Uvod

Dobro došli u knjigu *PHP i MySQL: razvoj aplikacija za veb*. Na stranice ove knjige preneli smo sažeta iskustva koja smo stekli koristeći PHP i MySQL – dve najvažnije i najkorišćenije alatke za izradu veb aplikacija.

U uvodu su obrađene sledeće teme:

- Zašto bi trebalo da pročitate ovu knjigu
- Šta ćete znati da uradite kada savladate njen sadržaj
- Šta su PHP i MySQL i zbog čega su tako izuzetni
- Šta se promenilo u najnovijim verzijama PHP-a i MySQL-a
- Kako je knjiga organizovana

Zašto bi trebalo da pročitate ovu knjigu

Iz ove knjige ćete naučiti da pravite interaktivne veb aplikacije – od vrlo jednostavnog obrasca za unošenje porudžbina, do složenih i bezbednih veb lokacija. Što je još lepše, naučićete kako se to radi uz primenu tehnologija otvorenog koda.

Ova knjiga je namenjena čitaocima koji poznaju barem osnove HTML-a i imaju određeno iskustvo s nekim savremenim programskim jezikom, ali možda nisu programirali veb aplikacije niti koristili relacione baze podataka. Ako ste programer početnik, ova knjiga će vam verovatno ipak biti korisna, ali će vam možda trebati malo više vremena da je savladate. Trudili smo se da ne izostavimo nijedan osnovni koncept, ali ih objašnjavamo samo ukratko. Tipičan čitalac ove knjige je neko ko pokušava da savlada PHP i MySQL kako bi napravio složenu ili komercijalnu veb lokaciju. Možda već radite na nekom drugom jeziku za razvoj veb aplikacija; u tom slučaju, ova knjiga će vam omogućiti da brzo steknete potrebno znanje.

Prvo izdanje ove knjige napisali smo jer su nam dosadile knjige o PHP-u koje se u suštini svode na opisivanje njegovih funkcija. Takve knjige su korisne, ali slabo pomažu kada vam šef ili klijent kaže: „Napravite mi korpu za kupovinu“. U ovoj knjizi, učinili smo sve što je bilo u našoj moći da svaki primer bude koristan. Mnoge blokove koda možete upotrebiti direktno na svojoj veb lokaciji, dok drugi postaju upotrebljivi nakon manjih izmena.

Šta ćete naučiti iz ove knjige

Iz ove knjige naučićete da pravite dinamičke veb aplikacije koje rade u stvarnom okruženju. Ako ste dosad pravili veb lokacije isključivo pomoću HTML-a, verovatno ste već upoznali ograničenja takvog rešenja. Statički sadržaj koji prikazuje veb lokacija napravljena pomoću čistog HTML koda upravo je takav – statičan. Taj sadržaj se ne menja dok ga

ručno ne izmenite. Korisnici takve veb lokacije ni na koji način ne mogu da utiču na prikazan sadržaj.

Upotreba programskog jezika kao što je PHP, i baze podataka kao što je MySQL, omogućava izradu dinamičkih veb lokacija, čiji je sadržaj prilagodljiv i koje prikazuju podatke u realnom vremenu.

U ovoj knjizi smo se namerno usredsredili na praktične aplikacije iz stvarnog života, čak i u uvodnim poglavljima. Počinjemo od jednostavnog sistema a zatim obrađujemo razne aspekte PHP-a i MySQL-a. Upoznajemo vas i sa integrativnim klijentskim i serverskim tehnologijama tako što razmatramo JavaScript i ulogu koju on može da ima u razvoju vaših aplikacija.

U poslednjem delu prikazujemo efikasno projektovanje aplikacija koje treba da rade na vebu i vodimo vas kroz faze projektovanja, planiranja i izrade sledećih projekata:

- Identifikovanje i personalizovanje korisnika
- Veb klijent za e-poštu
- Integrisanje društvenih mreža

Svaki od navedenih projekata možete koristiti bez izmena ili ga možete prilagoditi svojim potrebama. Izabrali smo ih zato što smatramo da predstavljaju neke od najčešćih vrsta današnjih veb aplikacija. Čak i ako su vaše potrebe drugačije, trebalo bi da vam ova knjiga pomogne da postignete svoje ciljeve.

Šta je PHP?

PHP je jezik za pisanje skriptova koji rade na serveru, osmišljen za upotrebu na vebu. Unutar HTML stranica može se ugraditi PHP kôd koji se izvršava kada posetilac veb lokacije zatraži stranicu. Veb server tumači PHP kôd ugrađen u stranicu i generiše HTML kôd ili drugu vrstu izlaznih podataka koje posetilac stranice može da vidi.

Početna verzija PHP-a napravljena je 1994. godine. To je bio rezultat truda samo jednog čoveka, Rasmusa Lerdorfa. Potom su ga prihvatili i drugi. Pretrpeo je nekoliko značajnih revizija dok nije postao složen i zreo proizvod kakvog danas poznajemo. Prema istraživanjima koje je obavio Greg Michillie iz Googlea u maju 2013, PHP se koristio na više od tri četvrtine veb lokacija širom sveta, a taj udeo je porastao na preko 82% do jula 2016.

PHP je projekat otvorenog koda, što znači da imate pristup izvornom kodu koji možete slobodno koristiti, menjati i dalje distribuirati.

PHP je prvobitno bila skraćenica za *Personal Home Page*, ali je značenje kasnije izmenjeno u skladu s rekurzivnim GNU (GNU = Gnu's Not Unix) pravilima za davanje imena, tako da je PHP sada skraćenica za *PHP Hypertext Preprocessor*.

Tekuća glavna verzija PHP-a je 7, s novom Zend mašinom, napisanom iz početka; osim toga, ova verzija donosi i nekoliko suštinskih poboljšanja jezika. Sav kôd u ovoj knjizi ispitan je i proveren u verziji PHP 7 koja je bila najnovija u vreme pisanja knjige, kao i u najnovijoj verziji iz porodice PHP 5.6, koja je i dalje zvanično podržana.

Zvanična veb lokacija PHP-a nalazi se na adresi http://www.php.net.

Početna stranica firme Zend Technologies nalazi se na adresi http://www.zend.com.

Šta je MySQL?

MySQL (izgovara se maj es ku el) veoma je brz i robustan *sistem za upravljanje relacionim bazama podataka* (RDBMS). Baza podataka omogućava efikasno skladištenje, pretraživanje, sortiranje i učitavanje podataka. Server MySQL upravlja pristupom podacima kako bi se omogućilo da s njima radi više korisnika istovremeno, te da pristup podacima bude brz i dozvoljen samo ovlašćenim korisnicima. Zbog toga je MySQL višekorisnički i višenitni server. Podržava SQL (*Structured Query Language*) – standardni jezik za upite u baze podataka. MySQL je na raspolaganju korisnicima od 1996. godine, ali je njegov razvoj započet još 1979. To je najpopularnija baza podataka otvorenog koda na svetu. Više puta je dobijala nagradu Reader's Choice Award, koju dodeljuje časopis *Linux Journal*.

MySQL je sada na raspolaganju s dve vrste licence. Možete ga besplatno koristiti pod uslovima licence otvorenog koda (GPL), ali uz obavezu da i sami poštujete uslove te licence. Ako svoju aplikaciju u koju je ugrađen MySQL želite da distribuirate pod uslovima različitim od onih koji važe za GPL, možete kupiti licence za komercijalnu upotrebu.

Zašto biste koristili PHP i MySQL?

Ako nameravate da napravite veb lokaciju, na raspolaganju vam je veći broj alatki za tu namenu.

Moraćete da izaberete sledeće komponente:

- Mesto gde će se izvršavati vaši veb serveri: u oblaku, na virtuelnim privatnim serverima ili na stvarnom hardveru
- operativni sistem
- softver za veb server
- sistem za upravljanje bazama podataka ili drugo skladište podataka
- programski jezik ili jezik za skriptove

Možda ćete koristiti hibridnu arhitekturu s više skladišta podataka. Neke od pomenutih odluka zavise od drugih. Na primer, ne rade svi operativni sistemi na svakom hardveru, ne podržavaju svi veb serveri sve programske jezike itd.

U ovoj knjizi se ne bavimo mnogo hardverom, operativnim sistemima niti softverom za veb server jer nema potrebe. Jedna od najkorisnijih odlika i PHP-a i MySQL-a jeste to što mogu da rade pod svakim poznatijim operativnim sistemom, kao i pod mnogim manje popularnim.

PHP kôd je najčešće prenosan – može se napisati tako da se može koristiti na različitim operativnim sistemima i veb serverima. Postoje PHP funkcije specifične za određeni sistem datoteka (engl. *filesystem*) koji zavisi od operativnog sistema, ali su one jasno obeležene kao takve i u priručniku za PHP i u ovoj knjizi.

Bez obzira na hardver, operativni sistem i veb server koji izaberete, ubeđeni smo kako bi trebalo da ozbiljno razmotrite upotrebu PHP-a i MySQL-a.

Neke od prednosti PHP-a

Glavni konkurenti PHP-u su Python, Ruby (on Rails ili drugi), Node.js, Perl, Microsoft .NET i Java.

U poređenju s navedenim proizvodima, PHP ima više prednosti, na primer:

- Performanse
- Proširivost
- Mogućnost povezivanja s velikim brojem sistema za upravljanje bazama podataka
- Ugrađene biblioteke za obavljanje mnogih poslova koji su uobičajeni u veb aplikacijama
- Niska cena
- Lako se uči i upotrebljava
- Dobra podrška za objektno orijentisano programiranje
- Prenosivost
- Fleksibilnost pristupa razvoju aplikacija
- Izvorni kôd je dostupan svima
- Široko dostupna podrška i dokumentacija

Slede detaljnija objašnjenja navedenih prednosti.

Performanse

PHP je veoma brz. Jedan relativno jeftin server može da obradi više miliona zahteva dnevno. Pogodan je i za najmanje obrasce za e-poštu i za izuzetno obimne veb lokacije kao što su Facebook i Etsy.

Proširivost

PHP ima ono što njegov autor, Rasmus Lerdorf, često naziva „shared-nothing“ arhitektura. To znači da možete efikasno i jeftino horizontalno proširiti sistem na veliki broj običnih, jeftinih servera za podatke.

Integrisanost sa sistemom za upravljanje bazama podataka

PHP standardno omogućava uspostavljanje veza s više sistema za upravljanje bazama podataka. Osim s MySQL-om, moguće je uspostavljanje direktnih veza sa sistemima PostgreSQL, Oracle, MongoDB i MSSQL – između ostalih. U PHP 5 i PHP 7 ugrađen je i SQL interfejs za pristupanje običnim tekstualnim datotekama, nazvan SQLite.mSQL.

Pomoću standarda ODBC (*Open Database Connectivity*) mogu se uspostavljati veze sa svakom bazom podataka za koju postoji ODBC upravljački program. To važi za Microsoftove proizvode, kao i za mnoge druge.

Pored biblioteka, PHP ima i sloj apstrakcije za pristup bazama podataka koji se zove PHP Database Objects (PDO); on omogućava konsistentan pristup bazama i pospešuje pisanje bezbednog koda.

Ugrađene biblioteke

Budući da je projektovan za upotrebu na vebu, PHP sadrži brojne funkcije koje omogućavaju obavljanje mnogih poslova potrebnih u veb aplikacijama. Možete generisati slike tokom rada aplikacije, uspostavljati veze s veb servisima i drugim mrežnim uslugama, koristiti XML dokumente, slati poruke e-poštom, raditi s kolačićima, generisati dokumente u formatu PDF – i sve to pomoću nekoliko redova programskog koda.

Niska cena

PHP je besplatan. Najnoviju verziju uvek možete preuzeti sa adrese http://www.php.net, potpuno besplatno.

Lako se uči

Sintaksa PHP-a slična je sintaksama drugih programskih jezika, prvenstveno jezika C i Perl. Ako već poznajete C ili Perl, ili neki drugi jezik sličan C-u, kao što je C++ ili Java, gotovo odmah ćete moći da koristite PHP za konkretne projekte.

Podrška za objektno orijentisano programiranje

Verzija PHP 5 je uvela objektno orijentisane mogućnosti koje su dodatno poboljšane u verziji PHP 7. Ako poznajete programiranje na jezicima Java ili C++, prepoznaćete elemente (a uglavnom i sintaksu) koje biste i očekivali, kao što su nasleđivanje, privatni i zaštićeni atributi i metode, apstraktne klase i metode, interfejsi, konstruktori i destruktori. Otkrićete i neke ređe korišćene mogućnosti, kao što je upotreba iteratora i svojstvenosti (engl. *traits*).

Prenosivost

PHP je na raspolaganju za mnoge operativne sisteme. PHP kôd možete pisati pod jednim od operativnih sistema iz porodice Unix, kao što su Linux i FreeBSD, pod komercijalnim verzijama Unixa ili raznim verzijama Microsoftovog Windowsa.

Kôd koji napišete najčešće će raditi bez izmena i u drugim operativnim sistemima u kojima je instaliran PHP.

Fleksibilnost pristupa razvoju aplikacija

U PHP-u se mogu lako realizovati jednostavni zadaci, a isto tako i velike aplikacije, uz upotrebu radnog okvira (frejmvorka) zasnovanog na projektnim obrascima (engl. *design patterns*) kao što je Model-View-Controller (MVC).

Izvorni kôd

Izvorni kôd PHP-a je javno dostupan. Za razliku od komercijalnih proizvoda zatvorenog koda, ako želite da izmenite nešto ili dodate nešto jeziku PHP, to možete slobodno učiniti.

Ne morate čekati da proizvođač objavi zakrpe (servisne pakete) niti brinuti o tome da li će proizvođač bankrotirati ili prestati da podržava proizvod.

Dostupnost podrške i dokumentacije

Kompanija Zend Technologies (www.zend.com), koja je autor mašine na kojoj se zasniva PHP, finansira razvoj PHP-a zaradom od podrške i srodnog softvera koji distribuira na komercijalnoj osnovi.

Dokumentacija o PHP-u i zajednica korisnika PHP-a ozbiljni su i bogati izvori informacija koje svi mogu da koriste.

Najvažnije odlike verzije PHP 7

U decembru 2015, javnosti je postala dostupna dugoočekivana verzija PHP 7. Kao što je pomenuto u ovom uvodu, ova knjiga obrađuje i PHP 5.6 i PHP 7, pa biste se mogli zapitati „šta se dogodilo s PHP-om 6?“ Kratak odgovor glasi: PHP 6 ne postoji i nikada nije ni postojao za široku javnost. Postojao je proces posvećen razvoju baze koda, koji je nosio oznaku „PHP 6“, ali on nikada nije dospeo do faze gotovog proizvoda; bilo je mnogo ambicioznih planova i posledičnih komplikacija koje su razvojnom timu otežavale nastavak rada. PHP 7 *nije* PHP 6 i ne sadrži mogućnosti i kôd iz pomenutog razvojnog procesa; PHP 7 je zasebna verzija sa sopstvenim fokusom – konkretno, fokusom na performansama.

Ispod haube, PHP 7 sadrži poboljšanu Zend mašinu koja ga pokreće, što je dovelo do značajnog poboljšavanja performansi mnogih veb aplikacija – ponekad i za čitavih 100%! Bolje performanse i smanjeno korišćenje memorije ključni su za PHP 7, a isto važi i za kompatibilnost sa starijim verzijama. U stvari, uvedeno je relativno malo izmena koje nisu kompatibilne s ranijim verzijama. One su razmotrene u odgovarajućem kontekstu kroz celu knjigu, tako da se gradivo iz poglavlja može koristiti i za PHP 5.6 i za PHP 7, pošto PHP 7 još nije masovno primenjen na komercijalnim veb lokacijama.

Neke prednosti MySQL-a

Najvažniji takmaci MySQL-a u oblasti relacionih baza podataka jesu PostgreSQL, Microsoft SQL Server i Oracle. U sveti veb aplikacija, postoji i rastući trend ka korišćenju NoSQL/nerelacionih baza podataka – kao što je MongoDB. Pogledajmo zašto je MySQL i dalje dobar izbor u mnogim slučajevima. MySQL ima više dobrih osobina, uključujući i sledeće:

- odlične performanse
- niska cena
- lako se konfiguriše i uči
- prenosivost
- izvorni kôd je javno dostupan
- široko dostupna podrška.

U narednim odeljcima opširnije opisujemo navedene osobine.

Performanse

MySQL je neosporno brz. Na veb lokaciji http://www.mysql.com/ nalazi se stranica s rezultatima uporednih testova.

Niska cena

MySQL se može nabaviti potpuno besplatno, pod uslovima licence za otvoren izvorni kôd, a na raspolaganju je i komercijalna licenca po veoma niskoj ceni. Takva licenca vam je neophodna ako želite da distribuirate MySQL kao sastavni deo svoje aplikacije koju ne distribuirate pod uslovima licence otvorenog koda. Ukoliko ne nameravate da dalje distribuirate svoju aplikaciju – što je uobičajeno za veb aplikacije – ili je delite besplatno, odnosno pravite aplikacije otvorenog koda, ne morate platiti licencu za upotrebu MySQL-a.

Jednostavnost upotrebe

Većina savremenih baza podataka koristi jezik SQL. Ako već imate iskustva s nekim drugim sistemom za upravljanje bazama podataka, neće vam biti teško da pređete na MySQL. Osim toga, MySQL se podešava i koristi znatno lakše od mnogih drugih proizvoda slične namene.

Prenosivost

MySQL se može koristiti u mnogim varijantama Unixa, kao i u Microsoftovom Windowsu.

Izvorni kôd

Kao i za PHP, izvorni kôd MySQL-a može se besplatno nabaviti i slobodno menjati. Mada je ova mogućnost nevažna mnogim korisnicima, u većini slučajeva pruža vam osećaj sigurnosti i saznanje da proizvod ima budućnost i da niste bespomoćni ako se pojave problemi.

Štaviše, sada postoji nekoliko derivata (engl. *forks*) i zamena za MySQL o čijoj upotrebi možda možete razmisliti – uključujući derivat MariaDB, koji su napisali originalni autori MySQL-a, uključujući Michael 'Monty' Wideniusa (https://mariadb.org).

Podrška

Ne stoji iza svakog proizvoda otvorenog koda kompanija koja obezbeđuje podršku, obuku, konsultantske usluge i sertifikovanje znanja, ali sve to možete dobiti od kompanije Oracle (koja je preuzela MySQL kada je kupila Sun Microsystems, koji je prethodno otkupio kompaniju osnivača, MySQL AB).

Šta je novo u MySQL-u (5.x)?

U vreme pisanja ove knjige, najnovija verzija MySQL-a bila je 5.7.

Među mogućnosti dodate MySQL-u u poslednjih nekoliko verzija spadaju

- širok opseg poboljšanja koja se odnose na bezbednost
- FULLTEXT podrška za InnoDB tabele
- API u NoSQL stilu za InnoDB
- podrška za particionisanje
- poboljšanja replikacije, uključujući i replikaciju zasnovanu na redovima i identifikatore transakcija (GTID)
- postojanje rezerve niti
- modularna provera identiteta
- prilagodljivost radu u višejezgarnom sistemu

- bolje dijagnostičke alatke
- InnoDB kao podrazumevana mašina baze podataka
- podrška za IPv6
- API u obliku dodatnog modula
- planiranje događaja
- automatizovano ažuriranje

Od ostalih izmena, pomenućemo bolju usklađenost sa ANSI standardom i bolje performanse.

Ako još uvek koristite neku od ranijih verzija 4.x ili verziju 3.x MySQL servera, korisno je da znate da su u razne verzije posle 4.0 dodate sledeće mogućnosti:

- prikazi
- uskladištene procedure
- okidači i kursori
- podrška za podupite
- GIS tipovi podataka za skladištenje geografskih podataka
- bolja podrška za internacionalizaciju
- mašina InnoDB, koja omogućava transakcije, sada je standardna
- MySQL-ova ostava (heš) za upite; ona značajno povećava brzinu upita koji se izvršavaju više puta zaredom, što je čest slučaj u veb aplikacijama.

Kako je knjiga organizovana?

Ova knjiga je podeljena na pet delova:

U prvom delu knjige, „PHP“, najvažniji delovi PHP-a opisani su na više primera. Svaki primer potiče iz stvarnog projekta veb lokacije za e-trgovinu – znači, nije izmišljen. Ukoliko ste već koristili PHP, dovoljno je da samo površno prelistate prvo poglavlje, „Kratak kurs PHP-a“. Ako ste početnik u PHP-u ili učite da programirate, bilo bi korisno da posvetite više vremena ovom delu knjige.

U drugom delu knjige, „MySQL“, opisujemo koncepte i načine projektovanja aplikacija kada se koriste sistemi za upravljanje relacionim bazama podataka, kao što je MySQL. Između ostalog, objašnjavamo upotrebu SQL-a, povezivanje MySQL-ove baze podataka sa spoljnim svetom pomoću PHP-a i naprednije teme u vezi s MySQL-om, kao što su zaštita i optimizovanje baze podataka.

U trećem delu, „Bezbednost veb aplikacija“, uopšteno razmatramo razvoj veb aplikacija, nevezano za programski jezik. Zatim objašnjavamo kako da upotrebite PHP i MySQL da biste proverili identitet korisnika aplikacije i kako da bezbedno primate, šaljete i skladištite podatke.

Četvrti deo knjige, „Napredne tehnike korišćenja PHP-a“, sadrži iscrpna objašnjenja najvažnijih standardnih funkcija PHP-a. Izabrali smo grupe funkcija za koje smatramo da su najkorisnije pri izradi veb aplikacija. Naučićete i da komunicirate sa serverom i mrežom, generišete slike, obrađujete podatke tipa datum/vreme i upravljate sesijama.

Peti deo, „Praktični projekti“ najviše volimo. Tu rešavamo konkretne zadatke, kao što su upravljanje opsežnim projektima i otklanjanje grešaka. Naveli smo i primere celovitih projekata koji ilustruju moć i raznovrsnost PHP-a i MySQL-a.

Prateći sadržaj na vebu

Na veb strani prikaza ove knjige (`www.mikroknjiga.rs/store/prikaz.php?ref=978-86-7555-421-9&pismo=latinica`) pronaći ćete:

- izvorni kôd za poglavlja iz knjige
- zadatke za proveru znanja, sa rešenjima
- kviz za proveru znanja, sa rešenjima
- korisne linkove za instaliranje Apachea, PHP-a i MySQL-a
- ispravke uočenih grešaka, ako se pojave.

Završna reč

Nadamo se da će vam se knjiga dopasti i da ćete uživati učeći PHP i MySQL kao i mi kada smo počeli da ih koristimo. Pravo je zadovoljstvo raditi u PHP-u i MySQL-u. Uskoro ćete moći da se pridružite hiljadama programera koji koriste ove moćne alatke za izradu dinamičkih veb aplikacija.

PHP

1 Kratak kurs PHP-a
2 Snimanje i učitavanje podataka
3 Upotreba nizova
4 Rad sa znakovnim nizovima i regularnim izrazima
5 Višekratna upotreba koda i pisanje funkcija
6 Objektno orijentisani PHP
7 Obrada grešaka i izuzetaka

1

Kratak kurs PHP-a

Ovo poglavlje sadrži kratak pregled sintakse i jezičkih konstrukcija PHP-a. Ako već programirate na PHP-u, popunićete praznine u svom znanju, a ako ste ranije koristili C, Perl, Python ili neki drugi programski jezik, ovo poglavlje će vam pomoći da brzo pređete na PHP.

Radićete na primerima iz stvarnog sveta, zasnovanim na iskustvu koje smo stekli pri izradi pravih veb lokacija, i tako naučiti kako da koristite PHP. Programerski udžbenici vas često uče osnovnoj sintaksi, ali s veoma jednostavnim primerima. Mi smo odlučili da ne radimo tako. Svesni smo da čitaoci žele da što pre nešto naprave i pokrenu, da shvate kako se jezik koristi, umesto da se muče s još jednom sintaksom i objašnjenjima funkcija koja nisu ništa bolja od uputstava na vebu.

Isprobajte naredne primere – otkucajte ih ili ih preuzmite na veb strani prikaza ove knjige (`www.mikroknjiga.rs/store/prikaz.php?ref=978-86-7555-421-9&pismo=latinica`), menjajte ih, rastavljajte i učite kako da ih popravite.

Poglavlje počinje primerom on-lajn narudžbenice iz koga ćete naučiti kako se u PHP-u koriste promenljive, operatori i izrazi. Govorićemo o tipovima promenljivih i prioritetima operatora. Naučićete da pristupate promenljivama obrasca izračunavajući ukupnu vrednost i porez na porudžbini.

Zatim ćemo napraviti obrazac za unošenje podataka u kojem ćemo pomoću PHP skripta proveriti ulazne podatke. Proučićemo koncept logičkih vrednosti i primere upotrebe naredbi `if`, `else`, operatora `?:` i naredbe `switch`. I najzad, napisaćemo petlje koje generišu HTML tabele.

U ovom poglavlju obrađene su sledeće teme:

- Ugradnja PHP-a u HTML
- Generisanje dinamičkog sadržaja
- Pristup promenljivama obrasca
- Identifikatori
- Promenljive
- Tipovi promenljivih
- Dodela vrednosti promenljivama
- Konstante
- Oblast važenja promenljivih
- Operatori i prioriteti

- Izrazi
- Funkcije
- Donošenje odluka pomoću naredbi `if`, `else` i `switch`
- Iteracija: petlje `while`, `do` i `for`

Pre nego što počnete: pristupanje PHP-u

Da biste obradili primere iz ovog poglavlja i ostatka knjige, morate imati pristup veb serveru na koji je instaliran PHP. Da biste iz tih primera naučili što više, trebalo bi da ih pokrenete i pokušate da ih izmenite. Za to će vam biti potreban server na kome ćete eksperimentisati.

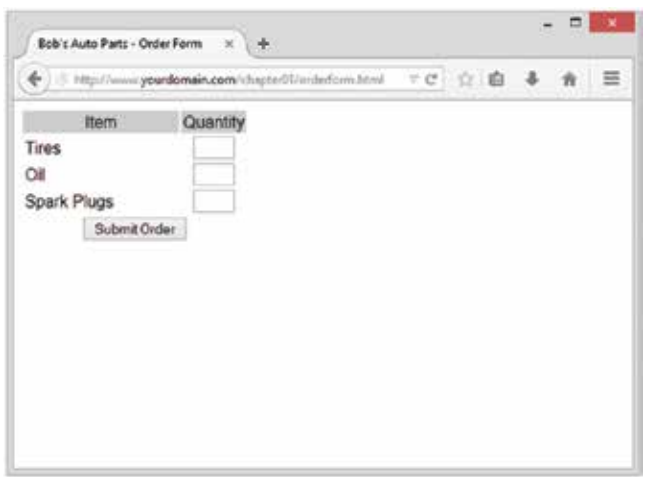
Ako PHP nije instaliran na vašem računaru, moraćete prvo da ga instalirate ili da se za to obratite administratoru sistema. Uputstvo za instaliranje pronaći ćete u dodatku A.

Probna aplikacija: Bobovi auto-delovi

Jedna od najčešćih primena serverskih skript jezika jeste obrada HTML obrazaca. Učenje PHP-a počećete tako što ćete napraviti narudžbenicu za fiktivnu veb prodavnicu „Bobovi auto-delovi“. Izvorni kôd primera iz ovog poglavlja nalazi se na veb strani prikaza ove knjige (`www.mikroknjiga.rs/store/prikaz.php?ref=978-86-7555-421-9&pismo=latinica`), u direktorijumu `Chapter01`.

Izrada obrasca za narudžbenicu

Veb dizajner je napravio narudžbenicu za auto-delove koje Bob prodaje. To je relativno jednostavna narudžbenica (slika 1.1), slična mnogima koje ste do sada verovatno sretali na vebu. Bob bi želeo da zna šta su njegovi kupci naručili i da izračuna ukupnu vrednost naručene robe i iznos poreza na tu robu.



Slika 1.1 Početna verzija narudžbenice beleži samo proizvode i količine.

Deo HTML koda narudžbenice prikazan je u listingu 1.1.

Listing 1.1 **orderform.html – HTML kôd osnovne verzije narudžbenice**

```
<form action="processorder.php" method="post">
<table style="border: 0px;">
<tr style="background: #cccccc;">
  <td style="width: 150px; text-align: center;">Item</td>
  <td style="width: 15px; text-align: center;">Quantity</td>
</tr>
<tr>
  <td>Tires</td>
  <td><input type="text" name="tireqty" size="3"
 maxlength="3" /></td>
</tr>
<tr>
 <td>Oil</td>
 <td><input type="text" name="oilqty" size="3"
  maxlength="3" /></td>
</tr>
<tr>
 <td>Spark Plugs</td>
 <td><input type="text" name="sparkqty" size="3"
    maxlength="3" /></td>
</tr>
<tr>
 <td colspan="2" style="text-align: center;"><input type="submit" value="Submit Order" /></td>
</tr>
</table>
</form>
```

Obratite pažnju na to da je u atributu `action` obrasca naveden PHP skript koji će obraditi porudžbinu (kasnije ćemo napisati taj skript). Vrednost atributa `action` je URL koji će biti učitan kada korisnik pritisne dugme Submit Order. Podaci koje je korisnik uneo u obrazac biće poslati stranici na toj adresi HTML metodom zadatom u atributu `method`. To može biti metoda `get` (podaci se dodaju na kraj URL-a) ili metoda `post` (podaci se šalju zasebno).

Sledeće što treba uočiti jesu imena polja u obrascu – `tireqty`, `oilqty` i `sparqty`. Ta imena ćemo upotrebiti i u PHP skriptu. Zbog toga je važno da poljima u obrascu date smislena imena kojih ćete se lako setiti dok budete pisali PHP skript (na primer, `gumekol`, `uljekol` i `modelkol`). Neki editori za HTML daju poljima podrazumevana imena – recimo, `field23` – koja se teško pamte. Ako imena polja asociraju na podatke koji se u njih unose, lakše ćete programirati.

Trebalo bi da razmotrite i usvajanje određenog standarda za imena polja, koji ćete primenjivati na svim stranicama. Tako će vam biti lakše da zapamtite da li ste, na primer, u imenu polja skratili reč ili ste umesto razmaka stavili donju crtu.

Obrada podataka sa obrasca

Da biste obradili podatke sa obrasca, treba da napravite skript pomenut u atributu `action` oznake `form`. Taj skript će se zvati `processorder.php`. Pokrenite svoj editor teksta i napravite tu datoteku. Otkucajte zatim sledeći kôd:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Bob's Auto Parts – Order Results</title>
  </head>
  <body>
    <h1>Bob's Auto Parts</h1>
    <h2>Order Results</h2>
  </body>
</html>
```

Obratite pažnju na to da je sve što ste do sada otkucali samo običan HTML. Vreme je da dodate nešto jednostavnog PHP koda u skript.

Ugrađivanje PHP koda u HTML kôd

Ispod naslova `<h2>` u datoteci, dodajte sledeće redove:

```
<?php
  echo '<p>Order processed.</p>';
?>
```

Snimite datoteku i pogledajte je u čitaču veba (engl. *web browser*) tako što ćete popuniti obrazac i pritisnuti dugme Submit Order. Trebalo bi da se pojavi rezultat nalik na onaj sa slike 1.2.



Slika 1.2 Tekst prosleđen PHP naredbi `echo` poslat je čitaču veba.

Obratite pažnju na to da je PHP kôd koji ste napisali ugrađen u običnu HTML datoteku. Pogledajte izvorni kôd stranice u čitaču veba. Trebalo bi da vidite nešto slično sledećem:

```
<html>
  <head>
    <title>Bob's Auto Parts – Order Results</title>
  </head>
  <body>
```

```
    <h1>Bob's Auto Parts</h1>
    <h2>Order Results</h2>
    <p>Order processed.</p>
  </body>
</html>
```

PHP naredbe se ne vide zato što ih je interpreter PHP koda zamenio rezultatom. To znači da se iz PHP koda dobija čisti HTML kôd, koji se može gledati pomoću svakog čitača; drugim rečima, čitač ne mora da razume PHP.

Prethodni primer ilustruje način rada serverskih skriptova. PHP kôd se interpretira i izvršava na veb serveru, za razliku od JavaScripta i drugih klijentskih jezika koji se prevode i izvršavaju unutar čitača veba na korisnikovom računaru.

Kôd koji se sada nalazi u PHP datoteci sadrži četiri vrste tekstualnih elemenata:

- HTML oznake (engl. *HTML tags*)
- PHP oznake (engl. *PHP tags*)
- PHP naredbe (engl. *PHP statements*)
- Beline (engl. *whitespace*)

Možete dodati i komentare.

Većinu redova u navedenom primeru čini običan HTML kôd.

PHP oznake

PHP kôd iz prethodnog primera počeo je sa `<?php` i završio se sa `?>`. To je slično HTML oznakama jer sve one počinju znakom „manje od“ (<) i završavaju se znakom „veće od“ (>). Te kombinacije znakova nazivaju se *PHP oznake* (engl. *PHP tags*) – one veb serveru pokazuju gde PHP kôd počinje i gde se završava. Ceo tekst između te dve oznake smatra se PHP kodom. Tekst izvan tih oznaka samo se prosleđuje čitaču jer predstavlja običan HTML kôd. PHP oznake omogućavaju da *izađemo* (engl. *escape*) iz bloka HTML koda.

Postoje dva stila PHP oznaka. Sledeći primeri koda međusobno su ekvivalentni.

- **XML stil**

  ```
  <?php echo '<p>Order processed.</p>'; ?>
  ```

  Ovaj stil oznaka koristimo u knjizi, a to je i stil čija se upotreba preporučuje. Pošto ga administrator servera ne može isključiti, on će sigurno biti dostupan na svim serverima, što je naročito korisno kada se aplikacija koristi na više različitih servera. Može se koristiti s XML dokumentima. U opštem slučaju, preporučujemo da koristite ovaj stil oznaka.

- **Skraćeni stil**

  ```
  <? echo '<p>Order processed.</p>'; ?>
  ```

  Ovaj stil je jednostavniji i oznake liče na instrukcije za obradu u SGML-u. Ako hoćete da ga koristite, potrebno je da u konfiguracijskoj datoteci PHP okruženja aktivirate opciju `short_tags` ili da prevedete PHP sa uključenim kratkim oznakama. Više informacija o upotrebi skraćenog stila oznaka naći ćete u dodatku A. Međutim, upotreba ovog stila se ne preporučuje u kodu koji će se distribuirati jer neće raditi u mnogim okruženjima pošto više nije podrazumevano uključen.

PHP naredbe

PHP naredbe (engl. *PHP statements*) nalaze se između početnih i završnih oznaka i određuju šta interpreter PHP koda treba da uradi. U navedenom primeru upotrebili smo samo jedan tip naredbe:

```
echo '<p>Order processed.</p>';
```

Kao što ste verovatno pretpostavili, naredba `echo` je veoma jednostavna: ona u čitaču ispisuje tekst koji joj prosledite. Na slici 1.2 vidite da se tekst `Order processed` pojavljuje u prozoru čitača.

Obratite pažnju na to da se na kraju naredbe `echo` nalazi znak tačka i zarez. On razdvaja izraze u PHP-u – kao što se u govornom jeziku za razdvajanje rečenica koristi tačka. Ako ste programirali na C-u ili na Javi, takva upotreba tačke i zareza biće vam poznata.

Izostavljanje tačke i zareza česta je sintaksna greška koja se lako pravi. Srećom, takva greška se isto tako lako otkriva i ispravlja.

Beline

Znakovi za razdvajanje – poput znaka za novi red (za povratak na početak reda), razmaka i tabulatora – nazivaju se *beline* (engl. *white spaces*). Kao što verovatno već znate, čitači veba zanemaruju beline u HTML kodu. Isto tako postupa i izvršno okruženje za PHP. Pogledajte ova dva bloka HTML koda:

```
<h1>Welcome to Bob's Auto Parts!</h1><p>What would you like to order today?</p>
```

i

```
<h1>Welcome             to Bob's
Auto Parts!</h1>
<p>What would you like
to order today?</p>
```

Oba bloka koda daju identičan rezultat zato što čitaču izgledaju isto. Preporučljivo je da u HTML kodu razumno koristite beline kao pomoćno sredstvo kojim ćete kôd stranica učiniti čitljivijim. Isto važi i za PHP. Nema potrebe da između PHP naredbi stoje beline, ali se kôd s njima lakše čita. Na primer, rezultati naredbi:

```
echo 'hello ';
echo 'world';
```

i

```
echo 'hello ';echo 'world';
```

identični su, ali se prva verzija lakše čita.

Komentari

Komentari su upravo to što im samo ime kaže i služe kao obaveštenja ljudima koji čitaju kôd. Komentari se mogu upotrebiti da objasne skript, da opišu ko ga je napisao, zašto je baš tako napisan, kada je poslednji put izmenjen itd. Komentare ćete nalaziti gotovo u svim PHP skriptovima, osim u onim najjednostavnijim.

U suštini, pošto PHP interpreter zanemaruje tekst u komentaru, može se smatrati da je ekvivalentan belini.

PHP podržava komentare u stilu jezika C, C++ i skriptova komandnog okruženja.

Sledi višeredni komentar u stilu jezika C, koji bi mogao da se pojavi na početku PHP skripta:

```
/* Pera Petrović
   Izmenjen: 10. aprila
   Skript obrađuje narudžbine.
*/
```

Višeredni komentari treba da počinju sa /* i završavaju se sa */. Isto kao u jeziku C, višeredni komentari ne mogu biti ugnežđeni jedni u druge.

Za komentare u jednom redu možete koristiti stil jezika C++:

```
echo '<p>Order processed.</p>'; // Štampaj narudžbenicu
```

ili stil skriptova komandnog okruženja:

```
echo '<p>Order processed.</p>'; # Štampaj narudžbenicu
```

U oba stila, sve što se nalazi iza simbola jednorednog komentara (# ili //) zanemaruje se do kraja reda ili do završne PHP oznake.

U narednom redu koda, tekst ispred završne PHP oznake, `ovo je komentar`, deo je komentara. Tekst iza završne PHP oznake, `a ovo nije`, obrađivaće se kao HTML kôd jer se nalazi iza završne PHP oznake:

```
// ovo je komentar ?> a ovo nije
```

Dodavanje dinamičkog sadržaja stranici

Sve što smo do sada uradili u PHP-u, mogli smo da uradimo i pomoću običnog HTML koda.

Glavni razlog upotrebe serverskog skript jezika jeste mogućnost prikazivanja dinamičkog sadržaja. To je vrlo važna primena, zato što će sadržaj koji se menja periodično ili prema potrebama korisnika, podstaći posetioce da se vraćaju na veb lokaciju. PHP omogućava laku izradu dinamičkog sadržaja.

Počnimo jednostavnim primerom. Zamenimo PHP u datoteci `processorder.php` sledećim kodom:

```
<?php
  echo "<p>Order processed at ";
  echo date('H:i, jS F Y');
  echo "</p>";
?>
```

Isto možete napisati i u jednom redu, koristeći operator nadovezivanja (.), kao

```
<?php
  echo "<p>Order processed at ".date('H:i, jS F Y')."</p>";
?>
```

U ovom kodu koristimo ugrađenu PHP funkciju `date()`, pomoću koje kupcu saopštavamo datum i vreme kada je njegova porudžbina obrađena. Prilikom svakog izvršavanja skripta biće ispisane različite vrednosti. Rezultat posle jednog izvršenja skripta prikazan je na slici 1.3.

II

MySQL

8 Projektovanje baze podataka za veb

9 Izrada baze podataka za veb

10 Rad sa MySQL-ovom bazom podataka

11 Pristup MySQL bazi podataka s veba pomoću PHP-a

12 Administriranje MySQL servera

13 Naprednije programiranje na MySQL-u

8

Projektovanje baze podataka za veb

Pošto smo opisali osnove programiranja na PHP-u, preći ćemo na integrisanje baze podataka u PHP skriptove. U poglavlju 2 naveli smo prednosti upotrebe baza podataka umesto običnih datoteka. Te prednosti su:

- Brži pristup podacima.
- Upotreba upita za jednostavno izdvajanje podataka koji ispunjavaju određene uslove.
- Ugrađeni su mehanizmi za rešavanje problema zbog istovremenog pristupa podacima tako da programer ne mora o tome da vodi računa.
- Obezbeđen nasumičan pristup podacima.
- Ugrađeni su sistemi ovlašćenja za pristup podacima.

Na primer, baza podataka brzo i jednostavno pruža odgovore na upite o tome odakle su kupci, koji proizvodi se najbolje prodaju i koji tipovi kupaca najviše troše. Informacije te vrste mogu vam pomoći da doterate veb lokaciju kako biste privukli što više korisnika, ali se veoma teško izdvajaju iz obične datoteke.

U ovom delu knjige opisaćemo bazu podataka MySQL. Pre nego što u sledećem poglavlju pređemo na specifičnosti MySQL-a, moramo razmotriti sledeće:

- Osnovne pojmove i terminologiju relacionih baza podataka
- Projektovanje baze podataka za veb
- Arhitekturu baze podataka za veb

U narednim poglavljima obrađujemo sledeće teme:

- Poglavlje 9, „Izrada baze podataka za veb“, govori o osnovnoj konfiguraciji potrebnoj za povezivanje MySQL-ove baze podataka na veb. Tu ćete naučiti da pravite korisničke naloge, baze podataka, tabele i indekse, a biće reči i o MySQL-ovim mašinama za baze podataka.
- Poglavlje 10, „Rad sa MySQL-ovom bazom podataka“, objašnjava kako se s komandne linije zadaju upiti, te dodaju, brišu i ažuriraju zapisi.

- Poglavlje 11, „Pristup MySQL bazi podataka s veba pomoću PHP-a“, objašnjava kako se PHP i MySQL povezuju tako da se preko veb interfejsa koristi i administrira baza podataka. Saznaćete da to možete raditi na dva načina: korišćenjem MySQL-ovog izvornog drajvera i korišćenjem interfejsa PDO (PHP Data Objects) nezavisnim od vrste baze podataka (engl. *database agnostic*).
- Poglavlje 12, „Naprednije tehnike administriranja MySQL servera“, detaljnije objašnjava kako se administrira MySQL; između ostalog, opisan je sistem prava i ovlašćenja, zaštita od neovlašćenog pristupa i optimizovanje MySQL-a.
- Poglavlje 13, „Naprednije tehnike programiranja u MySQL-u“, podrobnije opisuje mašine baze podataka – uključujući transakcije, tekstualno pretraživanje i uskladištene procedure.

Koncepti relacionih baza podataka

Relacione baze podataka – koje su najpopularnija vrsta baza podataka – zasnivaju se na teorijama relacione algebre. Da biste ih koristili, nije neophodno da poznajete pomenute teorije (mada ne može da vam škodi), ali treba da shvatite neke osnovne koncepte.

Tabele

Relacione baze podataka sastoje se od relacija (odnosa između podataka), koje se najčešće nazivaju tabelama. Tabela je upravo to što joj ime govori – tabela s podacima. Ako ste koristili program za tabelarna izračunavanja – koristili ste tabelu.

Slika 8.1 je primer tabele koja sadrži imena i adrese kupaca u knjižari Book-O-Rama.

KUPCI

ŠiraKupca	Ime	Adresa	Grad
1	Julija Savić	Banatska 25	Zrenjanin
2	Alen Vojnović	Bačka 47	Subotica
3	Milena Avramović	Raška 357	Kikinda

Slika 8.1 Podaci o kupcima knjižare Book-O-Rama čuvaju se u tabeli.

Tabela ima ime – Kupci (`Customers`), kolone u kojima su smeštene određene vrste podataka i redove koji opisuju pojedinačne kupce.

Kolone

Svaka kolona u tabeli ima jedinstveno ime i sadrži odgovarajući tip podataka. Na primer, u tabeli sa slike 8.1 kolona `CustomerID` (šifra kupca) celobrojna je vrednost, dok ostale kolone sadrže znakovne nizove. Kolone tabele se ponekad nazivaju *polja* ili *atributi* tabele.

Redovi

Svaki red u tabeli predstavlja po jednog kupca. Zbog tabelarnog formata, svaki red ima iste atribute. Redovi tabela se nazivaju i *zapisi* (engl. *records*) ili *torke* (engl. *tuples*).

Vrednosti

Svaki red se sastoji od skupa pojedinačnih vrednosti koje se nalaze u kolonama. Svaka vrednost mora imati tip podataka određen kolonom u kojoj se nalazi.

Ključevi

Neophodno je da postoji način nedvosmislene identifikacije svakog kupca. Imena osoba obično nisu pogodna za tu namenu jer može postojati više kupaca sa istim imenom i/ili prezimenom.

Kupca Juliju Savić iz tabele sa slike 8.1 možete da razlikujete od ostalih na nekoliko načina. Ona je najverovatnije jedina Julija Savić na navedenoj adresi, ali navoditi uvek ime, prezime i punu adresu previše je nezgrapno i previše podseća na otuđeni jezik birokratije. Osim toga, taj oblik zahteva upotrebu više od jedne kolone u tabeli.

Zato smo u ovom primeru (a to ćete verovatno i vi činiti u svojim aplikacijama), dodelili svakom kupcu jedinstven identifikacioni broj ili šifru kupca (`CustomerID`), po istom principu po kojem u banci dobijate jedinstven broj računa. To omogućava da se podaci lakše razlikuju u bazi podataka. Može se garantovati da će veštački dodeljen identifikacioni broj biti jedinstven, što je znatno ređe za stvarne podatke, čak i ako se kombinuju.

Identifikaciona kolona u tabeli naziva se *ključ* (engl. *key*) ili *primarni ključ* (engl. *primary key*). Ključ ne mora da se sastoji samo od jedne kolone. Ako, na primer, odlučimo da Juliju zovemo „Julija Savić, Banatska 25, Zrenjanin“, ključ će se sastojati od kolona Ime, Adresa i Grad, bez garancije da će biti jedinstven.

Baze podataka se obično sastoje od više tabela za čije povezivanje se koriste ključevi. Na slici 8.2, dodali smo bazi podataka još jednu tabelu. Druga tabela – Porudžbine (`Orders`) – sadrži porudžbine kupaca. Svaki red u tabeli `Orders` predstavlja jednu porudžbinu jednog kupca. Zahvaljujući šifri kupca, `CustomerID`, znamo čija je svaka porudžbina. Na primer, porudžbina broj 2 pripada kupcu čija je šifra 1. Ako pogledate tabelu `Customers`, videćete da je to Julija Savić.

KUPCI

BrojKupca	Ime	Adresa	Grad
1	Julija Savić	Banatska 25	Zrenjanin
2	Alen Vojnović	Bačka 47	Subotica
3	Milena Avramović	Raška 357	Kikinda

PORUDŽBINE

BrojPorudžbine	BrojKupca	Iznos	Datum
1	3	27,50	02-Apr-2007
2	1	12,99	15-Apr-2007
3	2	74,00	19-Apr-2007
4	3	6,99	01-Maj-2007

Slika 8.2 Svaka porudžbina u tabeli Porudžbine (Orders) povezana je sa određenim kupcem u tabeli Kupci (Customers).

U relacionoj bazi podataka, za takvu vrstu relacija koristi se izraz *spoljni ključ* (engl. *foreign key*). Kolona `CustomerID` primarni je ključ u tabeli `Customers`, ali kada postoji i u nekoj drugoj tabeli, kao što je tabela `Orders`, u toj tabeli je spoljni ključ.

Možda se pitate zbog čega smo odlučili da radimo s dve tabele, umesto da adresu kupca Julije Savić smestimo u tabelu `Orders`. O tome ćemo detaljnije govoriti u sledećem odeljku.

Šema baze podataka

Skup struktura svih tabela u bazi naziva se *šema* (engl. *schema*) baze podataka. Ona predstavlja „nacrt" ili „plan" baze podataka. Šema se sastoji od opisa tabela i njihovih kolona, tipova podataka u tim kolonama, primarnog ključa svake tabele i eventualnih spoljnih ključeva. Šema ne sadrži podatke, ali određene podatke možete prikazati uz šemu kao primere, ako želite da bolje objasnite čemu ti podaci služe. Šema se može prikazati u obliku neformalnih dijagrama koje koristimo u ovoj knjizi, zatim u obliku *dijagrama relacija i entiteta* (koji nije objašnjen u ovoj knjizi) i u tekstualnom obliku, na sledeći način:

```
Customers(CustomerID, Name, Address, City)
Orders(OrderID, CustomerID, Amount, Date)
```

Primarni ključevi su podvučeni, a spoljni su istaknuti kurzivom.

Relacije

Spoljni ključevi predstavljaju relacije (veze) između podataka u dve tabele. Na primer, veza između tabela `Orders` i `Customers` predstavlja odnos ili povezanost između reda u tabeli `Orders` i reda u tabeli `Customers`.

U relacionoj bazi podataka postoje tri osnovne vrste relacija. One se klasifikuju prema broju povezanih redova na svakoj strani relacije. Tipovi relacija su: „jedan prema jedan", „jedan prema više" i „više prema više".

Relacija tipa „jedan prema jedan" povezuje po jedan red u obe tabele. Ako biste, na primer, adrese izdvojili iz tabele `Customers` i stavili ih u zasebnu tabelu, odnos između te dve tabele bio bi „jedan prema jedan". Spoljni ključ bi mogao da povezuje red iz tabele `Adresa` s redom u tabeli `Customers` ili obrnuto (nije obavezno da postoje oba).

U relaciji tipa „jedan prema više", jedan red u jednoj tabeli povezan je s više redova u drugoj tabeli. U ovom primeru, to znači da jedan kupac (iz tabele `Customers`) može da ima više porudžbina (u tabeli `Orders`). U toj vrsti relacije, tabela na strani „više" povezana je preko spoljnog ključa s tabelom na strani „jedan". Na slici 8.2, tu vezu smo prikazali tako što smo kolonu za šifru kupca (`CustomerID`) dodali u tabelu `Customers`.

U relaciji tipa „više prema više", više redova u jednoj tabeli povezano je sa više redova u drugoj. Na primer, ako imate dve tabele, `Knjige` i `Autori`, može se desiti da su neku knjigu napisala dva autora, od kojih je svaki napisao i neke druge knjige – samostalno ili kao koautor. Takav tip veza obično se predstavlja pomoću dodatne tabele, tako da bi postojale tabele `Naslovi`, `Autori` i `Autor_naslov`. Treća tabela, `Autor_naslov`, sadržala bi samo parove povezanih spoljnih ključeva iz prve dve i označavala bi koji su autori napisali koje knjige.

Projektovanje baze podataka za rad na vebu

Nije uvek očigledno kada je potrebna nova tabela i šta bi trebalo da bude njen ključ. Postoji obilje informacija o dijagramima relacija i entiteta kao i o normalizaciji baza podataka – koje nisu tema ove knjige – ali u većini slučajeva biće dovoljno nekoliko osnovnih principa koje ćemo predstaviti. Pogledajmo ih u kontekstu knjižare Book-O-Rama.

Razmotrite objekte iz stvarnog sveta čije modele pravite

Pravljenje baze podataka podrazumeva izradu modela objekata iz stvarnog sveta i njihovih relacija, te prikupljanje podataka o tim objektima i njihovim relacijama.

Obično je za svaku klasu objekata čiji model pravite potrebna po jedna tabela. Razmislite o sledećem problemu: želite da evidentirate iste podatke o svim kupcima. Ako imate skup podataka koji čini određeni „oblik“, lako možete napraviti tabelu koja odgovara tim podacima.

U našem primeru, u knjižari Book-O-Rama, želimo da evidentiramo podatke o kupcima, knjigama i porudžbinama. Svaki kupac ima ime i adresu. Svaka porudžbina ima datum, ukupan iznos i naslove naručenih knjiga. Svaka knjiga ima jedinstven ISBN broj, autora, naslov i cenu.

Na osnovu navedenih podataka može se zaključiti da baza podataka treba da ima tri tabele: `Kupci (Customers)`, `Porudžbine (Orders)` i `Knjige (Books)`. Početna šema prikazana je na slici 8.3.

KUPCI

BrojKupca	Ime	Adresa	Grad
1	Julija Savić	Banatska 25	Zrenjanin
2	Alen Vojnović	Bačka 47	Subotica
3	Milena Avramović	Raška 357	Kikinda

PORUDŽBINE

BrojPorudžbine	BrojKupca	Iznos	Datum
1	3	27,50	02-Apr-2007
2	1	12,99	15-Apr-2007
3	2	74,00	19-Apr-2007
4	3	6,99	01-Maj-2007

KNJIGE

ISBN	Ime autora	Naslov	Cena
0-672-31697-8	Michael Morgan	Java 2 for Professional Developers	34,99
0-672-31745-1	Thomas Down	Installing GNU/Linux	24,99
0-672-31509-2	Pruitt.et al.	Teach Yourself GIMP in 24 Hours	24,99

Slika 8.3 Početna šema se sastoji od tabela Kupci (`Customers`), Porudžbine (`Orders`) i Knjige (`Books`).

Za sada se na osnovu modela ne može reći koje knjige pripadaju kojoj porudžbini. To ćemo rešiti kasnije.

Izbegavajte suvišne podatke

U ovom poglavlju smo se već jednom zapitali: zašto adresu Julije Savić ne bismo čuvali u tabeli `Orders`?

Ako Julija kupi knjige više puta (čemu se nadamo), svaki put bismo morali da ponovo upišemo njene podatke, pa bi tabela `Orders` mogla da izgleda kao na slici 8.4.

BrojKupca	Iznos	Datum	BrojKupca	Ime	Adresa	Grad
12	199,50	25-Apr-2007	1	Julija Savić	Banatska 25	Zrenjanin
13	43,00	29-Apr-2007	1	Julija Savić	Banatska 25	Zrenjanin
14	15,99	30-Apr-2007	1	Julija Savić	Banatska 25	Zrenjanin
15	23,75	01-Maj-2007	1	Julija Savić	Banatska 25	Zrenjanin

Slika 8.4 Baza podataka u kojoj se čuvaju suvišni podaci zauzima više prostora i može da bude uzrok neusklađenosti podataka.

Takvo rešenje prouzrokuje dva osnovna problema:

- Prostor se nepotrebno rasipa. Zašto nečije podatke upisivati triput ako je dovoljno samo jednom?
- Mogu se pojaviti *neusklađenosti pri ažuriranju* (engl. *update anomalies*) – izmene podataka koje mogu dovesti do njihove neusklađenosti. Narušava se integritet podataka i više se ne zna koji su podaci tačni a koji nisu. To obično dovodi do gubljenja podataka.

Treba izbegavati tri vrste neusklađenosti: pri izmenama, unošenju i brisanju podataka.

Ako se Julija preseli na novu adresu pre nego što joj dostavite knjige koje je kupila, njenu novu adresu treba uneti na tri mesta umesto na jedno, što znači triput više posla. Tu činjenicu lako možete prevideti i uneti njenu novu adresu na samo jedno mesto, usled čega podaci u bazi postaju neusklađeni (što je veoma loše). Ta vrsta problema zove se *neusklađenost izmena* (engl. *modification anomalies*) zato što nastaju pri pokušaju izmene baze podataka.

Pošto vas sadašnje rešenje prisiljava da podatke o Juliji ponovo unosite kad god dobijete novu porudžbinu od nje, svaki put morate proveriti da li su podaci koje ste uneli u skladu s postojećim redovima u tabeli. Ako to ne proverite, moglo bi se desiti da imate dva reda s različitim podacima o Juliji. Na primer, jedan red bi mogao da pokazuje da Julija živi u Zrenjaninu, a u drugom ime naselja može biti ZrenjaninMelenci. To se zove *neusklađenost pri unošenju* (engl. *insertion anomaly*) zato što nastaje pri unošenju podataka.

Treću vrstu čine *neusklađenosti pri brisanju* (engl. *deletion anomaly*) i nastaju prilikom brisanja redova u bazi podataka. Pretpostavimo da smo kupcu dostavili naručene knjige i da smo nakon toga tu porudžbinu obrisali iz baze podataka, tj. iz tabele `Orders`. To znači da više nemamo adresu kupca Julija i ne možemo da joj šaljemo obaveštenja o sniženjima ili specijalnim ponudama. Osim toga, kada ona sledeći put nešto naruči, moraćemo ponovo da unosimo njenu adresu i ostale podatke.

Zbog svega navedenog, bazu podataka treba projektovati tako da se pomenute neusklađenosti ne pojavljuju.

Kolone treba da sadrže proste (nedeljive) podatke

To znači da svaki atribut u svakom redu treba da sadrži samo podatak koji se ne može dalje razbiti na još prostije činioce (engl. *atomic column values*). Pretpostavimo da treba da utvrdite koje knjige pripadaju kojoj porudžbini. To možete saznati na nekoliko načina.

Jedno rešenje bi bilo da tabeli `Orders` dodate kolonu u kojoj ćete navesti sve naručene knjige, kao na slici 8.5.

PORUDŽBINE

BrojPorudžbine	BrojKupca	Iznos	Datum	Naručene knjige
1	3	27,50	02-Apr-2007	0-672-31697-8
2	1	12,99	15-Apr-2007	0-672-31745-1. 0-672-31509-2
3	2	74,00	19-Apr-2007	0-672-31697-8
4	3	6,99	01-Maj-2007	0-672-31745-1. 0-672-31509-2. 0-672-31697-8

Slika 8.5 U ovakvom idejnom rešenju, atribut `Books Ordered` (Naručene knjige) u svakom redu sadrži više vrednosti.

To rešenje nije dobro iz više razloga. Unutar jedne kolone ugnežđujemo zapravo celu jednu novu tabelu – onu koja povezuje porudžbine s knjigama. Ako kolonu definišete tako da sadrži kombinovane podatke, teško ćete naći odgovor na pitanje tipa „koliko je prodato primeraka knjige *Java 2 for Professional Developers*?“. Sistem više ne može da potraži odgovor tako što će prebrojati polja koja ispunjavaju uslov, već mora pojedinačno da analizira svaku vrednost atributa kako bi utvrdio da li sadrži traženi naslov.

Umesto da pravite tabelu unutar tabele, trebalo bi da napravite novu tabelu. Ta tabela se zove Stavke_porudžbine (`Order_Items`) i prikazana je na slici 8.6.

STAVKE_PORUDŽBINE

BrojPorudžbine	ISBN	Količina
1	0-672-31697-8	1
2	0-672-31745-1	2
2	0-672-31509-2	1
3	0-672-31697-8	1
4	0-672-31745-1	1
4	0-672-31509-2	2
4	0-672-31697-8	1

Slika 8.6 Ovakvo idejno rešenje olakšava utvrđivanje broja prodatih primeraka određenog naslova.

Prikazana tabela povezuje tabele `Orders` i `Books`. To je uobičajen tip tabele kada između dva objekta postoji relacija „više prema više“. U ovom slučaju, jedna porudžbina može da sadrži više knjiga, kao što i više osoba može da naruči istu knjigu.

Kada rešavate problem za koji su zaista potrebne složenije vrednosti u kolonama, trebalo bi da – umesto relacione baze podataka – koristite bazu koja je projektovana baš za takav tip podataka. Takve baze podataka nisu relacione, često se nazivaju NoSQL baze podataka ili *skladišta podataka* (engl. *datastores*) i nisu obrađene u ovoj knjizi.

Izaberite logične ključeve

Vodite računa o tome da ključevi koje ste izabrali budu jedinstveni. U ovom slučaju napravili smo poseban ključ za kupce (`CustomerID`) i za porudžbine (`OrderID`), zato što ti objekti u stvarnosti ne moraju imati identifikatore koji će garantovano biti jedinstveni. Za knjige ne morate praviti veštački jedinstven identifikator jer takav već postoji. To je ISBN broj. Tabeli `Order_Items` možete dodati još jedan ključ – ako želite – ali će kombinacija atributa `OrderID` i `ISBN` biti jedinstvena, pod uslovom da se svi primerci iste knjige u porudžbini upisuju u isti red. Zbog toga tabela `Order_Items` ima kolonu `Količina` (`Quantity`).

Mislite o tome šta hoćete da saznate iz baze podataka

Jedno od takvih pitanja moglo bi da bude: koje se knjige iz knjižare najbolje prodaju? Da bi mogla da odgovori na pitanja, baza podataka mora da sadrži sve neophodne podatke, a između tabela moraju da postoje odgovarajuće veze.

Izbegavajte strukture tabela s mnogo praznih atributa

Kada biste u bazi podataka želeli da evidentirate i prikaze knjiga, to biste mogli uraditi na barem dva načina (slika 8.7).

KNJIGE

ISBN	Ime autora	Naslov	Cena	Prikaz
0-672-31697-8	Michael Morgan	Java 2 for Professional Developers	34,99	
0-672-31745-1	Thomas Down	Installing GNU/Linux	24,99	
0-672-31509-2	Pruitt et al.	Teach Yourself GIMP in 24 Hours	24,99	

PRIKAZI_KNJIGA

ISBN	Prikaz

Slika 8.7 Prikaze knjiga možete uključiti u bazu tako što ćete dodati odgovarajuću kolonu tabeli Knjige (`Books`) ili dodati novu tabelu samo za tu namenu.

Prvo rešenje je da tabeli `Books` dodate kolonu `Prikaz` (`Review`). To podrazumeva da za svaku knjigu treba popuniti i polje `Review`. Ako u bazi postoji mnogo knjiga i ako recenzent ne planira da baš za svaku napiše prikaz, taj atribut neće imati vrednost u mnogim redovima tabele, tj. polje `Review` imaće vrednost *null*.

Nije dobro da baza podataka sadrži mnogo vrednosti Null jer se tako rasipa prostor na disku i nastaju problemi pri izračunavanju ukupnih iznosa i upotrebi drugih funkcija u numeričkim kolonama. Kada u tabeli vidi prazno polje, korisnik ne zna razlog – da li atribut nije relevantan, da li postoji greška u bazi ili podaci još nisu uneti.

Taj problem se najčešće izbegava tako što se koristi drugačija struktura baze podataka. U ovom slučaju, bolje je drugo rešenje sa slike 8.7. Tu su u tabeli Prikazi_knjiga (`Book_Reviews`) navedene samo knjige za koje postoje prikazi i tekstovi prikaza.

Imajte u vidu da to rešenje polazi od pretpostavke da postoji samo jedan recenzent koji je saradnik knjižare, tj. da između knjiga i prikaza postoji veza tipa „jedan prema jedan". Ako želite da više osoba piše prikaze za istu knjigu, to bi bila veza tipa „jedan prema više" i morali biste da se opredelite za drugo rešenje. Osim toga, ako za svaku knjigu može postojati samo po jedan prikaz, ISBN broj možete iskoristiti kao primarni ključ u tabeli `Book_Reviews`. Ukoliko dozvoljavate da postoji više prikaza po knjizi, moraćete da uvedete jedinstven identifikator svake od njih.

Kratak pregled vrsta tabela

Šema baze podataka najčešće se svodi na dve vrste tabela:

- Osnovne tabele, koje predstavljaju stvarne objekte. Te tabele mogu da sadrže i ključeve koji ih povezuju s drugim takvim tabelama, s kojima su u vezama tipa „jedan prema jedan" ili „jedan prema više". Na primer, jedan kupac može da ima više porudžbina, ali jedna porudžbina može da pripada samo jednom kupcu. Zato se u porudžbinu stavlja odrednica koja je povezuje sa odgovarajućim kupcem.
- Spojne tabele (engl. *linking tables*), pomoću kojih se uspostavljaju veze tipa „više prema više" između dve osnovne tabele – kao što je relacija između tabela `Orders` i `Books`. Te tabele često služe za obavljanje raznih vrsta transakcija iz stvarnog sveta.

Arhitektura baze podataka za veb

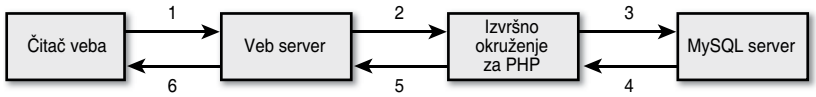
Pošto smo objasnili unutrašnju strukturu baze podataka, preći ćemo na spoljnu arhitekturu sistema baze podataka za veb i reći nešto o metodologiji njegovog razvoja.

Vrlo sažet opis principa rada veb servera prikazan je na slici 8.8. Sistem se sastoji od dva objekta: čitača veba i veb servera, između kojih mora da postoji komunikaciona veza. Čitač veba šalje zahtev serveru, koji potom vraća odgovor. Takva arhitektura sasvim odgovara za server koji isporučuje isključivo statičke stranice. Međutim, arhitektura veb lokacije čiji se sadržaj učitava iz baze podataka, donekle je složenija.



Slika 8.8 Klijentsko-serverski odnos između čitača veba i veb servera zahteva komunikaciju između njih.

Aplikacije baza podataka za veb koje ćemo praviti u ovoj knjizi slede opštu strukturu prikazanu na slici 8.9. Trebalo bi da vam ta struktura bude uglavnom poznata.



Slika 8.9 Osnovna arhitektura baze podataka za veb sastoji se od čitača veba, veb servera, mašine za izvršavanje skriptova i servera baze podataka.

Tipična transakcija u bazi podataka na vebu sastoji se od koraka numerisanih na slici 8.9. Objasnićemo ih na primeru knjižare Book-O-Rama.

1. Korisnikov čitač veba šalje HTTP zahtev za određenu veb stranicu. Na primer, korisnik je HTML obrascem zatražio da vidi sve knjige određenog autora koje se prodaju u knjižari. Stranica koja prikazuje rezultate pretraživanja zove se `results.php`.
2. Veb server prima zahtev za prikazivanje stranice `results.php`, učitava datoteku u kojoj se ona nalazi i prosleđuje sadržaj te datoteke PHP-ovom izvršnom okruženju na obradu.
3. PHP-ovo izvršno okruženje počinje sintaksnu analizu skripta. Skript sadrži naredbe za uspostavljanje veze s bazom podataka i izvršenje upita koji traži sve knjige određenog autora. PHP otvara vezu sa MySQL serverom i šalje mu odgovarajući upit.
4. MySQL server prima upit, obrađuje ga i zatim rezultat – listu traženih knjiga – vraća PHP-ovom izvršnom okruženju.
5. PHP-ovo izvršno okruženje za PHP završava obradu skripta, što obično podrazumeva i formatiranje rezultata upita u HTML kôd. Zatim skript prosleđuje veb serveru.
6. Dobijeni HTML kôd veb server prosleđuje čitaču veba koji je poslao zahtev, što korisniku omogućava da vidi spisak traženih knjiga.

Postupak je u suštini isti, bez obzira na to koju mašinu za izvršavanje skriptova ili server baze podataka koristite. Ponekad, softver za veb server, PHP-ovo izvršno okruženje i server baze podataka rade na istom računaru. Nije neuobičajeno ni da server baze podataka radi na nekom drugom računaru zbog bezbednosnih razloga, većeg kapaciteta i boljeg raspoređivanja opterećenosti. Iz razvojne perspektive, druga opcija se gotovo ne razlikuje od prve, ali može da pruži značajno bolje performanse.

Ako aplikacija počne da raste i da postaje sve složenija, moraćete je raspodeliti na više *slojeva*; to će najčešće biti sloj baze podataka koji direktno komunicira s MySQL-om, zatim sloj poslovne logike koji čini srž aplikacije i sloj za prezentaciju podataka koji se stara o generisanju izlaznog HTML koda. Međutim, osnovna arhitektura, prikazana na slici 8.9, i dalje važi; jedina razlika je to što PHP komponenta postaje složenija.

Dodatna literatura

U ovom poglavlju naveli smo neke smernice za projektovanje relacionih baza podataka. Ako vas zanima teorija na kojoj se zasnivaju relacione baze podataka, više ćete saznati iz knjiga neprikosnovenih stručnjaka iz te oblasti, kao što je C. J. Date. Imajte, međutim, u vidu da taj materijal može biti isuviše teorijski i – kao takav – ne mnogo značajan za komercijalnog veb programera. Prosečne baze podataka za veb obično nisu toliko složene.

Šta sledi

U sledećem poglavlju počinjemo izradu MySQL baze podataka. Prvo ćete naučiti kako da napravite takvu bazu podataka, kako da joj postavljate upite i kako da joj upite postavljate iz PHP-a.

15

Izrada bezbednih veb aplikacija

U ovom poglavlju nastavljamo razmatranje zaštite veb aplikacija i to tako što prelazimo na opsežniju temu zaštite cele veb aplikacije. Pošto svaki pojedinačni deo veb aplikacija moramo zaštititi od mogućih zloupotreba (nenamernih ili namernih), potrebno je da definišemo određene strategije koje će omogućiti da naše aplikacije uvek budu zaštićene.

U ovom poglavlju razmotrićemo sledeće ključne teme:

- Strategije zaštite
- Zaštita programskog koda
- Zaštita veb servera i PHP-a
- Zaštita servera baze podataka
- Zaštita mreže
- Planiranje za slučaj katastrofe.

Strategije zaštite

Jedna od najkorisnijih mogućnosti interneta – otvorenost i mogućnost pristupanja svih računara svima ostalima – pokazuje se i kao jedna od najvećih glavobolja s kojom vi, kao autor veb aplikacije, morate da se izborite. Uz toliki broj računara na svetu, verovatno je da će neki među njihovim korisnicima imati sve samo ne plemenite namere. Kad pomislimo na sve opasnosti koje vrebaju na internetu, može biti zastrašujuće razmišljati o tome da se u globalnu mrežu postavi aplikacija koja radi s potencijalno poverljivim podacima kao što su brojevi kreditnih kartica, podaci o bankovnim računima ili zdravstveni kartoni. Ali, firme moraju da posluju, a mi kao autori aplikacija moramo gledati dalje od zaštite samo onih delova naših aplikacija koji se direktno odnose na e-trgovinu i moramo razviti način za planiranje i realizovanje zaštite. Suština je to da nađemo ravnotežu između potrebe da se zaštitimo i potrebe da se poslovanje nesmetano obavlja i da aplikacija radi bez problema.

Počnite s pravim načinom razmišljanja

Zaštita aplikacije nije tek jedna od mogućnosti. Kada pišete veb aplikaciju i razmatrate listu mogućnosti koje želite da ugradite, zaštita nije nešto što „onako, za svaki slučaj“ stavljate na listu i određujete jednog programera da na tome radi nekoliko dana. To mora

neprekidno biti sastavni deo osnovne strukture aplikacije i posao koji nikad ne prestaje, čak ni pošto aplikaciju pustite u redovan rad i smanjite obim razvoja, ili ga sasvim prekinete.

Ako od samog početka razmišljamo o zaštiti i planiranju aplikacije, kao i o načinima na koje bi se sistem mogao zloupotrebiti ili na koje bi napadači mogli pokušati da provale u njega, programski kôd možemo projektovati tako da se smanji verovatnoća pojavljivanja tih problema. To će nas takođe poštedeti potrebe da kasnije sve ponovo projektujemo iz početka kada najzad počnemo da se bavimo problemom zaštite (kada ćemo gotovo sigurno prevideti više drugih potencijalnih problema).

Ravnoteža između zaštite i upotrebljivosti

Jedna od najvećih briga tokom projektovanja sistema jesu lozinke korisnika. Korisnici će često birati lozinke koje nije posebno teško pogoditi pomoću softvera, naročito ako upotrebljavaju uobičajene reči iz govornog jezika. Potreban nam je način da smanjimo rizik pogađanja korisnikove lozinke i provale u sistem na taj način.

Jedno od mogućih rešenja je zahtevati da korisnik prođe kroz četiri prijavna dijaloga, svaki s drugom lozinkom. Možemo takođe tražiti da korisnik menja te četiri lozinke barem jednom mesečno i obezbediti da nikad ne ponovi lozinku koju je već koristio. Mogli bismo zahtevati da lozinke budu veoma dugačke i da sadrže različite vrste znakova (na primer, velika slova, mala slova, znakove interpunkcije i brojeve). Tako bi sistem postao znatno zaštićeniji, a zlonamerni hakeri bi morali da posvete mnogo više vremena zaobilaženju postupka prijavljivanja i provale u sistem.

Nažalost, takav sistem bi bio toliko zaštićen da ga niko ne bi koristio – pre ili kasnije bi korisnik pomislio da nije vredno truda. Znači, podjednako je važno da brinemo o zaštiti, koliko i o njenom uticaju na upotrebljivost sistema. Lako upotrebljiv i slabo zaštićen sistem može se pokazati kao privlačan za korisnika, ali će posledice biti i veća verovatnoća pojave bezbednosnih problema i mogući prekidi poslovanja. Slično tome, sistem čija je zaštita toliko robusna da je sistem na granici upotrebljivosti, privući će mali broj korisnika i takođe će negativno uticati na poslovanje firme.

Kao projektanti veb aplikacija, moramo uvek tražiti načine da poboljšamo zaštitu, ali da pritom ne umanjimo preterano upotrebljivost sistema. Kao i u oblasti korisničkog interfejsa, budući da ne postoje stroga i univerzalna pravila koja bi trebalo slediti, moramo se osloniti na sopstvene procene, testove upotrebljivosti i probne grupe da bismo utvrdili kako korisnici reaguju na naše prototipove i projekte.

Nadgledanje zaštite

Kada završimo razvoj veb aplikacije i postavimo je na servere kako bi ljudi počeli da je koriste, posao još nije završen. Deo zaštite aplikacije je i nadgledanje sistema tokom rada, kroz pregledanje dnevnika i drugih datoteka da bismo videli koliko dobro sistem radi i kako se koristi. Samo pažljivim praćenjem rada sistema (ili pisanjem alatki koje će obavljati deo tog posla umesto nas), možemo utvrditi postoje li bezbednosni problemi i otkriti oblasti kojima moramo posvetiti vreme da bismo razvili bezbednija rešenja.

Zaštita je, nažalost, neprekidna bitka i u izvesnom hiperboličnom smislu, bitka u kojoj ne možemo nikad pobediti. Neprekidan oprez, poboljšavanje sistema i brzo reagovanje na svaki problem čine cenu koja se mora platiti da bi veb aplikacija glatko radila.

Osnovni pristup problemu

Da bismo došli do najpotpunijeg rešenja zaštite, uz razumnu količinu truda i vremena, u ovoj knjizi ćemo opisati dvodelni pristup problemu zaštite. Prvi deo se nadovezuje na ono o čemu je dosad već bilo reči: kako planirati zaštitu aplikacije i kako u nju ugraditi mogućnosti koje će očuvati tu zaštitu. Kada bismo bili takvi da na sve lepimo etikete, takav pristup bismo mogli nazvati *pristup odozgo nadole*.

Nasuprot tome, drugi deo naseg pristupa problemu zastite mogli bismo nazvati *pristup odozdo nagore*. O tome će biti reči u ovom poglavlju. U toj fazi razmotrićemo pojedinačne komponente aplikacije, kao što su server baze podataka, sam server i mreža u kojoj se on nalazi. Pokušaćemo da ne samo rad s tim komponetama bude zaštićen, nego i da njihovo instaliranje i podešavanje bude zaštićeno. Pošto se mnogi proizvodi instaliraju s podrazumevanim konfiguracijama koje ostaju otvorene za napade, korisno je da saznamo gde su „rupe“ i da ih zatvorimo.

Zaštita programskog koda

Da biste se zaista pobrinuli za zaštitu svog koda, morate razmišljati na nivou pojedinačnih komponenata, tj. ispitati svaku komponentu pojedinačno i razmisliti o tome kako poboljšati njenu zaštitu. U ovom odeljku počećemo razmatranje onoga što možemo uraditi da bismo zaštitili svoj programski kôd. Mada ne možemo da vam pokažemo sve što biste možda mogli uraditi kako biste se zaštitili od svih mogućih bezbednosnih pretnji (tim temama posvećene su brojne knjige), možemo barem da navedemo nekoliko opštih smernica i usmerimo vas i dobrom pravcu.

Filtriranje podataka koje korisnik šalje

Jedna od najvažnijih stvari koje možemo uraditi u svojim veb aplikacijama da bismo ih bolje zaštitili jeste da *filtriramo sve podatke koje korisnik šalje*.

Projektanti aplikacija moraju da filtriraju sve podatke koji dolaze iz spoljnih izvora. To ne znači da bi trebalo projektovati sistem s pretpostavkom da su svi naši korisnici prevaranti. Mi i dalje želimo da se oni osećaju dobrodošli i da ih podstičemo da koriste našu veb aplikaciju. Samo želimo da budemo spremni za svaki pokušaj zloupotrebe našeg sistema.

Ako obezbedimo efikasno filtriranje ulaznih podataka, znatno ćemo umanjiti broj spoljnih pretnji i poboljšati robusnost sistema. Čak i ukoliko smo sasvim ubeđeni da možemo verovati korisnicima, ne možemo biti sigurni da oni nemaju određenu vrstu špijunskog ili drugog softvera koji menja postojeće ili šalje nove zahteve našem serveru. Znači, nikada ne treba verovati korisniku.

Zbog važnosti filtriranja podataka koje dobijamo od spoljnih korisnika, trebalo bi da razmotrimo načine na koje se to može uraditi.

Proveravanje da li smo zaista dobili jednu od očekivanih vrednosti

U nekim slučajevima ponudićemo korisniku skup vrednosti između kojih može da bira, za podatke kao što su način dostave (običnom poštom, preporučeno, kao specijalan paket), poštanski broj ili grad itd. A sada zamislite da imamo sledeći jednostavan obrazac, prikazan u listingu 15.1:

Listing 15.1 **simple_form.html – jednostavan obrazac**

```
<!DOCTYPE html>
<html>
<head>
   <title>What be ye laddie?</title>
</head>
<body>
<h1>What be ye laddie?</h1>

<form action="submit_form.php" method="post">

<p>
<input type="radio" name="gender" id="gender_m" value="male" />
   <label for="gender_m">male</label><br/>

<input type="radio" name="gender" id="gender_f" value="female" />
   <label for="gender_f">female</label><br/>

<input type="radio" name="gender" id="gender_o" value="other" />
   <label for="gender_o">other</label><br/>
</p>

<button type="submit" name="submit">Submit Form</button>
</form>

</body>
</html>
```

Taj obrazac može biti nalik onome na slici 15.1. Na tom obrascu pretpostavili bismo da – kad god ispitamo vrednost promenljive `$_POST['gender']` na stranici *submit_form.php* – možemo dobiti samo jednu od tri moguće vrednosti: `'male'`, `'female'` ili `'other'` – što je potpuno pogrešno.



Slika 15.1 Jednostavan obrazac za unošenje pola osobe.

Kao što je već rečeno, veb radi pomoću HTTP-a, jednostavnog tekstualnog protokola. Navedeni obrazac bio bi poslat na naš server u obliku tekstualne poruke sa strukturom nalik sledećoj:

```
POST /submit_form.php HTTP/1.1
Host: www.yourdomain.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:40.0) Gecko/20100101 Firefox/40.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 11
gender=male
```

Međutim, ne postoji baš ništa što bi nekog sprečilo da uspostavi vezu s našim serverom i šalje putem obrasca koje god hoće vrednosti. Prema tome, neko može da nam pošalje i sledeće:

```
POST /submit_form.php HTTP/1.1
Host: www.yourdomain.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:40.0) Gecko/20100101 Firefox/40.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 22
gender=I+like+cookies.
```

Ako napišemo sledeći kôd:

```
<?php
echo "<h1>
      The user's gender is: ".$_POST['gender']. ".
      </h1>";
?>
```

mogli bismo kasnije dobiti zbunjujuće rezultate. Znatno bolja strategija je da proverimo da li je poslata vrednost jedna od očekivanih/dozvoljenih vrednosti, kao u listingu 15.2:

Listing 15.2 **submit_form.php – provera podataka unetih u obrazac**

```
<?php
switch ($_POST['gender']) {
   case 'male':
   case 'female':
   case 'other':

      echo "<h1>Congratulations!<br/>
           You are: ".$_POST['gender']. ".</h1>";
   break;

   default:

      echo "<h1><span style=\"color:  red;\">WARNING:</span><br/>
           Invalid input value specified.</h1>";
   break;
}
?>
```

Sada je programski kôd nešto složeniji – sadrži listu prihvatljivih unosa – ali barem možemo biti sigurni da uvek dobijamo ispravne vrednosti, a to će postati znatno važnije kada budemo obrađivali finansijske vrednosti, koje su svakako poverljivije od pôla korisnika. Pravilo glasi: ne možemo nikad poći od pretpostavke da će vrednost koju dobijemo preko obrasca biti unutar skupa očekivanih vrednosti – to moramo prvo proveriti.

Filtrirajte čak i osnovne vrednosti

Elementima na HTML obrascu nije pridružen tip podataka i većina njih se prosleđuju serveru u obliku znakovnih nizova (koji predstavljaju podatke kao što su datumi, vremena ili brojevi). To znači sledeće: ako imate numeričko polje, ne možete pretpostaviti, niti očekivati da je podatak u polju zaista broj. Čak i u okruženjima gde se posebno moćnim programskim kodom na klijentskoj strani može pokušati da se obezbedi unošenje vrednosti samo određene vrste, nema garancije da te vrednosti neće nikad biti direktno poslate serveru, kao što smo videli u prethodnom odeljku.

Jednostavan način da obezbedite da vrednost bude određenog tipa jeste da je preslikate ili pretvorite u taj tip i da zatim radite s rezultujućom vrednošću, na sledeći način:

```
$broj_nocenja = (int)$_POST['br_nocenja'];
if ($broj_nocenja == 0)
{
  echo "GREŠKA: Pogrešan broj noćenja!";
  exit;
}
```

Ako korisnik unosi datum u nekom lokalnom formatu, kao što je *mm/dd/gg* za korisnike iz SAD, pomoću PHP-ove funkcije `checkdate()` možemo napisati kôd koji proverava da li imamo ispravan datum. Ta funkcija preuzima vrednosti *meseca*, *dana* i *godine* (dvocifrena godina) i pokazuje da li njihova kombinacija čini ispravan datum:

```
$mmddyy = explode($_POST['datum_polaska'], '/');
if (count($mmddyy) != 3)
{
  echo "GREŠKA: Zadali ste neispravan datum!";
  exit;
}
// obrada godina kao sto su 02 ili 95
if ((int)$mmddyy[2] < 100)
{
  if ((int)$mmddyy[2] > 50)
    $mmddyy[2] = (int)$mmddyy[2] + 1900;
  else if ((int)$mmddyy[2] >= 0)
    $mmddyy[2] = (int)$mmddyy[2] + 2000;
  // inače je datum < 0, što će otkriti funkcija checkdate()
}
if  (!checkdate($mmddyy[0],  $mmddyy[1],  $mmddyy[2]))
{
  echo "GREŠKA: Zadali ste neispravan datum!";
  exit;
}
```

Ako ulazne podatke filtriramo i proveravamo njihovu ispravnost, ne samo što ćemo obezbediti otkrivanje očekivanih grešaka (npr. da li je datum početka leta na avionskoj karti ispravan datum), što ionako moramo obezbediti, nego time možemo i poboljšati zaštitu celog sistema.

Priprema bezbednih tekstova SQL upita

Još jedan slučaj kada treba da obradimo znakovne nizove kako bismo ih učinili bezbednim jeste sprečavanje napada koji nastaju umetanjem SQL naredbi (engl. *SQL injection attacks*) i koje smo pomenuli kada smo razmatrali upotrebu MySQL-a u PHP-u. U tim napadima, zlonamerni korisnik pokušava da iskoristi propuste u loše zaštićenom programskom kodu i ovlašćenjima korisnika da bi izvršio dodatan SQL kôd. Ako ne obratimo pažnju, ime korisnika kao što je sledeće

```
mala_maca; DELETE FROM kupci;
```

može nam prouzrokovati priličan problem.

Tu vrstu narušavanja zaštite možete sprečiti paralelnom primenom dve osnovne metode:

- Koristite parametrizovane upite kad god je moguće. U takvim upitima, SQL kôd je razdvojen od podataka. Doduše, to neće pomoći za imena kolona i tabela, jer se ona ne mogu proslediti pomoću parametrizovanog upita. Međutim, pošto unapred znate svoju šemu, možete primeniti metodu bele liste sa odgovarajućim vrednostima.
- Proverite da li su svi ulazni podaci u skladu sa onim što očekujete da budu. Ako je pravilo da se imena korisnika sastoje od najviše 50 znakova i sadrže isključivo slova i brojeve, možemo biti sigurni da `"; DELETE FROM kupci"` na kraju imena verovatno nije nešto što bismo dozvolili. Ukoliko napišemo PHP kôd kojim se obezbeđuje da su ulazni podaci u skladu sa ispravnim i dozvoljenim vrednostima, pre nego što ih pošaljemo serveru baze podataka, znači da ćemo moći da dobijemo znatno razumljiviji opis greške nego što bi nam baza podataka prikazala (kada bi i proveravala tu vrstu stvari) i smanjićemo rizike.

Proširenje *mysqli* pruža dodatnu bezbednosnu prednost jer se dozvoljava izvršavanje samo po jednog upita pomoću metoda `mysqli_query` ili `mysqli::query`. Da biste izvršili više upita, morate upotrebiti metodu `mysqli_multi_query` ili `mysqli::multi_query`, čime se sprečava izvršavanje dodatnih, potencijalno štetnih naredbi ili upita.

Priprema izlaznih podataka

Gotovo podjednako važno kao filtriranje ulaznih podataka jeste *pripremanje izlaznih podataka*. Pošto u sistemu pripremimo podatke namenjene korisniku, od ključne je važnosti da oni budu takvi da ne mogu načiniti nikakvu štetu, niti biti uzrok neželjenih posledica. To postižemo pomoću nekoliko ključnih funkcija kojima se obezbeđuje da klijentski čitač tumači podatke koje mu šaljemo isključivo kao tekst koji treba da prikaže, a ne kao programski kôd koji treba da izvrši.

Mnoge veb aplikacije preuzimaju podatke koje je korisnik uneo a zatim ih prikazuju na određenoj veb stranici. Odlični primeri toga su stranice na kojima korisnik može da dodaje komentare na objavljen tekst, ili stranice sistema za razmenu poruka između korisnika (forumi). U takvim slučajevima moramo sprečiti korisnike da umeću zlonameran HMTL kôd u tekst.

Jedan od najlakših načina da to postignemo jeste pomoću funkcija `htmlspecialchars` ili `htmlentities`. Te funkcije preuzimaju određene znakove u ulaznom tekstu i pretvaraju ih u HTML *entitete*. Ukratko, HTML entitet je specijalan niz znakova, koji počinje znakom

ampersend (&), a svrha mu je da predstavi određeni specijalan znak koji se ne može lako predstaviti u HTML kodu. Iza znaka ampersend sledi ime entiteta i završni znak tačka i zarez (;). Entitet može biti i u obliku ASCII koda tastera, zadatog kao znak # kojem sledi broj, kao što je / što predstavlja kosu crtu (/).

Na primer, budući da su svi HTML elementi omeđeni znakovima < i >, može biti problem kako te znakove predstaviti u izlaznom sadržaju (jer će ih čitač veba standardno tumačiti kao graničnike HTML elemenata, a ne kao tekst). Da bismo rešili taj problem, koristimo entitete < i >. Slično tome, ako u HTML tekstu želimo da prikažemo znak ampersand, upotrebićemo entitet &. Polunavodnike i navodnike predstavlja entitet ', odnosno ". HTML klijent (čitač veba) pretvara entitete u običan tekst koji prikazuje i ne tumači ih kao delove HTML elemenata.

Razlika između funkcija `htmlspecialchars` i `htmlentities` jeste sledeća: prva funkcija zamenjuje samo znakove &, < i >, a mogu joj se zadati i opcioni parametri za polunavodnike i navodnike. Nasuprot tome, druga funkcija zamenjuje odgovarajućim entitetom sve što se može predstaviti tim imenovanim entitetom. Primeri takvih entiteta su simbol za autorska prava ©, koji se predstavlja pomoću entiteta © i simbol za novčanu jedinicu evro, koji se predstavlja kao €. Međutim, ta funkcija ne pretvara znakove u numeričke entitete.

Drugi parametar obe funkcije je vrednost koja određuje kako se tretiraju polunavodnici/navodnici i neispravne sekvence koda, a treći – opcioni – parametar obe funkcije jeste skup znakova u koji se kodira ulazni tekst (što je od ključne važnosti za nas jer želimo da ta funkcija ispravno radi s našim znakovnim nizovima u formatu UTF-8). Drugi parametar najčešće ima jednu od sledećih šest vrednosti:

- `ENT_COMPAT` (podrazumevana vrednost) – Navodnici se zamenjuju entitetom " ali ne i polunavodnici.
- `ENT_QUOTES` – Zamenjuju se i polunavodnici i navodnici, entitetima ' odnosno ".
- `ENT_NOQUOTES` – Funkcija ne zamenjuje ni polunavodnike ni navodnike.
- `ENT_IGNORE` – Neispravne sekvence koda se tiho zanemaruju.
- `ENT_SUBSTITUTE` – Neispravne sekvence koda zamenjuju se nekim Unicode zamenskim znakom (Unicode Replacement Character) umesto da se vrati prazan znakovni niz.
- `ENT_DISALLOWED` – Neispravne sekvence koda zamenjuju se nekim Unicode zamenskim znakom umesto da se ostave takve kakve su.

Pogledajte sledeći odlomak koda:

```
$input_str  =  "<p  align=\"center\">Korisnik je platio  \"15000?\".</p>
               <script type=\"text/javascript\">
               // ovde dolazi zlonameran JavaScript kôd.
               </script>";
```

Ako ga izvršimo u sledećem PHP skriptu (izlazni tekst prosleđujemo funkciji `nl2br` kako bismo bili sigurni da će ga klijentski čitač lepo formatirati):

```
<?php
  $str = htmlspecialchars($input_str, ENT_NOQUOTES, "UTF-8");
  echo nl2br($str);
  $str = htmlentities($input_str, ENT_QUOTES, "UTF-8");
  echo nl2br($str);
?>
```

dobili bismo sledeći tekst:

```
&lt;p align="center"&gt;Korisnik je platio "15000?".&lt;/p&gt;<br />
<br />
&lt;script type="text/javascript"&gt;<br />
// ovde dolazi zlonameran JavaScript kôd.
&lt;/script&gt;&lt;p align=&quot;center&quot;&gt;Korisnik je platio
&quot;15000&euro;&quot;.&lt;/p&gt;<br />
<br />
&lt;script type=&quot;text/javascript&quot;&gt;<br />
// ovde dolazi zlonameran JavaScript kôd.
&lt;/script&gt;
```

U prozoru čitača, tekst bi izgledao ovako:

```
<p align="center">Korisnik je platio "15000?".</p>

<script type="text/javascript">
// ovde dolazi zlonameran JavaScript kôd.
</script><p align="center">Korisnik je platio "15000?".</p>

<script type="text/javascript">
// ovde dolazi zlonameran JavaScript kôd.
</script>
```

Obratite pažnju na to da je funkcija `htmlentities` zamenila simbol za novčanu jedinicu evro (€) entitetom (`€`), dok funkcija `htmlspecialchars` to nije učinila.

U slučajevima kada korisnicima dozvoljavamo da unose i HTML kôd, kao na forumima za razmenu poruka, gde bi ljudi možda voleli da koriste znakove koji definišu font, boju i stil teksta (podebljana ili kurzivna slova), moraćemo da ispitujemo ulazne znakovne nizove kako bismo otkrili te znakove i ostavili ih u tekstu.

Organizovanje programskog koda

Dobar osnovni pristup je sledeći: nijedna datoteka koja nije direktno dostupna korisniku preko interneta, ne bi trebalo da se nalazi u stablu direktorijuma u kome su smešteni dokumenti veb aplikacije. Na primer, ako je `/home/httpd/messageboard/www` korenski direktorijum nase veb lokacije za razmenu poruka, sve datoteke za umetanje trebalo bi da stavimo na neko mesto poput `/home/httpd/messageboard/lib`. Zatim, kada nam u programskom kodu zatreba da umetnemo sadržaj iz tih datoteka, možemo napisati:

```
require_once('../lib/user_object.php);
```

Za tu meru opreza postoji nekoliko razloga.

Prvo, razmotrite šta se događa kada zlonamerni korisnik pošalje zahtev za datoteku koja nije `.php` ili `.html`. Mnogi veb serveri – ukoliko nisu ispravno konfigurisani – samo će proslediti ceo sadržaj tražene datoteke u izlazni tok. Ako datoteku `neka_biblioteka.inc` čuvate negde u stablu direktorijuma za dokumente veb lokacije, može se dogoditi da korisnik vidi ceo vaš programski kôd u svom čitaču veba. Tako bi korisnik mogao da vidi podatke ili putanje servera i da otkrije bezbednosne propuste koje ste možda prevideli.

Da bismo to sprečili, moramo podesiti veb server tako da prihvata samo zahteve u kojima se traže `.php` i `.html` datoteke, dok za druge vrste datoteka (kao što su `*.inc`, `*.mo`, `*.txt` itd.) treba da prijavljuje grešku.

Drugo, čak i ako sve vaše datoteke imaju oznaku tipa `.php`, neke od njih koje su namenjena za umetanje u kôd pomoću komande `include` mogu izazvati nepoželjne posledice ukoliko se učitaju van konteksta. Razmotrite, na primer, biblioteku koda za administriranje. Možete tražiti proveru identiteta u uobičajenom kontekstu, ali ako se datoteka učita sama – ta provera se može narušiti.

Slično tome, sve ostale datoteke – kao što su one s lozinkama, tekstualne datoteke, konfiguracione datoteke ili specijalni direktorijumi – moraju se čuvati izvan stabla direktorijuma javno dostupnih dokumenata. Čak i ako mislite da je veb server ispravno podešen, možda ste nešto prevideli; ili, ako kasnije premestite veb aplikaciju na nov server koji nije pravilno konfigurisan, može se dogoditi da otvorite mogućnost provale.

Ukoliko u datoteci `php.ini` uključite opciju `allow_url_fopen` – a imajte na umu da je ona podrazumevano uključena – teorijski biste mogli da umećete ili zahevate datoteke sa udaljenih servera. Pošto bi to bila još jedna mogućnost narušavanja zaštite, trebalo bi da izbegavate izvršavanje datoteka koje se nalaze na udaljenim mašinama, naročito na onim nad kojima nemate potpunu kontrolu. Slično tome, odluke o tome koje ćete datoteke umetati ili zahtevati, ne bi trebalo da zasnivate na podacima koje je korisnik poslao jer i u tom slučaju zlonamerni ili pogrešni podaci mogu biti uzrok problema.

Šta treba da se nalazi u programskom kodu

Mnogi blokovi koda za pristupanje bazama podataka koje ste dosad videli sadrže u samom kodu ime baze podataka, korisničko ime i lozinku, sve u obliku otvorenog teksta, kao u sledećem primeru:

```
$conn = @new mysqli("localhost", "mika", "tajna", "nekabaza");
```

Mada je to zgodno i jednostavno, prilično je nebezbedno jer ako se zlonamernici domognu ove datoteke, dobiće širom otvoren pristup bazi podataka, sa svim ovlašćenjima koje ima korisnik `mika`.

Znatno je bolje da korisničko ime i lozinku čuvate u datoteci koja se ne nalazi u stablu direktorijuma za dokumente veb aplikacije i da je umećete u skript kad vam zatreba, na sledeći način:

```
<?php
  // ovo je dbconnect.php
  $db_server = 'localhost';
  $db_korisnik = 'mika';
  $db_lozinka = 'tajna';
  $db_ime = 'nekabaza';
?>
```

Ova datoteka se onda može pozvati ovako:

```
<?php
  include('../code/dbconnect.php');
  $conn = @new mysqli($db_server, $db_korisnik, $db_lozinka,
                      $db_ime);
  // itd.
?>
```

Preporučljivo je da uradite isto i za sve ostale slične poverljive podatke za koje bi vam trebao dodatan sloj zaštite.

VI

Dodatak

A Instaliranje Apache servera, PHP-a i MySQL-a

A

Instaliranje Apache servera, PHP-a i MySQL-a

Apache, PHP i MySQL na raspolaganju su za mnogobrojne kombinacije operativnih sistema i veb servera. U ovom dodatku objašnjavamo kako da instalirate i podesite Apache, PHP i MySQL na UNIX platformi (gotovo) od početka, i dajemo smernice za instaliranje tih tehnologija na Windows i Mac OS X sisteme.

U ovom dodatku obrađujemo sledeće teme:

- Podešavanje PHP-a za rad kao CGI programa ili kao modula
- Instaliranje Apachea, SSL-a, PHP-a i MySQL-a u Unixu
- Instaliranje Apachea, PHP-a i MySQL-a koristeći objedinjene instalacione pakete
- Instaliranje PEAR-a
- Razmatranje drugih konfiguracija veb servera

> **Napomena**
>
> Dodavanje PHP-a Microsoft Internet Information Serveru ili drugim veb serverima nije opisano u ovom dodatku. Preporučujemo da koristite veb server Apache kad god je moguće. Ipak, smernice za konfigurisanje drugih veb servera date su na kraju dodatka.

Ovaj dodatak je vodič za instaliranje veb servera koji omogućava rad više veb lokacija na istom serveru. U slučaju nekih veb lokacija koje ćete praviti, potreban je protokol SSL (Secure Socket Layer) za sve aplikacije iz oblasti elektronske trgovine. Većina radi tako što skriptovi uspostavljaju veze sa serverom za baze podataka da bi izdvajali i obrađivali podatke. Zbog toga dajemo i uputstva za uobičajeno podešavanje PHP-a, MySQL-a i Apache servera uz korišćenje SSL-a na UNIX mašini.

Pošto većina korisnika PHP-a neće nikada morati da instalira PHP, izdvojili smo ovo gradivo u dodatak umesto da ga predstavimo u prvom poglavlju. Najlakši način da dobijete pristup pouzdanom serveru koji ima brzu vezu sa internetom i na kome je već instaliran PHP jeste da otvorite nalog na nekoj od hiljada veb lokacija koje pružaju usluge ugošćavanja (hostinga) aplikacija. Ako i vi tako radite, obavezno proverite da li vaš davalac usluga koristi najnovije verzije Apachea, PHP-a i MySQL-a, jer ćete inače biti izloženi bezbednosnim rizicima na koje ne možete da utičete (da ne pominjemo kako nećete moći da koristite najnovije i najbolje mogućnosti tih tehnologija).

U zavisnosti od razloga zbog kojih instalirate PHP, moraćete da donesete određene odluke. Na primer, ukoliko imate računar koji je stalno povezan sa internetom i koji nameravate da koristite kao server za aplikacije, performanse sistema će vam svakako biti važne. Ako postavljate razvojni (probni) server koji će vam služiti da na njemu pišete i testirate programski kôd, najvažnije će vam biti da imate sličnu konfiguraciju kao onu na veb serveru koji ćete stvarno koristiti za aplikacije.

> **Napomena**
>
> PHP interpreter može raditi kao modul ili kao zaseban CGI (Common Gateway Interface) izvršni program. Najčešće se koristi verzija u obliku modula zbog boljih performansi. CGI verzija se ponekad koristi na serverima za koje ne postoji verzija u obliku modula ili zato što CGI verzija omogućava korisnicima Apachea da učitavaju različite PHP stranice pod različitim korisničkim nalozima.
>
> U ovom dodatku bavićemo se najviše verzijom u obliku modula kao metodom upotrebe PHP-a.

Instaliranje Apache servera, PHP-a i MySQL-a u Unixu

U zavisnosti od vaših potreba i nivoa iskustva koje imate sa Unix sistemima, možete se opredeliti da instalirate binarnu verziju ili da programe prevodite iz izvornog koda. Svaki pristup pruža određene prednosti.

Za instaliranje iz binarne verzije, stručnjaku će biti potrebno nekoliko minuta, a početniku nešto malo više, ali će konačan rezultat biti sistem koji će kasniti verziju ili dve za najnovijim izdanjem, i sistem za čije su konfigurisanje korišćene opcije koje je neko drugi izabrao. Ukoliko ste pročitali dokumente o izmenama u novim verzijama i znate šta propuštate, ili ako opcije koje je koristila kompanija koja održava binarnu distribuciju zadovoljavaju vaše potrebe, onda svakako obavite binarnu instalaciju.

Mada će vam za instaliranje uz korišćenje izvornog koda trebati dodatno vreme jer morate preuzeti izvorni kôd sa interneta, instalirati ga i konfigurisati – a osim toga, postupak pomalo plaši prvih nekoliko puta – ako se opredelite za to rešenje, imaćete potpunu kontrolu nad konfiguracijom svih tih tehnologija. Sami birate šta ćete instalirati, koju verziju ćete upotrebiti i koje su vam konfiguracione opcije na raspolaganju.

Instaliranje iz binarne verzije

Većina distribucija Linuxa sadrži i već konfigurisan veb server Apache s ugrađenim PHP-om. Šta ćete tačno dobiti, zavisi od distribucije i verzije za koju se opredelite.

Jedan od nedostataka binarnih verzija jeste to što retko kada dobijate najnoviju verziju programa. U zavisnosti od toga koliko su vam važne ispravke grešaka, činjenica da nemate baš najnoviju verziju možda vam i neće predstavljati problem. Ipak, ako nameravate da koristite unapred konfigurisane binarne instalacije PHP-a, MySQL-a, Apache servera i bilo kakvih dodatnih biblioteka, preporučujemo da – pre nego što počnete da ih koristite – obavezno ažurirate sve te pakete primenjujući metodu ažuriranja koja je uobičajena za vašu distribuciju (recimo, koristeći `apt-get`, `yum` ili druge programe za upravljanje paketima).

Najveći problem s binarnom instalacijom jeste to što ne možete da birate opcije s kojima će se vaši programi prevoditi. Najprilagodljivije i najpouzdanije rešenje jeste da sve programe koji vam trebaju prevedete iz njihovog izvornih verzija. Budući da će vam za to biti potrebno malo više vremena nego za instaliranje paketa pomoću nekog programa za upravljanje paketima, razumemo da ćete možda odlučiti da koristite binarne pakete kada su na raspolaganju; štaviše, ako vam treba samo osnovna konfiguracija, vrlo je verovatno da će zvanične binarne verzije – unapred konfigurisane za vaš sistem – odgovarati vašim potrebama.

Instaliranje iz izvornog koda

Da biste instalirali Apache, PHP i MySQL iz izvornog koda u Unix okruženju, prvo morate odlučiti koje dodatne module ćete instalirati. Pošto se u nekim primerima navedenim u ovoj knjizi koristi zaštićen server za transakcije na vebu, trebalo bi da instalirate server koji podržava i protokol SSL (Secure Socket Layer).

Naša PHP konfiguracija biće manje-više slična standardnoj konfiguraciji, ali dodaćemo joj i biblioteku `gd2`.

`gd2` je samo jedna od brojnih biblioteka raspoloživih za PHP. Opisaćemo instaliranje gd2 kako biste stekli sliku o tome šta treba uraditi da bi se PHP-u pridružile dodatne biblioteke. Prevođenje većine programa koji treba da rade u Unixu odvija se na sličan način, ali – kao što ćete videti dok budete napredovali kroz ovaj odeljak – ponekad je ionako lakše instalirati unapred kompajlirane pakete.

Budući da PHP obično morate ponovo da prevedete pošto instalirate novu biblioteku, ako unapred znate šta vam sve treba, instalirajte prvo sve neophodne biblioteke, a zatim prevedite PHP modul.

U ovom dodatku opisujemo instaliranje iz izvornog koda na Linux server Ubuntu, ali je opis dovoljno opšti da se postupak može primeniti i na druge Unix servere.

Najpre ćemo prikupiti datoteke koje su neophodne za instaliranje. Trebaće vam sledeći programi:

- Apache (http://httpd.apache.org/): veb server
- OpenSSL (http://www.openssl.org/): Paket otvorenog koda koji realizuje protokol SSL (Secure Sockets Layer)
- MySQL (http://www.mysql.com/): Relaciona baza podataka
- PHP (http://www.php.net/): Jezik za skriptove koji rade na serveru
- JPEG (http://www.ijg.org/): Biblioteka JPEG, potrebna za rad biblioteke gd2.
- PNG (//www.libpng.org/pub/png/libpng.html): Biblioteka PNG, potrebna za rad biblioteke gd2.
- zlib (//www.zlib.net/): Biblioteka zlib, potrebna za rad biblioteke PNG.
- IMAP (//ftp.cac.washington.edu/imap/): IMAP C klijent, potreban za rad biblioteke IMAP.

Ako želite da koristite funkciju `mail()`, potreban vam je i program tipa MTA (Mail Transfer Agent), mada ga ovde ne opisujemo.

Pretpostavićemo da imate administratorska ovlašćenja i da na svom sistemu imate instalirane sledeće alatke:

- `gzip` ili `gunzip`
- `gcc` i GNU `make`

Kada budete spremni da započnete postupak instaliranja servera Apache, PHP-a i MySQL-a, trebalo bi da prvo preuzmete sve datoteke sa izvornim kodom i da ih smestite u neki privremeni direktorijum. Mi smo kao privremeni direktorijum izabrali `/usr/src`. Najbolje je da datoteke preuzmete kao administrator da biste izbegli probleme sa ovlašćenjima.

Instaliranje MySQL-a

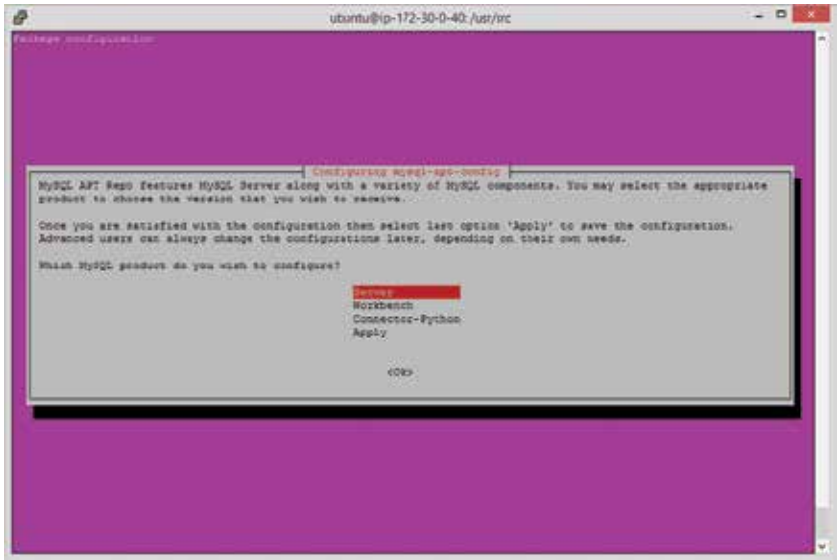
U ovom odeljku opisujemo kako se MySQL instalira iz binarne verzije. Instaliranje zvanične distribuirane binarne verzije MySQL-a, specifične za vaš sistem, jeste preporučeni postupak. Ona u sebi sadrži unapred određene opcije koje će vam najverovatnije trebati, za koje se pokazalo da su najčešće korišćene u praksi. Mada je sasvim u redu da preuzmete i prevedete (engl. *build*) izvorni kôd za MySQL, a i prateća uputstva su prilično dobra, brže ćete završiti posao oko instalacije, bez ikakve razlike u funkcionalnosti, ako instalirate odgovarajuću binarnu verziju.

U ovom primeru koristimo spremište (engl. *repository*) `apt` za Ubuntu 14.04, ali na lokaciji http://www.mysql.com/downloads/ možete naći spremište binarnih datoteka koje odgovara vašem sistemu. Taj način instaliranja automatski postavlja datoteke na više raznih lokacija i upravlja opcijama za konfigurisanje sistema.

Preuzeli smo paket `mysql-apt-config_0.3.2-1ubuntu14.04_all.deb` a zatim ga instalirali pomoću naredbe

```
# sudo dpkg -i mysql-apt-config_0.3.2-1ubuntu14.04_all.deb
```

Na ovom mestu, program će vas pitati koje komponente želite da instalirate, kao što je prikazano na slici A.1.



Slika A.1 Konfiguracione opcije za MySQL.

Izaberite „Server“ i nastavite, a zatim odaberite verziju servera u sledećem koraku. Pritisnite „Apply“ da biste primenili izmene, a onda „Ok“ da biste potvrdili konfiguraciju. Nakon toga, sistem je spreman da koristi `apt` kako bi instalirao baš onu alikaciju koju želite da koristite, pa zadajte sledeću komandu za ažuriranje:

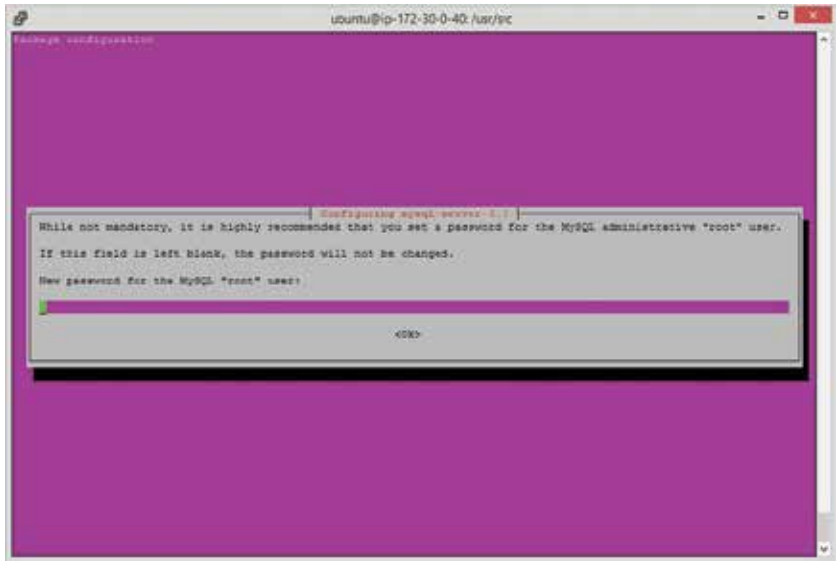
```
# sudo apt-get update
```

Kada se taj proces završi, instalirajte MySQL server:

```
# sudo apt-get install mysql-server
```

Možda će vam se ponuditi da instalirate dodatne potrebne biblioteke; u tom slučaju, izaberite Y da biste nastavili.

U nastavku instalacionog procesa, trebalo bi da se pojavi zahtev za zadavanje administratorske lozinke za MySQL, kao što je prikazano na slici A.2.



Slika A.2 Poziv za unošenje administratorske lozinke za MySQL.

Nakon što upišete lozinku, tražiće vam se da je potvrdite, a zatim će se proces instalacije nastaviti.

Kada instalirate MySQL, automatski se generišu tri baze podataka. Jedna je `mysql` tabela, koja sadrži podatke o korisnicima, računarima i ovlašćenjima za rad. Druge dve su `information_schema` i `performance_schema`, u kojima se čuvaju dodatni metapodaci o MySQL serveru. Postojanje tih baza podataka možete proveriti s komandne linije, na način sličan sledećem:

```
# mysql -u root –p
Enter password:
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| performance_schema |
+--------------------+
3 rows in set (0.00 sec)
```

Upišite `quit` ili `\q` da biste zatvorili klijenta za MySQL.

Podrazumevana konfiguracija MySQL-a omogućava korisnicima da pristupaju sistemu bez zadavanja korisničkog imena i lozinke. To je svakako nepoželjno, pa vas proces instalacije vodi kroz korak zadavanja lozinke za administratora.

Poslednji obavezni korak podešavanja MySQL-a jeste brisanje eventualnih anonimnih naloga, koji mogu biti deo distribuirane baze podataka. To ćete obaviti ako pređete u komandni prozor i upišete sledeće redove:

```
# mysql -u root –p
Enter password:
mysql> use mysql
mysql> delete from user where User='';
mysql> quit
```

Potom treba da upišete

```
mysqladmin -u root -p reload
```

kako bi počele da važe izmene koje ste uneli u sistem.

U ovom trenutku imate funkcionalnu i bezbednu instalaciju MySQL-a koju možete koristiti s PHP-om.

Instaliranje PHP-a i Apache servera

Imajte na umu da se u dole navedenim primerima reč `VERSION` koristi kao rezervacija za onu skorašnju verziju softvera koja je dostupna za vaš sistem. Instalacioni proces je isti bez obzira na verziju. Kada budete izvršavali navedene komande, zamenite reč `VERSION` brojem verzije softvera koju ste preuzeli s veba.

Pre instaliranja PHP-a konfigurisaćemo i instalirati tzv. vanila (bez ikakvih opcija) verziju Apache servera, da bi proces instaliranja PHP-a mogao da pronađe sve što se odnosi na Apache. Ne brinite, na temu potpunog konfigurisanja i instaliranja Apache servera vratićemo se kasnije u ovom odeljku. Prvo proverite da li se nalazite u direktorijumu sa izvornim kodom date distribucije.

```
# cd /usr/src
# sudo gunzip httpd-VERSION.tar.gz –
# sudo tar –xvf httpd-VERSION.tar
# cd httpd-VERSION
# sudo ./configure --prefix=/usr/local/apache2 –-enable-so
```

Drugi identifikator konfiguracije obezbeđuje da će se modul `mod_so` prevesti u Apache. Taj modul – čije ime ukazuje na format UNIX-ovog deljenog objekta (`*.so`) – omogućava upotrebu dinamičkih modula kao što je PHP sa serverom Apache.

Nastavimo proces zadavanjem sledećih komandi:

```
# sudo make
# sudo make install
```

U ovoj tački, osnovna verzija servera Apache konfigurisana je na vašem sistemu. Da biste nastavili postavljanje PHP-a, moraćete da stvorite dodatne biblioteke koje će PHP i Apache koristiti u našem primeru (JPEG, PNG, zlib, OpenSSL i IMAP) tako da konfiguracija PHP-a ispravno ukazuje na te biblioteke.

Da biste instalirali biblioteku JPEG, pratite sledeće korake:

```
# cd /usr/src
# sudo gunzip jpegsrc.VERSION.tar.gz
# sudo tar –xvf jpegsrc.VERSION.tar.gz
```

```
# cd jpeg-VERSION
# sudo ./configure
# sudo make
# sudo make install
```

Nakon poslednjeg koraka – osim ukoliko su se tokom postupka pojavile poruke o greškama – trebalo bi da je biblioteka JPEG instalirana u direktorijum `/usr/local/lib`. Ako je bilo poruka o greškama, pratite uputstva u njima ili proučite dokumentaciju za biblioteku JPEG.

Ponovite proces `gunzip`, `tar`, `configure`, `make` i `make install` za izvorne datoteke biblioteka PNG i zlib, i zabeležite instalacione direktorijume za obe.

Za instaliranje biblioteke IMAP C klijenta, možete preuzeti i izgraditi izvorne datoteke na sasvim sličan način, ali postoje i neke specifičnosti koje mogu rezultovati razlikama od jednog sistema do drugog. Preporučujemo da sledite najnovije smernice objavljene u priručniku za PHP, na adresi http://php.net/manual/en/imap.requirements.php.

Možda ćete zaključiti i da je znatno jednostavnije instalirati unapred kompajliran paket za vaš tip UNIX servera, kao što mi radimo ovde za Ubuntu 14.04:

```
# sudo  sudo apt-get install libc-client-dev
```

Pošto su sve biblioteke spremne za upotrebu, vratimo se na podešavanje PHP-a. Izdvojite izvorne datoteke i pređite u odgovarajući direktorijum:

```
# cd /usr/src
# sudo gunzip  php-VERSION.tar.gz
# sudo tar –xvf php-VERSION.tar# cd php-VERSION
```

PHP-ova komanda `configure` ima više opcija. Zadajte `./configure -help` da biste saznali šta sve možete zadati. U ovom slučaju želimo da PHP-u dodamo podršku za MySQL, Apache i gd2.

Imajte u vidu da sledeći primer predstavlja jednu komandu. Možete je napisati celu u jednom redu ili – kao u ovom slučaju – upotrebite znak za nastavljanje, obrnutu kosu crtu (\), što omogućava da komandu raspodelite u više redova da bi se lakše čitala.

```
# ./configure  --prefix=/usr/local/php
               --with-mysqli=mysqlnd \
               --with-apxs2=/usr/local/apache2/bin/apxs \
               --with-jpeg-dir=/usr/local/lib \
               --with-png-dir=/usr/local/lib \
               --with-zlib-dir=/usr/local/lib \
               --with-imap=/usr/lib \
               --with-kerberos \
               --with-imap-ssl \
               --with-gd
```

Prvi indikator konfiguracije zadaje lokaciju instalacionog direktorijuma za PHP – u ovom slučaju, `/usr/local/php`. Drugi indikator obezbeđuje da PHP bude izgrađen sa izvornim drajverom za MySQL. Treći indikator saopštava PHP-u lokaciju alatke `apxs2` (Apache Extension), koja se koristi za izgradnju Apache modula – od kojih će jedan biti PHP. Lokacija alatke `apxs2` mora da se konfiguriše u procesu podešavanja instalacije PHP-a kako bi mogla da se napravi odgovarajuća izvršna verzija PHP-a za Apache modul, zbog čega je neophodno da se unapred konfiguriše i instalira makar osnovna verzija Apache servera pre formiranja izvršne verzije PHP-a.

Sledeći skup indikatora konfiguracije usmerava konfiguracionu alatku za PHP na lokaciju prethodno instaliranih biblioteka (JPEG, PNG, zlip i IMAP). Indikatori `--with-kerberos` i `–with-imap-ssl` dodatni su indikatori koji se odnose na korišćenje biblioteke IMAP; zavisno od unapred kompajliranog paketa koji koristite u svom sistemu, možda vam neće trebati ovi indikatori ili će vam trebati dodatni indikatori. Proces konfiguracije će vam reći koji indikatori nedostaju.

Nakon završene konfigurcije, videćete poruku sličnu sledećoj:

```
Generating files
configure: creating ./config.status
creating main/internal_functions.c
creating main/internal_functions_cli.c
+--------------------------------------------------------------------+
| License:                                                           |
| This software is subject to the PHP License, available in this     |
| distribution in the file LICENSE.  By continuing this installation |
| process, you are bound by the terms of this license agreement.     |
| If you do not agree with the terms of this license, you must abort |
| the installation process at this point.                            |
+--------------------------------------------------------------------+

Thank you for using PHP.

config.status: creating php7.spec
config.status: creating main/build-defs.h
config.status: creating scripts/phpize
config.status: creating scripts/man1/phpize.1
config.status: creating scripts/php-config
config.status: creating scripts/man1/php-config.1
config.status: creating sapi/cli/php.1
config.status: creating sapi/cgi/php-cgi.1
config.status: creating ext/phar/phar.1
config.status: creating ext/phar/phar.phar.1
config.status: creating main/php_config.h
config.status: main/php_config.h is unchanged
config.status: executing default commands
```

Sistem je spreman za stvaranje i instaliranje binarnih datoteka; zadajte komande `make` i `make install` da biste dobili binarni oblik prema specifikacijama koje ste zadali:

```
# make
# make install
```

Nakon završnog koraka – osim ako je usput bilo poruka o greškama – PHP će biti napravljen i instaliran, i Apache modul PHP biće napravljen, instaliran i smešten u odgovarajući direktorijum unutar strukture direktorijuma Apache servera. Poslednji delić procesa konfigurisanja koji ćete možda želeti da obavite jeste da obezbedite da datoteka `php.ini` bude smeštena na stabilnom i uobičajenom mestu:

```
# sudo cp php.ini-development /usr/local/php/lib/php.ini
```

ili

```
# sudo cp php.ini-production /usr/local/php/lib/php.ini
```

Obe verzije datoteke `php.ini` imaju u predloženim komandama drugačije podešene opcije. Prva datoteka, `php.ini-development`, predviđena je za upotrebu u razvojnim sistemima. Na primer, u njoj je opcija `display_errors` podešena na On. To olakšava razvijanje aplikacija, ali nije pogodno za računar na kome rade poslovne aplikacije. Kada u ovoj knjizi pominjemo standardne vrednosti opcija u datoteci `php.ini`, mislimo na vrednosti u ovoj verziji datoteke `php.ini`. Druga verzija, `php.ini-production`, predviđena je za upotrebu na stvarnim, (ne razvojnim) serverima.

Da biste podesili opcije za PHP, možete ažurirati datoteku `php.ini`. Postoji veliki broj opcija koje možete zadati, ali nekoliko njih je posebno vredno pomena. Može vam zatrebati da podesite vrednost opcije `sendmail_path` ako želite da iz svojih skriptova šaljete poruke e-poštom.

Kada je PHP potpuno podešen, vreme je da se nakratko vratite Apacheovim izvornim datotekama i ponovo ih konfigurišete i prevedete na način pogodniji za razvoj veb aplikacija (znači, dalje od ranije konfigurisanih osnova). Pored konfiguracione opcije `--enable-so`, koja omogućava naknadno pojedinačno dodavanje modula Apachea, upotrebićemo i `--enable-ssl` da bismo omogućili upotrebu modula `mod_ssl`. Uz to, podešavamo osnovni direktorijum za OpenSSL, koji smo konfigurisali i instalirali ranije u ovom odeljku.

```
# cd /usr/local/httpd-VERSION
# sudo SSL_BASE=../openssl-VERSION \
       ./configure \
       --prefix=/usr/local/apache2 \
       --enable-so \
       --enable-ssl
# sudo make
# sudo make install
```

Ako sve prođe kako treba, vratićete se na komandnu liniju bez poruka o greškama i moći ćete da napravite finalne izmene konfiguracije u svojoj instalaciji. Ukoliko se u procesu instaliranja pojave greške, potražite pomoć u dokumentaciji za Apache HTTP Server 2.4, na adresi http://httpd.apache.org/docs/2.4/.

Osnovne izmene konfiguracije Apache servera

Za najveći deo konfiguracije Apache servera koristi se datoteka `httpd.conf`. Ako ste pratili prethodna uputstva, vaša datoteka `httpd.conf` biće u direktorijumu `/usr/local/apache2/conf`. Da bi se vaš server pokrenuo i koristio PHP i SSL, morate ponešto izmeniti:

- Pronađite red koda koji počinje sa `#ServerName` i izmenite ga u `ServerName imevašegservera.com`
- Pronađite blok redova koda koji počinje sa AddType i dodajte sledeće kako biste bili sigurni da se datoteke sa oznakom tipa `php` ispravno usmeravaju kroz PHP modul radi obrade:

```
AddType application/x-httpd-php .php
AddType application/x-httpd-php-source .phps
```

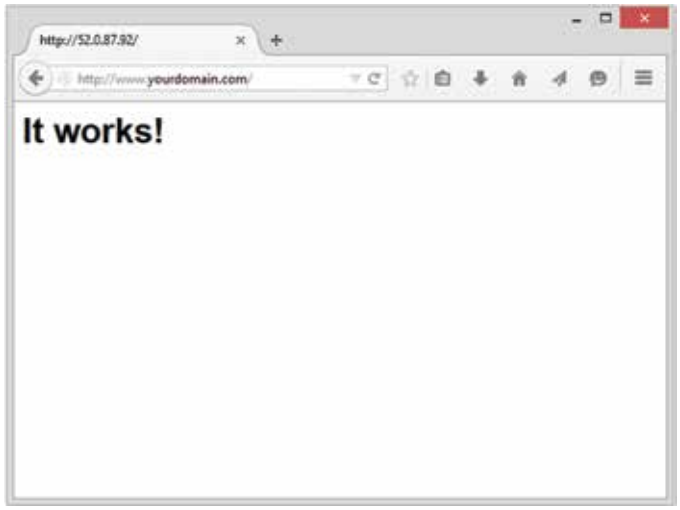
Sada ste spremni da pokrenete Apache server kako biste videli da li radi s PHP-om. Prvo pokrenimo server bez podrške za SSL i pogledajmo da li se uopšte pokreće. U narednim odeljcima proverićemo da li su podržani PHP i SSL kako bismo videli da li sve funkcioniše.

Možete upotrebiti `configtest` da proverite da li je konfiguracija dobro postavljena:

```
# cd /usr/local/apache2/bin
# sudo ./apachectl configtest
Syntax OK
# sudo ./apachectl start
./apachectl start: httpd started
```

Ukoliko je sve u redu, videćete nešto slično onome sa slike A.3 kada se preko čitača veba povežete na server.

> **Napomena**
>
> Sa serverom se možete povezati preko imena domena ili preko stvarne IP adrese datog računara. Povežite se na oba načina da biste se uverili da sve funkcioniše kako treba.



Slika A.3 Podrazumevana probna stranica koju daje Apache.

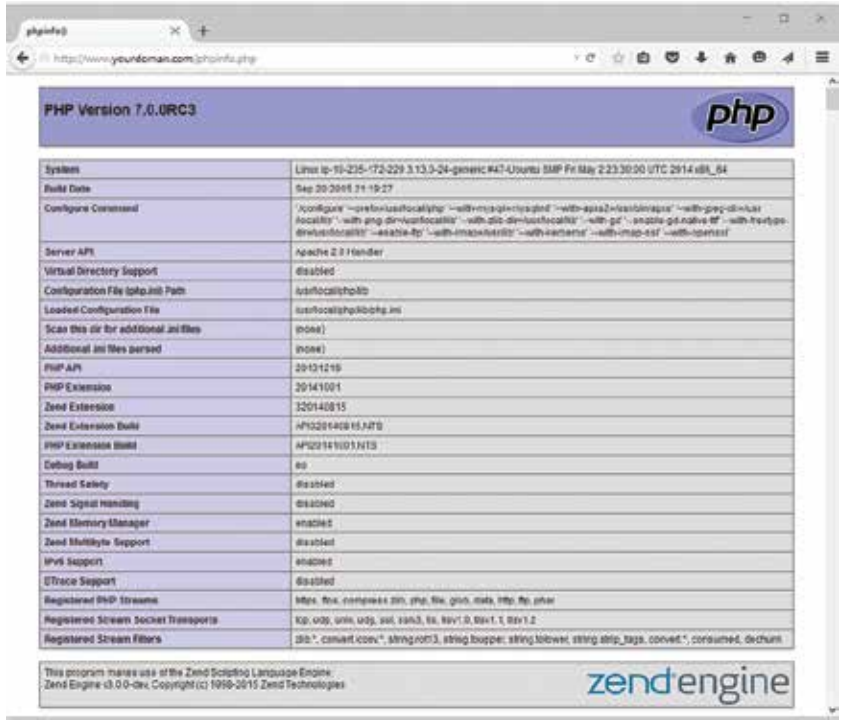
Da li radi podrška za PHP?

Pošto ste sada sigurni da Apache veb server radi sam za sebe, možete ispitati da li podržava PHP. Napravite datoteku pod imenom `test.php` na korenskoj putanji dokumenta – što bi trebalo da bude `/usr/local/apache/htdocs` ako ste pratili prethodno navedena uputstva za instaliranje. Imajte u vidu da tu putanju možete promeniti u datoteci `httpd.conf`.

Datoteka bi trebalo da sadrži samo sledeći red koda:

```
<?php phpinfo(); ?>
```

Prikaz na ekranu trebalo bi da bude sličan onom sa slike A.4.



Slika A.4 Funkcija `phpinfo()` daje korisne informacije o konfiguraciji.

Da li radi SSL?

U ovoj fazi, SSL *ne* bi trebalo da radi na vašem računaru, zato što nismo napravili SSL sertifikate niti ključeve za njega. Apache je konfigurisan i spreman za rad, ali sertifikati koji garantuju bezbednost vaših podataka još ne postoje.

Za potrebe razvoja, možete napraviti sertifikate koje ćete sami potpisati, koristeći OpenSSL; ukoliko ćete datu veb lokaciju koristiti u stvarnim uslovima na vebu, trebalo bi da primenjujete legitimno potpisane sertifikate koje izdaje neko zvanično telo (Certificate Authority). Na primer, Let's Encrypt je besplatno, automatizovano sertifikaciono telo koje koristi otvoren kôd i koje – radi opšte, javne dobrobiti – vodi organizacija Internet Security Research Group (ISRG). Više o tome saznajte na adresi https://letsencrypt.org/.

Da biste napravili sertifikat koji ćete sami potpisati, zadajte sledeću komandu kako biste dobili sertifikat i ključ, i smestili ih tamo gde ih Apache očekuje:

```
# sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
      -keyout /usr/local/apache2/conf/server.key \
     -out /usr/local/apache2/conf/server.crt
```

Skript za izradu sertifikata i ključa tražiće neke informacije – kao što su država, ime kompanije i ime domena; možete dati lažne ili prave podatke, po želji. Nakon izvršavanja skripta, imaćete privremeni sertifikat koji ste sami potpisali i koji će važiti 365 dana, smešten u direktorijum `/usr/local/apache2/conf` – zajedno s pripadajućim ključem.

Apacheov modul SSL ima sopstvenu konfiguracionu datoteku, smeštenu u `/usr/local/apache2/conf/extra/httpd-ssl.conf`. U ovoj fazi, možete je ostaviti neizmenjenu i samo restartovati Apache da bi se primenile sve izmene, ili možete nastaviti tako što ćete izmeniti konfiguraciju po svojoj želji. Dokumentaciju o mogućim izmenama Apacheove datoteke `httpd-ssl.conf` naći ćete na adresi http://httpd.apache.org/docs/2.4/mod/mod_ssl.html.

U verziji Apache 2.4, da biste aktivirali SSL na nivou servera, treba samo da isključite komentar za datoteku `httpd-ssl.conf` u datoteci `httpd.conf`. Umesto da u datoteci `httpd.conf` red glasi ovako:

```
# Include conf/extra/httpd-ssl.conf
```

treba da glasi ovako:

```
Include conf/extra/httpd-ssl.conf
```

Nakon što završite menjanje konfiguracije, samo restartujte server:

```
# sudo /usr/local/apache2/bin/apachectl restart
```

Proverite da li radi tako što ćete se povezati sa serverom pomoću čitača veba i izabrati protokol `https`, ovako:

`https://`*`vašserver.vašdomen`*`.com`

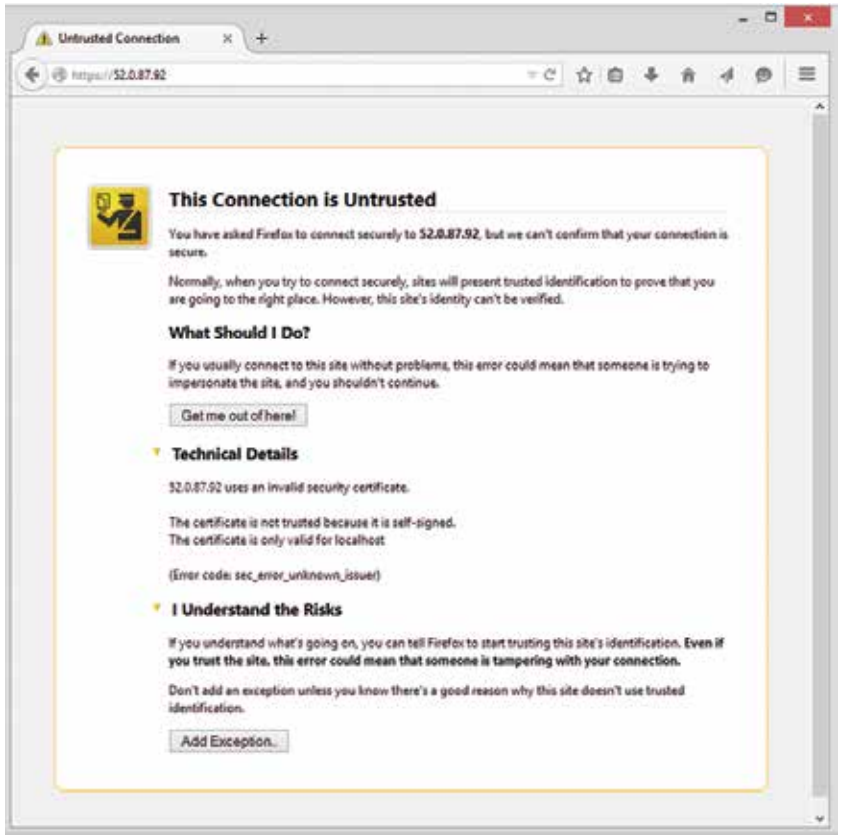
Isprobajte i IP adresu svog servera, ovako:

`https://`*`xxx.xxx.xxx.xxx`*

ili

`http://`*`xxx.xxx.xxx.xxx`*`:443`

Ako i konfiguracija i samopotpisani sertifikati funkcionišu, server će poslati sertifikate čitaču veba da bi se uspostavila bezbedna veza. Pošto je reč o sertifikatu koji ste sami potpisali, čitač će vam prikazati i upozorenje pre nego što vam dozvoli da nastavite rad (slika A.5). Kada biste imali sertifikat izdat od sertifikacionog tela kome vaš čitač veba veruje od ranije, čitač ne bi prikazao upozorenje.



Slika A.5 Kada koristite sertifikat koji ste sami potpisali, čitač veba će tražiti da pritisnete dugme Add Exception i tako potvrdite da može da prihvati taj sertifikat.

Instaliranje Apache servera, PHP-a i MySQL-a za Windows i Mac OS X korišćenjem objedinjenih instalacionih paketa

Ako imate odgovarajuće razvojne alatke, možete instalirati PHP, Apache i MySQL iz izvornog koda i na Windows i na Mac OS X sistemima. Možete, takođe, ručno instalirati PHP, Apache i MySQL pakete pojedinačno, kao što je opisano na stranama priručnika za PHP posvećenim instalaciji, na adresi http://php.net/manual/en/install.php. Međutim, da biste brzo osposobili svoje razvojno okruženje, možete koristiti neki objedinjeni instalacioni paket nezavisnog proizvođača, pomoću kojeg možete instalirati sve tri komponente odjednom.

Popularan i dobro održavan instalacioni paket za Apache, MySQL i PHP jeste XAMPP – „X“ ukazuje na to da je namenjen za više platformi; u stvari, besplatno možete preuzeti paket XAMPP za Linux, Windows ili Mac OS X sa lokacije https://www.apachefriends.org/download.html:

Postoje još dva dobra instalaciona paketa nezavisnih proizvođača za Apache, MySQL i PHP, namenjena za određene operativne sisteme:

- WAMP – Instalira Apache, MySQL i PHP na Windows. Više informacija na adresi http://www.wampserver.com.
- MAMP – Instalira Apache, MySQL i PHP na Mac. Više informacija na adresi http://www.mamp.info/.

Da biste podesili PHP, Apache i MySQL na svom Windows ili Mac OS X sistemu, pratite instrukcije iz instalacionog paketa koji ste odabrali. Verovatno će to biti instalacioni proces u kome ćete u svakom koraku jednom pritiskati taster miša ili jednostavan postupak zasnovan na čarobnjaku, koji će vas pažljivo voditi kroz korake instaliranja i konfigurisanja.

Proverite šta ste uradili

Kada završite s korišćenjem objedinjenog paketa za instaliranje i konfigurisanje, pokrenite veb server i proverite da li PHP radi kako treba. Napravite datoteku `test.php` i upišite u nju sledeći red:

```
<?php phpinfo(); ?>
```

Postavite tu datoteku u korenski direktorijum za dokumente (na primer, u Windowsu, to je najčešće `c:\Program File\Apache Software Foundation\Apache2.4\htdocs`), a zatim je otvorite u čitaču veba, na sledeći način:

```
http://localhost/test.php
```

ili

```
http://vaša-lokalna-ip-adresa/test.php
```

Ako se prikaže stranica nalik na onu sa slike A.4, to znači da PHP radi sa Apache serverom.

Instaliranje biblioteka PEAR

PEAR (PHP Extension and Appplication Repository), predstavlja sistem za distribuciju višekratno upotrebljivih komponenata PHP-a, sačinjen od priloga zajednice PHP programera. Trenutno je na raspolaganju više od 200 paketa za PHP 5+, a svi oni proširuju funkcionalnost osnovne instalacije PHP-a.

Da biste koristili PEAR, prvo proverite imate li program za instaliranje paketa PEAR. Taj program će već postojati i biti dostupan nakon instaliranja PHP-a u UNIX sistem, ali da biste instalirali paket PEAR pod Windowsom, na komandnu liniju upišite `:c:\php\go-pear`.

Skript `go-pear` traži da unesete nekoliko jednostavnih podataka – na primer, gde želite da bude smešten instalacioni program za pakete i standardne PEAR klase, a zatim će sa interneta preuzeti i instalirati te datoteke. U ovoj fazi, trebalo bi da imate instaliranu verziju instalacionog programa za paket PEAR i osnovne PEAR biblioteke. Potom možete instalirati pakete ako otkucate:

```
pear install paket
```

gde *paket* treba da zamenite imenom paketa koji želite da instalirate.

Da biste dobili listu svih paketa na raspolaganju, otkucajte:

```
pear list-all
```

Da biste videli šta sve trenutno imate instalirano, zadajte:

```
pear list
```

Ako hoćete da proverite postoje li novije verzije bilo kojih instalacionih paketa, otkucajte:

```
pear upgrade imepaketa
```

Ukoliko iz bilo kog razloga ne uspete da primenite opisani postupak, predlažemo da PEAR pakete sami preuzmete direktno sa interneta umesto da ih instalirate pomoću instalacionog programa. Da biste to uradili, idite na veb lokaciju http://pear.php.net/packages.php. Sa te lokacije možete se kretati kroz različite raspoložive pakete dok ne pronađete onaj koji vam treba i koji onda možete preuzeti i ručno smestiti datoteke u PHP-ov direktorijum PEAR na svom sistemu.

Instaliranje PHP-a s drugim veb serverima

Mada je korišćenje PHP-a sa veb serverom Apache gotovo podrazumevana, standardna konfiguracija – imajući u vidu koliko već dugo te dve tehnologije uspešno rade zajedno (više od 15 godina) – PHP možete podesiti i tako da radi s nekoliko drugih veb servera. Na primer, odnedavno je sve popularnija konfiguracija PHP plus Nginx (http://nginx.org). Detaljne informacije o instaliranju PHP-a na Nginx server, date su u priručniku za PHP, na adresi http://php.net/manual/en/install.unix.nginx.php.

Uputstva za konfigurisanje dodatnih veb servera za UNIX sisteme takođe su data u priručniku za PHP, na adresi http://php.net/manual/en/install.unix.php.