način identifikuje korisnik i njegova *sesija*. Svaki put kada čitač uputi zahtev za određenim resursom, on šalje žeton, i svaki put kada se Web server odazove, on vraća žeton čitaču Weba. Softver u srednjem sloju pomoću žetona obnavlja podatke o korisniku od njegovog prethodnog zahteva (npr. kom je meniju u aplikaciji poslednji put pristupljeno). Razmena žetona omogućava da se aplikaciji dodaju strukture za koje je neophodno čuvanje stanja, kao što su meniji, koraci i praćenje procesa rada.

Lagani klijenti

Pošto znamo da Web aplikacije troslojne arhitekture po prirodi stvari nisu pogodne za HTTP, možete se zapitati čemu onda uopšte upotreba takvog modela? Odgovor prvenstveno treba tražiti u prednostima koje pružaju *lagani klijenti*. Čitači Weba su vrlo lagani klijenti: vrlo mali deo logike aplikacije je uključen u klijentski sloj. Čitač samo šalje HTTP zahteve za resursima i prikazuje odgovore, koji mahom sadrže HTML dokumente.

Troslojni model znači da ne morate da pravite, instalirate ili podešavate klijentski sloj. Svaki posetilac koji ima čitač Weba može da koristi Web aplikaciju koja čita podatke iz baze, obično bez potrebe da instalira dodatni softver, da koristi specifičan operativni sistem ili određenu hardversku platformu. To znači da aplikacija može da opsluži bilo koji broj različitih, geografski rasejanih korisnika. Ova prednost je toliko značajna da je ova knjiga u potpunosti usredsređena na troslojna rešenja i arhitekturu sa laganim klijentom – čitačem Weba.

Međutim, šta je alternativa laganom klijentu? Namenski napisan Java aplet primer je težeg klijenta koji se ipak može uklopiti u troslojni model: korisnik preuzima aplet i izvršava više aplikacione logike na svojoj platformi nego u slučaju laganog klijenta. Aplet i dalje komunicira sa srednjim slojem koji, pak predstavlja interfejs ka sloju baze podataka. U ovom slučaju prednost je prilagođenost određenoj nameni: umesto da se koristi generički čitač, koristi se namensko rešenje koje eliminiše mnoge probleme oko čuvanja stanja, bezbednosti i pomanjkanja fleksibilnosti Weba. Aplet uopšte ne mora da koristi HTTP protokol za komuniciranje sa logikom aplikacije u srednjem sloju.

Težak klijent je deo klasičnog dvoslojnog rešenja, koje je takođe poznato pod nazivom *klijent/server arhitektura*. Većina klasičnih aplikacija koje pristupaju bazama podataka (poput onih u bankama) ima samo dva sloja. Klijentski sloj sadrži najveći deo logike aplikacije, dok je serverski sloj, zapravo, sam DBMS. Prednost ovakve arhitekture je to što mogu da se projektuju namenska rešenja koja potpuno zadovoljavaju specifične aplikacione zahteve, bez ikakvih kompromisa. Mane su nedostatak fleksibilnosti po pitanju hardvera i operativnog sistema, kao i potreba da se svakom korisniku instalira softver.

Klijentski sloj

Klijentski sloj u modelu troslojne arhitekture obično je čitač Weba. Čitač Weba obrađuje i prikazuje HTML resurse, šalje HTTP zahteve za resursima i obrađuje HTTP odgovore. Kao što je već pomenuto, upotreba čitača Weba kao tankog klijentskog sloja pruža značajne prednosti, među kojima su jednostavan razvoj i podrška za širok opseg različitih platformi.

Na raspolaganju je veliki broj čitača Weba i svaki od njih ima drugačije osobine. Dva najpopularnija čitača zasnovana na okruženju sa prozorima jesu Netscape i Internet Explorer. Nećemo opisivati sve mogućnosti čitača Weba, pomenućemo samo njihove zajedničke odlike:

- Svi čitači Weba su HTTP klijenti koji šalju zahteve i prikazuju odgovore dobijene od Web servera (obično u nekom grafičkom okruženju).
- Svi čitači tumače sadržaj stranica sa HTML kodom kada prikazuju stranicu, tj. oni korisnicima prikazuju zaglavlja, slike, hiperveze itd.
- Neki čitači prikazuju slike, reprodukuju filmove i zvuk i vizuelizuju druge tipove objekata.
- Mnogi čitači mogu da izvršavaju JavaScript kôd ugrađen u HTML stranice. JavaScript se koristi za, na primer, proveru ispravnosti podataka u HTML obrascu `<form>` ili promenu izgleda i sadržaja stranice u zavisnosti od korisnikovih akcija.
- Neki čitači Weba mogu da pokreću komponente razvijene pomoću programskih jezika Java i ActiveX. Te komponente često daju dodatne animacije, alatke koje nije moguće izraditi pomoću HTML-a ili, druge, još složenije mogućnosti.
- Nekoliko čitača može da primenjuje kaskadne liste stilova (engl. *Cascading Style Sheets*, CSS) na HTML stranice kako bi upravljali prikazivanjem HTML elemenata.

Postoje manje (a nekad i ne baš tako male) razlike u načinima na koje pojedini čitači Weba prikazuju HTML stranice. Lynx je, na primer, čitač koji prikazuje isključivo tekst a ne prikazuje slike, niti izvršava JavaScript kôd. MultiWeb je čitač koji tekst na stranici reprodukuje kao zvuk (izgovorene reči), što omogućava pristup Webu osobama sa oštećenim vidom. Postoje mnoge manje, ali neprijatne razlike u podršci za CSS i mogućnosti najnovijeg HTML standarda, HTML 4.

Čitači Weba su najočigledniji primer *korisničkog agenta*, softverskog klijenta koji zahteva resurse od Web servera. U druge korisničke agente spadaju *Web pauci* (automatizovani softver koji krstari Webom i pronalazi Web stranice) i *posredničke ostave (keš)*, softverski sistemi koji učitavaju Web stranice i lokalno ih skladište kako bi one bile na raspolaganju drugim korisničkim agentima.

Pošto ova knjiga nije priručnik za jezik HTML, mogućnosti HTML-a ćemo razmatrati kroz primere koji se pojavljuju. Lista resursa koji opisuju jezik HTML, način izrade Web stranica i standarde za njih, nalazi se u dodatku E. Sa JavaScript kodom za proveru ispravnosti unetih podataka na klijentskoj strani i s upotrebom čitača Weba upoznaćemo se u poglavlju 7.

Srednji sloj

U većini troslojnih sistema baza podataka na Webu, najveći deo logike aplikacije nalazi se u srednjem sloju. Klijentski sloj prikazuje podatke korisniku i prihvata podatke koje on unosi; sloj baze podataka skladišti i učitava podatke. Srednji sloj obavlja većinu preostalih zadataka pomoću kojih se objedinjuju ostali slojevi: upravlja strukturom i sadržajem podataka koji se prikazuju korisniku i obrađuje podatke dobijene od korisnika pretvarajući ih u upite za učitavanje ili upisivanje u bazu podataka. Takođe, ovaj sloj omogućava upravljanje stanjem uz upotrebu HTTP protokola. Logika aplikacije ugrađena u srednji sloj integriše Web sa sistemom za upravljanje bazom podataka.

U ovoj knjizi koristićemo aplikacioni model u kome su komponente srednjeg sloja Web server, programski jezik za pisanje Web skriptova i mašina za izvršavanje skriptova (engl. *engine*). Web server obrađuje HTTP zahteve i formuliše odgovore. U slučaju Web aplikacija, ti zahtevi su obično upućeni programima koji komuniciraju sa osnovnim sistemom za upravljanje bazom podataka. U ovoj knjizi kao Web server koristimo Apache HTTP server, proizvod organizacije Apache Software Foundation. To je Web server otvorenog koda i koristi se na više od 60% računara koji su stalno povezani na Internet.*

Za pisanje skriptova koji rade u srednjem sloju koristimo programski jezik PHP. Budući da je PHP nastao kao projekat otvorenog koda iza koga stoji Apache Software Foundation, ne iznenađuje što je to najpopularniji dodatni modul za Apache HTTP server, sa oko 40% Apache HTTP servera koji podržavaju PHP.** PHP je izuzetno

* Prema *Netcraftovom istraživanju Web servera, http://www.netcraft.com/survey/* (April 2001).

** Prema izveštaju o modulima za Apache, iz istraživanja Web servera koje je obavio Security Space, *http://www.securityspace.com/s_survey/data/index.html* (april 2001).

prikladan za Web aplikacije zahvaljujući alatkama za integraciju Weba i okruženja baze podataka. Izuzetna fleksibilnost skriptova ugrađenih u HTML stranice omogućava laku integraciju s klijentskim slojem. Podrška za integraciju sa slojem baze podataka takođe je odlična zahvaljujući više od 15 biblioteka koje omogućavaju povezivanje sa skoro svim poznatim sistemima za upravljanje bazama podataka.

Web serveri

Web serveri se često nazivaju i *HTTP serverima*. Izraz „HTTP server" je dobar sažetak funkcije koju obavljaju: njihov osnovni zadatak je da otkrivaju pristizanje HTTP zahteva iz mreže, primaju HTTP zahteve koje šalju korisnički agenti (obično čitači Weba), obrađuju te zahteve i vraćaju HTTP odgovore koji sadrže tražene resurse.

U suštini postoje dva tipa zahteva koji se šalju Web serveru: prvi, kada se zahteva slanje određene datoteke (obično statičke HTML Web stranice ili slike) i drugi, kada se zahteva pokretanje programa i slanje izlaznih podataka tog programa korisničkom agentu. Jednostavni zahtevi za datotekama podrobnije su objašnjeni u dodatku B.

Zahtevi za pokretanje Web skriptova koji pristupaju bazi podataka primeri su HTTP zahteva kada je neophodno da server pokrene određeni program. Pomoću softvera koji koristimo u ovoj knjizi upućivaćemo HTTP zahteve za resursima tipa PHP skriptova, nakon čega se pokreće PHP Zend mašina koja učitava i izvršava skript, a potom prihvata njegove izlazne podatke.

Apache HTTP server, verzija 1.3

Kao i većina korisnika Apache HTTP servera, i mi ga nazivamo *Apache*. Apache je Web server otvorenog koda. U vreme kada smo pisali ovu knjigu, tekuća verzija je bila 1.3.20.

Instaliranje i podešavanje Apache servera za većinu Web aplikacija koje rade s bazama podataka vrlo je jednostavno. Sažeto uputstvo za instaliranje operativnog sistema Linux nalazi se u dodatku A. Apache možete preuzeti sa *http://www.apache.org*; ostali resursi za Apache navedeni su u dodatku E.

Apache je brz i prilagodljiv. On može istovremeno da opsluži više zahteva korisničkih agenata i projektovan je za rad na operativnim sistemima koji podržavaju istovremeni rad više programa (engl. *multitasking*) kao što su Linux i 32-bitne verzije Microsoft Windowsa. Takođe, Apache spada u „lakše" programe, ima male zahteve po pojedinačnom procesu, efikasno se prilagođava promenama opterećenja i radi brzo čak i na skromnom hardveru.

Apache (barem konceptualno) nije komplikovan. Web server je u stvari skup nekoliko procesa gde jedan proces koordinira rad ostalih. Proces koji koordinira obično se pokreće s pravima *administratora* (engl. *superuser*) ili korisnika *root user* na računarima pod Unixom, i on sam ne obrađuje zahteve. Ostali procesi, koje obično na mnogo bezbedniji način pokreću korisnici s manjim pravima, obaveštavaju server za koordinisanje o svojoj (ne)zauzetosti opsluživanjem zahteva. Ako nema dovoljno servera da opsluži prispele zahteve, server za koordinisanje može pokrenuti nove servere; ako ih je previše slobodno, može isključiti prekobrojne servere da bi štedeo resurse.

Pomoću konfiguracione datoteke podešava se način na koji Apache prati stanje u mreži i opslužuje prispele zahteve. Administrator servera može da podešava ponašanje Apachea pomoću više od 150 naredaba koje se odnose na zahteve za resursima, vreme odziva, prilagođavanje različitim opterećenjima, bezbednost, način na koji se postupa sa HTTP zahtevima, način na koji se oni beleže, i na mnoge druge aspekte njegovog funkcionisanja. Bitno je pažljivo podesiti ove parametre zbog performansi, a više detalja o konfigurisanju Apachea možete pronaći u listi resursa koja se nalazi u dodatku E.

Apache HTTP server, verzija 2.0

Verzija 1.3 Apache servera ima neke nedostatke koji će biti ispravljeni u verziji 2.0. Verziju 2.0 možete preuzeti, ali u vreme kada smo pisali ovu knjigu, ona se nalazila u fazi beta testiranja. Zasad samo dvadesetak Web lokacija koristi ovu beta verziju.

Značajna poboljšanja Apachea 2.0 su:

- Korišćenje lakših procesa ili *niti* u sprezi s procesnim modelom prethodnih verzija. To će najverovatnije omogućiti značajno poboljšanje performansi prilikom pokretanja novih servera i umanjiti ukupnu količinu memorije koja je neophodna za već pokrenute servere.
- Bolja podrška, performanse i stabilnost na računarima koji rade pod Unixom.
- Dodatni moduli za filtriranje koji omogućavaju izmenu podataka dok ih Web server obrađuje.
- Podrška za IPv6, novu verziju IP protokola iz programskog paketa mrežnog protokola TCP/IP.

Upotreba PHP-a za Web skriptove

PHP se nametnuo kao komponenta mnogih dinamičkih Web aplikacija srednjeg i velikog obima. To ne znači da drugi jezici za skriptove nemaju izvanredne mogućnosti. Međutim, postoje mnogi razlozi zbog kojih je PHP dobar izbor, uključujući sledeće:

- PHP softver je otvorenog koda, što znači da je potpuno besplatan. Zbog tog razloga nastojanja programerske zajednice da ga unapredi i održava nisu ugrožena komercijalnim potrebama.
- U statičke HTML datoteke može se ugraditi jedan ili više PHP skriptova, što značajno pojednostavljuje integraciju sa klijentskim slojem. S druge strane, to može dovesti do mešanja skriptova s prezentacijom rezultata; ipak, primenom tehnika sa šablonima opisanim u poglavlju 13 može se rešiti većina navedenih problema.
- Postoji preko 15 biblioteka za osnovni, brz pristup sloju baze podataka.
- Brzo izvršavanje skriptova. S novim poboljšanjima Zend mehanizma za obradu skriptova, izvršavanje je brzo i sve komponente rade unutar glavnog memorijskog prostora PHP-a (za razliku od drugih okruženja za izvršavanje skriptova, u kojima su komponente zasebni moduli). Empirijski dokazi svedoče da PHP obavlja zadatke srednje složenosti brže od drugih poznatih okruženja za skriptove.

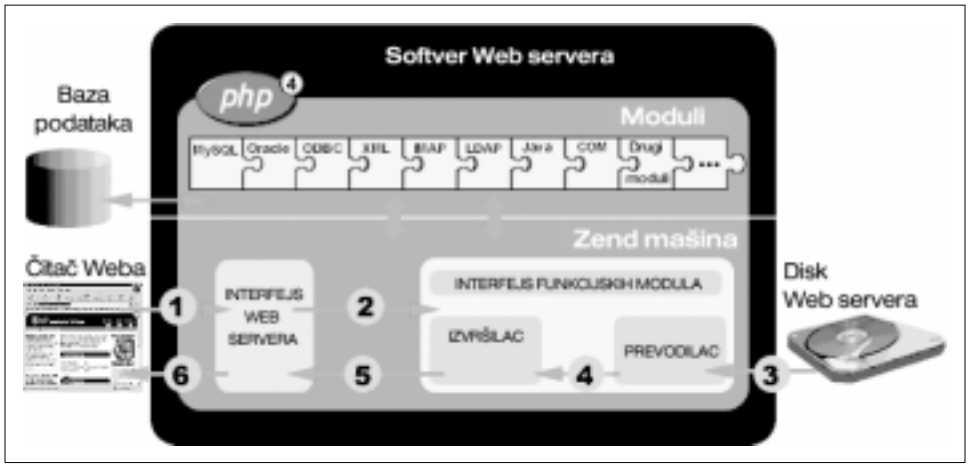
- Fleksibilnost po pitanju platformi i operativnih sistema. Apache radi na različitim platformama i na nekoliko operativnih sistema; PHP radi na svemu tome, ali i na mnogim drugim kada se integriše s drugim Web serverima.
- PHP je podesan za razvoj složenih sistema. To je potpun programski jezik sa preko 50 funkcijskih biblioteka.

Najnovija verzija PHP-a je verzija 4 (kroz najveći deo ove knjige nazivaćemo je PHP), a poslednje izdanje, u vreme kada smo pisali ovu knjigu, jeste PHP 4.0.6.

PHP4 koristi u potpunosti iznova napisanu mašinu za izvršavanje skriptova u odnosu na onu koja se koristi u PHP3. Značajna razlika je u tome što je promenjen model po kome mašina izvršava skriptove. Mašina korišćena u PHP3 ujedno je bila i *prevodilac*. Svaki red programskog koda skripta čita se, analizira i izvršava. Ako se iskaz u telu petlje izvršava 100 puta, red programskog koda se 100 puta prevodi kada se koristi PHP3. Ovaj model je spor za složenije skriptove ali brz za kraće skriptove.

PHP4 koristi drugačiji model za obradu skriptova i on je projektovan za složenije aplikacije. Skript se čita, analizira i prevodi u međuformat, a zatim se kôd međuformata izvršava pomoću *izvršioca* PHP4 Zend mašine za izvršavanje skriptova. To znači da se svaki red u skriptu iz izvornog oblika prevodi samo jedanput, čak i kad se izvršava stotinama puta. Osim toga, prevođenje omogućava optimizaciju segmenata koda. Rezultat toga je poboljšanje performansi PHP4 za sve sem za vrlo jednostavne skriptove.

Struktura okruženja za izvršavanje skriptova PHP4 prikazana je na slici 1-3 (slika je preuzeta od kompanije Zend Technologies Inc.). Kao što je prikazano, PHP4 je softverski modul Web servera. PHP softver je podeljen u dve komponente: funkcijske biblioteke (module) i Zend mašinu.



Slika 1-3. Struktura PHP4 okruženja za izvršavanje skriptova.

Kada korisnički agent uputi zahtev Web serveru za PHP skriptom, proces se odvija u sledećih šest koraka:

1. Web server prosleđuje zahtev do Web server interfejsa Zend mašine.
2. Interfejs Web servera poziva Zend mašinu i prosleđuje joj parametre.
3. Mašina preuzima PHP skript sa diska.
4. Prevodilac prevodi skript.
5. Izvršilac pokreće prevedeni kôd koji može da sadrži pozive funkcijskih modula. Izlazni podaci izvršioca prosleđuju se interfejsu Web servera.
6. Interfejs Web servera prosleđuje izlazne podatke Web serveru (koji, na kraju, prosleđuje izlazne podatke korisničkom agentu u obliku HTTP odgovora).

Način kojim se upravlja i na koji funkcioniše PHP mašina za izvršavanje skriptova zavisi od načina na koji je PHP modul uključen tokom instalacije Apache Web servera. U uputstvu koje se nalazi u dodatku A, biblioteka PHP modula je *statički* povezana sa binarnom izvršnom datotekom `httpd`. To znači da se PHP mašina za izvršavanje skriptova učitava svaki put kada se pokrene Apache, što kao krajnju posledicu ima brži rad PHP mašine. Nedostaci su to što Apache sa statičkom PHP bibliotekom okupira više memorije nego kada se modul učitava *dinamički*, kao i to što je postupak nadogradnje modula manje fleksibilan.

Odrednice u vezi sa Web resursima, knjigama i komercijalnim proizvodima za PHP razvoj navedene su u dodatku E.

Sloj baze podataka

Sloj baze podataka je osnova Web aplikacija koje rade s bazama podataka. Razumevanje sistemskih zahteva, odabir softvera za sloj baze podataka, projektovanje baza podataka i izrada sloja prvi su koraci uspešnog projektovanja aplikacija. U dodatku C razmatraćemo tehnike za modeliranje sistemskih zahteva, pretvaranje modela u bazu podataka i principe tehnologija baza podataka. U ovom odeljku usredsredićemo se na komponente sloja baze podataka i upoznati se sa softverom baza podataka upoređujući ga s drugim tehnikama skladištenja podataka. U poglavlju 3 detaljnije su opisani standardi i softver koji smo koristili.

U aplikaciji sa troslojnom arhitekturom, sloj baze podataka zadužen je za upravljanje podacima. Upravljanje podacima obično podrazumeva skladištenje i učitavanje podataka, kao i upravljanje postupkom ažuriranja, pri čemu više procesa iz srednjeg sloja može simultano, tj. *istovremeno* da pristupa, zatim, zaštitu podataka, očuvanje njihovog integriteta i obezbeđivanje usluga za podršku, poput izrade rezervnih kopija. U mnogim bazama podataka za Web sve te poslove obavlja sistem RDBMS i podaci smešteni u relacionoj bazi podataka.

Upravljanje relacionim podacima u trećem sloju zahteva složeni RDBMS softver. Na sreću, većina RDBMS-ova je projektovana tako da se složenost softvera ne primećuje. Da biste efikasno koristili DBMS, potrebno je da znate da projektujete bazu podataka i da formulišete komande i upite za DBMS. U većini DBMS-ova jezik za upite je SQL. Većini korisnika nije potrebno da razumeju unutrašnju arhitekturu DBMS-a.

U ovoj knjizi koristićemo MySQL RDBMS za upravljanje podacima. Poput rasprava o tome koji programski jezik treba koristiti za skriptove u srednjem sloju, raspravlja se i o tome koji je DBMS najprikladniji za određenu aplikaciju. MySQL ima potpuno zasluženu dobru reputaciju o brzini i posebno je dobro projektovan za aplikacije gde se podaci mnogo češće učitavaju nego što se ažuriraju i za slučajeve gde su manja, jednostavnija ažuriranja uobičajena vrsta izmena. To su tipične karakteristike većine aplikacija koje rade s bazama podataka. Osim toga, poput PHP-a i Apachea, MySQL spada u softver otvorenog koda. Ipak, MySQL ima i izvesne mane koje ćemo razmatrati kasnije u ovom odeljku.

Postoje i druga, nerelaciona DBMS softverska rešenja za skladištenje podataka u sloju baze podataka. Tu spadaju mašine za pretraživanje, sistemi za upravljanje dokumentima i jednostavniji mrežni servisi, poput softvera za elektronsku poštu. U ovoj knjizi razmatramo prvenstveno tehnologije relacionih baza podataka u sloju baze podataka.

Sistemi za upravljanje bazama podataka

Sistem za upravljanje bazama podataka skladišti podatke, pretražuje ih i upravlja njima.

Baza podataka je skup povezanih podataka. Uskladišteni podaci mogu se sastojati od svega nekoliko ulaznih podataka ili *redova* koji čine jednostavan adresar sa imenima, adresama i telefonskim brojevima. Nasuprot tome, baza podataka može da sadrži i milione zapisa koji opisuju katalog, kupovine, porudžbine i platni spisak velike kompanije. Baza podataka koja se nalazi u osnovi slučaja koji proučavamo, Hugh and Dave's Online Wines, primer je baze srednje veličine koja se nalazi između ta dva ekstremna slučaja.

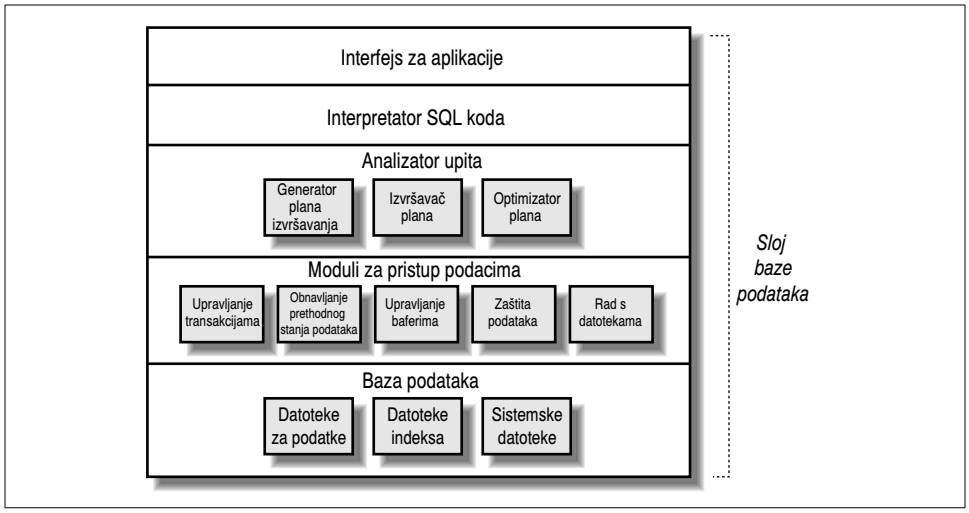
DBMS je skup komponenata za definisanje, izradu i korišćenje baze podataka. Kada pomenemo sistem za upravljanje bazom podataka (DBMS), uglavnom mislimo na relacioni DBMS tj. RDBMS. Relacione baze podataka skladište veze između podataka i upravljaju njima. Na primer, kupci poručuju proizvode, porudžbine kupaca sadrže artikle iz proizvodne linije ili se vinarije nalaze u određenom vinarskom kraju.

Slika 1-4 prikazuje pojednostavljenu strukturu tipičnog DBMS-a.

DBMS se sastoji od nekoliko komponenata:

Interfejs za aplikacije
: Biblioteke za komunikaciju sa DBMS-om. Većina DBMS-ova ima jednostavan prevodilac u obliku komandne linije koji često koristi te biblioteke za prenošenje zahteva unetih sa tastature do DBMS-a i za prikazivanje odgovora. U Web aplikacijama zasnovanim na korišćenju baza podataka, prevodilac u obliku komandne linije obično je zamenjen funkcijskom bibliotekom koja je deo jezika za skriptove srednjeg sloja.

Interpretator SQL koda
: Sintaksni analizator proverava sintaksu prosleđenih upita i prevodi ih u interni prikaz.



Slika 1-4. Struktura tipičnog DBMS-a.

Analizator upita
: Generiše razne planove izvršavanja upita na osnovu statistike baze podataka i njenih svojstava, odabira jedan od planova i prevodi ga u akcije koje izvršava na niskom nivou.

Pristup podacima
: U module koji upravljaju pristupom podacima uskladištenim na disku spadaju modul za upravljanje transakcijama, modul za upravljanje postupkom obnavljanja stanja, modul za upravljanje baferom glavne memorije, modul za zaštitu podataka i modul za upravljanje pristupom datotekama.

Baza podataka
: To su, zapravo, sami podaci fizički smešteni u datoteke. Podaci pored toga obuhvataju i *indeksne* datoteke za brz pristup, kao i statističke podatke o bazi podataka i sistemu, koji se prvenstveno koriste za generisanje i optimizovanje planova za izvršavanje upita.

Projektantima baza podataka za Web bitne komponente su baza podataka i aplikacioni interfejs. Obično nije potrebno poznavanje i konfigurisanje ostalih komponenata DBMS-a, sem kada se radi o obimnoj aplikaciji.

Zašto treba koristiti DBMS?

Pitanje koje se često postavlja glasi: zašto koristiti složeni DBMS za upravljanje podacima? Postoji nekoliko razloga. Oni se mogu objasniti poređenjem baze podataka sa programima za unakrsne proračune, jednostavnom tekstualnom datotekom ili namenski izrađenim metodom skladištenja podataka. Kasnije u ovom odeljku opisano je nekoliko situacija i primera kada treba, a kada ne treba koristiti DBMS.

Uzmimo program za unakrsne proračune kao primer. Radni listovi programa za unakrsne proračune obično su dizajnirani za određene aplikacije. Ako dva korisnika skladište imena i adrese, vrlo je verovatno da će organizovati podatke na različit način (zavisno od potreba) i razviti metode za premeštanje i sumiranje podataka. U ovakvoj situaciji program i podaci nisu nezavisni: premeštanje kolone može značiti i prepisivanje preko makroa ili formule, a razmena podataka između aplikacija s kojima rade ta dva korisnika može biti složena. Nasuprot tome, DBMS i baza podataka obezbeđuju nezavisnost datoteke i programa, gde metoda skladištenja, redosled uskladištenih podataka i način upravljanja podacima na disku ne zavise od softvera koji im pristupa.

Upravljanje složenim vezama je prilično teško u proračunskim tabelama ili tekstualnim datotekama. Razmotrimo, na primer, našu prodavnicu vina: ako želimo da skladištimo podatke o kupcima, možemo odvojiti nekoliko kolona u tabeli da bismo sačuvali kućnu adresu svakog kupca. Ukoliko hoćemo da dodamo adresu na poslu i poštansku adresu, trebaće nam još kolona i složena obrada da bismo, na primer, poslali pismo kupcima. Ako želimo da uskladištimo podatke o kupovini koju su obavili naši kupci, proračunska tabela postaće šira i pojaviće se problemi. Teško je, na primer, odrediti maksimalan broj kolona potrebnih za skladištenje porudžbina i projektovati metod za njihovu obradu i generisanje izveštaja.

Proračunske tabele i tekstualne datoteke ne funkcionišu dobro kada su uskladišteni podaci spojeni ili povezani. Nasuprot tome, DBMS-ovi su projektovani za upravljanje složenim *relacionim* podacima. DBMS-ovi takođe predstavljaju kompletno rešenje: ako koristite DBMS, neće biti potrebe za rešenjima sa namenski projektovanim proračunskim tabelama ili datotekama. Metode pristupa podacima (najčešće pomoću jezika za upite SQL) ne zavise od vrste fizičkog skladištenja i načina obrade podataka.

DBMS obično dozvoljava višekorisnički rad. DBMS-ovi srednjeg i velikog obima mogu sistemski kontrolisati upisivanje podataka više različitih korisnika. Nasuprot tome, samo jedan korisnik može da otvori proračunsku tabelu i da u nju upisuje podatke. Ako neki drugi korisnik otvori tu istu tabelu, neće moći da vidi izmene koje u to vreme unosi prvi korisnik. U najboljem slučaju, proračunske tabele ili tekstualne datoteke s deljenim pristupom dozvoljavaju vrlo ograničen istovremeni pristup.

Dodatna prednost DBMS-a jeste njegova brzina. Nije u potpunosti tačna tvrdnja da baza podataka omogućava mnogo brže pretraživanje podataka u odnosu na programe za unakrsne proračune ili namenski sistem datoteka. U mnogim slučajevima, pretraživanje u proračunskoj tabeli ili datoteci posebne namene može biti savršeno prihvatljivo ili čak brže ako su one pažljivo projektovane a količina podataka mala. Međutim, za upravljanje velikom količinom povezanih podataka, fundamentalna struktura DBMS-a za pretragu obezbeđuje brzo pretraživanje, a ako su potrebe za podacima kompleksne, DBMS može da optimizuje metodu njihovog pronalaženja.

Postoje i druge prednosti DBMS-a, uključujući i zaštitu podataka i prava pristupa u zavisnosti od korisnika, softver za administriranje, i podršku za restauraciju podataka. Praktična dobit je u skraćenom vremenu razvoja aplikacija: sistem je već izgrađen, potrebni su samo podaci i upiti za pristup podacima.

Primeri kada treba koristiti DBMS

DBMS treba koristiti za upravljanje podacima u sledećim slučajevima:

- Ako postoji više korisnika koji moraju da pristupaju podacima u isto vreme.
- Ako postoji barem osrednja količina podataka. Na primer, mi ćemo imati potrebu da održavamo podatke o nekoliko stotina kupaca.
- Ako postoje veze između uskladištenih stavki. Na primer, svakom kupcu može pripadati ma koji broj porudžbina.
- Ako postoji više vrsta zapisa podataka. Na primer, u prodavnici na Internetu mogu postojati podaci o kupcima, porudžbinama, popisu robe i drugi podaci.
- Ako za podatke postoje ograničenja koja se moraju striktno poštovati, poput dužine polja, tipova polja, jedinstvenih brojeva kupaca itd.
- Ako postoji potreba da se novi ili zbirni podaci izvode iz osnovnih, srodnih podataka; to znači da podaci moraju da se filtriraju da bi se izvodili izveštaji i rezultati.
- Ako postoji velika količina podataka koja se mora brzo pretraživati.
- Ako je važna bezbednost, tj. ako postoji potreba da se zadaju pravila ko može da pristupa podacima.
- Ako je dodavanje, brisanje ili izmena podataka složen postupak.

Primeri kada ne treba koristiti DBMS

Postoje situacije kada relacioni DBMS nije potreban ili je nepodesan. Evo primera:

- Postoji samo jedan tip stavki podataka i podaci se ne pretražuju. Na primer, ako se evidentira kada se korisnik prijavi na sistem i odjavi sa njega; dodavanje zapisa na kraj jednostavne tekstualne datoteke sasvim je dovoljno.
- Upravljanje podacima je jednostavno. U tom slučaju, podaci se mogu ubaciti u Web skript iz srednjeg sloja, umesto da se sistem opterećuje pristupanjem bazi podataka svaki put kada su podaci potrebni.
- Podaci zahtevaju složenu analizu. Programski paket za unakrsne proračune ili statistički softver mogu biti daleko pogodniji za analizu.

MySQL DBMS

MySQL je DBMS srednjeg obima, sa većinom odlika koje imaju sistemi velikog obima i mogućnošću da upravlja velikom količinom podataka. Projektovan je tako da je savršeno pogodan za upravljanje bazama podataka koje se obično koriste u dinamičkim Web aplikacijama.

Razlika između MySQL-a i nekih drugih sistema jeste u tome što MySQL ne podržava pojedina filtriranja i pruža ograničene mogućnosti istovremenog upravljanja podacima u višekorisničkom okruženju. To znači da desetine procesa iz srednjeg sloja mogu da

pristupaju bazi podataka u isto vreme, ali ne i stotine procesa. Nisu podržane dve tehnike izrade upita (posebno *ugnežđeni upiti* i *prikazi*), ali podrška se planira u bliskoj budućnosti, u verziji 4 MySQL-a. Postoje i druga, mnogo manja ograničenja koja su obično nebitna za razvoj Web aplikacije.

Ograničenja MySQL-a obično vrlo malo utiču na razvoj Web aplikacija. Ipak, za sisteme s velikim protokom podataka, velikim brojem istovremenih korisnika ili aplikacijama koje često ažuriraju bazu podataka, najbolje je razmotriti druge sisteme. Naš drugi izbor bio bi PostgreSQL, koji je nešto sporiji, ali podržava istovremeni rad više korisnika. Više informacija o PosgreSQL-u možete pronaći na *http://www.postgresql.org*.

U vreme kada smo pisali ovu knjigu, tekuća verzija MySQL-a bila je 3.23, a aktuelno izdanje 3.23.38. MySQL resursi navedeni su u dodatku E.

SQL

SQL je standardan jezik za interakciju s relacionim bazama podataka. Skoro svi sistemi relacionih baza podataka, uključujući i MySQL, podržavaju SQL kao alatku za izradu, zaštitu i pretraživanje baze podataka, kao i za upravljanje njome. SQL je mnogo više od običnog jezika za upite; to je u potpunosti usavršena alatka za sve aspekte održavanja baze podataka.

Istorijat

SQL je imao komplikovan život. Nastao je ranih sedamdesetih godina u IBM-ovoj istraživačkoj laboratoriji u San Hoseu, gde je bio poznat pod nazivom *Sequel*; neki korisnici ga i dalje tako zovu, iako je ispravnije koristiti akronim od tri slova, SQL. Nakon skoro 16 godina razvoja i neusaglašenih realizacija, organizacije za standarde ANSI i ISO ustanovile su 1986. godine SQL standard. Godinu dana kasnije IBM je ustanovio drugačiji standard!

Od sredine osamdesetih, ANSI i ISO su ustanovili tri naredna standarda. Prvi, SQL--89, najrasprostranjeniji je i u potpunosti implementiran SQL standard u većini poznatih baza podataka. Mnogi sistemi implementiraju samo neke mogućnosti narednih izdanja, SQL-2 ili SQL-92, a skoro nijedan sistem nema realizovane mogućnosti najnovije verzije standarda, SQL-99 ili SQL-3.

Usredsredićemo se na mogućnosti na kojima je zasnovan MySQL DBMS. MySQL podržava osnovni nivo standarda SQL-92.

SQL komponente

SQL ima četiri glavna dela, od kojih ćemo dva – jezik za definisanje podataka (engl. *Data Definition Language*, DDL) i jezik za manipulisanje podacima (engl. *Data Manipulation Language*, DML) – detaljno razmatrati u poglavlju 3.

Četiri glavne komponente SQL-a su:

Jezik za definisanje podataka (DDL)
: DDL je skup SQL komandi koje formiraju i brišu bazu podataka, dodaju i uklanjaju tabele, formiraju indekse i unose izmene na svakom od pobrojanih elemenata. DDL komande se uglavnom koriste samo tokom izrade baze podataka. Indeksi su strukture za brz pristup podacima i za njihovo brzo ažuriranje.

Jezik za manipulisanje podacima (DML)
: DML je skup komandi koje rade sa DBMS-om i bazom podataka. U DML komande spadaju komande za pretraživanje, umetanje i brisanje podataka. Te komande služe za interakciju s bazom podataka tokom njenog uobičajenog korišćenja.

Upravljanje transakcijama
: SQL sadrži komande za označavanje grupe komandi kao celine ili *transakcije*. Korišćenjem ovih alatki, transakcije se mogu opozvati ili *poništiti*.

Napredne mogućnosti
: DML i DDL pružaju napredne mogućnosti za ugrađivanje SQL-a u programske jezike opšte namene (umnogome nalik na način na koji se SQL komande ugrađuju u PHP, što možete videti u poglavlju 4) i definisanje prikaza postojećih podataka za specijalne namene, kao i dodeljivanje i oduzimanje prava pristupa DBMS-u i bazi podataka. Oni, takođe, sadrže komande za obezbeđivanje integriteta, tj. komande koje obezbeđuju ispravnost podataka i poštovanje relacionih ograničenja.

Upravljanje transakcijama i napredne mogućnosti SQL-a razmotrene su ukratko u poglavljima 3 i 6, kao i u dodatku C. U dodatku E možete pronaći reference o SQL-u.

Naš primer studije

Principi baza podataka praktično su ilustrovani u knjizi aplikacijom Hugh and Dave's Online Wines. U knjizi je nazivamo *prodavnica vina*.

Aplikacija Prodavnica vina ima mnoge komponente tipične za Web aplikacije koje rade s bazama podataka, uključujući:

- Web stranice ispunjene podacima iz baze
- pretraživanje pomoću upita, pri čemu korisnik zadaje ograničavajuće parametre za pretraživanje baze podataka
- unos podataka i proveru njihove ispravnosti. HTML obrasci prihvataju podatke, dok JavaScript kôd na klijentskoj strani i PHP skriptovi na serverskoj strani proveravaju njihovu ispravnost
- praćenje korisnika, tj. upotrebu tehnika za upravljanje sesijom koje protokolu HTTP dodaju čuvanje stanja
- proveru identiteta korisnika i upravljanje njihovim aktivnostima.
- izveštavanje.

Pogledajmo opis prodavnice vina i funkcionalne zahteve sistema. (Postupak modeliranja tih zahteva u obliku entiteta i relacija (engl. *entity relationship*, ER) u relacionoj bazi podataka i pretvaranje tog modela u SQL naredbe teme su kojima je posvećen dodatak C. Kompletan ER model prodavnice vina i SQL naredbe za formiranje baze podataka naći ćete u poglavlju 3. Komponente prodavnice vina koristimo kao primere počev od poglavlja 4. Konačne verzije komponenata aplikacije Prodavnica vina razmatraju se od poglavlja 10 do 13.)

Šta je Hugh and Dave's Online Wines?

Hugh and Dave's Online Wines predstavlja fiktivnu prodavnicu vina na Internetu. U ovom odeljku ukratko ćemo opisati ciljeve i funkcije prodavnice vina, a potom ćemo razmotriti sistemske zahteve izvedene na osnovu toga. Upoznaćemo se, takođe, s tehničkim komponentama prodavnice vina i istaći poglavlja knjige u kojima su te komponente detaljno opisane. Odeljak završavamo diskusijom o nedostacima prodavnice vina i temama koje nisu obrađene u knjizi. Gotovu verziju prodavnice vina opisane u ovom odeljku možete pronaći na Web lokaciji ove knjige.

Prodavnica vina je otvorena za javnost: anonimni korisnici imaju ograničen pristup sistemu, a da bi mogli da poručuju robu, moraju prethodno da se učlane. Cilj Web lokacije je da bude privlačna, jednostavna i upotrebljiva; ipak, pošto su je dizajnirala dva kompjuterska naučnika, nismo uspeli da je učinimo privlačnom! Tehnički ciljevi su mnogo bolje ispunjeni: prodavnica vina upravlja sa preko 1000 vina, informacijama o zalihama i bazom podataka sa oko 1000 kupaca i njihovih porudžbina.

Svaki korisnik sa čitačem Weba može da pristupi lokaciji, pregleda i pretražuje vina koja postoje na zalihama i pogleda podatke o njima. U podatke o vinima spadaju ime, godina berbe, tip vina, sorta grožđa i, u nekim slučajevima, recenzija eksperta. Anonimni korisnik može da doda odabrana vina u svoju korpu za kupovinu. Korisnici aplikacije mogu, takođe, da budu već učlanjeni, a prilikom učlanjenja daju podatke o sebi kao i u većini drugih mrežnih prodavnica.

Da bi kupili vina, korisnici moraju da se prijave tako što daju podatke o članstvu. Ako se korisnik tek učlanio, on automatski biva prijavljen. Kada izabere vino, korisnik može da ga poruči. Porudžbina se smesta otprema, a potvrda šalje e-poštom.

U pozadini, sistem omogućava poslovođama magacina da u bazu podataka dodaju nove isporuke vina. Administrator Web lokacije takođe može da u prodavnicu vina dodaje nova vina, vinarije, vinogradarske regione i ostale podatke. Postoje i ograničene mogućnosti za izradu izveštaja.

Sistemski zahtevi

Sledeći zahtevi se obično mogu sačiniti iz dokumenta o planiranju, razgovora sa kupcima itd. Ali, naravno, ova knjiga ne opisuje postupak softverskog inženjeringa, već prikazuje opšte zahteve koji čine osnovu pomenutih praktičnih primera. Neki aspekti sistemskih zahteva su uprošćeni, određeni aspekti trgovinskih radnji su izostavljeni, a neki detalji su opširno opisani i verno odražavaju realne situacije.

Pobrojani zahtevi su dati samo kao opšte teze; komercijalna aplikacija u realnoj situaciji bila bi detaljno opisana funkcionalnim sistemskim zahtevima. Aplikaciji profesionalnog kvaliteta takođe bi bila pridružena projektna dokumentacija u kojoj bi bila opisana struktura baze podataka, izgled ekrana i tokovi podataka.

Evo pregleda funkcionalnih i sistemskih zahteva:

- Prodavnica vina na Internetu prvenstveno je namenjena za e-poslovanje čiji je cilj prodaja vina.
- Sistem ne omogućava knjigovodstvo, upravljanje zalihama, obračun plata, naručivanje i druge slične poslove.
- Korisnici mogu da odaberu vina i da ih dodaju u korpu za kupovinu. Korisnici mogu da kupe stavke iz svojih korpi najkasnije jedan dan nakon što su u korpu dodali prvu stavku. Korisnici imaju samo jednu korpu za kupovinu i mogu je isprazniti kad god požele.
- Korisnici Web lokacije mogu biti anonimni sve do trenutka kada pristanu da kupe stavke iz svoje korpe.
- Da bi kupili stavke iz korpe, korisnici moraju da se prijave na sistem. Da bi se prijavio, korisnik mora da ima nalog. Da bi dobio nalog, korisnik mora da pruži sledeće podatke o sebi: prezime, ime, adresu (minimum jedan red), grad, poštanski broj, državu, datum rođenja, adresu e-pošte i šifru. Adresa e-pošte služi kao korisničko ime za prijavljivanje na sistem. Korisnik može da pruži dodatne podatke o srednjem slovu, tituli, zatim, tu su dva dodatna polja za adresu, kao i mesta za podatke o delu države (pokrajina, region), broj telefona i broj faksa.
- Kada korisnik kupi vina, njegova porudžbina se arhivira.
- Korisnik može da dobije popust na vrednost porudžbine. Popust se može odobriti određenog dana, na određenu minimalnu količinu, ili redovnom kupcu.
- U porudžbinu može biti uključena cena dostave koja se određuje zavisno od adrese korisnika i načina dostave. U načine dostave spadaju dostava pomorskim saobraćajem, običnom poštom i ekspresnom poštom. Porudžbina može da sadrži i napomenu koja je namenjena kompaniji za dostave; na primer, u napomeni može da se naznači da „vina ostavite ispred zadnjih vrata kuće".
- Vina su po opštoj podeli klasifikovana na crna, bela, penušava, slatka i ojačana. Vina, pored toga, imaju ime, godinu berbe i opis; opisi su neobavezni tekstovi slobodne forme u kojima se obično ocenjuje vino, slično opisu na etiketi.
- Vina se prave od različitih sorti grožđa kao što su Šardone, Semijon, Merlo itd. Jedno vino se može praviti od više sorti, a redosled navođenja sorti je bitan. Na primer, vino napravljeno od dve sorte grožđa, Kaberne i Merlo – Kaberne Merlo – različito je od Merlo Kabernea.
- Korisnici mogu da pregledaju vina prema vrsti ili poreklu.
- Određeno vino proizvodi određena vinarija.

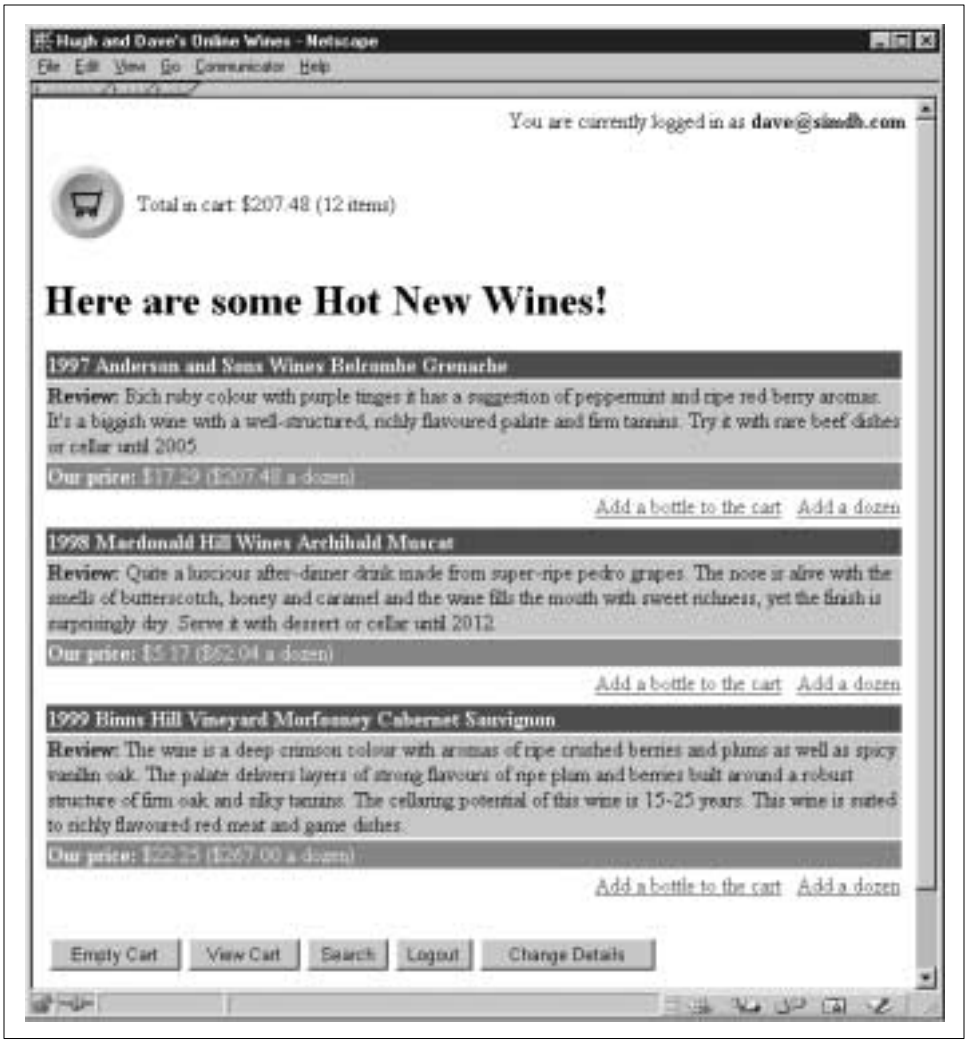
- Vinarije imaju opise, koji se obično sastoje od mišljenja stručnjaka, kao i broj telefona i faksa.
- Vinarije se nalaze u određenim krajevima. Krajevi se nalaze u oblastima – na primer, dolina Barosa u južnoj Australiji – kao i svaka oblast – ima opis i, po mogućstvu, sliku ili mapu.
- Korpa za kupovinu je nezavršena porudžbina koja sadrži stavke. Ona može postati završena porudžbina nakon što se korisnik prijavi na sistem. Svaka stavka u porudžbini sadrži određeno vino, količinu tog vina i cenu po boci. Cena vina je uvek cena boce vina koja je prvobitno stavljena u korpu za kupovinu, što je ujedno i uvek najniža moguća cena na zalihama.
- Korisnik može da menja količinu vina u korpi za kupovinu i stavke mogu da se uklone iz korpe.
- Vina koja se prodaju nalaze se na zalihama robe. Svaki zapis zaliha sadrži datum prispeća i odnosi se na određeno vino. Zalihe sadrže količine razvrstane po ceni boce i ceni gajbe. Može postojati više zapisa zaliha za određeno vino; tada se oni odnose na različite isporuke koje su u prodavnicu vina stigle različitog datuma ili imaju različite cene.
- Korisnicima se uvek nudi najniža cena sa zaliha svakog vina. Kada korisnik doda vino u korpu za kupovinu, njemu je garantovana ta cena.
- Korisnik može da kupi samo ona vina koja su na zalihama.
- Kada korisnik pretvori svoju korpu za kupovinu u porudžbinu, proverava se da li na zalihama postoji dovoljno robe da se ona može poručiti. Ako nema dovoljno vina, korisnik biva obavešten i količina u njegovoj korpi se ažurira; do toga može doći ako korisnik doda u korpu više vina nego što ga ima na zalihama.
- Kada postoji dovoljno robe da se obavi porudžbina, količina vina na zalihama umanjuje se nakon što se porudžbina okonča. Na zalihama se uvek umanjuje najstariji zapis određenog vina.

Komponente prodavnice vina

Ovaj odeljak opisuje principe i praktične tehnike za razvoj svake komponente prodavnice vina opisane u ovoj knjizi. Kompletna aplikacija se razmatra od poglavlja 10 do 13.

Izvršavanje upita u bazi podataka

U poglavlju 4 upoznaćemo se sa tehnikama za povezivanje sa DBMS-om, izvršavanjem srednje složenih SQL upita, učitavanjem rezultata i njihovom obradom. Da bismo ilustrovali te tehnike, smestićemo pano Hot New Wines na početnu stranicu prodavnice vina. Završen pano je prikazan na slici 1-5. Na panou su tri najnovija vina koja su dodata u bazu podataka i koje je ocenio ekspert za vino. Detaljan opis korpe za kupovinu kao komponente nalazi se u poglavlju 11 i on, takođe, sadrži kôd panoa napravljenog u poglavlju 4.

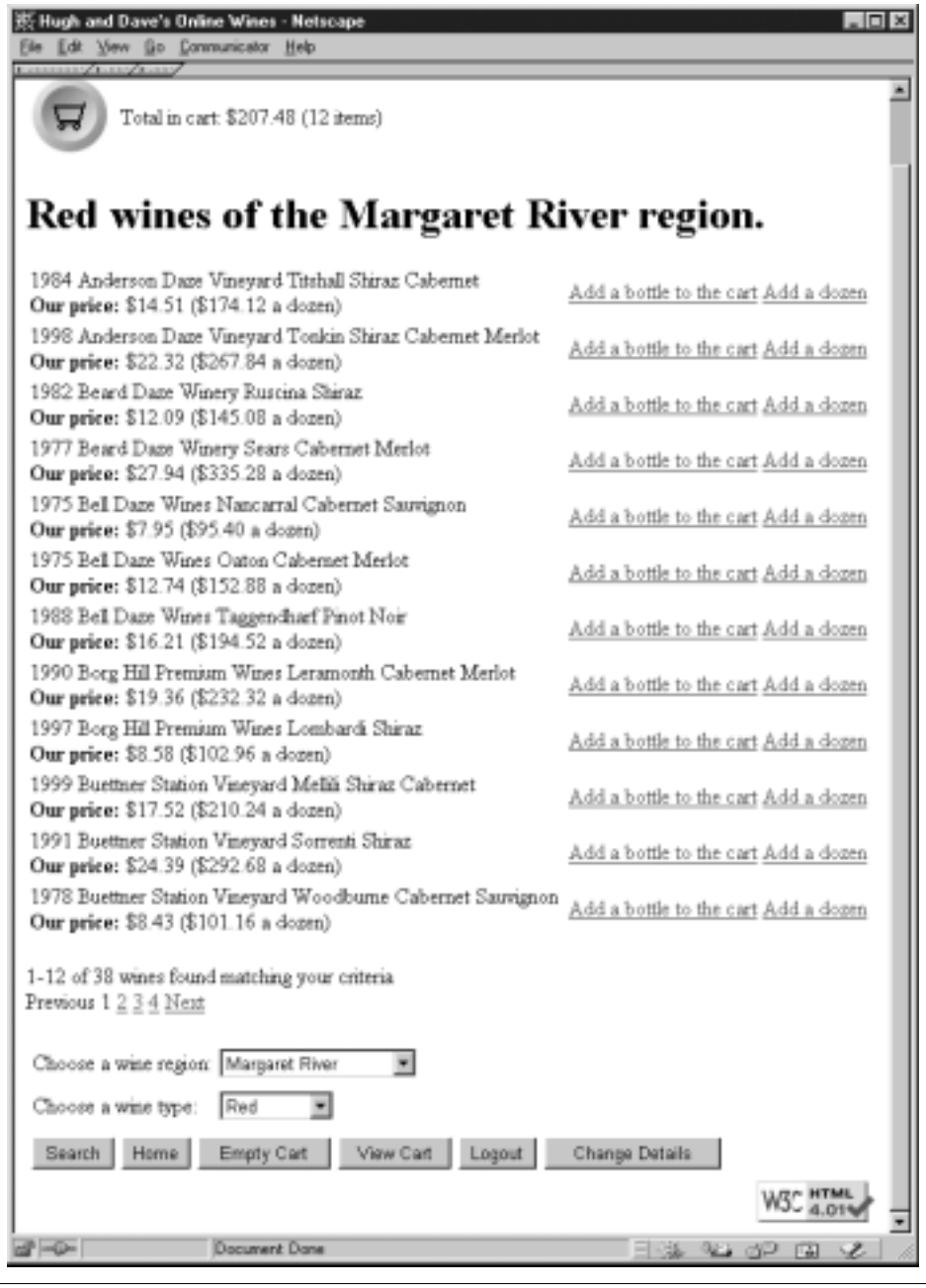


Slika 1-5. Završen pano Hot New Wines na početnoj stranici.

Slanje upita čije parametre zadaju korisnici

Korisnici mogu da vide odabrana vina iz postojećih zaliha prodavnice ako upišu jednostavan uslov za pretragu.

Rezultat koji dobiju pošto pritisnu dugme Search kada odaberu vrstu vina „Red“ ili oblast „Margaret River“, prikazan je na slici 1-6. Na ekranu su rezultati prvih 12 od 38 vina koja zadovoljavaju uslov, a pri dnu se nalaze veze pomoću kojih korisnik može da pregleda ostale rezultate.



Slika 1-6. Veze pri dnu stranice čitača omogućavaju korisniku da pregleda preostale rezultate.

U poglavlju 5 prikazane su tehnike za prikupljanje ulaznih podataka pomoću HTML elemenata `<form>` (obrasca), sastavljanje upita pomoću podataka koje je uneo korisnik i pregledanje rezultata pretraživanja. Tu ćemo se upoznati sa osnovama zaštite baze podataka pomoću prethodne obrade ulaznih podataka. Kompletan kôd ovog modula nalazi se u poglavlju 13.

Unošenje podataka i smeštanje zapisa u bazu podataka

U poglavlju 6 upoznaćemo se s tehnikama unošenja podataka u bazu. Ilustrovaćemo principe unošenja podataka tako što ćemo u poglavljima 6, 7 i 8 razviti jednostavan obrazac za učlanjivanje korisnika. Kompletna realizacija postupka učlanjenja kupaca objašnjena je u poglavlju 10, a završen obrazac prikazan je na slici 1-7.

Unošenje podataka zahteva pažljivo razmatranje načina na koji drugi korisnici istovremeno rade s bazom podataka. U poglavlju 6 upoznaćemo se s teorijom i praksom unošenja podataka u baze, kao i sa PHP funkcijama za upravljanje postupkom unošenja podataka i izveštavanje o njemu.

Provera ispravnosti podataka u klijentskom i srednjem sloju

U poglavlju 7 nastavljamo s razvojem pojednostavljenog obrasca za podatke o kupcu, gde ćemo se upoznati s proverom ispravnosti podataka u klijentskom i srednjem sloju. Provera ispravnosti podataka u oba sloja vrlo je bitna. Provera ispravnosti podataka na klijentskoj strani pomoću JavaScripta smanjuje opterećenje Web servera u srednjem sloju, brza je za korisnika i nije opterećena mrežnom komunikacijom.

Provera ispravnosti podataka na serverskoj strani takođe je vrlo bitna: korisnici mogu da zaobiđu proveru u klijentskom sloju jer ona može biti pogrešno konfigurisana, može se desiti da je čitač Weba ne podržava, a potpuna i složena provera ispravnosti podataka moguća je samo u srednjem sloju.

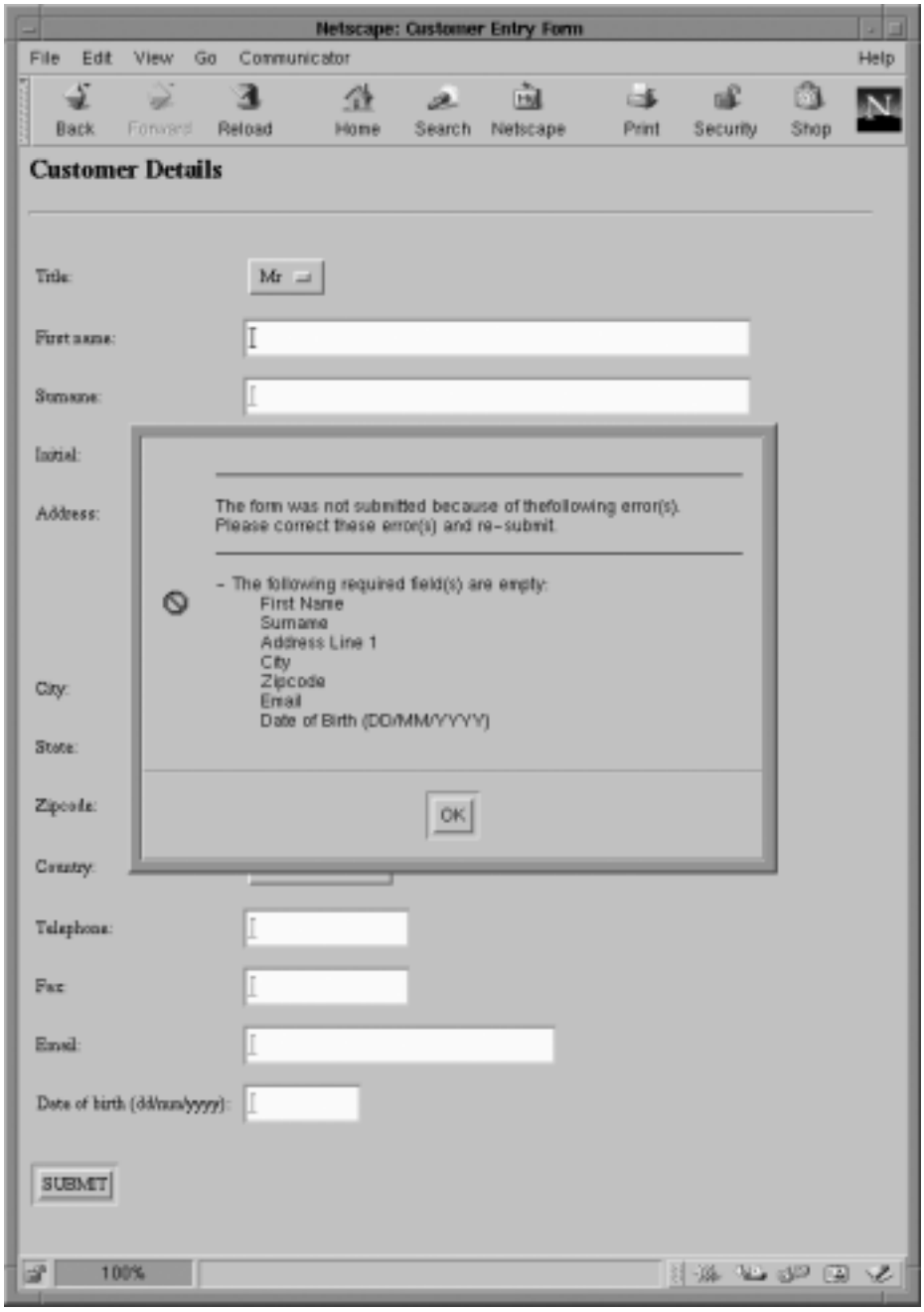
Na slici 1-8 prikazan je obrazac za podatke o kupcu kako prijavljuje grešku prilikom provere ispravnosti podataka pomoću JavaScript tehnika na klijentskoj strani, objašnjenih u poglavlju 7.

Praćenje korisnika i upravljanje sesijama

Tema poglavlja 8 jeste dodavanje stanja protokolu HTTP, a tu ćemo se upoznati i sa PHP tehnikama za rad sa sesijama koje upravljaju transakcijama korisnika u prodavnici vina. Objasnićemo odlike ovih tehnika i ilustrovati slučajeve kada ih treba, a kada ne treba koristiti. U dodatku D razmatraćemo ostale mogućnosti za upravljanje sesijama koje koriste sloj baze podataka za održavanje stanja.

Ilustrovaćemo PHP sesije proširujući obrazac za podatke o kupcu iz poglavlja 6 i 7. Prikazaćemo praktičan primer upisivanja podataka i njihovog ponovnog prikazivanja kada se korisnik vrati da ispravi pogrešno unete podatke tokom provere ispravnosti. Na kraju poglavlja 8 završićemo jednostavan obrazac za unos podataka o kupcima. Potpuna implementacija obrasca korišćenjem istih tehnika tema je poglavlja 10, a sesije se koriste u primerima koda od poglavlja 10 do 13.

Slika 1-7. Pomoću HTML obrasca unose se, sakupljaju i ažuriraju podaci o članovima.



Slika 1-8. JavaScript greška pri proveri ispravnosti podataka unetih u obrazac za podatke o kupcu u prodavnici vina.

Provera identiteta

Provera identiteta je identifikacija dveju strana koje komuniciraju. U poglavlju 9 razmatraćemo principe bezbednosti i provere identiteta. Principe ćemo ilustrovati primerima postupka prijavljivanja u prodavnicu vina i odjavljivanja. Kompletan postupak prijavljivanja u prodavnicu vina i odjavljivanja opisan je u poglavlju 10.

Kompletna aplikacija

Prodavnica vina sadrži nekoliko kompletnih komponenata opisanih u poglavljima 10 do 13:

- Realizacija korpe za kupovinu opisana je u poglavlju 11.
- Prikaz obavljanja porudžbine, potvrđivanja prijema porudžbine putem e-pošte i obaveštavanja o dostavi nalazi se u poglavlju 12.
- Ažuriranje količine robe u korpi za kupovinu opisano je u poglavlju 11.
- Ceo postupak učlanjenja, dopunjavanja podataka o kupcu, prijavljivanja i odjavljivanja opisan je u poglavlju 10.
- Implementacija složenog postupka poručivanja koji upravlja zalihama robe opisana je u poglavlju 12.
- Administrativno održavanje, razdvajanje prezentacije od sadržaja i pretraživanje prodavnice opisani su u poglavlju 13.