I

PHP

1

Kratak kurs PHP-a

OVO POGLAVLJE SADRŽI KRATAK PREGLED sintakse i jezičkih konstrukcija PHP-a. Ako već programirate na PHP-u, ono će vam pomoći da popunite praznine u svom znanju, a ako ste ranije koristili C, ASP (Active Server Pages) 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 Web lokacija za elektronsku trgovinu 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 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 Webu.

Isprobajte naredne primere – otkucajte ih ili ih učitajte sa CD-a, menjajte ih, rastavljajte i učite kako da ih popravite.

Poglavlje počinje primerom 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 komandi `if`, `else`, operatora `?:` i iskaza `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 iskaza `if`, `else` i `switch`
- Iteracija: `while`, `do` i petlje `for`

Upotreba PHP-a

Da biste obradili primere iz ovog poglavlja i ostatka knjige, morate imati pristup Web 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 „Instaliranje PHP-a i MySQL-a". Sve što vam treba da biste instalirali PHP pod Unixom ili Windowsom NT naći ćete na pratećem CD-u.

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 firmu „Bobovi auto-delovi". Celokupan kôd primera koji se pominju u ovom poglavlju nalazi se na pratećem disku u direktorijumu `chapter01`.

Izrada obrasca za narudžbenicu

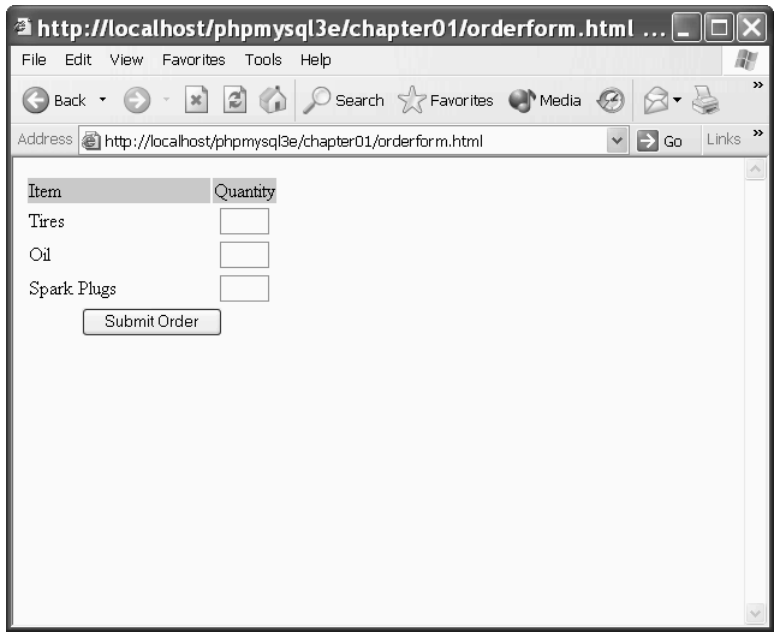
Web dizajner je napravio narudžbenicu za auto-delove koje Bob prodaje. To je relativno jednostavna narudžbenica (slika 1.1), slična mnogima koje ste do sada verovatno sretali na Webu. Bob bi želeo da zna šta su njegovi kupci naručili i da izračuna ukupnu vrednost naručene robe i iznos poreza na tu robu.

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 border="0">
<tr bgcolor="#cccccc">
  <td width="150">Item</td>
  <td width="15">Quantity</td>
</tr>
<tr>
  <td>Tires</td>
  <td align="center"><input type="text" name="tireqty" size="3"
     maxlength="3" /></td>
</tr>
```

```
<tr>
  <td>Oil</td>
  <td align="center"><input type="text" name="oilqty" size="3"
     maxlength="3" /></td>
</tr>
<tr>
  <td>Spark Plugs</td>
  <td align="center"><input type="text" name="sparkqty" size="3"
     maxlength="3" /></td>
</tr>
<tr>
  <td colspan="2" align="center"><input type="submit" value="Submit
Order" /></td>
</tr>
</table>
</form>
```



Slika 1.1 Početna verzija narudžbenice beleži samo proizvode i količine.

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 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, na primer `field23`, koja se teško pamte. Ako imena polja odražavaju unete podatke, olakšaćete sebi programiranje.

Možda bi vredelo 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 znak za podvlačenje.

Obrada podataka sa obrasca

Da bismo obradili podatke sa obrasca, treba da napravimo 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:

```
<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.

Ugradnja PHP koda u HTML kôd

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

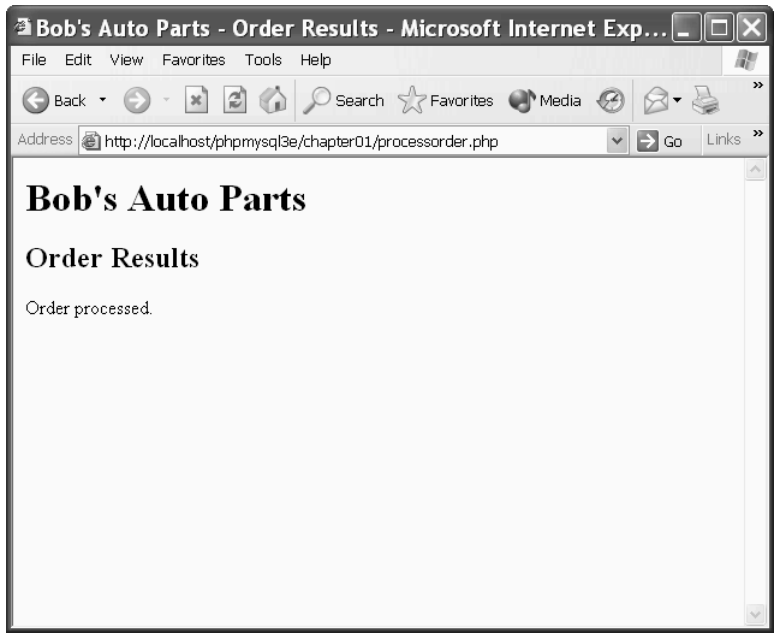
```
<?php
  echo '<p>Order processed.</p>';
?>
```

Snimite datoteku i pogledajte je u čitaču Weba tako što ćete popuniti obrazac i pritisnuti dugme Submit Order. Trebalo bi da se pojavi rezultat nalik na onaj sa slike 1.2.

Obratite pažnju na to da je PHP kôd naredaba koji ste napisali ugrađen u običnu HTML datoteku. Pogledajte izvorni kôd stranice u čitaču Weba. Trebalo bi da vidite nešto nalik na sledeće:

```
<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>
```



Slika 1.2 Tekst prosleđen naredbi `echo` poslat je čitaču Weba.

PHP naredbe se ne vide zato što je interpretator PHP koda zamenio naredbe 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 Web serveru, za razliku od JavaScripta i drugih klijentskih jezika koji se prevode i izvršavaju unutar čitača Weba na korisnikovom računaru.

Kôd koji se sada nalazi u PHP datoteci sastoji se od četiri vrste elemenata:

- HTML oznake
- PHP oznake
- PHP iskazi
- beline

Možemo da dodamo i

- komentare

Većinu redova u navedenom primeru čini običan HTML.

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* – one Web 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* iz bloka HTML koda.

Postoje različiti tipovi PHP oznaka koje ćemo u nastavku detaljnije objasniti.

Stilovi PHP oznaka

Postoje četiri stila PHP oznaka (engl. *PHP tags*). 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. Ako na Web lokaciji upotrebljavate XML dokumente, 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. Iako je on standardno uključen, administratori sistema ga ponekad isključuju jer može biti uzrok pogrešnog tumačenja deklaracija u XML dokumentima.

- **stil SCRIPT**

  ```
  <SCRIPT LANGUAGE='php'> echo '<p>Order processed.</p>'; </script>
  ```

  Ovaj stil oznaka je najduži i biće vam poznat ako ste koristili JavaScript ili VBSrcipt. Možete ga upotrebiti ako vam HTML editor pravi probleme s drugim stilovima oznaka.

- **ASP stil**

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

  Ovaj stil oznaka koristi se u okruženjima ASP i ASP.NET. Da biste mogli da ga koristite, aktivirajte konfiguracijski parametar `asp_tags`. Ovaj stil oznaka dobro će vam doći ako koristite editor za izradu ASP i ASP.NET stranica ili već programirate na ASP-u ili ASP.NET-u. Imajte u vidu da je ovaj stil oznaka standardno isključen.

PHP iskazi

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

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

Kao što ste verovatno pretpostavili, upotreba naredbe `echo` veoma je jednostavna: ona štampa (ili ispisuje) u čitaču tekst koji joj prosledite. Na slici 1.2 vidite da je rezultat to da se tekst „`order processed` pojavljuje u prozoru čitača.

Obratite pažnju na to da se na kraju iskaza `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 novog reda (znak za povratak na početak reda), razmaka i tabulatora, nazivaju se *beline* (engl. *white spaces*). Čitači zanemaruju beline u HTML kodu. Isto tako postupa i izvršno okruženje za PHP. Pogledajte ova dva bloka 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. U HTML kodu možete da koristite beline kao pomoćno sredstvo kojim ćete kôd stranica učiniti čitljivijim, što vam i preporučujemo. Isto se odnosi i na PHP. Nema potrebe da između PHP iskaza stoje beline, ali se kôd s njima lakše čita. Na primer, rezultati iskaza:

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

i

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

identični su, ali se prvi 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 interpretator 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

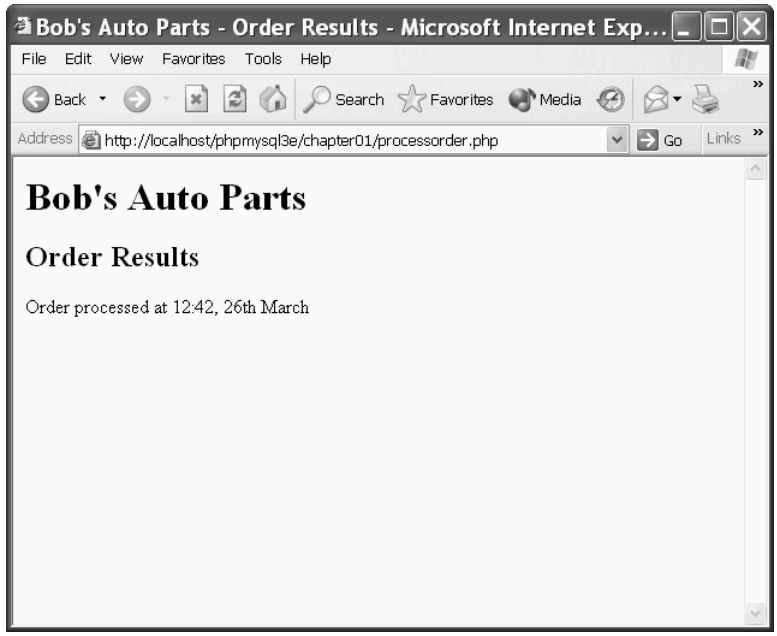
Sve što smo do sada uradili u PHP-u, mogli smo da uradimo i pomoću običnog HTML koda.

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 Web 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');
  echo '</p>';
?>
```

U ovom kodu koristimo ugrađenu funkciju PHP-a `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.



Slika 1.3 PHP funkcija `date()` vraća formatiran datum.

Pozivanje funkcija

Pogledajmo poziv funkcije `date()`. To je opšti oblik pozivanja funkcija. PHP sadrži bogatu biblioteku funkcija, koje možete da koristite u Web aplikacijama. Većini tih funkcija ili treba proslediti neke podatke, ili one vraćaju neke podatke.

Pogledajmo naredni poziv funkcije:

```
date()('H:i, jS F')
```

Obratite pažnju na to da funkciji prosleđujemo niz znakova (tekst) između zagrada. Element naveden između zagrada zove se *argument* ili *parametar* funkcije. Argumenti su ulazni podaci na osnovu kojih funkcija daje određen rezultat.

Upotreba funkcije date()

Funkcija `date()` očekuje kao argument znakovni niz koji predstavlja format rezultata koji želite. Svako slovo u argumentu predstavlja jedan element datuma i vremena. `H` je sat u 24-časovnom formatu, `i` je minut, s vodećom nulom ako treba, `j` je dan u mesecu bez vodeće nule, `S` predstavlja redni sufiks na engleskom jeziku (u ovom slučaju „th“), a `F` je puno ime meseca.

Kompletan spisak formata koje podržava funkcija `date()` potražite u poglavlju 20.

Pristup promenljivama obrasca

Svrha obrasca za porudžbine jeste prikupljanje porudžbina koje kupci šalju. U PHP-u se lako pristupa podacima koje je kupac uneo, ali način zavisi od verzije PHP-a koju koristite, i od parametara u datoteci `php.ini`.

Promenljive obrasca

Unutar PHP skripta, svakom polju na obrascu možete pristupiti preko istoimene PHP promenljive (engl. *variable*). Imena promenljivih u PHP-u lako se prepoznaju jer svako počinje znakom za dolar (`$`).(Izostavljanje ovog znaka je uobičajena programerska greška.)

U zavisnosti od toga koju verziju PHP-a koristite i kako je ona podešena, postoje tri načina pristupanja podacima na obrascu pomoću promenljivih. Pošto te metode nemaju zvanična imena, dali smo im sledeće nadimke: *kratki*, *srednji* i *dugi* stil. U svim slučajevima, svako polje obrasca poslatog PHP skriptu dostupno je u skriptu.

Sadržaju polja `tireqty` možete pristupiti na tri načina:

```
$tireqty                        // kratki stil
$_POST['tireqty']               // srednji stil
$HTTP_POST_VARS['tireqty']      // dugi stil
```

U ovom primeru, a i u nastavku knjige, koristili smo srednji stil (tj. `$_POST['tireqty']`) za promenljive koje predstavljaju polja obrasca, ali smo radi lakše upotrebe pravili kratke verzije promenljivih. (To je preporučeni oblik od verzije PHP-a 4.2.0 nadalje.)

Vi ćete se možda opredeliti za drugačiji pristup, pa ćemo opisati sve tri metode.

- Kratki stil (`$tireqty`) je praktičan, ali zahteva da konfiguracijski parametar `register_globals` bude podešen na vrednost on (da li on standardno ima tu vrednost ili ne, zavisi od verzije PHP-a). U svim verzijama PHP-a od 4.2.0 nadalje, ovaj parametar ima vrednost off. Pre toga, standardna vrednost parametra bila je on, što je bio uzrok mnogih zabuna u vreme uvođenja izmene. Ovaj stil dozvoljava da pravite greške zbog kojih kôd neće biti bezbedan, pa se zato više ne preporučuje.
- Srednji stil (`$_POST['tireqty']`) sada se preporučuje. Prilično je praktičan, ali upotrebljiv je tek od verzije PHP-a 4.1.0, što znači da na starijim instalacijama neće raditi.
- Dugi stil (`$HTTP_POST_VARS['tireqty']`) najopširniji je. Imajte u vidu da je zastareo i da će zbog toga, pre ili kasnije, verovatno biti ukinut. Ovaj stil je nekada bio podržan u najvećem broju verzija, ali se može isključiti pomoću konfiguracijske direktive `register_long_arrays`, što poboljšava performanse.

Kada koristite kratki stil, imena promenljivih u skriptu ista su kao imena polja na HTML obrascu. Nije potrebno deklarisati promenljive, niti uraditi bilo šta drugo da biste napravili promenljive u skriptu. One se skriptu prosleđuju na isti način kao što se argumenti prosleđuju funkciji. Ako koristite kratki stil, treba samo da navedete ime

promenljive, npr. `$tireqty`. Polju `tireqty` na obrascu odgovara promenljiva `$tireqty` u skriptu.

Ovako jednostavan oblik pristupa promenljivama na prvi pogled izgleda privlačan, ali pre nego što parametar `register_globals` podesite na on, korisno je razmotriti zašto ga razvojni tim PHP-a nije tako podesio.

Direktan pristup promenljivama poput navedenog oblika vrlo je praktičan, ali, nažalost, dozvoljava da pravite programerske greške koje bi mogle da ugroze bezbednost skriptova. Kada se sve promenljive obrasca pretvore u globalne promenljive kao u navedenom primeru, ne postoji jasno razdvajanje između promenljivih koje ste vi napravili u svom kodu i nepouzdanih promenljivih koje stižu direktno od korisnika.

Ako zaboravite da svim promenljivama koje ste napravili u kodu dodelite početne vrednosti, korisnici skriptova moći će da prosleđuju svoje promenljive i vrednosti preko obrasca i da ih mešaju s vašim u kodu skripta. Ako se opredelite za praktični, kratki oblik pristupa promenljivama, zapamtite da morate obavezno zadati početne vrednosti svim lokalnim promenljivama.

Srednji stil podrazumeva učitavanje vrednosti promenljivih obrasca iz jednog od nizova, `$_POST`, `$_GET` ili `$_REQUEST`. Jedan od nizova, `$_POST` ili `$_GET`, sadržaće vrednosti svih polja obrasca. Koji niz će biti upotrebljen zavisiće od metode koja se koristi za slanje obrasca, `GET` ili `POST`. Osim toga, svi podaci poslati pomoću metode `GET` ili `POST` biće dostupni u nizu `$_REQUEST`.

Ukoliko je obrazac poslat metodom `POST`, podaci iz polja `tireqty` nalaziće se u elementu `$_POST['tireqty']`. Ako je obrazac poslat metodom `GET`, podaci će se nalaziti u elementu `$_GET['tireqty']`. U oba slučaja, podaci će biti dostupni i u elementu `$_REQUEST['tireqty']`.

Ovi nizovi su neki od novih, takozvanih *superglobalnih* nizova. Vratićemo se na njih kada budemo govorili o opsegu važenja promenljivih.

Ako koristite neku stariju verziju PHP-a, možda nećete moći da pristupite nizovima `$_POST` ili `$_GET`. Pre verzije 4.1.0, korišćeni su nizovi `$HTTP_POST_VARS` i `$HTTP_GET_VARS`. To nazivamo dugim stilom. Kao što smo već napomenuli, ovaj stil je zastareo. U njemu ne postoji ekvivalent nizu `$_REQUEST`.

Ako koristite dugi stil, podacima koje su uneli korisnici moći ćete da pristupate preko elemenata `$HTTP_POST_VARS['tireqty']` ili `$HTTP_GET_VARS['tireqty']`.

Primeri u knjizi testirani su na verziji PHP-a 5.0 i ponekad neće biti kompatibilni s verzijama starijim od 4.1.0. Preporučujemo da, gde god je moguće, koristite tekuću verziju.

Pogledajmo još jedan primer. Budući da su dugi i srednji oblik imena promenljivih pomalo nezgrapni i podrazumevaju rad s *nizovima*, koji nisu detaljnije opisani pre poglavlja 3, pravićemo kopije promenljivih koje se jednostavnije upotrebljavaju.

Da biste vrednost jedne promenljive kopirali u drugu, koristite operator za dodelu, a to je u PHP-u znak jednakosti (=). Sledeći iskaz pravi novu promenljivu po imenu `$tireqty` i kopira u nju sadržaj elementa `$_POST['tireqty']`:

```
$tireqty = $_POST['tireqty'];
```

Umetnite sledeći blok koda na početak skripta. Sličan blok koda nalaziće se na početku svih ostalih skriptova u ovoj knjizi koji obrađuju podatke sa obrazaca. Budući da taj blok koda ne ispisuje ništa, svejedno je da li ćete ga postaviti ispred ili iza oznake `<html>` i drugih početnih HTML oznaka. Takav blok koda obično stavljamo na početak skripta da bismo lakše mogli da ga pronađemo.

```
<?php
  //formira promenljive s kratkim imenima
  $tireqty = $_POST['tireqty'];
  $oilqty = $_POST['oilqty'];
  $sparkqty = $_POST['sparkqty'];
?>
```

Navedeni kôd pravi tri nove promenljive, `$tireqty`, `$oilqty` i `$sparkqty`, koje sadrže podatke poslate iz obrasca metodom `POST`.

Da bi skript davao vidljiv rezultat, dodajte na kraj sledeće redove:

```
echo '<p>Your order is as follows: </p>';
echo $tireqty.' tires<br />';
echo $oilqty.' bottles of oil<br />';
echo $sparkqty.' spark plugs<br />';
```

Zasad još niste ispitali vrednosti promenljivih da biste znali da li su u sva polja obrasca unete prihvatljive vrednosti. Namerno unesite pogrešne podatke i posmatrajte šta se dešava. Pošto pročitate preostali deo ovog poglavlja, možda ćete pokušati da u ovom skriptu dodate kôd za proveru ispravnosti podataka. Ako sada otvorite ovaj skript u svom čitaču, trebalo bi da vidite rezultat nalik na onaj sa slike 1.4. Razume se, prikazane vrednosti zavisiće od onih koje unesete na obrascu.

U narednim odeljcima opisano je nekoliko zanimljivih elemenata u vezi sa ovim skriptom.

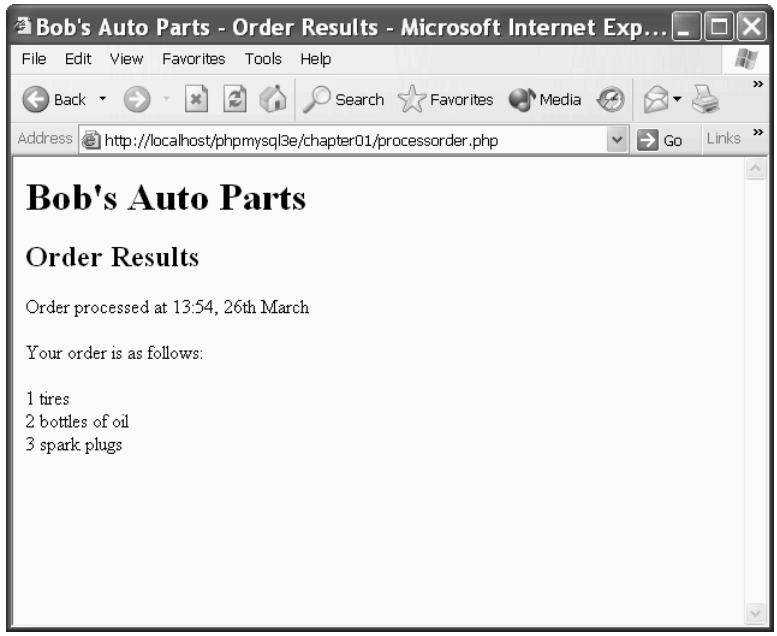
Nadovezivanje tekstualnih podataka

Navedeni skript ispisuje pomoću komande `echo` vrednost koju je korisnik uneo u polje obrasca, kojoj sledi određeni opisni tekst. Ako pažljivije pogledate iskaze `echo`, videćete da se između imena promenljive i pratećeg teksta nalazi tačka (.), kao u sledećem iskazu:

```
echo $tireqty.' tires<br />';
```

Tačka je operator za nadovezivanje znakovnih podataka (blokova teksta) (engl. *string concatenation*). Često ćete je koristiti kada komandom `echo` prosledite izlazne podatke čitaču. Tako nećete morati da pišete više komandi `echo`.

Ime promenljive (koja nije niz) možete upisati između navodnika, zajedno s drugim tekstom koji prosleđujete komandi `echo`. PHP će u tom slučaju zameniti ime promenljive odgovarajućom vrednošću. (Pošto su nizovi nešto složeniji, kombinovanje nizova i znakovnih podataka objasnićemo u poglavlju 4, „Rad sa znakovnim vrednostima i regularnim izrazima“.)



Slika 1.4 Vrednostima promenljivih koje je korisnik uneo u obrazac lako se pristupa u skriptu `processorder.php`.

Razmotrite sledeće:

```
echo "$tireqty tires<br />";
```

Navedeni iskaz je ekvivalentan prvom iskazu u ovom odeljku. Oba formata su ispravna i od vas zavisi koji ćete koristiti. Ovaj postupak – zamena imena promenljive njenom vrednošću unutar bloka teksta – zove se interpoliranje.

Imajte u vidu da se interpoliranje primenjuje isključivo u blokovima teksta omeđenim navodnicima. Imena promenljivih umetnuta u tekst napisan između polunavodnika ne daju isti rezultat. Sledeći iskaz:

```
echo '$tireqty tires<br />';
```

šalje čitaču tekst `"$tireqty tires<br />"` . Ime promenljive unutar teksta napisanog između navodnika biće zamenjeno njenom vrednošću. U tekstu napisanom između polunavodnika, ime promenljive ili svaki drugi tekst ostaju neizmenjeni.

Promenljive i literali

Promenljive i tekst koji spajate u svakom iskazu `echo` navedenog skripta jesu elementi različitog tipa. Promenljive su simboli za podatke, dok je tekst vrednost podataka. Sirovi podaci se u programima nazivaju *literali* (engl. *literals*) da bi se razlikovali od promenljivih (engl. *variables*). Ime `$tireqty` je promenljiva – koja predstavlja podatke

koje je uneo korisnik. S druge strane, `' tires<br />'` je literal čija se vrednost tumači doslovno. U stvari, ne sasvim doslovno. Sećate li se drugog primera? PHP je zamenio ime promenljive `$tireqty` vrednošću promenljive.

Ne zaboravite da u PHP-u postoje dve vrste znakovnih podataka: zadate između navodnika i između polunavodnika. PHP će pokušati da izračuna vrednost znakovnih podataka zadatih između navodnika, što ste već videli. Znakovni podaci zadati između polunavodnika obrađuju se kao pravi literali.

Nedavno je dodat i treći način zadavanja znakovnih podataka. U PHP4 dodata je heredoc sintaksa (<<<), poznata korisnicima Perla. Heredoc sintaksa omogućava zadavanje dužih znakovnih podataka u lako razumljivom obliku, tako što se zada i graničnik kojim se znakovni podatak završava. U narednom primeru, znakovni podatak se deli u tri reda pre nego što ga obradi komanda `echo`:

```
echo <<<kraj
  prvi red
  drugi red
  treći red
kraj
```

Graničnik `kraj` može biti bilo šta.Važno je samo da se ne pojavljuje u tekstu. Da biste zadali znakovnu vrednost u heredoc formatu, dodajte graničnik na početak reda.

Znakovne vrednosti zadate u heredoc formatu interpoliraju se isto kao one zadate između navodnika.

Identifikatori

Identifikatori su imena promenljivih. (Imena funkcija i klasa takođe su identifikatori; o funkcijama i klasama govorimo u petom i šestom poglavlju.) U vezi sa upotrebom identifikatora, trebalo bi da znate sledeća osnovna pravila:

- Identifikatori mogu biti bilo koje dužine i mogu da se sastoje od slova, brojeva, podvlaka.
- Identifikatori ne mogu da počinju cifrom.
- U PHP-u se pravi razlika između malih i velikih slova u imenima identifikatora. `$tireqty` nije isto što i `$TireQty`, a mešanje jednog s drugim tipična je programerska greška. Imena funkcija su izuzetak od ovog pravila – njihova imena se mogu pisati i malim i velikim slovima.
- Promenljiva može da ima isto ime kao i funkcija. Međutim, ta tehnika dovodi do zabune i trebalo bi je izbegavati. Ne možete napraviti funkciju koja će imati isto ime kao i postojeća funkcija.

Promenljive koje deklariše korisnik

Osim promenljivih koje su prosleđene iz HTML obrasca, u skriptovima možete da koristite i promenljive koje sami deklarišete.

Jedna od karakteristika PHP-a jeste to što ne treba da deklarišete promenljive pre nego što ih upotrebite. Promenljiva će biti napravljena kada joj prvi put dodelite vrednost – dodatne informacije potražite u sledećem odeljku.

Dodeljivanje vrednosti promenljivama

Promenljivama dodeljujete vrednost pomoću operatora za dodelu vrednosti, =, kao što ste to uradili kada ste kopirali vrednost jedne promenljive u drugu. Na Bobovoj Web lokaciji, želite da izračunate ukupan broj naručenih artikala i njihovu ukupnu vrednost. Za čuvanje tih iznosa možete da napravite dve promenljive. Na početku ćete obema dodeliti vrednost nula.

Na kraj PHP skripta dodajte sledeće redove:

```
$totalqty = 0;
$totalamount = 0.00;
```

Svaki od navedena dva reda pravi promenljivu i daje joj vrednost literala. Promenljivama možete da dodelite i vrednosti drugih promenljivih, na primer:

```
$totalqty = 0;
$totalamount = $totalqty;
```

Tipovi promenljivih

Tip promenljive se odnosi na vrstu podataka koji se u njoj nalaze. PHP-ov skup tipova podataka neprestano se širi.

Tipovi podataka u PHP-u

PHP podržava sledeće osnovne tipove podataka:

- **integer** – celobrojni, koristi se za cele brojeve
- **double** ili **float** – pokretni zarez dvostruke preciznosti, koristi se za realne brojeve
- **string** – znakovni, koristi se za znakovne podatke
- **boolean** – logički, koristi se za podatke tipa tačno ili netačno
- **array** – niz, koristi se za čuvanje više podataka istog tipa (videti poglavlje 3, „Upotreba nizova“)
- **object** – objekat, koristi se za čuvanje primeraka (instanci) klasa (videti poglavlje 6, „Objektno orijentisani PHP“)

Postoje i dva specijalna tipa podataka: NULL i resurs. Promenljive kojima nije dodeljena vrednost, koje su nedefinisane ili kojima je izričito dodeljena vrednost NULL, tipa su NULL. Neke ugrađene funkcije (npr. za rad s bazama podataka) vraćaju promenljive tipa resurs, koje predstavljaju spoljne resurse (kao što su veze s bazama podataka). Skoro je izvesno da s tim tipom nećete direktno raditi, ali se često koriste za vrednosti koje funkcije vraćaju ili koje im se prosleđuju.

Provera tipa podataka

PHP je jezik s veoma slabom proverom tipa podataka. U većini programskih jezika, promenljive mogu da sadrže samo jedan tip podataka koji se pre upotrebe promenljive mora deklarisati, kao što je slučaj u C-u. U PHP-u se tip promenljive određuje na osnovu vrednosti koja joj je dodeljena.

Na primer, kada smo napravili promenljive `$totalqty` i `$totalamount`, njihovi početni tipovi bili su određeni ovako:

```
$totalqty = 0;
$totalamount = 0.00;
```

S obzirom na to da smo promenljivoj `$totalqty` dodelili vrednost 0, što je ceo broj, to je sada promenljiva celobrojnog tipa. Na isti način, promenljiva `$totalamount` sada je u formatu pokretnog zareza dvostruke preciznosti.

Skriptu bismo sada mogli da dodamo sledeći red:

```
$totalamount = 'Hello';
```

Promenljiva `$totalamount` bi u tom slučaju bila znakovnog tipa. PHP menja tip promenljive prema onome što se u njoj nalazi.

Mogućnost promene tipa podataka u hodu može biti veoma korisna. Ne zaboravite da PHP „automatski“ zna koji ste tip podataka stavili u promenljive. PHP će vratiti podatak upisanog tipa prilikom učitavanja vrednosti promenljive.

Konverzija tipa podataka

Operator za konverziju omogućava privremenu promenu tipa promenljive ili vrednosti. Postupak se odvija identično kao u C-u. Novi privremeni tip upišite u zagradu ispred promenljive koju želite da konvertujete.

Na primer, u deklaraciji dve promenljive iz prethodnog odeljka možete primeniti konverziju tipa podataka:

```
$totalqty = 0;
$totalamount = (double)$totalqty;
```

Drugi red znači „pročitaj vrednost promenljive `$totalqty`, tumači je kao vrednost tipa double i smesti je u promenljivu `$totalamount`“. Tip promenljive `$totalamount` biće double. Tip konvertovane promenljive se ne menja, tako da `$totalqty` ostaje celobrojnog tipa.

Promenljive promenljive

PHP podržava još jedan tip promenljive – promenljivu promenljivu (engl. *variable variable*). Takve promenljive omogućavaju da ime promenljive menjate dinamički.

Kao što vidite, PHP u ovoj oblasti daje dosta slobode – svi programski jezici puštaju da menjate vrednost promenljive, ali nema mnogo jezika koji dopuštaju da menjate tip promenljive. Još ih manje prihvata da menjate ime promenljive.

Promenljiva promenljiva se dobija kada se vrednost jedne promenljive koristi kao ime druge. Na primer:

```
$varname = 'tireqty';
```

Umesto promenljive `$tireqty`, možete da koristite `$$varname`. Na primer, promenljivoj `$tireqty` možete da dodelite vrednost sledećim iskazom:

```
$$varname = 5;
```

što je ekvivalentno izrazu:

```
$tireqty = 5;
```

Upotreba ove vrste promenljivih možda vam zasad nije sasvim jasna, ali ćemo ih objasniti nešto kasnije. Umesto da svaku promenljivu obrasca navodite i koristite zasebno, možete da upotrebite petlju i promenljivu i da ih sve obradite automatski. U odeljku o petljama `for` nalazi se primer koji ilustruje tu tehniku.

Deklarisanje i upotreba konstanti

Kao što ste videli ranije, vrednost koja se čuva u promenljivoj možete da menjate kad god vam zatreba. Osim toga, možete da deklarišete i konstante. Konstanta predstavlja određenu vrednost, isto kao i promenljiva, ali se ta vrednost zadaje jednom i potom se više ne može menjati.

Na primer, cene artikala koji se prodaju možete čuvati kao konstante. Te konstante možete da definišete pomoću funkcije define.

```
define('TIREPRICE', 100);
define('OILPRICE', 10);
define('SPARKPRICE', 4);
```

Dodajte navedene redove u skript. Sada imate tri konstante pomoću kojih možete izračunati ukupan iznos porudžbine koju je kupac poslao.

Obratite pažnju na to da su sva imena konstanti napisana velikim slovima. Ta konvencija, pozajmljena iz C-a, olakšava razlikovanje promenljivih i konstanti. Ona nije obavezna, ali će olakšati čitanje i održavanje koda.

Važna razlika između konstanti i promenljivih jeste ta da se ispred konstante ne piše simbol za dolar. Ako želite da upotrebite vrednost konstante, upišite samo njeno ime. Na primer, da biste upotrebili konstantu koju ste upravo napravili, napisaćete:

```
echo TIREPRICE;
```

U izvršnom okruženju za PHP definisan je velik broj konstanti. Možete ih pregledati pomoću komande `phpinfo()`:

```
phpinfo();
```

koja prikazuje spisak unapred definisanih konstanti i promenljivih, kao i neke druge korisne informacije. O nekima od njih biće reči kasnije.

Još jedna razlika između konstanti i promenljivih jeste ta da konstante mogu sadržati samo podatke logičkog, celobrojnog ili znakovnog tipa i numeričke podatke s pokretnim zarezom. Navedeni tipovi podataka poznati su pod zajedničkim imenom skalarni tipovi.

Opseg važenja promenljive

Izraz *opseg važenja* (engl. *scope*) odnosi se na mesta u skriptu na kojima je data promenljiva vidljiva. U PHP-u postoji šest mogućnosti opsega važenja:

- Ugrađene superglobalne promenljive vide se u celom skriptu.
- Konstante, pošto ih deklarišete, uvek imaju globalnu vidljivost; to znači da su upotrebljive unutar i izvan funkcija.
- Globalne promenljive, deklarisane u skriptu vidljive su svuda u skriptu, ali *ne i unutar funkcija*.
- Promenljiva koju napravite unutar funkcije, a date joj isto ime kao promenljivoj koja je deklarisana kao globalna, upućuje na globalnu promenljivu istog imena.
- Promenljive koje napravite unutar funkcije, a deklarišete kao statičke promenljive, ne vide se izvan funkcija, ali čuvaju svoje vrednosti između dva izvršavanja funkcije. (Ovu ideju detaljnije objašnjavamo u poglavlju 5, „Višekratna upotreba koda i pisanje funkcija“.)
- Promenljive koje napravite unutar funkcije lokalne su za funkciju i prestaju da postoje kada prestane izvršavanje funkcije.

Od PHP-a 4.1 nadalje, za nizove `$_GET` i `$_POST`, i za neke druge posebne promenljive, važe drugačija pravila za opseg važenja. To su superglobalne promenljive koje se vide svuda, u funkcijama i van njih.

Kompletan spisak superglobalnih promenljivih je:

- `$GLOBALS` – niz svih globalnih promenljivih. (Slično rezervisanoj reči `global`, ovaj niz omogućava pristupanje globalnim promenljivama; na primer: `$GLOBALS['nekapromenljiva']`)
- `$_SERVER` – niz serverskih promenljivih
- `$_GET` – niz promenljivih prosleđenih skriptu metodom `GET`
- `$_POST` – niz promenljivih prosleđenih skriptu metodom `POST`
- `$_COOKIE` – niz kolačića
- `$_FILES` – niz promenljivih koje se odnose na datoteke poslate s klijentskog računara
- `$_ENV` – Niz promenljivih okruženja
- `$_REQUEST` – Niz svih promenljivih koje je korisnik poslao; obuhvata i nizove `$_GET`, `$_POST` i `$_COOKIE`
- `$_SESSION` – Niz promenljivih sesije.

Sve navedene promenljive opisaćemo kasnije, kako se budu pojavljivale u knjizi. Opseg važenja promenljivih detaljnije ćemo objasniti kada budemo govorili o funkcijama. Za sada će sve promene koje koristimo biti globalne promenljive, osim ako je drugačije naglašeno.

Operatori

Operatori su simboli koji omogućavaju izvršavanje operacija nad vrednostima i promenljivama. Neke od njih ćete upotrebiti da biste na porudžbini izračunali ukupan iznos i porez.

Već smo spomenuli dva operatora, operator dodele i operator nadovezivanja znakovnih vrednosti. Pogledajmo sada kompletan spisak.

Operatori mogu imati jedan, dva ili tri argumenta, s tim što većina ima dva. Na primer, operator dodele ima dva argumenta – memorijsku lokaciju na levoj strani simbola = i izraz na desnoj strani. Ti argumenti se nazivaju *operandi* – vrednosti kojima se operiše.

Aritmetički operatori

Aritmetički operatori su prilično jednostavni – to su obični znakovi za računske operacije. Prikazani su u tabeli 1.1.

Tabela 1.1 Aritmetički operatori jezika PHP

Operator	Ime	Primer
`+`	sabiranje	`$a + $b`
`-`	oduzimanje	`$a - $b`
`*`	množenje	`$a * $b`
`/`	deljenje	`$a / $b`
`%`	modulo	`$a % $b`

Rezultat operacije svakog operatora može se sačuvati u nekoj promenljivoj. Na primer,

```
$result = $a + $b;
```

Sabira se i oduzima kao u matematici. Rezultat tih operacija je zbir, odnosno razlika vrednosti koje se nalaze u promenljivama `$a` i `$b`.

Znak za oduzimanje (-) možete da koristite i kao unarni operator (operator sa jednim argumentom, odnosno operandom) za negativne brojeve; na primer:

```
$a = -1;
```

Množenje i deljenje takođe se odvijaju kao i matematičke operacije istog imena. Obratite pažnju na to da se kao znak za množenje koristi zvezdica (umesto uobičajenog znaka množenja), a kao znak za deljenje služi kosa crta.

Operator modulo vraća ostatak celobrojnog deljenja promenljive `$a` promenljivom `$b`. Pogledajte sledeći odlomak:

```
$a = 27;
$b = 10;
$result = $a%$b;
```

Vrednost promenljive `$result` biće ostatak celobrojnog deljenja 27 sa 10, dakle 7.

Imajte u vidu da se aritmetički operatori obično primenjuju na brojeve. Ako ih primenite na znakovne podatke, PHP će pokušati da pretvori znakovnu vrednost u broj. Ukoliko znakovna vrednost sadrži slovo „e“ ili „E“, tumačiće se kao broj u naučnoj notaciji i biće pretvorena u tip float; u protivnom, biće pretvorena u celobrojni tip. PHP će upotrebiti cifre na početku teksta da bi izveo vrednost – ako ih nema, vrednost pretvorenog znakovnog podatka biće nula.

Operatori za znakovne vrednosti

Već ste videli i upotrebili jedini operator za znakovne vrednosti (engl. *string operator*). Operator nadovezivanja znakovnih vrednosti možete upotrebiti za spajanje dve znakovne vrednosti, otprilike isto kao što biste upotrebili operator sabiranja za sabiranje dva broja.

```
$a = "Bob's ";
$b = 'Auto Parts';
$result = $a.$b;
```

Promenljiva `$result` će u tom slučaju sadržati znakovnu vrednost „`Bob's Auto Parts`“.

Operatori dodele

Već ste upoznali osnovni operator dodele (=). O tom znaku uvek govorite kao o operatoru dodele i čitajte ga kao „dobija vrednost“. Na primer:

```
$totalqty = 0;
```

Navedeno bi trebalo čitati kao „`$totalqty` dobija vrednost nula“. Razlog ćemo objasniti kada budemo govorili o operatorima poređenja u nastavku ovog poglavlja.

Rezultujuća vrednost operacije dodele

Operator dodele takođe vraća određenu vrednost, slično drugim operatorima. Na primer, kada zadate:

```
$a + $b
```

vrednost ovog izraza je zbir promenljivih `$a` i `$b`. Slično tome, vrednost izraza

```
$a = 0;
```

je nula.

To omogućava da napišete izraze poput:

```
$b = 6 + ($a = 5);
```

Promenljiva `$b` dobija vrednost 11. Za sve operatore dodele uglavnom važi sledeće pravilo: vrednost celog iskaza dodele je vrednost koja je dodeljena operandu na levoj strani.

Kada izračunavate vrednost određenog izraza, možete upotrebiti zagrade da biste povećali prioritet izračunavanja podizraza, kao što smo ovde uradili. Princip je potpuno isti kao u matematici.

Kombinovani operatori dodele

Osim jednostavnog operatora dodele, postoji i skup kombinovanih operatora dodela. Svaki od njih je skraćenica za izvršavanje određene operacije nad promenljivom i upisivanja rezultata u tu promenljivu. Na primer:

```
$a += 5;
```

isto je kao da ste napisali:

```
$a = $a + 5;
```

Kombinovani operatori dodele postoje za svaki aritmetički operator i za operator nadovezivanja znakovnih vrednosti.

Kombinovani operatori dodele i njihovi rezultati prikazani su u tabeli 1.2.

Tabela 1.2 Kombinovani operatori dodele

Operator	Upotreba	Ekivalentan izrazu
`+=`	`$a += $b`	`$a = $a + $b`
`-=`	`$a -= $b`	`$a = $a - $b`
`*=`	`$a *= $b`	`$a = $a * $b`
`/=`	`$a /= $b`	`$a = $a / $b`
`%=`	`$a %= $b`	`$a = $a % $b`
`.=`	`$a .= $b`	`$a = $a . $b`

Prefiksno i sufiksno uvećanje i umanjenje

Prefiksni i sufiksni operatori uvećanja `++`, i umanjenja `--`, slični su operatorima `+=` i `-=`, uz neke specifičnosti.

Operatori uvećanja imaju dva efekta – oni prvo povećavaju tekuću vrednost, a zatim je dodeljuju. Pogledajte ovo:

```
$a=4;
echo ++$a;
```

U drugom redu upotrebljen je prefiksni operator uvećanja, koji se tako zove zato što se `++` pojavljuje ispred `$a`. Prvo se vrednost `$a` povećava za 1, a zatim se ta uvećana vrednost dodeljuje istoj promenljivoj. U ovom slučaju vrednost promenljive `$a` povećana je na 5, a zatim se vrednost 5 učitava iz promenljive i ispisuje.Vrednost celog izraza je 5 (imajte u vidu da je početna vrednost `$a` izmenjena: rezultat nije samo `$a + 1`).

Međutim, ako je `++` posle `$a`, to je sufiksni operator uvećanja, koji deluje drugačije. Pogledajte ovo:

```
$a=4;
echo $a++;
```

U ovom slučaju, efekat je suprotan, tj. tekuća vrednost promenljive `$a` prvo se učitava i ispisuje, a zatim se povećava.Vrednost celog izraza je 4 i vrednost se ispisuje. Vrednost `$a` posle ovog iskaza je 5.

Kao što verovatno pretpostavljate, slično se ponaša i operator `--`, s tim što se vrednost `$a` umanjuje, a ne povećava.

Reference

Operator referenca (`&`, ampersend), može se koristiti u kombinaciji s operatorima dodele. Kada se jedna promenljiva dodeljuje drugoj, obično se pravi kopija prve promenljive, koja se smešta negde drugde u memoriju. Na primer:

```
$a = 5;
$b = $a;
```

Navedeni redovi koda prave kopiju vrednosti u promenljivoj `$a` i smeštaju je u promenljivu `$b`. Ukoliko zatim izmenite vrednost promenljive `$a`, vrednost `$b` se neće promeniti.

```
$a = 7; // $b je i dalje 5
```

Kopiranje možete da izbegnete pomoću operatora reference `&`. Na primer:

```
$a = 5;
$b = &$a;
$a = 7; // $a i $b sada su 7
```

Reference mogu malo da zbune. Imajte u vidu da se referenca ponaša sličnije drugom imenu nego pokazivaču. I `$a` i `$b` upućuju na isti blok memorije. To možete izmeniti ako poništite definiciju jedne od promenljivih, na sledeći način:

```
unset($a);
```

Poništavanjem definicije promenljive `$b` (7) ne menja se njena vrednost, ali se prekida veza između `$a` i vrednosti 7 koja se čuva u memoriji.

Operatori poređenja

Operatori poređenja se koriste za poređenje dve vrednosti. Izrazi u kojima upotrebite te operatore vraćaju logičku vrednost `true` ili `false`, u zavisnosti od rezultata poređenja.

Operator jednakosti

Operator poređenja (`==`, dva znaka jednakosti) omogućava da ispitate da li su dve vrednosti jednake. Na primer:

```
$a == $b
```

ispituje da li su vrednosti `$a` i `$b` iste. Rezultat koji vraća ovaj izraz biće `true` ako su jednaki, a `false` ako nisu.

Ovaj izraz se lako brka sa operatorom dodele (`=`). Ako upotrebite pogrešan operator, neće doći do greške, ali obično nećete dobiti ni željeni rezultat. Uglavnom, numeričke vrednosti različite od nule odgovaraju logičkoj vrednosti `true`, dok nula odgovara logičkoj vrednosti `false`. Na primer, ako dve promenljive imaju sledeće početne vrednosti:

```
$a = 5;
$b = 7;
```

rezultat izraza `$a = $b` biće `true`. Zašto? Vrednost izraza `$a = $b` vrednost je dodeljena levoj strani, što je u ovom slučaju 7. Pošto je to vrednost različita od nule, rezultat izraza je `true`. Ukoliko biste napisali to umesto izraza `$a == $b`, što daje `false`, uveli biste u kôd logičku grešku koju ćete možda veoma teško pronaći. Uvek proverite kako koristite ova dva operatora i da li ste upotrebili ispravan operator.

Upotreba operatora za dodelu umesto operatora za ispitivanje jednakosti je greška koja se lako pravi i verovatno ćete je i vi napraviti više puta.

Ostali operatori poređenja

PHP podržava još neke operatore poređenja, koji su prikazani u tabeli 1.3.

Obratite pažnju na operator identičnosti (`===`), koji vraća `true` samo ako su oba operanda jednaka i istog tipa. Na primer, rezultat poređenja `0 == '0'` je `true`, ali `0 === '0'` nije, jer je prva nula celobrojnog tipa, a tip druge nule je znakovni.

Tabela 1.3 Operatori poređenja

Operator	Ime	Upotreba
`==`	jednako	`$a == $b`
`===`	identično	`$a === $b`
`!=`	različito	`$a != $b`
`!==`	nije identično	`$a !== $b`
`<>`	različito	`$a <> $b`
`<`	manje od	`$a < $b`
`>`	veće od	`$a > $b`
`<=`	manje od ili jednako	`$a <= $b`
`>=`	veće od ili jednako	`$a >= $b`

Logički operatori

Logički operatori se koriste za kombinovanje rezultata logičkih izraza. Na primer, želite da utvrdite da li je vrednost promenljive `$a` između 0 i 100. Rešenje je da proverite da li su ispunjena oba uslova: `$a >= 0` i `$a <= 100`. To omogućava operator logičke konjunkcije (`AND`):

```
$a >= 0 && $a <=100
```

PHP podržava logičke operacije konjunkcije (AND), disjunkcije (OR), isključive disjunkcije (XOR) i negacije (NOT).

Skup logičkih operatora i njihova upotreba prikazani su u tabeli 1.4.

Tabela 1.4 PHP-ovi logički operatori

Operator	Ime	Upotreba	Rezultat
`!`	negacija	`!$b`	Vraća `true` ako je `$b` `false` i obrnuto
`&&`	konjunkcija	`$a && $b`	Vraća `true` ako su `$a` i `$b` `true`; u protivnom vraća `false`

Tabela 1.4 PHP-ovi logički operatori (nastavak)

Operator	Ime	Upotreba	Rezultat
`||`	disjunkcija	`$a || $b`	Vraća `true` ako su `$a` ili `$b` ili oba `true`; u protivnom vraća `false`
`and`	konjunkcija	`$a and $b`	Isto kao `&&`, ali s nižim prioritetom
`or`	disjunkcija	`$a or $b`	Isto kao `||`, ali s nižim prioritetom

Operatori `and` i `or` imaju niži prioritet od operatora `&&` i `||`. O prioritetima operatora biće više reči u nastavku ovog poglavlja.

Operatori nad bitovima

Operatori nad bitovima omogućavaju da ceo broj obrađujete kao grupu bitova koja ga predstavlja u memoriji.

U PHP-u verovatno ih nećete često upotrebljavati. Operatori nad bitovima prikazani su u tabeli 1.5.

Tabela 1.5 Operatori nad bitovima

Operator	Ime	Upotreba	Rezultat
`&`	konjunkcija	`$a & $b`	Bitovi koji su aktivni u `$a` i `$b` aktivni su u rezultatu
`|`	disjunkcija	`$a | $b`	Bitovi koji su aktivni u `$a` ili `$b` aktivni su u rezultatu
`~`	negacija	`~$a`	Bitovi koji su aktivni u `$a` nisu aktivni u rezultatu i obrnuto
`^`	isključiva disjunkcija	`$a ^ $b`	Bitovi koji su aktivni ili u `$a` ili u `$b` ali ne u oba, aktivni su u rezultatu
`<<`	pomeranje ulevo	`$a << $b`	Pomera bitove `$a` ulevo za `$b` mesta
`>>`	pomeranje udesno	`$a >> $b`	Pomera bitove `$a` udesno za `$b` mesta

Ostali operatori

Osim operatora koje smo do sada predstavili, postoje i drugi.

Zarez se koristi za razdvajanje argumenata funkcije i stavki na spisku.

Dva posebna operatora, `new` i `->`, koriste se za pravljenje objekta, odnosno pristupanje članovima objekata. O njima će više reči biti u poglavlju 6.

Postoje još tri operatora koje ćemo ukratko opisati.

Uslovni operator

Operator `?:` deluje isto kao u C-u. Oblik mu je sledeći:

```
uslov ? vrednost ako je uslov ispunjen : vrednost ako nije ispunjen
```

Uslovni operator deluje na sličan način kao jedna od verzija iskaza `if-else`, o kome govorimo kasnije u ovom poglavlju.

Jednostavan primer:

```
($ocena > 5 ? 'Položio' : 'Pao');
```

Ovaj izraz dodeljuje `'Položio'` ili `'Pao'`, u zavisnosti od ocene studenta.

Operator zanemarivanja greške

Operator zanemarivanja greške, `@`, može se upotrebiti ispred svakog izraza. Na primer,

```
$a = @(57/0);
```

Bez operatora `@`, izvršno okruženje bi generisalo upozorenje „delite nulom". Ako upotrebite ovaj operator, greška se zanemaruje.

Ukoliko na ovaj način zanemarite upozorenja o greškama, trebalo bi da napišete kôd za obradu grešaka. Ako je uključena konfiguracijska opcija `track_errors`, tekst poruka o grešci nalaziće se u globalnoj promenljivoj `$php_errormsg`.

Operator izvršenja

Reč je zapravo o dva operatora – paru inverznih polunavodnika (``). Inverzni polunavodnik nije isto što i običan polunavodnik (apostrof); obično se nalazi na istom tasteru gde i simbol ~.

Sve što se nalazi između inverznih polunavodnika PHP pokušava da izvrši kao komandu zadatu na komandnoj liniji servera. Rezultat komande predstavlja vrednost izraza.

Na primer, u operativnim sistemima tipa Unix možete da upotrebite:

```
$out = `ls -la`;
echo '<pre>'.$out.'</pre>';
```

Ekvivalent tome na Windows serveru bilo bi:

```
$out = `dir c:`;
echo '<pre>'.$out.'</pre>';
```

Obe verzije će generisati spisak datoteka u direktorijumu i dodeliti ga promenljivoj `$out`.

Postoje i drugi načini izvršavanja komandi na serveru. O njima ćemo detaljnije govoriti u poglavlju 18.

Operatori za rad s nizovima

Postoji više operatora za rad s nizovima. Operator za element niza (`[]`) omogućava pristup pojedinim elementima niza. Operator `=>` može se, u nekim kontekstima, primeniti i na nizove. Operatori za rad s nizovima opisani su u poglavlju 3.

Na raspolaganju su i drugi operatori za rad s nizovima, koji su takođe detaljno opisani u poglavlju 3,,,Upotreba nizova", a ovde ih navodimo samo radi informacije.

Obratite pažnju na to da svaki operator iz tabele 1.6 ima odgovarajući ekvivalent koji deluje na promenljive skalarnog tipa. Ako ne gubite iz vida da operator + zbraja skalarne tipove i formira uniju kad se primeni na nizove, čak i ako vas ne zanima aritmetika na kojoj se zasniva to ponašanje, delovanje tog operatora biće vam sasvim jasno i logično.

Tabela 1.6 PHP-ovi operatori za rad s nizovima

Operator	Ime	Upotreba	Rezultat
`+`	unija	`$a + $b`	Vraća niz koji se sastoji od svih elemenata `$a` i `$b`.
`==`	jednako	`$a = $b`	Vraća `true` ako `$a` i `$b` imaju jednake elemente.
`===`	identično	`$a === $b`	Vraća `true` ako `$a` i `$b` imaju jednake elemente u jednakom redosledu.
`!=`	različito	`$a != $b`	Vraća `true` ako je `$a` različit od `$b`.
`<>`	različito	`$a <> $b`	Vraća `true` ako je `$a` različit od `$b`.
`!==`	nije identično	`$a !== $b`	Vraća `true` ako `$a` nije identičan `$b`.

Operator za utvrđivanje tipa

Za utvrđivanje tipa objekta postoji samo jedan operator: `instanceof`. Koristi se u objektno orijentisanom programiranju, a ovde ga pominjemo samo radi informacije. (Objektno orijentisano programiranje opisano je u poglavlju 6.)

Operator `instanceof` omogućava da ispitate da li je određeni objekat primerak (instanca) zadate klase, kao u sledećem primeru:

```
class nekaKlasa{};
nekiObjekat = new nekaKlasa();
if (nekiObjekat instanceof nekaKlasa)
   echo "nekiObjekat je primerak klase nekaKlasa";
```

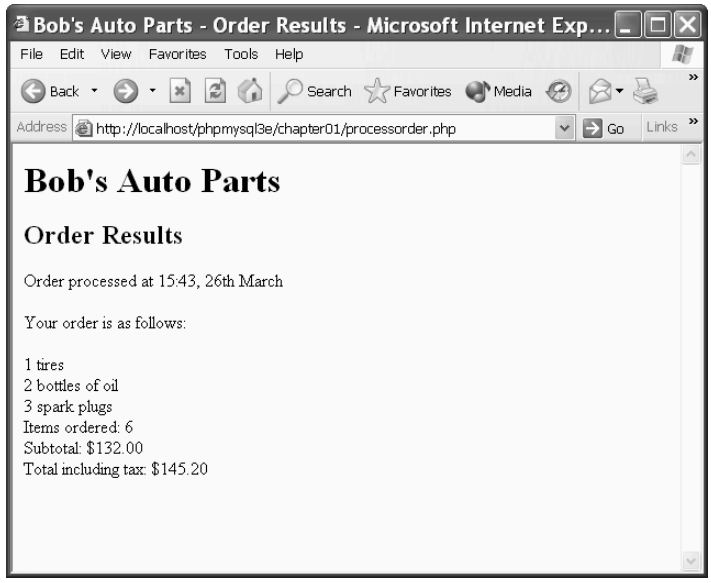
Upotreba operatora: izračunavanje ukupnog iznosa na obrascu

Pošto sada znate kako da koristite operatore u PHP-u, možete izračunati ukupan iznos i iznos poreza u porudžbini.

Na kraj PHP skripta dodajte sledeći blok koda:

```
$totalqty = 0;
$totalqty = $tireqty + $oilqty + $sparkqty;
echo 'Items ordered: '.$totalqty.'<br />';

$totalamount = 0.00;

define('TIREPRICE', 100);
define('OILPRICE', 10);
define('SPARKPRICE', 4);

$totalamount = $tireqty * TIREPRICE
             + $oilqty * OILPRICE
             + $sparkqty * SPARKPRICE;

echo 'Subtotal: $'.number_format($totalamount,2).'<br />';

$taxrate = 0.10;  // stopa poreza je 10%
$totalamount = $totalamount * (1 + $taxrate);
echo 'Total including tax: $'.number_format($totalamount,2).'<br />';
```

Ako osvežite stranicu u prozoru čitača, trebalo bi da vidite rezultat nalik na onaj sa slike 1.5.



Slika 1.5 Ukupan iznos je izračunat, formatiran i prikazan.

Kao što vidite, upotrebili smo nekoliko operatora: operatore sabiranja (+) i množenja (*) – za izračunavanje zbira, i operator nadovezivanja (.) – za definisanje rezultata.

Osim toga, upotrebili smo funkciju `number_format` za formatiranje zbira. To je funkcija iz PHP-ove biblioteke Math.

Ako pažljivo pogledate izraze, možda ćete se zapitati zašto su računati baš tim redosledom. Pogledajte, na primer, sledeći iskaz:

```
$totalamount = $tireqty * TIREPRICE
             + $oilqty * OILPRICE
             + $sparkqty * SPARKPRICE;
```

Izgleda da je ukupan zbir tačan, ali zašto su množenja izvršena pre sabiranja? Odgovor je u prioritetima operatora, dakle u redosledu njihovog izračunavanja.

Prioriteti i asocijativnost: izračunavanje izraza

Svaki operator ima određeni prioritet, ili redosled kojim se izračunava.

Operatori takođe imaju asocijativnost, tj. redosled kojim se operatori jednakog prioriteta izračunavaju. Asocijativnost može biti sleva nadesno (skraćeno se naziva *leva*), zdesna nalevo (skraćeno *desna*), ili *nije bitna* (u tabeli n\b).

Tabela 1.7 prikazuje prioritete i asocijativnost operatora.

U ovoj tabeli, operatori pri vrhu imaju najmanji prioritet, a on raste ka dnu tabele.

Tabela 1.7 Prioritet operatora

Asocijativnost	Operatori
leva	`,`
leva	`or`
leva	`xor`
leva	`and`
desna	`print`
leva	`= += -= *= /= .= %= &= |= ^= ~= <<= >>=`
leva	`? :`
leva	`||`
leva	`&&`
leva	`|`
leva	`^`
leva	`&`
n/b	`== != ===`
n/b	`< <= > >=`
leva	`<< >>`
leva	`+ - .`
leva	`* / %`
desna	`! ~ ++ -- (int) (double) (string) (array) (object) @`
desna	`[]`
n/b	`new`
n/b	`()`

Obratite pažnju na to da još nismo pomenuli operator s najvišim prioritetom: obične zagrade. One deluju tako što povećavaju prioritet svega što se nalazi između njih. Pomoću njih možete namerno da zaobiđete pravila prioriteta kad je potrebno.

Pogledajte ovaj deo prethodnog primera:

```
$totalamount = $totalamount * (1 + $taxrate);
```

Da ste napisali:

```
$totalamount = $totalamount * 1 + $taxrate;
```

prvo bi se izvršila operacija množenja, budući da ima viši prioritet od operacije sabiranja, usled čega biste dobili pogrešan rezultat. Zagrade čine da se prvo izračunava podizraz `1 + $taxrate`.

U izrazu možete upotrebiti neograničen broj zagrada, a uvek će se prvo izračunavati one koje su najdublje ugnežđene.

Obratite pažnju na još jedan operator koji nismo pomenuli, a to je komanda jezika PHP, `print`, koja je ekvivalentna komandi `echo`. Obe komande generišu izlazne podatke.

U ovoj knjizi najčešće koristimo `echo`, ali slobodno koristite `print` ako ste navikli na tu komandu. Ni `echo`, ni `print` nije prava funkcija, ali se obe mogu pozivati kao funkcije s parametrima zadatim između zagrada. Obe se mogu smatrati i operatorima: tekst na koji one treba da deluju napišite iza rezervisane reči `echo` ili `print`.

Kada `print` pozovete u obliku funkcije, dobijate povratnu vrednost (1). Ta mogućnost je korisna kada želite da generišete određene izlazne podatke unutar nekog složenijeg izraza, ali to znači da će `print` biti neznatno sporija od komande `echo`.

Funkcije za rad s promenljivama

Pre nego što napustimo svet promenljivih i operatora, pogledaćemo PHP-ove funkcije za rad s promenljivama. Postoji biblioteka funkcija koja omogućava da na razne načine obrađujete i ispitujete promenljive.

Ispitivanje i menjanje tipova promenljivih

Većina funkcija za rad s promenljivama služi za ispitivanje tipova podataka.

Dve najopštije funkcije su `gettype()` i `settype()`. One imaju sledeće prototipove – prototipovi opisuju koje argumente funkcija očekuje i šta vraća:

```
string gettype(mixed promenljiva);
int settype (mixed promenljiva, tip);
```

Funkcija `gettype()` utvrđuje tip promenljive koju joj prosledite i vraća znakovnu vrednost s imenom tipa, koje može biti `boolean`, `integer`, `double` (za vrednosti s pokretnim zarezom), `string array`, `object`, `resource` ili `NULL`. Ako to nije nijedan od standardnih tipova, funkcija vraća tekst `uknown type`.

Funkcija `settype()` menja tip promenljive koju joj prosledite u novi tip naveden u obliku znakovne vrednosti u drugom argumentu. To može biti jedan od tipova sa prethodne liste.

> **Napomena**
>
> U ovoj knjizi i u php.net dokumentaciji pominje se tip podataka `mixed`. Takav tip podataka zapravo ne postoji, ali budući da je PHP tako fleksibilan pri obradi tipova podataka, mnoge funkcije prihvataju ulazne argumente više različitih tipova podataka (ili čak sve). Argumenti koji mogu imati više tipova podataka navedeni su kao pseudo-tip `mixed`.

Primer upotrebe funkcija:

```
$a = 56;
echo gettype($a).'<br />';
settype($a, 'double');
echo gettype($a).'<br />';
```

U prvom pozivu funkcije `gettype`, promenljiva `$a` je celobrojnog tipa. Nakon pozivanja funkcije `settype`, tip promenljive `$a` je promenjen u `double`.

Postoje i funkcije za ispitivanje tipova podataka. Argument svake je promenljiva i svaka vraća logičku vrednost. Te funkcije su:

- `is_array`
- `is_double`, `is_float`, `is_real` (sve je to ista funkcija)
- `is_long`, `is_int`, `is_integer` (sve je to ista funkcija)
- `is_string`
- `is_object`
- `is_resource()`
- `is_null()`
- `is_scalar()` - Ispituje da li je promenljiva skalarnog tipa, tj. da li je u pitanju podatak numeričkog, logičkog ili znakovnog tipa.
- `is_numeric()` - Ispituje da li je promenljiva bilo koja vrsta broja ili znakovna vrednost koja se može pretvoriti u broj.
- `is_callable` - Ispituje da li je vrednost promenljive ime postojeće funkcije.

Ispitivanje stanja promenljive

Postoji nekoliko funkcija za ispitivanje stanja promenljive. Funkcija `isset` ima sledeći prototip:

```
boolean isset(promenljiva);
```

Funkcija `isset` vraća `true` ako je promenljiva definisana, a u suprotnom vraća `false`. Funkciji možete proslediti i listu imena promenljivih razdvojenih zarezima, a `isset` će vratiti `true` ako su definisane sve promenljive na listi.

Definiciju promenljive možete da poništite pomoću funkcije `unset` koja ima sledeći prototip:

```
void unset(mixed promenljiva);
```

Funkcija `unset` poništava definiciju promenljive koju joj prosledite.

Funkcija `empty` ispituje da li promenljiva postoji i da li ima vrednost koja nije nula odnosno nije prazna, i vraća `true` ili `false`. Funkcija `empty` ima sledeći prototip:

```
boolean empty(mixed var);
```

Pogledajmo primer u kojem se koriste navedene funkcije.

Skriptu privremeno dodajte sledeći kôd:

```
echo 'isset($tireqty): '.isset($tireqty).'<br>';
echo 'isset($nothere): '.isset($nothere).'<br>';
echo 'empty($tireqty): '.empty($tireqty).'<br>';
echo 'empty($nothere): '.empty($nothere).'<br>';
```

Osvežite stranicu da biste videli rezultat.

Trebalo bi da za promenljivu `$tireqty` funkcija `isset` vrati 1 (`true`) bez obzira na vrednost koju ste uneli u to polje u obrascu, pa čak i ako niste ništa uneli. Vrednost koju će vratiti funkcija `empty` zavisi od toga šta ste uneli.

Budući da promenljiva `$nothere` ne postoji, rezultat funkcije `isset` biće `false`, a rezultat funkcije `empty` biće `true`.

To može biti korisno ako želite da proverite da li je korisnik popunio odgovarajuća polja u obrascu.

Ponovno interpretiranje promenljivih

Sledeće tri funkcije imaju isti efekat kao konverzija promenljive:

```
int intval(mixed promenljiva[, int baza]);
float floatval(mixed promenljiva);
string strval(mixed promenljiva);
```

Sve tri funkcije vraćaju vrednost prosleđene promenljive, konvertovanu u odgovarajući tip (redom: u ceo broj, u realan broj i u znakovnu vrednost). Kada je ulazna promenljiva znakovnog tipa, funkcija `intval` omogućava da zadate bazu ciljnog sistema u koji konvertujete argument. (Tako možete da konvertujete, na primer, heksadecimalne vrednosti u celobrojne.)

Upravljačke strukture

Upravljačke strukture omogućavaju upravljanje tokom izvršenja programa ili skripta. Grupisali smo ih u uslovne (strukture s grananjem) i strukture s ponavljanjem ili petlje. U narednim odeljcima opisaćemo te PHP-ove strukture.

Donošenje odluka u uslovnim strukturama

Ako želite da smisleno obradite podatke koje je korisnik uneo, kôd mora donositi odluke. Konstrukcije koje programu omogućavaju da donosi odluke nazivaju se *uslovne strukture*.

Iskaz `if`

Iskaz `if` možete da upotrebite za donošenje odluke u zavisnosti od određenog uslova. Ako je uslov ispunjen, izvršava se blok koda koji sledi iskazu `if` – u suprotnom biće preskočen. Uslove u iskazima `if` morate napisati između zagrada.

Na primer, kada korisnik ne popuni obrazac kako treba i pritisne dugme Submit Order, umesto obaveštenja korisniku da je njegova porudžbina prihvaćena, stranica bi mogla da prikaže neku korisniju poruku.

Kada posetilac ne naruči ništa, možete da ga obavestite „Niste ništa naručili“. To se lako postiže sledećim iskazom `if`:

```
if( $totalqty == 0 )
   echo 'You did not order anything on the previous page!<br />';
```

Uslov koji smo ovde zadali je `$totalqty == 0`. Ne zaboravite da se operator jednakosti (==) ponaša drugačije od operatora dodele (=).

Uslov `$totalqty == 0` biće ispunjen ako je `$totalqty` nula. Ako `$totalqty` nije nula, uslov će biti `false`. Kada uslov ima vrednost `true`, biće izvršen iskaz `echo`.

Blokovi naredaba

Često unutar jednog uslovnog iskaza, kao što je `if`, treba da se izvrši više iskaza. Nema potrebe da se ispred svakog stavlja nov iskaz `if`. Umesto toga, možete grupisati iskaze u *blok*. Da biste grupu iskaza deklarisali kao blok, stavite je u vitičaste zagrade:

```
if( $totalqty == 0 )
{
  echo '<font color=red>';
  echo 'You did not order anything on the previous page!<br />';
  echo '</font>';
}
```

Tri reda unutar vitičaste zagrade sada čine blok. Kada je uslov ispunjen, izvršavaju se sva tri. Kada uslov nije ispunjen, ne izvršava se nijedan.

> **Napomena**
>
> Kao što smo već rekli, PHP zanemaruje praznine u kodu. Radi bolje čitljivosti, trebalo bi da uvlačite iskaze. Uvlačenje se obično primenjuje da biste lakše uočili redove koji će biti izvršeni ako su ispunjeni uslovi, iskaze koji su grupisani u blokove i iskaze koji su deo petlji ili funkcija. U prethodnim primerima vidite da je uvučen iskaz koji zavisi od iskaza `if` i iskazi koji čine blok.

Iskazi `else`

Često valja odlučiti ne samo da li određena akcija treba da se izvede, nego morate zadati koja grupa od više mogućih akcija treba da se izvede.

Iskaz `else` omogućava da definišete alternativnu akciju koju treba preduzeti ako uslov iskaza `if` nije ispunjen. Na primer, želite da upozorite korisnike da nisu ništa naručili. S druge strane, ako naruče nešto, želite da im umesto upozorenja prikažete to što su naručili.

Ako drugačije rasporedite kôd i dodate jedan iskaz `else`, moći ćete da prikažete upozorenje ili opis naručenog.

```
if( $totalqty == 0 )
{
  echo 'You did not order anything on the previous page!<br />';
}
else
{
  echo $tireqty.' tires<br />';
  echo $oilqty.' bottles of oil<br />';
  echo $sparkqty.' spark plugs<br />';
}
```

Složenije logičke procese možete da gradite tako što ćete jedne iskaze `if` smeštati unutar drugih. U sledećem primeru će sažetak biti prikazan jedino ako je uslov `$totalqty == 0` ispunjen, a svaki red sažetka prikazuje se samo ako je ispunjen uslov za njega.

```
if( $totalqty == 0)
{
  echo 'You did not order anything on the previous page!<br />';
}
else
{
  if ( $tireqty>0 )
    echo $tireqty.' tires<br />';
  if ( $oilqty>0 )
    echo $oilqty.' bottles of oil<br />';
  if ( $sparkqty>0 )
    echo $sparkqty.' spark plugs<br />';
}
```

Iskazi `elseif`

Za mnoge odluke koje donosite postoje više od dve opcije. Pomoću iskaza `elseif` možete da napravite redosled obrade više opcija. Taj iskaz je kombinacija iskaza `else` i `if`. Kada zadate više uslova, program može da ispituje svaki uslov po zadatom redosledu dok ne nađe onaj koji je ispunjen.

Za veće porudžbine guma, Bob daje popust:

- Manje od 10 kupljenih guma - nema popusta
- 10-49 kupljenih guma - 5% popusta
- 50-99 kupljenih guma – 10% popusta
- 100 ili više kupljenih guma – 15% popusta

Možete da napišete kôd koji izračunava popust pomoću uslova i iskaza `if` i `else`. Dva uslova kombinujete u jedan pomoću operatora konjunkcije (`&&`).

```
if( $tireqty < 10 )
  $discount = 0;
elseif( $tireqty >= 10 && $tireqty <= 49 )
  $discount = 5;
elseif( $tireqty >= 50 && $tireqty <= 99 )
  $discount = 10;
elseif( $tireqty >= 100 )
  $discount = 15;
```

Reč `elseif` možete da kucate tako ili s razmakom (`else if`) – oba oblika su ispravna.

Ako nadovezujete više iskaza `elseif`, imajte u vidu da će samo jedan blok ili iskaz biti izvršeni. To u ovom primeru nije bilo važno zato što su svi uslovi bili uzajmno isključivi – samo jedan može biti ispunjen. Ako su uslovi takvi da više njih može biti istovremeno ispunjeno, izvršava se samo blok ili iskaz iza prvog ispunjenog uslova.

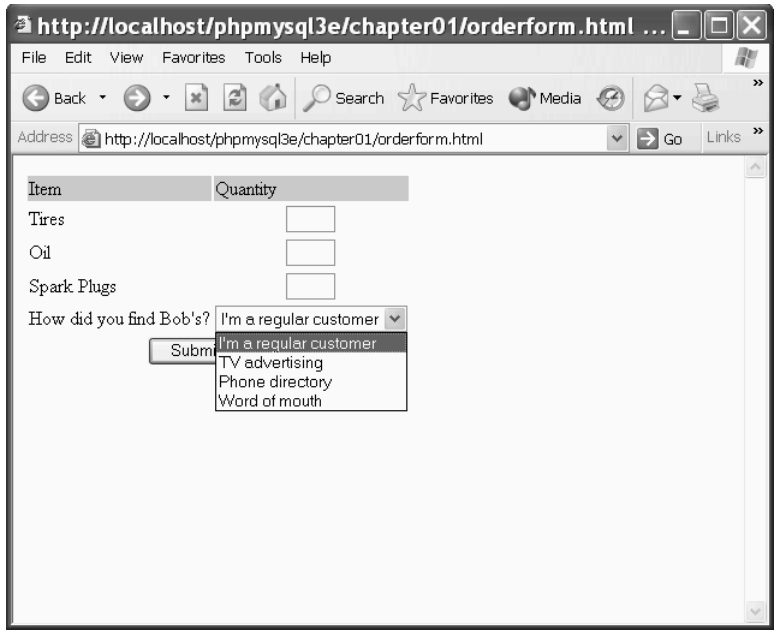
Iskaz `switch`

Iskaz `switch` deluje slično kao iskaz `if`, ali omogućava da uslov ima više od dve vrednosti. U iskazu `if`, uslov može biti samo `true` ili `false`. U iskazu `switch` uslov može imati više različitih vrednosti, koje moraju biti skalarnog tipa (integer, string ili float). Za svaku vrednost koju želite da obradite treba da napišete iskaz `case` i, eventualno, podrazumevani iskaz `case` za sve ostale vrednosti.

Pošto Bob želi da zna koja reklama privlači najviše kupaca, treba da dodate jedno pitanje na narudžbenicu.

Ubacite sledeći HTML kôd u obrazac narudžbenice, da bi izgledao kao na slici 1.6.

```
<tr>
  <td>How did you find Bob's</td>
  <td><select name="find">
        <option value = "a">I'm a regular customer</option>
        <option value = "b">TV advertising</option>
        <option value = "c">Phone directory</option>
        <option value = "d">Word of mouth</option>
      </select>
  </td>
</tr>
```



Slika 1.6 U narudžbenici pitamo posetioce kako su pronašli prodavnicu.

Navedeni HTML blok dodao je novu promenljivu obrasca čija će vrednost biti `'a'`, `'b'`, `'c'` ili `'d'`. Tu promenljivu možete da obradite nizom iskaza `if` i `else` na sledeći način:

```
if($find == 'a')
  echo '<p>Regular customer.</p>';
elseif($find == 'b')
  echo '<p>Customer referred by TV advert.</p>';
elseif($find == 'c')
  echo '<p>Customer referred by phone directory.</p>';
elseif($find == 'd')
  echo '<p>Customer referred by word of mouth.</p>';
else
  echo '<p>We do not know how this customer found us.<p>';
```

Druga mogućnost je da napišete iskaz `switch`

```
switch($find)
{
  case 'a' :
    echo '<p>Regular customer.</p>';
    break;
  case 'b' :
    echo '<p>Customer referred by TV advert.</p>';
    break;
  case 'c' :
    echo '<p>Customer referred by phone directory.</p>';
    break;
  case 'd' :
    echo '<p>Customer referred by word of mouth.</p>';
    break;
  default :
    echo '<p>We do not know how this customer found us.</p>';
    break;
}
```

(Imajte u vidu da se u oba primera podrazumeva da ste promenljivu `$find` učitali iz niza `$_POST`.)

Iskaz `switch` se ponaša malo drugačije od iskaza `if` i `else`. Iskaz `if` deluje na samo jedan iskaz, osim kada upotrebite vitičaste zagrade da biste napravili blok iskaza. Iskaz `switch` se ponaša drugačije. PHP izvršava iskaze od odgovarajuće oznake `case` do iskaza `break`. Bez iskaza `break`, izvršavaju se svi iskazi iza odgovarajuće oznake `case`. Kada se dođe do iskaza `break`, izvršavanje se nastavlja od reda koda iza iskaza `switch`.

Poređenje uslovnih iskaza

Ako vam iskazi opisani u prethodnim odeljcima nisu mnogo bliski, možda se sada pitate: „Koji je najbolji?“.

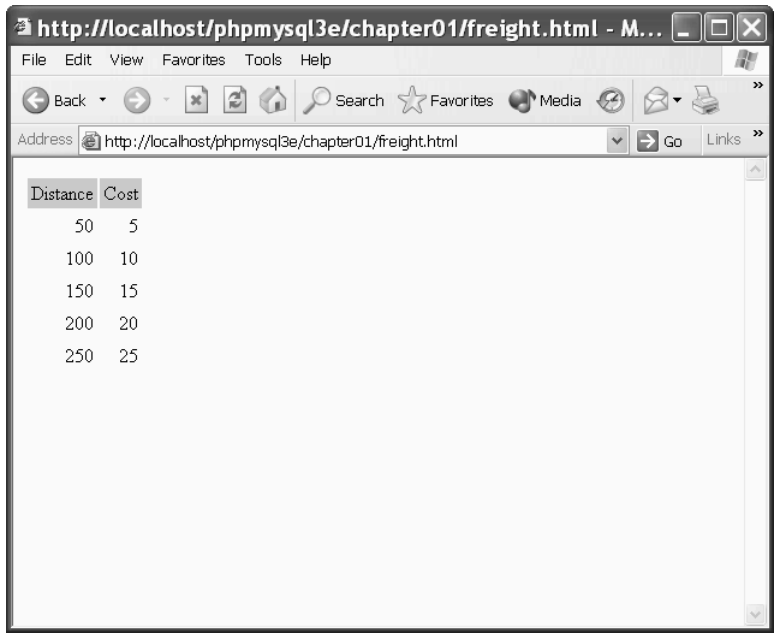
Na to pitanje ne možemo da odgovorimo. Ne postoji ništa što možete da uradite s jednim ili više iskaza `else`, `elseif` ili `switch`, a ne možete sa skupom iskaza `if`. Trebalo bi da pokušate da koristite uslovne iskaze koji se u datom slučaju najlakše čitaju. S vremenom ćete steći iskustvo i osećaj koji iskaz treba da upotrebite.

Petlje: ponavljanje akcija

Računari su oduvek bili veoma dobri za jednu primenu – automatizovanje radnji koje se ponavljaju. Ako postoji nešto što treba na isti način da se uradi mnogo puta, možete da upotrebite petlju za ponavljanje određenog bloka koda u programu.

Bob želi da prikaže tabelu sa troškovima transporta koji će biti uključeni u cenu naručene robe. Troškovi transporta zavise od razdaljine koju paket treba da pređe i izračunavaju se po jednostavnoj formuli.

Tabela treba da izgleda kao na slici 1.7.



Slika 1.7 Ova tabela prikazuje kako troškovi transporta rastu i s povećanjem razdaljine.

Listing 1.2 sadrži HTML kôd koji prikazuje ovu tabelu i koji je, kao što vidite, dugačak i sadrži delove koji se ponavljaju.

Listing 1.2 `freight.html` – HTML kôd za tabelu troškova transporta

```
<html>
<body>
<table border="0" cellpadding="3">
<tr>
  <td bgcolor="#CCCCCC" align="center">Distance</td>
  <td bgcolor="#CCCCCC" align="center">Cost</td>
</tr>
<tr>
```

```
  <td align="right">50</td>
  <td align="right">5</td>
</tr>
<tr>
  <td align="right">100</td>
  <td align="right">10</td>
</tr>
<tr>
  <td align="right">150</td>
  <td align="right">15</td>
</tr>
<tr>
  <td align="right">200</td>
  <td align="right">20</td>
</tr>
<tr>
  <td align="right">250</td>
  <td align="right">25</td>
</tr>
</table>
</body>
</html>
```

Bilo bi korisno kada bi se za generisanje HTML koda upotrebljavao jeftin i neumoran računar, a ne čovek kome treba platiti i kome brzo postane dosadno.

Petlje nalažu PHP-u da ponavlja iskaze ili blokove iskaza.

Petlja `while`

`while` je najjednostavnija vrsta petlje u PHP-u. Kao i iskaz `if`, ona zavisi od uslova. Razlika između petlje `while` i iskaza `if` leži u tome što iskaz `if` izvršava blok koda ako je uslov ispunjen, a petlja `while` izvršava blok dokle god je uslov ispunjen.

Obično koristimo petlju `while` kada ne znamo unapred koliko će iteracija biti potrebno. Kada je broj ciklusa unapred poznat, koristi se petlja `for`.

Osnovna struktura petlje `while` je:

```
while( uslov ) izraz;
```

Sledeća petlja `while` prikazaće brojeve od 1 do 5.

```
$num = 1;
while ($num <= 5 )
{
  echo $num."<br />";
  $num++;
}
```

Na početku svake iteracije ispituje se uslov. Ako uslov nije ispunjen, izraz neće biti izvršen i petlja će se završiti, a zatim se prelazi na iskaz koji sledi iza petlje.

Petlju tipa `while` možete da upotrebite za nešto korisnije, kao što je prikazivanje tabele transportnih troškova (slika 1.7).

U listingu 1.3 koristi se petlja `while` za pravljenje tabele transportnih troškova.

Listing 1.3 `freight.php` – izrada tabele troškova transporta u PHP-u

```
<body>
<table border="0" cellpadding="3">
<tr>
  <td bgcolor="#CCCCCC" align="center">Distance</td>
  <td bgcolor="#CCCCCC" align="center">Cost</td>
</tr>
<?php
$distance = 50;
while ($distance <= 250 )
{
  echo "<tr>\n  <td align='right'>$distance</td>\n";
  echo "  <td align='right'>". $distance / 10 ."</td>\n</tr>\n";
  $distance += 50;
}
?>
</table>
</body>
</html>
```

Da bi generisani HTML kôd bio čitljiv, mora da sadrži znakove za nov red i razmake. Kao što je već rečeno, čitači će zanemariti te znake, ali su za ljude oni važni. Često ćete morati da pregledate generisani HTML kôd ako ne dobijate ono što ste očekivali.

U listingu 1.3 videćete `\n`. Kada je pod navodnicima, taj niz znakova predstavlja znak za novi red.

Petlje `for` i `foreach`

Način na koji smo upotrebili petlju `while` u prethodnom odeljku veoma je uobičajen. Prvo zadamo početnu vrednost brojača i onda pre svake iteracije ispitujemo taj brojač u uslovu. Na kraju svake iteracije menjamo brojač.

Takvu vrstu petlje možete da napišete i u sažetijem obliku pomoću petlje `for`. Osnovna struktura petlje `for` je:

```
for( izraz1; uslov; izraz2)
  izraz3;
```

- *`izraz1`* se izvršava jednom na početku. U njemu obično zadajete početnu vrednost brojača.
- *`uslov`* se ispituje pre svake iteracije. Ako nije ispunjen, izvršavanje petlje prestaje. U uslovu obično ispitujete da li je brojač došao do određene granice.
- *`izraz2`* se izvršava na kraju svake iteracije. On obično menja vrednost brojača.
- *`izraz3`* se izvršava jednom u svakoj iteraciji. Taj izraz je obično blok koda i sadrži najveći deo koda petlje.

Petlju `while` iz listinga 1.3 sada možete da napišete kao petlju `for`:

```
<?php
for($distance = 50; $distance <= 250; $distance += 50)
{
  echo "<tr>\n  <td align='right'>$distance</td>\n";
  echo "  <td align='right'>". $distance / 10 ."</td>\n</tr>\n";
}
?>
```

Obe verzije koda, `for` i `while`, funkcionalno su identične. Petlja `for` je nešto sažetija jer je kraća za dva reda.

Te petlje su ravnopravne – nema bolje i lošije. Koristite onu koja više odgovara.

Napomenimo da možete kombinovati promenljive promenljivih s petljom `for` radi iteracije kroz niz polja obrasca. Ako se, na primer, polja zovu `name1`, `name2`, `name3` itd., možete da ih obradite u petlji na sledeći način:

```
for ($i=1; $i <= $numnames; $i++)
{
  $temp= "name$i";
  echo $$temp.'<br />'; // obrađivanje vrednosti iz polja
}
```

Kada dinamički generišete imena promenljivih, možete redom da pristupite svakom polju.

Osim petlje `for`, postoji i petlja `foreach`, namenjena radu s nizovima. O njenoj upotrebi govorićemo u poglavlju 3, „Upotreba nizova“.

Petlje `do..while`

Poslednji tip petlje koji ćemo spomenuti ponaša se nešto drugačije. Opšta struktura petlje `do..while` je:

```
do
  izraz;
while( uslov );
```

Petlja `do..while` se razlikuje od petlje `while` zbog toga što se uslov ispituje na kraju. To znači da se u petlji `do..while` iskaz ili blok unutar petlje uvek izvršavaju najmanje jednom.

Čak i ako na početku uslov nije ispunjen i ako nikada ne može da bude ispunjen, petlja će biti izvršena jednom pre nego što ispita uslov i završi se.

```
$num = 100;
do
{
  echo $num.'<br />';
}
while ($num < 1 );
```

Izlazak iz upravljačke strukture

Postoje tri načina da prekinete izvršavanje bloka koda:

Da biste prekinuli izvršenje petlje, upotrebite iskaz `break`, kao što smo naveli u odeljku o iskazu `switch`. Ako u petlji koristite iskaz `break`, izvršenje skripta će se nastaviti od prvog reda iza petlje.

Ako želite da započnete novu iteraciju petlje, upotrebite iskaz `continue`.

Da biste prekinuli izvršavanje celog PHP skripta, upotrebite komandu `exit`. To je obično korisno kada se utvrdi greška. Na primer:

```
if( $totalqty == 0)
{
  echo 'You did not order anything on the previous page!<br />';
  exit;
}
```

Iskaz `exit` prekida izvršavanje skripta.

Upotreba alternativnih oblika sintakse za upravljačke strukture

Za sve upravljačke strukture koje smo opisali postoji alternativni oblik sintakse, u kojem se početna vitičasta zagrada ({) zamenjuje dvotačkom (:), a završna vitičasta zagrada (}) zamenjuje se novom ključnom rečju koja će biti `endif`, `endswitch`, `endwhile`, `endfor` ili `endforeach`, u zavisnosti od toga toga koja je upravljačka struktura u pitanju.

Na primer, sledeći blok koda:

```
if( $totalqty == 0)
{
  echo 'You did not order anything on the previous page!<br />';
  exit;
}
```

može se drugačije napisati u sledećem obliku:

```
if( $totalqty == 0):
  echo 'You did not order anything on the previous page!<br />';
  exit;
endif;
```

Struktura `declare`

Jedna od PHP-ovih upravljačkih struktura koja se ne koristi tako često kao druge, jeste struktura `declare`. Opšti oblik te strukture izgleda ovako:

```
declare (direktiva)
{
// blok
}
```

Ta struktura omogućava da zadate *direktive za izvršavanje bloka* koda, tj. pravila o tome kako treba da se izvrši kôd koji sledi. Zasad je na raspolaganju samo jedna izvršna direktiva, `ticks`. Zadaje se u obliku `ticks=`*n* i znači da se unutar bloka koda izvršava određena funkcija u svakom *n*-tom redu bloka, što je korisno prvenstveno pri ispitivanju performansi ili otkrivanju grešaka.

Upravljačku strukturu `declare` pominjemo na ovom mestu samo radi informacije. U poglavljima 24, „Upotreba PHP-a i MySQL-a u većim projektima“ i 25, „Otklanjanje grešaka“, naći ćete primere upotrebe funkcije `tick`.

Šta sledi

Sada znate kako da primite porudžbinu i radite s njom. U sledećem poglavlju naučićete da sačuvate porudžbinu tako da ona kasnije može da se učita i obradi.