Vodič kroz PHP i MySQL

Achilles
achilles333@gmail.com

Pre nego što počnete da čitate ovu skriptu, hteo bi da vam skrenem pažnju na nekoliko pojedinosti vezanih za istu. Pre svega, ova je samo deo velike skripte, odnosno "Vodiča kroz PHP i MySQL", koji će se, nadam se, uskoro pojaviti, a sadržaće zatno više materijala, mahom, namenjenog već iskusnim PHP programerima. Takođe ova skripa je urađena u relativno kratkom vremenskom periodu, tako da su moguće pravopisne greške, pa vam se zato, kao autor skripte unapred izvinjvam.

Naravno sve kritike, komentari, saveti su dobrodošli i možete ih poslati na achilles333@gmail.com . Takođe putem ove adrese mozete se informisati o kompletnoj verziji "Vodiča kroz PHP i MySQL".

Beograd, decembar 2004.
Autor

Sadržaj :

Logovanje u MySQL

Pre svega, da bi se povezali sa MySQL bazom, potreban nam je MySQL klijent, tj. program koji će izvršiti povezivanje. U okviru Windows i Unix platforme, već postoje takvi programi. Tako npr. u Linux-u program pozivamo komandom mysql, a njegova podrazumevana lokacija (default) je /usr/local/mysql/bin, pod Windows-om program pozivamo sa mysql.exe, a podrazumevana lokacija c:\mysql\bin.

Ukoliko želite da se povežete sa MySQL koji se ne nalazi na vašem kompjuteru (npr. sa *Web host's MySQL server*) postoje dva načina povezivanja. Prvi je korišćenjem telnet-a (da bi se ulogovali na vaš *Web host's MySQL server* i odatle pokrenuli mysql). Drugi način je da sa interneta preuzmete ***MySQL client software*** (*http://www.mysql.com/* , koji je kompatibilan i za Windows i za Linux platforme) i preko njega se povežete sa MySQL serverom na Internetu.

Ma koju platformu i program izabrali, većina stvari će biti ista, tako da mogu da počnem sa objašnjavanjem - povezivanja na MySQL server.

Da bi se povezali potrebno je da upište sledeću komandu :

```
mysql -h <hostname> -u <username> -p
```

Naravno, umesto polja <hostname> pišete host name ili IP adresu kompjutera na kome se nalazi MySQL server. Ukoliko pokrećete program na istom računaru, na kome se nalazi i server, možete izostaviti deo – h <hostname> iz komandne linije umesto da pišete `-h localhost`. <username> je vaš MySQL username, ako je hostname localhost, onda će userneme najverovatnije biti root.

Oznaka "-p", služi da bi pozvali upit za upisivanje password-a (u sledećoj komandnoj liniji).

Ako je sve urađeno kako treba MySQL program će izbaciti sledeću liniju :

```
mysql>
```

Sada smo se povezali sa bazom podataka, tako da sada preostaje da izaberemo bazu sa kojom ćemo da radimo. Prvo ćemo pogledati spisak baza na serveru. Da bi to uradili koristimo sledeću komandu (OBAVEZNO je **;** na kraju svake komande).

```
mysql> SHOW DATABASES;
```

Sada dobijamo spisak baza, koji bi mogao da izgleda ovako (ukoliko do sada nije bilo nikakvih promena)

```
+----------+
| Database |
+----------+
| mysql    |
| test     |
+----------+
2 rows in set (0.11 sec)
```

MySQL koristi prvu bazu, pozivom mysql, da bi sačuvao članove, njihove lozinke i dozvole. Za sada ćemo ostaviti prazne tabele i ponovo se vratiti na njih u narednim delovima ove skripte. Pošto nam druga baza ne treba, jer ćemo u nastavku kreirati

gomilu baza koje sa kojima ćemo raditi, iskoristićemo bazu test, da vidimo upotrebu naredbe DROP koja služi za brisanje baze.

```
mysql> DROP DATABASE test;
```

Nakon ove naredbe baza će biti obrisana. Važna napomena je da neće biti nikakvog upozorenja tj. pitanja tipa "da li ste sigurni ?". Zbog toga, preporučujem, da sa ovom naredbom budete operezni, jer se jednom komandom može obrisati baza koja sadrži jako bitne podatke, što bi moglo imati katasrtofalne posledice.

Pre nego što krenemo na nove komande , da ponovimo, da na kraju svake mora stajati ";" jer će u suprotnom MySQL misliti da komanda nije završena i očekivaće nastavak u novom redu:

```
mysql> SHOW
   -> DATABASES;
```

Takođe ako je vaša komanda dugačka tj. ako se sastoji od nekoliko redova u svakom sledećem će umesto mysql> biti samo ->

Ukoliko ste napravili grešku i želite da obustavite komandu (cancel) to možete uraditi komandom \c:

```
mysql> DROP DATABASE\c
mysql>
```

MySQL će ignorisati prethodnu komandu i vratiti se u početni položaj i čekati sledeću naredbu.

Konačno, ako želimo da izađemo iz MySQL klijenta koristimo ključnu reč quit ili exit. Inače, to je jedina komanda koja ne zahteva ";" na kraju:

```
mysql> quit
Bye
```

Kreiranje baza

Ono što prvo moramo da uradimo, da bi nastavili dalje, je da kreiramo bazu. Kreiranje baze je veoma jednostavno (kao i njeno brisanje). Baza se kreira uz pomoć ključne reči CREATE:

```
mysql> CREATE DATABASE vicevi;
```

Ovde smo izabrali da ime baze bude "vicevi" (naravno izbor imena baze zavisi izključivo od autora).

Pošto smo sada napravili našu bazu, da bi mogli da joj pristupimo, koristimo naredbu USE:

```
mysql> USE vicevi;
```

Tek sada možemo da raspolažemo sa našom bazom, koja je za sada prazna (dok ne ubacimo neku tabelu). Pravljenje tabele koja će sadržati "viceve" će biti naš prvi zadatak.

Kreiranje tabela

Da bi kreirali tabelu koristimo sledeći model:

```
mysql> CREATE TABLE <ime tabele> (
   ->  <ime kolone 1> <tip kolone 1> <dodatak kolone 1>,
   ->  <ime kolone 2> < tip kolone 2> <dodatak kolone 2>,
   ->  ...
   -> );
```

U našem primeru to bi izgledalo ovako: kreiramo tri kolone: ID (broj), VicText (tekst vica), VicDate (datum upisivanja vica). Komanda za kreiranje ove tabele bi izgledala ovako :

```
mysql> CREATE TABLE Vicevi (
    ->  ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    ->  VicText TEXT,
    ->  VicDate DATE NOT NULL
    -> );
```

Prva linija je krajnje jednostavna, u njoj obaveštavamo MySQL da želimo da kreiramo tabelu Vicevi.

U drugoj liniji želimo da napravimo kolonu koju ćemo nazvati ID koja će sadržati podatke tipa integer (int). Ova linija sadrži i neke specifičnosti vezane za prvu kolonu:
Prvo, kolona ne sme biti prazna (NOT NULL).
Zatim, automatski se dodeljuje vrednost svake sledeće vrste prilikom unosa (AUTO_INCREMENT).
I konačno, pošto će sve vrednosti sigurno biti različite to nam omogućava da datu kolonu označimo kao primarni ključ (PRIMARY KEY).

Treća linija je takođe veoma jednostavna, žalimo da napravimo kolonu VicText koja će sadržati tekstove viceva (tipa TEXT).

Četvrta kolona VicDate će sadržati tip podataka (DATE) i ona takođe nikada ne sme biti prazna, zato i za nju stavljamo NOT NULL.

Ako su sve komande korektne, MySQL će učitati sa Query OK, i naša prva tabela će biti kreirana. Ako smo napravili grešku MySQL će tu prijaviti problem sa našim upitom i probati da da neko objašnjenje u vezi sa problemom.

Sada možemo pogledati tabelu koju smo napravili. To radimo komandom:

```
mysql> SHOW TABLES;
```

i dobijamo sledeći odgovor

```
+------------------+
| Tables in Vicevi|
+------------------+
| Vicevi           |
+------------------+
1 row in set
```

Ona predstavlja listu svih tabela u našoj bazi. Sadrži samo tabelu vicevi. Pošto je sve OK, sada možemo pogledati i tabelu Vicevi :

```
mysql> DESCRIBE Vicevi;
+----------+---------+------+-----+------------+- -
| Field    | Type    | Null | Key | Default    | ...
+----------+---------+------+-----+------------+- -
| ID       | int(11) |      | PRI | 0          | ...
| VicText  | text    | YES  |     | NULL       |
| VicDate  | date    |      |     | 0000-00-00 |
+----------+---------+------+-----+------------+- -
3 rows in set
...
```

Takođe važna stavka je brisanje tabele, koja je takođe jednostavna (kao i sve prethodne). Tabelu ćemo obrisati ako napišemo sledeću komandu :

```
mysql> DROP TABLE <imeName>;
```

Ubacivanje sadržaja u tabelu

Pošto smo kreirali bazu i u njoj tabelu, sada preostaje da u tabelu unesemo neki sadržaj. Ključna reč za unošenje sadržaja u tabelu je INSERT. Postoje dva načina na koja možemo izvršiti upisivanje u tabelu :

```
mysql> INSERT INTO <ime tabele> SET
   ->  imeKolone1 = vrednost1,
   ->  imeKolone2 = vrednost2 ,
   ->  ...
   -> ;


mysql> INSERT INTO <table name>
   -> (imeKolone 1, imeKolone 2, ...)
   -> VALUES (vrednost 1, vrednost 2, ...);
```

Da bi ubacili neki vic u našu tabelu možemo izabrati bilo koji od ova dva načina :

```
mysql> INSERT INTO Vicevi SET
    -> VicText = "Zasto zec ima duge usi, a kratak rep?
Pa da je obrnuto bio bi veverica!",
    -> VicDate = "2004-05-18";
```

```
mysql> INSERT INTO Vicevi
    -> (VicText, VicDate) VALUES (
    -> " Zasto zec ima duge usi, a kratak rep?
Pa da je obrnuto bio bi veverica!",
    -> "2004-05-18"
    -> );
```

Pošto sada znamo kako se unose podaci u tabelu, mogli bi da pogledamo šta smo uneli...

Pregled sačuvanih podataka

Komanda za pregled sadržaja je SELECT. Ona je verovatno najkomplikovanija naredba u SQL-u. Razlog te kompleksnosti leži u tome što je ona glavna i od nje zavisi fleksibilnost odgovora (što ćemo uskoro pokazati).

Sada ćemo pogledati najjednostavniji oblik komande SELECT, koji ispisuje kompletan sadržaj tabele Vicevi:

```
mysql> SELECT * FROM Vicevi;
```

Ako bi pokušali da tumačimo ovu naredbu, to bi izgledalo otprilike ovako: "ispiši sve iz tabele Vicevi". Nakon ove komande dobijamo sledeći odgovor:

```
+----+--------------------------------------
------------------------+------------+
| ID | VicText
                        | VicDate    |
+----+--------------------------------------
------------------------+------------+
|  1 | Zasto zec ima duge usi, a kratak rep?
Pa da je obrnuto bio bi veverica! | 2000-04-01 |
+----+--------------------------------------
------------------------+------------+
1 row in set (0.05 sec)
```

Dobili smo jedan nepregledan pogled na našu tabelu iz razloga što je tekst vica predugačak za *ekran.* Zato ćemo u sledećem pozivu izostaviti tekst vica.

```
mysql> SELECT ID, VicDate FROM Vicevi;
```

posle čega dobijamo novi prikaz:

```
+----+------------+
| ID | VicDate    |
+----+------------+
|  1 | 2000-04-01 |
+----+------------+
1 row in set (0.00 sec)
```

Ovo nije loše, ali može i bolje. Želimo da ispišemo sve kolone, ali da budu pregledno prikazane na našem ekranu. Kako ćemo to uraditi ? Ograničićemo prikaz kolone VicText na određeni broj karaktera uz pomoć ključne reči LEFT (neka bude npr. 20):

```
mysql> SELECT ID, LEFT(VicText,20), VicDate FROM Vicevi;
```

```
+----+----------------------+------------+
| ID | LEFT(VicText,20)     | VicDate    |
+----+----------------------+------------+
|  1 | Zasto zec ima duge   | 2000-04-01 |
+----+----------------------+------------+
1 row in set (0.05 sec)
```

Druga korisna funkcija je COUNT, koja jednostavno vraća brojač kao rezultat. Tako npr. ako želimo da prebrojimo koliko je viceva sačuvano u našoj tabeli to možemo da uradimo sledećom komandom:

```
mysql> SELECT COUNT(*) FROM Vicevi;
```

```
+----------+
| COUNT(*) |
+----------+
|        1 |
+----------+
1 row in set (0.06 sec)
```

Dakle, imamo samo jedan vic u tabeli.

Sada možemo da uvedemo klauzulu WHERE koja nam služi da učitamo samo podatke koji ispunjavaju uslov zadatka. Tako dolazimo do sledećeg primera:

```
mysql> SELECT COUNT(*) FROM Vicevi
    -> WHERE VicDate >= "2004-01-01";
```

Kao rezultat ovog upita, dobićemo ispis svih viceva koji su unešeni u tabelu nakon željenog datuma.

Takođe, uz pomoć ključne reči LIKE , koja služi da prikažemo reči i rečenice koje u sebi sadrže datu reč, možemo napraviti varijaciju našeg upita:

```
mysql> SELECT VicText FROM Vicevi
    -> WHERE VicText LIKE "%zec%";
```

Dakle tražimo ispis svih Viceva koji sadrže reč *zec*.

Da bi još više precizirali upit, možemo da kombinujemo ove dve naredbe:

```
mysql> SELECT VicText FROM Vicevi WHERE
    -> VicText LIKE "%rec1 rec2%" AND
    -> VicDate >= "2004-04-01" AND
    -> VicDate < "2004-05-01";
```

Kako budemo dodavali sadržaj u tabelu vicevi, tako ćemo proširivati komandu SELECT.

Izmena sačuvanih podataka

Neke stvari koje unesemo u bazu želimo da menjamo. Sve promene obavljaju se uz pomoć kjučne reči UPDATE. Ova komanda sadrži elemente komande INSERT (za podešavanje vrednosti kolone) i elemente komande SELECT (za izbor sadržaja koji ćemo modifikovati). Uopšteno oblik komande UPDATE bi mogao da izgleda ovako:

```
mysql> UPDATE <imeTabele> SET
    -> <ime_kolone>=<nova_vrednost>, ...
    -> WHERE <uslov>;
```

Tako npr. ako želimo da promenimo datum nekog vica koji smo uneli koristimo sledeću komandu:

```
mysql> UPDATE Vicevi SET VicDate="2000-10-04" WHERE ID=1;
```

Takođe izmene možemo praviti i ako ubacimo ključnu reč WHERE:

```
mysql> UPDATE Vicevi SET VicDate="2000-10-04"
    -> WHERE VicText LIKE "%zec%";
```

Brisanje sačuvanih podataka

Brisanje sadržaja u SQL je "opasno lako" (o čemu je već bilo reči u prethodnom delu skripte). Komanda za brisanje sadržaja je sledeća:

```
mysql> DELETE FROM <imeTabele> WHERE <uslov>;
```

Takođe ako želimo možemo obrisati sve Viceve iz tabele koji sadrže odgovarajuću reč, npr:

```
mysql> DELETE FROM Vicevi WHERE VicText LIKE "%zec%";
```

Naravno postoji i komanda koja bi obrisala čitav sadržaj tabele Vicevi:

```
mysql> DELETE FROM Vicevi;
```

Krenimo priču o PHP

Do sada smo naučili kako uz pomoć MySQL možemo da čuvamo Viceve u jednostavnoj bazi (sastavljenoj od jedne tabele Vicevi). Koristili smo MySQL klijent za ispisivanje SQL komandi. Sada ćemo naučiti kako da to isto uradimo sa *server-side scripting* jezikom kao što je PHP. Uz neka proširenja osnovnih osobina, koje smo do sada naučili, ovaj jezik (PHP) u potpunosti podržava komunikaciju sa MySQL bazom.

Pretstavimo PHP

Kao što smo rekli PHP je *server-side scripting* jezik. *Server-side scripting* jezik je sličan JavaScript po mnogim segmentima, oni dopuštaju ubacivanje malih programa (skripti) u HTML u okviru Web strane. U izvršavanju takvih skripta, dopuštena je kontrola, koja će odmah pokazati u browser-u svaku izmenu koju je moguće izvršiti sa početnim HTML kodom.

Ključna razlika između JavaScript i PHP, je ta što web browser jednom tumači JavaScript i to prilikom učitavanja stranice, dok *server-side scripting* jezik, kao što je PHP svoje tumačenje obavlja na serveru, pre nego što stranicu pošalje u browser. Tumačenje PHP koda se na web strani zamenjuje standardnim HTML kodom. Skripta se u potpunosti izvršava na serveru. Otuda i sam naziv: *server-side scripting* jezik.

Sada ćemo dati jedan jednostavan primer PHP-a (nazovimo ga datum.php):

```
<HTML>
<HEAD>
<TITLE>Danasnji datum</TITLE>
</HEAD>
<BODY>
<P>Danasnji datum (na Web server-u) je
<?php

  echo( date("l, F dS Y.") );

?>
</BODY>
</HTML>
```

Većina predstavlja standardni HTML kod, dok je linija između <?php i ?> napisana u PHP. <?php označava početak PHP koda, a ?> kraj PHP koda. Server će tumačiti sve što se nalazi između ova dva "graničnika" i to prebaciti u standardan HTML kod, pre slanja na web stranu koju učitavamo. U samom browser-u će se pojaviti HTML sledeće sadžine:

```
<HTML>
<HEAD>
<TITLE>Danasnji datum</TITLE>
</HEAD>
<BODY>
<P>Danasnji datum (na Web server-u) je
Wednesday, June 17th 2003.</BODY>
</HTML>
```

Recimo i da će sve PHP oznake nestati i na njihovom mestu će se pojaviti izlazna skripta koja će izgledati kao standardni HTML. Ovaj primer pokazuje nekoliko prednosti server-side scripting:

Nema problema sa kompatibilnošću browser-a, pošto se PHP skripta izvršava na serveru i nigde više, to omogućava da ne moramo razmišljati šta podržava, a šta ne browser posetioca naše strane.

Pristupa resursima koji se nalaze na serveru. U prethodnom primeru smo ubacili datum saglasno sa serverom u web stranu. Da smo hteli to da uradimo uz pomoć npr. JavaScript, onda bi se na strani pokazalo vreme saglasno kompjuteru sa koga se pristupa našoj stranici. Pošto ovaj primer i nije baš zanimljiv što se tiče ekspoatacije *server-sid* resursa, mi možemo lako ubaciti neku drugu informaciju, koja će biti upotrebljiva samo za skript koji se pokreće na serveru – npr. informacije sačuvane u MySQL bazi pokreću se na serveru.

Smanjuje se dužina učitavanja stranice kod korisnika. JavaScript može značajno usporiti prikazivanje stranice u browser-u, na sporijim kompjuterima. Sa *server-side scripting* sav posao se obavlja na serverskoj mašini i korisnik čita običan HTML kod.

Osnovna sintaksa i komande

Svako ko je do sada radio (i razume) u programskim jezicima kao što su C, C++, Java, JavaScript, Perl (ili bilo koji jezik izveden od C) biće mu poznata i PHP sintaksa. PHP skrip se sastoji od serije komandi, ili iskaza, gde svaki mora biti u vezi sa serverom. PHP izkazi liče na iskaze u prethodno pomenutim jezicima i obavezno se završavaju sa ";".

Evo jednog karakterističnog iskaza:

```
echo( "Ovo je <B>test</B>!" );
```

Ova funkcija poziva na ugradnju u HTML kod echo i njen rezultat u HTML strani će biti: Ovo je *<B>test</B>!*. PHP raspolaže sa puno funkcija koje nam omogućavaju sve od slanja e-maila do rada sa informacijama sačuvanim u različitim tipovima baza.

Funkcija echo jednostavno uzima tekst i prosleđuje ga u HTML kod, na strani koja se učitava:

```
<HTML>
<HEAD>
<TITLE> Jednostavan primer PHP</TITLE>
</HEAD>
<BODY>
<P><?php echo("Ovo je <B>test</B>!"); ?></P>
</BODY>
</HTML>
```

ako sačuvano ovaj fajl kao test.php i postavimo ga na web server, browser će videti sledeću stranu:

```
<HTML>
<HEAD>
<TITLE> Jednostavan primer PHP </TITLE>
</HEAD>
<BODY>
<P>Ovo je <B>test</B>!</P>
</BODY>
</HTML>
```

Napomenimo da su HTML tagovi (kao npr. <B> i </B>) apsolutno dozvoljeni.

Navodnici se koriste da označe početak i kraj *niza znakova* u PHP, pa je njihovo postojanje neophodno. Zagrade imaju dvostruku namenu. Prvo, one obaveštavaju da je echo funkcija koju želimo da pozovemo. Drugo, označavaju početak i kraj liste "parametara" koje želimo da prosledimo funkciji da ih izvrši. U echo funkciji, jedino je neophodno dati *niz znakova* koji će se pokazati na strani.

Operatori i promenljive

Promenljive u PHP su iste kao i u svim drugim programskim jezicima. U narednom primeru kreiraćemo promenljivu `$testvariable` (sve promenljive u PHP moraju počinjati znakom "dolara") i dodeliti joj vrednost 3:

```
$testvariable = 3;
```

Jedna promenljiva u PHP, može sadržati razne tipove podataka (može biti broj, niz znakova ili neka druga vrednost) i može ih menjati za vreme "života". Pa ako sledeći izraz napišemo posle prethodnog, onda dodajemo novu vrednost u postojeći `$testvariable`. U tom procesu, promenljiva menja sadržaj i umesto broja uzaima *niz znakova*:

```
$testvariable = "Tri";
```

Isti znak koji smo uzeli u prethodna dva izraza (=) zove se "operator dodele" i koristimo ga da bi promenljivoj dodelili neku vrednost. Takođe možemo koristiti razne matematičke operacije:

```
$testvariable = 1 + 1; // Dodaje vrednost 2.
$testvariable = 1 - 1; // Dodaje vrednost 0.
$testvariable = 2 * 2; // Dodaje vrednost 4.
$testvariable = 2 / 2; // Dodaje vrednost 1.
```

Na kraju svake linije nalazi se komentar. Komentari nam služe da bi opisali šta radi naš kod. To radimo ubacivanjem objašnjenja u kod i naznakom da PHP treba da ih ignoriše. Komentari počinju sa // i završavaju se prelaskom u novi red.

Vratimo se sada na naše izraze. Operatori koje smo uzeli dozvoljavaju da dodamo, oduzmemo, pomnožimo i podelimo brojeve. Između ostalog, tu su takođe operatori za spajanje *niza znakova*:

```
// Spajanje niza "Hej ti!".
$testvariable = "Hej " . "ti!";
```

Promenljive možemo koristiti i na sledeći način:

```
$var1 = "PHP"; // Dodaje vrednost "PHP" u $var1
$var2 = 5; // Dodaje vrednost 5 u $var2
$var3 = $var2 + 1; // Dodaje vrednost 6 u $var3
$var2 = $var1; // Dodaje vrednost "PHP" u $var2
echo($var1); // Ispisuje "PHP"
echo($var2); // Ispisuje "PHP"
echo($var3); // Ispisuje 6
echo($var1 . " zakon!"); // Ispisuje "PHP zakon!"
echo("$var1 zakon!"); // Ispisuje "PHP zakon!"
echo('$var1 zakon!'); // Ispisuje '$var1 zakon!'
```

Posebno razmotrimo poslednje dve linije. Možemo uključiti ime promenljive unutar *niza znakova*. To možemo uradti na dva načina - između dvostrukih i jednostrukih navodnika, a prethodni primer nam pokazuje u čemu je razlika.

Korisnički uticaj (interakcija) i forme

Za većinu zanimljivih aplikacija u PHP jako je bitan uticaj korisnika koji posećuje web stranu. *Server-side scripting* jezik kao PHP može da ima mnoga ograničenja rada kada dolazi do uticaja posetioca sajta.

Ideja korisnikovih uticaja na PHP se bazira na slanju informacija sa zahtevom korisnika na novu stranu.

Najjednostavniji metod za slanje informacija na stranu je uz pomoć "URL query string". Ako ste nekad videli URL sa znakom pitanja iza imena fajla, videli ste primenu ove tehnike. Evo jedan jednostavan primer. Napravimo standardan HTML fajl i ubacimo sledeći link:

```
<A HREF="welcome.php?ime=Petar"> Zdravo ja sam Petar! </A>
```

Taj link ka fajlu nazovimo welcome.php, ali u dopuni za linkovanje smo takođe ubacili promenljivu sa zahtevom strane. Promenljivu smo nazvali *ime*, a njena vrednost je *Petar*. Da bi videli efekat, potrebno je da napravimo stanicu koja će prihvatiti vrednost *Petar* na mesto *ime* i to proslediti u HTML kod.

Napravimo sada novi HTML fajl, ali sa ekstenzijom .php, posto će sadržati i PHP kod. Dakle, nazovimo ga welcome.php:

```
<?php
  echo( "Dobrodosao na nas sajt, $ime!" );
?>
```

Sada ako kliknemo na link iz prvog fajla, on će pozvati drugi fajl i na ekranu ispisati *"Dobrodosao na nas sajt, Petar!"*. Vrednost promenljive je preko upita automatski prosleđena u PHP i postavljena na mesto promenljive `$ime` koja je deo *niza znakova* koji se prikazuje.

Takođe, ako želimo, možemo proslediti više vrednosti u upitu. Da bi to pokazali modifikovaćemo prethodni primer:

```
<A HREF="welcome.php?ime=Petar&prezime=Petrovic">
Zdravo, ja sam Petar Petrovic! </A>
```

Sada smo prosledili dve promenljive `ime` i `prezime`. Promenljive su u upitu odvojene znakom `&`. Takođe sada nam trebaju i dve promenljive u fajlu welcome.php:

```
<?php
  echo( " Dobrodosao na nas sajt,
$ime $prezime!" );
?>
```

Ovo je sve dobro, ali nedostaje korisnički uticaj, zato ćemo omogućiti korisniku da unese svoje podatke. Nastavićemo sa ovim primerom "poruke za dobrodošlicu"... Želimo da dopustimo da korisnik sam unese svoje podatke, koji će mu se pokazati u poruci. Da bi to uradili potreban nam je sledeći HTML kod:

```
<FORM ACTION="welcome.php" METHOD=GET>
Ime: <INPUT TYPE=TEXT NAME="ime"><BR>
Prezime: <INPUT TYPE=TEXT NAME="prezime">
<INPUT TYPE=SUBMIT VALUE="GO">
</FORM>
```

Ova forma ostavlja isti evekat kao i drugi link koji o kome smo malopre pričali, osim što sada korisnik sam upisuje svoje podatke. Kada klinkne na submit dugme (koje u ovom slučaju ima vrednost GO), browser ce učitati stranicu welcome.php i automatski promenljivama dodeliti vrednosti koje je korisnik uneo. Uzima ime promenljive NAME, sa dodatkom INPUT TYPE=TEXT i uzima vrednost koju je korisnik uneo u tekstualno polje.

Reč METHOD je dodatak taga FORM koji se koristi da kaže browser-u kako da pošalje promenljive i njihove vrednosti u zahtev. Vrednost GET (koju smo koristili u prethodnom primeru) kaže da zahtev ubaci u upit, ali ima i drugih alternativa. Tačnije, nije uvek poželjno (ili nije tehnički izvodljivo) imati vrednosti koje se pokazuju u upitu.

Šta da smo npr. ubacili TEXTAREA tag u našu formu i da korisnik unese tekst velike dužine. URL bi sadržao nekoliko paragrafa teksta u upitu, bio bi besmisleno dugačak, i prelazio bi maksimalnu dozvoljenu dužinu za upisivanje URL u browser-u. Alternativno rešenje, je "nevidljivo" ubacivanje informacija, iza "scene". Kod za to je potpuno isti, ali umesto podešavanja METHOD-a na GET mi postavljamo na POST.

```
<FORM ACTION="welcome.php" METHOD=POST>
Ime: <INPUT TYPE=TEXT NAME="ime"><BR>
Prezime: <INPUT TYPE=TEXT NAME="prezime">
<INPUT TYPE=SUBMIT VALUE="GO">
</FORM>
```

Ova forma funkcioniše identično kao prethodna. Jedina razlika je URL, koji neće imati upit, kada korisnik klikne na dugme "GO". Sada možemo ubaciti i velike i osetljive (npr. lozinke) vrednosti u slanju podataka iz forme, bez prikazivanja upita. Drugo, ako korisnik stranu sačuva u "*bookmarks*", strana će biti beskorisna i neće sadržati nikakve vrednosti.

Ovaj šablon je osnova za korišćenje formi, za izvršavanje elementarne interakcije između korisnika i PHP. Mi ćemo taj šablon unapređivati u narednim primerima.

Kontrola konstrukcije (strukture)

Svi dosadašnji primeri PHP koda mogu biti jednostavni, "jedno-iskazni" skriptovi kao izlazni *nizovi znakova* na web strani ili mogu imati seriju iskaza koji će se izvršavati posle neke naredbe. Ko je radio u programskim jezicima kao što su JavaScript, C, BASIC... zna da su ti programi ponekad veoma jednostavni.

PHP za razliku od drugih programskih jezika, sadrži specijalne iskaze koji dozvoljavaju "skretanje" nakon neke izvršne naredbe koja je dominanta u našim primerima. Takve izraze zovemo "kontrola strukture". Ako ništa niste razumeli, ne brinite, slede nekoliko primera, koji će vam sigurno pomoći da shvatite ovu priču.

Najjednostavniji i najčešći oblik "kontrole strukture" je if-else iskaz. Evo kako on izgleda :

```
if ( <uslov> ) {
  // izraz koji ce biti izvrsen
  // ako je <uslov> tacan.
} else {
  // (Opciono) izraz koji ce biti izvrsen
  // ako je <uslov> ne tacanis.
}
```

Ova kontrola strukture govori PHP da izvrši jedan ili drugi set iskaza, zavisno da li je uslov tačan ili netačan. Sada ćemo prikazati malopređašnji primer u malo izmenjenom obliku:

```
if ( $ime == "Petar" ) {
  echo( "Petre, poseban pozdrav za tebe!" );
} else {
  echo( "Dobrodosao, $ime!" );
}
```

Sada ako promenljiva ima vrednost Petar ispisaće se specijalna poruka, a ako nema pojaviće se standardna poruka.

Takođe u prethodnom primeru se pojavljuje klauzula else koja nam govori šta da radimo ako uslov (if) nije zadovoljen. Ona je opciona i možemo je izostaviti,

ako npr. želimo da ispišemo specijalnu poruku za određeno ime, a prazan ekran u suprotnom. Tada bi kod izgledao otprilike ovako:

```
if ( $ime == "Petar" ) {
  echo( "Petre, poseban pozdrav za tebe!" );
}
```

U prethodnom uslovu smo koristili == da bi uporedili dve vrednosti i videli da li su jednake. Važno je zapamtiti da se upotrebljava dvostruki znak jednakosti, zato što jednostruki služi za dodeljivanje vrednosti, o čemu smo ranije pričali, pa bi umesto upoređivanja dodelili novu vrednost promenljivoj.

Da bi se zaštitili od uobičajenih grešaka, primenjujemo trik zamene pozicija promenljive i konstante u poređenju, kao u sledećem primeru:

```
if ( "Petar" == $ime ) {
```

Dobijamo isti efekat, ali pogledajmo šta će se desiti ako greškom napišemo samo jedan znak jednakosti. PHP će pokušati da dodeli vrednost promenljive ($ime) konstantnoj vrednosti (Petar). Pošto ne može promeniti vrednost konstante, PHP će blokirati i na ekranu će se pojaviti poruka o grešci i odmah će nam skrenuti pažnju da smo zaboravili još jedan znak jednakosti.

Uslovi mogu biti i komplikovaniji. Sada ćemo malo modifikovati prethodni primer, da bi pokazali i malo složeniju varijantu:

```
if ( "Petar" == $ime and "Petrovic" == $prezime ) {
  echo( " Petre, poseban pozdrav za tebe!" );
}
```

Uslov će biti tačan ako i samo ako $ime ima vrednost Petar, a $prezime ima vrednost Petrovic. Reč and u prethodnom primeru, pravi ceo uslov tačan samo ako su obe vrednosti tačne. Drugi takav operator je `or` koji čini da uslov bude tačan, ako je tačan bar jedan od dva jednostavna uslova. Slično kao što su u JavaScript ili C `&&` i `||`.

Sada ćemo pogledati malo komplikovanija poredjenja. Za početak, osnovne sličnosti sa if-else iskazima su dovoljne.

Druga, takođe veoma često korišćena kontrola strukture je while petlja. Sa if-else iskazima mi odlučujemo da li ćemo i koji iskaz da upotrebimo, dok sa while određujemo koliko puta će se određeni set naredbi izvršiti. Evo jednog primera while petlje:

```
while ( <uslov> ) {
  // izraz(i) koji ce se izvrsavati
  // sve dok <condition>
  // bude bio tacan
}
```

Rad je veoma sličan if-else iskazu, ali bez else klauzule. Rzlika je u tome što umesto da nastavi izvršavanja sledećeg izkaza, rad se nastavlja u okviru while petlje i uslov se proverava ponovo. Ako je iskaz tačan set naredbi se izvršava drugi put, ako je opet tačan onda i treći, četvrti... Prvi put kada iskaz ne bude tačan, nastavlja se dalje od kraja bloka (odnosno od }).

Petlja se koristi kada se radi sa velikim listama, ali ćemo sada to ilistrovati na jednom trivijalnom primeru (brojanje do deset):

```
$count = 1;
while ($count <= 10) {
  echo( "$count " );
  $count++;
}
```

Sve to deluje lako, ali hajde da ipak sve to objasnim red po red.

Prva linija kreira promenljivu $count i dodeljuje joj vrednost 1. U drugoj liniji počinje while petlja i uslov da je vednost $count manja ili jednaka 10. Treća i četvrta linija čine telo while petlje i biće izvršavane, sve dok uslov bude bio tačan. Treći je jednostavno ispisivanje vrednosti sa $count blankom. Četvrta je povećavanje vrednosti za 1 (Inače $count++ je skraćenica za $count = $count +1).

Sada smo videli šta se dešava sa delovima koda pri izvršavanju. Prvo se proverava uslov, vrednost $count je 1, sto je očigledno tačno i ispisuje se vrednost $count(1). Sada $count dobija novu vrednost 2, ponovo se proverava i pošto je i ona tačna, onda se i ona ispisuje i nastavlja se dalje 3, 4, 5, 6, 7, 8, 9, 10. Konačno $count dobija vrednost 11 sto ne ispunjava uslov i tu se završava proces. Na kraju kao rezultat dobijamo niz "1 2 3 4 5 6 7 8 9 10".

Takođe mozemo koristiti i ostale poredbene operatore <= (manje i jednako), >= (veće i jednako), < (manje), > (veće), != (nije jednako). Poslednji se takođe koristi i kada se upoređuju tekstualni nizovi.

Višenamenske stranice

Pretpostavimo da želimo da konstruišemo sajt, tako da ime posetioca bude prikazano na vrhu svake strane. Sa našim primerom "poruke za dobrodošlicu", smo već na pola puta. Ovde treba da proširimo problem, pa bi bilo dobro da prvo vidimo kako ćemo to da uradimo:

- Potrebno nam je ime na svakoj strani sajta, a ne samo na jednoj
- Nemamo kontrolu koju će stranu posetilac prvo pogledati

Problem i nije tako teško savladati. Ako jednom uzmemo korisnikovo ime možemo ga postaviti kao promenljivu na stranicu i možemo ga ubaciti sa svakim zahtevom za narednu stranu uz dodatak imena upitu za svaki link. To bi izgledalo otprilike ovako:

```
<A HREF="newpage.php?name=<?php echo(urlencode($ime)); ?>"> Link </A>
```

Napomenimo da imamo pravo ugraditi u PHP kod izvesne HTML tagove. To je potpuno ispravno i sve će raditi savršeno. Upoznali smo se sa echo funkcijom, ali urlencode je nova. Sta se dešava ako funkcija naiđe na specijalne karaktere (kao što je npr. blanko) i konvertuje to u specijalne kodove, neophodne da bi mogao da se ispiše upit. Npr. ako je promenljiva $ime koja ima vrednost "Petar Petrovic", tada blanko ne može da se nađe u *nizu* upita, tada će izlaz urlencode biti "Petar+Petrovic", koji će se automatski konvertovati u $ime promenljivu na novoj strani.

OK sada možemo da ime posetioca sajta ubacimo na svaku stranicu. Sada sve što nam je potrebno je da uzmemo ime na prvom mestu. U npr. našoj "poruci za dobrodošlicu", mi smo imali HTML stranu sa koje smo dostavljali ime posetioca našeg sajta. Problem sa ovim je što ne možemo – niti želimo – tražiti od posetioca da upisuje to svaki put kada poseti naš sajt.

Rešenje je da na svakoj stranici našeg sajta, proveravamo da li je ime specificirano i tražiti ime posetiocu ako je potrebno. Taj način da se na svakoj stranici našeg sajta ispiše bilo koji sadržaj ili zatraži upis imena, zavisi da li promenljiva $ime ima neku vrednost. Kod za više-namensku stranu izgleda otprilike ovako:

```
<HTML>
<HEAD>
<TITLE> Visenamenska strana </TITLE>
</HEAD>
<BODY>


<?php if (<uslov>) { ?>

<!—sadrzaj HTML koji ce se ispisati ako je <condition> tacan -->

<?php } else { ?>



<!—sadrzaj HTML koji ce se ispisati ako je <condition> ne tacan -->

<?php } ?>

</BODY>
</HTML>
```

Može delovati malo zbunjujuće na prvi pogled, ali u suštini to je običan if-else iskaz sa delom HTML koda. Ovaj primer ilustruje jedan veliki adut PHP: da kad god poželis možeš izaći iz PHP koda. Tako oznaka <?php znači ulazak u "PHP mod", a ?> znaci izlazak iz njega i povratak u normalan HTML kod, a prethodni primer pokazuje kako se pravi odlična kombinacija te dve stvari.

To je izvesna naizmenična forma if-else iskaza, koji čini naš kod znatno čitljivijim u situacijama kao što je ova. Ovo je izlazna strana za višenamensku stranu koja koristi naizmenično if-else iskaze:

```
<HTML>
<HEAD>
<TITLE> Visenamenska strana </TITLE>
</HEAD>
<BODY>

<?php if (<uslov>): ?>

<!—sadrzaj HTML koji ce se ispisati ako je <condition> tacan -->

<?php else: ?>

<!—sadrzaj HTML koji ce se ispisati ako je <condition> ne tacan -->
<?php endif; ?>
</BODY>
</HTML>
```

OK, ajde da sada pogledamo ovu stranu sa svim dodacima koji su potrebni:

```
<HTML>
<HEAD>
<TITLE> Probna strana </TITLE>
```

```
</HEAD>
<BODY>

<?php if ( isset($ime) ): ?>

  <P>Ime: <?php echo($ime); ?></P>

  <P>Ovaj pasus sadrzi
<A HREF="newpage.php?name=<?php echo(urlencode
($ime)); ?>">link</A> ovaj pasus
je ime promenljive slececeg dokumenta.</P>

<?php else: ?>

  <FORM ACTION=<?php echo($PHP_SELF); ?> METHOD=GET>
  Upisite vase ime: <INPUT TYPE=TEXT NAME="ime">
  <INPUT TYPE=SUBMIT VALUE="GO">
  </FORM>

<?php endif; ?>

</BODY>
</HTML>
```

Ubacili smo dva nova trika u prethodni kod i dobili smo zgodniji i komforniji način rada. Prvo smo uveli novu funkciju isset u uslovu. Funkcija vraća (izlazno) tačnu vrednost, ako je promenljivoj dodata neka vrednost, i netačno ako promenljiva ne postoji. Drugi trik koji smo upotrebili je da je promenljiva $PHP_SELF naznačena kao dodatak ACTION u FORM tagu. Ova promenljiva je jedna od nekoliko kojima PHP automatski dodeljuje vrednost. Konkretno $PHP_SELF će uvek biti podešena kao aktuenlna strana. To nam omogućava da lakše kreiramo FORM, kada pošaljemo on će učitati veoma sličnu stranu, ali sada sa promenljivom $ime koja je specificirana.

Pomoću konstrukcije svih strana na našem sajtu, na ovaj način, posetilac će proslediti svoje ime na prvoj strani kojoj pristupi, ma koja to strana bila. Gore unešeno njegovo ime pritiskom na "GO" će pretstaviti sa tačnom stranom gde je zahtev. Ime koje se unese biće prosleđeno u upitnik svakog linka kojem bude hteo pristupiti.

Predstavljenje podataka iz MySQL baze na web

Sada ćemo naučiti kako se uzimaju podaci sačuvani u bazi i prikazuju na web strani. Pošto smo naucili kako se instalira i osnovno o MySQL, kao i kako se koriste neke osnovne komande PHP, *server-side scripting* jezika. Sada nastavljamo, videćemo kako možemo da ove dve stvari koristimo zajedno da bi kreirali bazu – koristeći web stranu !

Povezivanje sa MySQL pomoću PHP

Pre nego što preuzmemo sadržaj iz naše MySQL baze i ubacimo ga na naš web sajt, prvo moramo znati kako se uspostavlja veza sa MySQL bazom. Vratimo se na prethodnu lekciju (br. 1), koristili smo program mysql, koji je dozvoljavao da ostvarimo tu vezu. PHPu, međutim, nije potreban specijalan program da bi ostvario tu

vezu; u sam jezik je ugrađena podrška za povezivanje sa MySQL bazom. Sledeća PHP funkcija uspostavlja vezu:

mysql_connect(<address>, <username>, <password>);

Gde <address> predtsavlja IP ili hostname kompjutera na kome se nalazi MySQL sa kojim želimo da uspostavimo vezu ("localhost", ako se pokreće na istom kompjuteru), a <username> i <password> su isti kao oni o kojima smo pričali u lekciji br. 1.

Funkcija mysql_connect, vraća broj kao potrvdu da je uspostavljena veza. Pošto želimo da uspostavimo običnu vezu, mi ćemo prihvatiti tu vrednost.Evo primera kako možemo uspostaviti vezu sa nasim MySQL sevrverom:

$dbcnx = mysql_connect("localhost", "root", "mypasswd");

Kao što smo već rekli fukncija se sastoji od tri vrednosti koje su karakteristične za svaki MySQL server. Ono što je važno ovde primetiti je da se vrednost koju vraća mysql_connect, čuva u promenljivoj $dbcnx.

Posto je MySQL server potpuno zaseban deo, moramo predvideti mogućnost da je server "nedosutpan" ili da postoji posebna dozvola pristupa ili da username/password nije ispravan. U tom slucaju mysql_connect neće vratiti nikakav indikator (da veza nije uspostavljena). Umesto toga, prijaviće grešku. Da bi premostili taj problem koristimo if iskaz :

```
$dbcnx = @mysql_connect("localhost", "root", "mypasswd");
if (!$dbcnx) {
  echo( "<P>Trenutno nije moguce pristupiti " .
        "serveru sa bazom.</P>" );
  exit();
}
```

Uveli smo tri nova trika u ovom delu koda. Prvo, ubacili smo @ ispred mysql_connect funkcije. Mnogo funkcija, uključuje mysql_connect i automatski ispisuje "ružnu" poruku o grešci kada je neispravno. Ubacivanjem znaka @ isped imena funkcije, omogućavamo da mi sami upišemo našu porku za grešku (koja će imati mnogo lepši oblik).

Dalje, stavili smo znak uzvika pre promenljive $dbcnx u uslovu sa if iskazom. Operator ! u PHP znači negaciju, koja u suštinu prebacuje netačan iskaz u tačan ili obrnuto. Prema tome, ako je ova konekcija neuspešna vraća "netačno", $dbcnx će se vrednovati kao tačno i to će uticati da se iskaz iz našeg if iskaza izvrši.

Ako se uspostavi veza, identifikator konekcije, sačuvan u $dbcnx će pokazati "tačno" (svaki broj sem nule u PHP, predstavlja tačnu vrednost), pa će !$dbcnx biti netačno i if iskaz se neće izvršiti.

Treći trik je funkcija exit, koja je prvi primer funkcije koja ne zahteva nikakve parametre. Sve funkcije prestaju da se čitaju na tom mestu. To je dobra rekacija na loše uspostavljenu vezu sa bazom.

Sledeći korak je uspostavljanje veze sa bazom sa kojom želimo da radimo. Pošto je već kreirana, sada ćemo samo nastaviti rad sa našom bazom Vicevi. Izbor baze u PHP je uz pomoć sledeće konstrukcije :

mysql_select_db("vicevi", $dbcnx);

Koristimo promenljivu $dbcnx koja sadrži identifikator o povezanosti sa bazom da kaže funkciji koju bazu da upotrebi sa konekcijom. Taj parametar je opcioni. Kada krene, funkcija će automatski uzeti link identifikator za poslednju ostvarenu vezu. Funkcija vraća tačno, kada je uspešno i netačno kada se javi greška. Još jednom, treba biti oprezan prilikom korišćenja `if` iskaza za signaliziranje greške:

```
if (! @mysql_select_db("vicevi") ) {
  echo( "<P>Trenutno ne moze da locira " .
        "bazu vicevi.</P>" );
  exit();
}
```

Pošto je veza uspostavljena i izabrana baza, sada smo spremni da počnemo sa korišćenjem potadaka sačuvanih u bazi.

Prikazivanje SQL upita pomoću PHP

PHP ima sličan mehanizam onom koga smo opisali u provom delu skripte, a koji se ostvaruje uz pomoć mysql_query funkcije.

mysql_query(<query>, <connection id>);

Gde je <query> niz koji sadrži SQL komande koje će se izvršiti. Kao i sa mysql_select_db identifikator konekcije je opcion.

Šta će funkcija vratiti zavisi od tipa upita koji je poslat. Za većinu funkcija mysql_query će vratiti "tačno" ili "netačno", kao indikator uspeha ili neuspeha. Pogledajmo sledeći primer, koji treba da napravi tabelu Vicevi, koju smo mi kreirali u prethodnom delu.

```
$sql = "CREATE TABLE Vicevi ( " .
       "ID INT NOT NULL AUTO_INCREMENT PRIMARY KEY, " .
       "VicText TEXT, " .
       "VicDate DATE NOT NULL " .
       ")";
if ( mysql_query($sql) ) {
  echo("<P>Tabela Vicevi je uspesno kreirana!</P>");
} else {
  echo("<P>Greska prilikom kreiranja baze Vicevi: " .
       mysql_error() . "</P>");
}
```

Funkcija mysql_error koja je upotrebljena, vraća *niz* teksta sa opisom poslednje greške, poslate sa MySQL servera.

Za DELETE, INSERT i UPDATE upite (koji služe za promenu sačuvanih podataka), MySQL takođe, čuva *track* broja redova tabele (upisa) koji su obuhvaćeni upitom. Pogledajmo sledeću komandu, koju smo koristili u prethodnom delu da podesimo datume svih viceva koji sadrže reč "zec".

```
$sql = "UPDATE Vicevi SET VicDate='2000-10-04' " .
       "WHERE VicText LIKE '%zec%'";
```

Kada se izvrši upit, možemo koristiti funkciju mysql_affected_rows da vidimo broj redova koji su obuhvaćeni ovom promenom.

```
if ( mysql_query($sql) ) {
  echo("<P>Efekti azuriranja" .
       mysql_affected_rows() . " rows.</P>");
} else {
  echo("<P>Greska: " .
       mysql_error() . "</P>");
}
```

Manipulisanje rezultatima pomoću SELECT

Upiti SELECT su malo drugačiji u odnosu na ostale komande....

Za većinu SQL upita, funkcija mysql_query vraća uglavnom tačno (uspešno) i netačno (neuspešno). Za SELECT to nije dovoljno. Pozivamo SELECT funkciju da bi videli podatke sačuvane u bazi. U dodatku indikatora, da li je je upit uspešan ili neuspešan, PHP takođe mora da primi rezultat upita. Kao rezultat poziva SELECT, mysql_query vraća broj identifikator "podešenosti rezultata", dobija se lista redova (sadržaj baze). Greška se ne vraća, ako je upit loš iz bilo kog razloga.

```
      $result = mysql_query("SELECT VicText FROM Vicevi");
if (!$result) {
  echo("<P>Greska: " .
       mysql_error() . "</P>");
  exit();
}
```

Sa sigurnošću možemo reći da neće biti greške u upitu. Prethodni primer će u rezultatu sadržati tekst sa svim vicevima, sačuvanim u tabeli vicevi kroz promenljivu $result. Pošto praktično nemamo ograničenje broja viceva u bazi, rezultat koji dobijemo može biti previše veliki.

Da bi razjasnili while petlju, koju smo ranije spomenuli, pozabavićemo se sledećim primerom:

```
$row = mysql_fetch_array($result);
```

Funkcija mysql_fetch_array prihvata rezultat i setuje ga kao parametar (čuva u promenljivoj $result u ovom slučaju), i vraća sledeći red u rezultatu kao niz. Ako vam nije poznat koncept nizova sada ćemo ga ukratko objasniti. Kada nema više kolona rezultat se podešava na, mysql_fetch_array umesto vraćanja greške.

U prethodmom primeru, naznačena je promenljiva $row, ali u isto vreme čitav iskaz uzima istu vrednost. To je kao kada uzimamo uslov while u petlji. Pošto while petlje čuvaju petlju dok uslov ne postane netačan, petlja se dešva više puta u ispisivanju rezultata, sa $row se uzima vrednost sledeće kolone svakim prolazom kroz petlju. Svi ti izlazi se pojavljuju napolju.

Redovi se u rezultatu prikazuju kao nizovi. Niz je specijalni tip promenljive koji sadrži mnogo vrednosti. Ako zamislimo promenljivu kao kutiju koja sadrži

vrednosti, onda bi niz mogao da bude kao kutija sa odeljcima, gde svaki odeljak može da se sačuva kao posebna vrednost. U slučaju redova naše baze, odeljci su imena posle tabele kolona u našem rezultatu. Ako je $row red u našem rezultatu tada $row["VicText"] je vrednost u kolini VicText reda. Evo sada i kako bi izgledala while petlja ako bi hteli da ispišemo tekstove svih viceva iz naše baze :

```
while ( $row = mysql_fetch_array($result) ) {
  echo("<P>" . $row["VicText"] . "</P>");
}
```

Da bi sada sve to sumirali, pogledajmo kompletan kod PHP web strane, povezivanje sa bazom, preuzimanje teksta svih viceva iz naše baze i ispisivanje u HTML paragrafima:

```
<HTML>
<HEAD>
<TITLE> Lista nasih viceva </TITLE>
<HEAD>
<BODY>
<?php

  // Povezivanje sa serverom baze
  $dbcnx = @mysql_connect("localhost",
           "root", "mypasswd");
  if (!$dbcnx) {
    echo( "<P>Trenutno ne moze " .
          "da se pristupi bazi.</P>" );
    exit();
  }

  // Odabir baze vicevi
  if (! @mysql_select_db("vicevi") ) {
    echo( "<P>Trenutno ne moze da se " .
          "poveze sa bazom vicevi.</P>" );
    exit();
  }

?>
<P> Ovo su vicevi koji se nalaze u nasoj bazi: </P>
<BLOCKQUOTE>

<?php

  // Zahtev za tekstovima viceva
  $result = mysql_query(
            "SELECT VicText FROM Vicevi");
  if (!$result) {
    echo("<P>Greska: " .
         mysql_error() . "</P>");
    exit();
  }


  // Ispisivanje viceva
  while ( $row = mysql_fetch_array($result) ) {
    echo("<P>" . $row["VicText"] . "</P>");
  }

?>
```

```
</BLOCKQUOTE>
</BODY>
</HTML>
```

Ubacivanje podataka u bazu

U ovom delu videćemo kako možemo da koristimo sve razpoložive alate da bi omogućili posetiocima našeg sajta da ubace njihov vic u bazu. U ovom delu je jako malo novog materijala. U suštini to je skup svega sto smo učili do sada.

Ako želimo da omogućimo posetiocima našeg sajta da upišu svoj vic u bazu, neophodno je da obezbedimo formu.

```
      <FORM ACTION="<?php echo($PHP_SELF); ?>" METHOD=POST>
<P>Upisi vic:<BR>
<TEXTAREA NAME="victext" ROWS=10 COLS=40 WRAP></TEXTAREA><BR>
<INPUT TYPE=SUBMIT NAME="posalji" VALUE="SUBMIT">
</FORM>
```

Kao što smo ranije videli, kada pošaljemo ovu formu, učitaćemo veoma sličnu stranu (ponovo smo koristili promenljivu $PHP_SELF kao dodatak ACTION), ali sa dve promenljive dodate zahtevu. Prvo $victext sadrži tekst vica koji je unet u *text area*. Drugo $posalji sadrži vrednost SUBMIT koja služi da bi tekst bio poslat.

Za ubacivanje vica u bazu, koristimo samo mysql_query da bi pokrenuli INSERT upit, koristimo promenljivu $victext da bi poslali vrednost

```
if ("SUBMIT" == $posalji) {
  $sql = "INSERT INTO Vicevi SET " .
         "VicText='$victext', " .
         "VicDate=CURDATE()";
  if (mysql_query($sql)) {
    echo("<P>Tvoj vic je upisan u bazu.</P>");
  } else {
    echo("<P>Greska: " .
         mysql_error() . "</P>");
  }
}
```

Još jedan novi trik u prethodnom primeru. Iskoristili smo MySQL funkciju CURDATE() da bi upisali trenutan datum kao vrednost promenljive VicDate kolone. MySQL sadrži gomilu takvih funkcija, ali mi ćemo za sada uvesti samo ovu. Ostale funkcije, će se pored ostalog nalaziti u dodatku kompletne verzije "Vodiča kroz PHP i MySQL".

Sada imamo kod koji dozvoljava korisniku da unese vic i da ga ubaci u bazu podataka. Sada je ostalo još samo da ukomponujemo sve što smo naučili u jednu stranu web stranu. Evo kako bi to izgledalo:

```
<HTML>
...
<BODY>
<?php
  // Ako korisnik zeli da doda vic
  if (isset($dodajvic)):
?>
```

```
<FORM ACTION="<?php echo($PHP_SELF); ?>" METHOD=POST>
<P>Upis vic:<BR>
<TEXTAREA NAME="victext" ROWS=10 COLS=40 WRAP>
</TEXTAREA><BR>
<INPUT TYPE=SUBMIT NAME="posalji" VALUE="SUBMIT">
</FORM>


<?php
  else:

    // Povezivanje sa bazom na serveru
    $dbcnx = @mysql_connect("localhost",
             "root", "mypasswd");
    if (!$dbcnx) {
      echo( "<P>Trenutno nije moguce " .
            "ostvariti konekciju sa bazom.</P>" );
      exit();
    }

    // Biramo bazu vicevi
    if (! @mysql_select_db("vicevi") ) {
      echo( "<P>Trenutno nije moguce " .
            "pristupiti bazi vicevi.</P>" );
      exit();
    }

    // Ako je vic poslat,
    // dodajemo ga u bazu.
    if ("SUBMIT" == $posalji) {
      $sql = "INSERT INTO Vicevi SET " .
             "VicText='$victext', " .
             "VicDate=CURDATE()";
      if (mysql_query($sql)) {
        echo("<P> Tvoj vic je upisan u bazu.</P>");
      } else {
        echo("<P>Greska: " .
             mysql_error() . "</P>");
      }
    }

    echo("<P> Ovo su vicevi koji " .
         " se nalaze u nasoj bazi: </P>");

    // Zahtevamo ispis svih viceva iz baze
    $result = mysql_query(
              "SELECT VicText FROM Vicevi");
    if (!$result) {
      echo("<P>Greska: " .
           mysql_error() . "</P>");
      exit();
    }

    // Ispisujemo sve viceve iz baze na web stranu
    while ( $row = mysql_fetch_array($result) ) {
      echo("<P>" . $row["VicText"] . "</P>");
    }

    // Kada korisnik klikne na ovaj link
    // otvori mu se strana za upisivanje viceva.
```

```
    echo("<P><A HREF='$PHP_SELF?dodajvic=1'>" .
         "Dodaj viv!</A></P>");

  endif;

?>
</BODY>
</HTML>
```

To je to ! Sa jednim dokumentom koji sadrži mali PHP kod mi smo u mogućnosti da vidimo i dodamo viceve u našu MySQL bazu.