Uvod u PHP i MySQL

Na osnovu Uvod u PHP i MYSQL (Tečaj D350 Srce Zagreb)
https://www.srce.unizg.hr/tecajevi/popis-osnovnih-tecajeva/D350

Prof. dr Saša Salapura

Uvod

- upoznavanje sa programskim jezikom za dinamičko stvaranje web stranica PHP i

- upoznavanje sa sistemom za upravljanje relacionim bazama podataka MySQL

- potrebno je poznavanje osnova HTML

- poželjno je i poznavanje osnova rada sa bazama podataka SQL

- poželjno poznavanje osnova programiranja

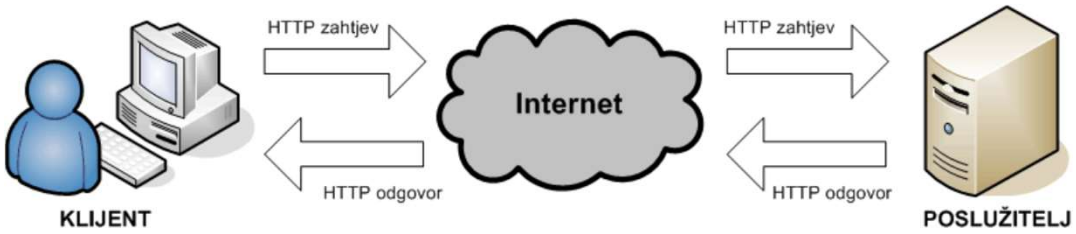
- polaznici će naučiti kako se programira u jeziku PHP, te kako se pomoću njega generišu web stranice, kako se radi sa bazama podataka na čemu se zasnivaju dinamičke i interaktivne web stranice i aplikacije.

1. Uvod u PHP i MySQL

PHP je skriptni jezik koji se izvršava na serveru, a glavna mu je namjena dinamičko stvaranje web stranica.

MySQL je sistem za upravljanje relacionim bazama podataka, i zajedno s programskim jezikom PHP predstavlja jedno od najpopularnijih rješenja za izradu dinamičkih

1.1. Klijent - server



Većina jezika koji se koriste u web programiranju (PHP, JavaScript, ASP i drugi) su podskup skriptnih jezika za koje je karakteristično da program nije izvršna (exe) datoteka, već se prevođenje naredbi obavlja pri svakom izvršavanju programa. Datoteke i dijelovi kôda napisani u takvim jezicima nazivaju se skriptama.

U komunikaciji na Internetu sudjeluju dvije strane: klijent i server.

Klijent (korisnikov web preglednik) šalje zahtjev - npr. adresu neke stranice.

Server vraća odgovor u kom se nalazi HTML kôd tražene stranice (ali i binarne datoteke, kao što su slike, video...) koji se u klijentovom web pregledniku (browser) prikazujekao web stranica.

Komunikacija između klijenta i servera odvija se putem HTTP protokola, koji se sastoji od HTTP zahtjeva i odgovora.

Komunikacija preko Interneta je dvosmjerna – klijent može unutar HTTP zahtjeva poslati neke podatke serveru.

Statičke i dinamičke HTML stranice

Statičke HTML stranice

Sa statičkim HTML stranicama, web stranica će uvijek biti ista, a mogućnosti joj se svode na prikazivanje teksta i slika, s mogućnošću povezivanja, preko linkova, s drugim stranicama.

HTML stranice s klijentskim skriptama

Klijentske skripte se zajedno s HTML kôdom (unutar HTML datoteka ili unutar zasebnih datoteka) nalaze na serveru. Kad stigne zahtjev, skripte se zajedno sa HTML kôdom šalju klijentu gdje se izvršavaju kad budu pozvane (nakon što se stranica učita u preglednik, kad korisnik klikne na dugme i slično).

Klijentske skripte nude nove mogućnosti web stranica, (najkorisnije su prikazivanje prozora sa porukama, trenutno prikazivanje i skrivanje određenih dijelova na stranici i drugi dinamični vizualni efekti - promjena slike pri prelasku mišem i slično).

HTML stranice stvorene pomoću serverskih skripti

Serverske skripte se također nalaze se na serverui i također mogu biti ubačene u HTML kôd (mogu se nalaziti i u zasebnim datotekama). Kad stigne zahtjev od klijenta, skripta se izvršava na poslužitelju, a kao rezultat izvršavanje dobiva se HTML kôd koji se šalje klijentu. Najveća prednost serverskih skripti je njihova mogućnost povezivanja sa bazama podataka, što je dovelo do pojave web stranica i web aplikacija sa dinamičkim sadržajem koji se čita iz baze podataka a koji korisnici mogu mijenjati preko web aplikacije.

Pregled mogućnosti klijentskih i serverskih skripti

Klijentske skripte	Serverske skripte
Izvršavaju se na klijentu	Izvršavaju se na serveru
nekompatibilnost sa web preglednicima (neke stvari neće raditi u potpunosti u svim preglednicima)	aplikacija će uvijek raditi na isti način, neovisno o pregledniku
nemogućnost povezivanja sa bazama podataka	povezivanje sa bazama podataka
mogućnost reagovanja na veliki broj događaja (povećanje prozora, desni klik miša i sl.)	aplikacija reaguje samo na klik na link ili na dugme
aplikacija reaguje trenutno, stranica se ne osvježava	veće vrijeme čekanja na odgovor, potrebno osvježavanje cijele stranice
server je rasterećen korištenjem resursa klijenta	uvijek se koriste resursi servera

Izvršavanje skripti na klijentskoj ili serverskoj strani ne isključuju jedno drugo – većina stvari koje se može postići jednim načinom programiranja ne može se postići drugim te se koriste kombinovano.

Glavni klijentski skriptni jezik je JavaScript (čija je sintaksa slična programskim jezicima C i Java), koji je podržan u najvećem dijelu web preglednika.

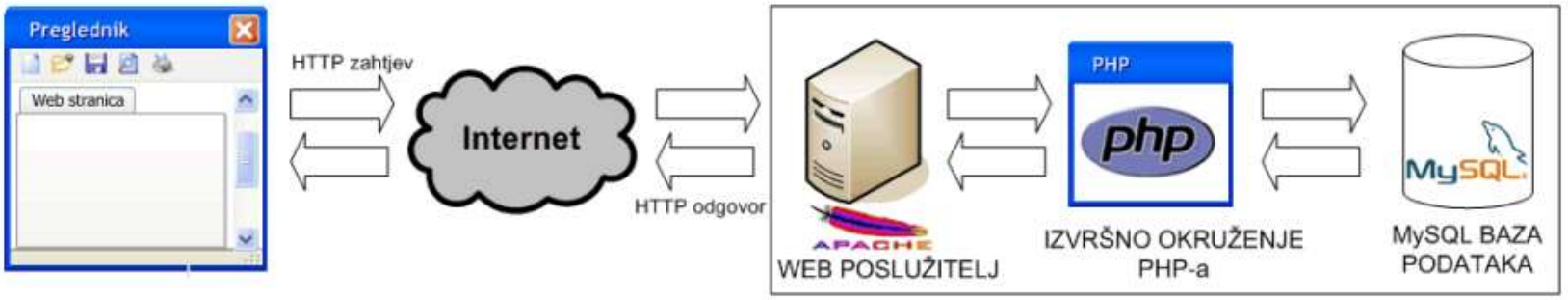
Osim njega, još postoje i koriste se JScript (Microsoftova verzija JavaScripta) i VBScript (skriptni jezik sa sintaksom sličnom programskom jeziku Visual Basic, također Microsoftov).

AJAX (Asynchronous JavaScript and XML) je tehnologija koja kombinuje klijentske i serverske skripte. Na određenu korisnikovu akciju klijentska skripta šalje pozadinski zahtjev skripti na serveru koja dohvata podataka iz baze podataka i šalje ih korisnikovom pregledniku, **bez osvježavanja** cijele stranice. Upotrebom tehnologije AJAX web aplikacije postaju brže i imaju bolji korisnički interfejs od aplikacija koje se temelje samo na poerverskim tehnologijama.

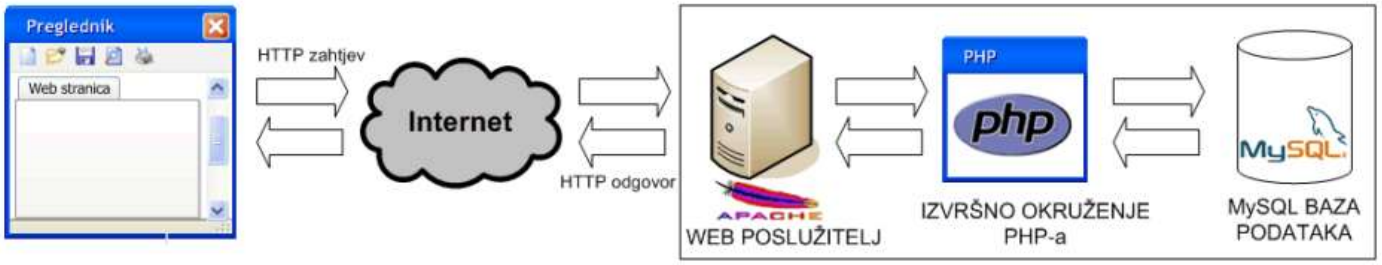
PHP

Originalna skraćenica PHP značila Personal Home Page Tools - kasnije je značenje promijenjeno pa danas skraćenica PHP znači Hypertext Preprocessor - opisuje glavnu funkciju jezika PHP - da na osnovu PHP naredbi generiše HTML.

Najčešća upotreba kroz LAMP platformu (Linux, Apache, MySQL i PHP). Web server Apache prima HTTP zahtjev, i ako se traži stranica sa ekstenzijom '.php', poziva se izvršno okruženje PHP-a, koje prevodi i izvršava PHP naredbe.



PHP



Ukoliko skripta treba dohvatiti neke podatke iz baze podataka, bazi se postavljaju upiti iz PHP-a.

Dobijeni podaci se prosljeđuju PHP-u koji ih uklapa unutar HTML kôda. Taj HTML kôd se unutar HTTP odgovora šalje natrag klijentu i podaci se prikazuju u korisnikovom web pregledniku.

Ostale serverske tehnologije

Konkurent skriptnom jeziku PHP bio je **ASP** (Active Server Pages), Microsoftova serverska tehnologija jednakih mogućnosti.

ASP je zastario, a PHP se nastavio razvijati – dobija podršku za objektno-orijentirano programiranje. Postoji nekoliko frameworka (razvojnih platformi) za PHP (Zend Framework, Symfony) čiji je cilj olakšati programiranje u njemu odvajanjem prezentacije (HTML) od logike (PHP kôd), te uvođenjem objekata koji predstavljaju HTML elemente i proširuju funkcionalnost web stranica, zatim objekata za rad s bazama podataka i drugih.

Veliki konkurent ostaje Microsoft **ASP.NET** koji koristi .NET Framework za koji je moguće razvijati u više .NET jezika, od kojih su najrašireniji C# i Visual Basic.NET. Prednost je veliki broj gotovih kontrola i objekata, i to što se njegove datoteke ne moraju prevoditi svaki put kod izvršavanja, već samo prvi put.

Ostale serverske tehnologije

Jaka konkurencija je i **JSP** (JavaServer Pages) za koji danas postoji velik broj frameworka (Spring, Struts, JavaServer Faces...). Jednako kao ASP.NET, ne mora da se prevodi svaki put kod izvršavanja, a vrlo je raširen u svijetu poslovnih aplikacija.

Perl je skriptni jezik UNIX sustava, i to jedan od prvih s kojima je započeo razvoj web aplikacija. Ima velike mogućnosti, a njegov razvoj i dalje traje pa se i danas koristi za razvoj web aplikacija.

Python je skriptni jezik koji se može upotrebljavati i za web te postoji nekoliko frameworka.

Ruby je jezik sličan jeziku Python, a u posljednje vrijeme je vrlo popularan u kombinaciji s Rails frameworkom koji omogućava brz i jednostavan razvoj aplikacija.

Jedan od konkurenata skriptnom jeziku PHP je i **ColdFusion**, poslužiteljski skriptni jezik i tehnologija u vlasništvu Macromedie.

Baze podataka - MySQL, MSSQL, Oracle

MySQL je najrasprostranjeniji open source susistem za upravljanje bazama podataka na webu. Neki od korisnika su YouTube, Filckr, Wikipedia i Wordpress.

Od ostalih open source sistema za upravljanje bazama podataka treba spomenuti **PostgreSQL** (više se koristi u poslovnim sistemima, a manje na webu).

Od komercijalnih sistema za upravljanje bazama podataka najznačajniji su **Microsoft SQL Server** i **Oracle Database**. I jedan i drugi upotrebljavaju se u zahtjevnim poslovnim okruženjima, pri čemu se prvi značajno koristi i na webu. Baze podataka napravljene u programu **Microsoft Access** se također često upotrebljavaju na webu, pogotovo u kombinaciji s ASP-om. No, treba napomenuti da su performanse takve baze podataka slabe u poređenju sa pravim sistemima za upravljanje bazama podataka.

Osnove PHP sintakse

PHP naredbe najčešće se nalaze umetnute unutar HTML kôda zajedno s HTML oznakama.

Da bi ih PHP prevodilac mogao prepoznati, moraju se nalaziti unutar posebnih oznaka. Najčešći i preporučeni način pisanja PHP oznaka je ovaj:

```
<?php


?>
```

Ili skraćeni način pisanja PHP oznaka:

```
<?


?>
```

Preporučuje se prvi način pisanja PHP oznaka jer skraćeni način pisanja ne mora biti podržan na svim serverima!

Svaka PHP naredba mora završiti sa znakom tačka-zarez (’**;**’), inače će se prilikom izvršavanja javiti sintaksna greška.

Primjer naredbe u jeziku PHP:

```
<?php
  echo "Dobar dan";
?>
```

Naredba **echo** ispisuje niz znakova, pa će gornji primjer ispisati tekst "Dobar dan".

Vježba 1.1 – Prva PHP skripta

U postojeći HTML kod

```
<html>
<head>
 <title>Prva PHP skripta</title>
</head>
<body>
 <h2>Prva PHP skripta</h2>
</body>
</html>
```

Ubaciti PHP kod (`v1.php`)

```
<html>
<head>
 <title>Prva PHP skripta</title>
</head>
<body>
 <h2>Prva PHP skripta</h2>
 <?php
     echo "Pozdrav svima!!!";
 ?>
</body>
</html>
```

U web pregledniku korisnika biće prikazano:

a sa 'View source' se vidi kako izgleda preuzeti HTML kod

Vježba 1.2 – Druga PHP skripta

```
<!DOCTYPE html>
<html>
<body onload="startTime()">
<h2>JavaScript Clock</h2>
<div id="txt"></div>
<h2>PHP Clock</h2>
<?php
 echo "Tačno vrijeme (prije promjene timezone): " . date("H:i:s");
 date_default_timezone_set("CET");
 echo "<br>" ;
 echo date_default_timezone_get();
//https://www.php.net/manual/en/timezones.php  //sve vrem.zone
 echo "<br>Tačno vrijeme (nakon promjene tz): " . date("H:i:s");
?>
```

```
<script>
function startTime() {
  const today = new Date();
  let h = today.getHours();
  let m = today.getMinutes();
  let s = today.getSeconds();
  m = checkTime(m);
  s = checkTime(s);
  document.getElementById('txt').innerHTML =  h + ":" + m + ":" + s;
  setTimeout(startTime, 1000);
}
function checkTime(i) {
  if (i < 10) {i = "0" + i};  // add zero in front of numbers < 10
  return i;
}
</script>
</body>
</html>
```

Javascript sat se izvršava na klijentu: stalno se osvježava i pokazuje klijentsko vrijeme

PHP sat se izvršava na serveru: uzima serversko vrijeme svaki put kada se stranica osvježi i prikazuje vrijeme vremenske zone servera. Također, PHP koristi vlastitu vremensku zonu koja se može imjenjati komandom `date_default_timezone_set("CET");`


```
<!DOCTYPE html>
<html>
<body onload="startTime()">
<h2>JavaScript Clock</h2>
<div id="txt"></div>
<h2>PHP Clock</h2>
Tačno vrijeme (prije promjene timezone): 10:32:42<br>CET<br>Tačno vrijeme (nakon promjene timezone): 12:32:42<script>
function startTime() {
  const today = new Date();
  let h = today.getHours();
  let m = today.getMinutes();
  let s = today.getSeconds();
  m = checkTime(m);
  s = checkTime(s);
  document.getElementById('txt').innerHTML =  h + ":" + m + ":" + s;
  setTimeout(startTime, 1000);
}
function checkTime(i) {
  if (i < 10) {i = "0" + i};  // add zero in front of numbers < 10
  return i;
}
</script>
</body>
</html>
```

ftp pristup

Za upload datoteka na web server, ukoliko nemate neku drugu opciju, možete koristiti sljedeće kredencijale:

ftp.salapura.com

Username: student

Password: pim_BL_2021-22

Tamo će vas dočekati 10 različitih foldera u kojima možete postaviti php/html dokumente i testirati svoj kod.

Varijable i operacije nad varijablama

Varijable

varijable - za privremeno spremanje podataka tokom izvođenja programa.

operatori - simboli za različite operacije (pridruživanje, aritmetičke operacije, operacije poređenja…).

tipovi podataka:

- cjelobrojni
- tekstualni
- logički
- decimalni
- složeni (polje, objekat)
- specijalni tip:
  - resource (mem.adresa, datoteka, zapis iz DB)
  - NULL (varijablja kojoj nije pridružena vrijednost)

Varijable u PHP-u prije upotrebe nije potrebno deklarisati - navesti njen tip (za razliku od većine drugih programskih jezika) - dovoljno je navesti njezino ime i pridružiti joj neku vrijednost (varijabla će sama steći odgovarajući tip podatka u zavisnosti od pridružene vrijednosti).

I ne samo to, moguće je tokom upotrebe varijable promijeniti njen tip (što se ipak ne preporučuje).

Ispred svake varijable u jeziku PHP navodi se znak $.

Imena varijabli (identifikatori) mogu sadržavati slova, cifre ili donju crtu (_), a prvi znak (poslije $) može biti samo slovo ili donja crta.

Primjeri ispravnih i neispravnih imena varijabli dani su u tablici:

Ispravno	Nije ispravno
$mojaVarijabla	*mojaVarijabla*
$varijabla1	*$1varijabla*
$moja_Varijabla	*$moja Varijabla*
$_varijabla	*$v@rijabla*

Imena varijabli su case-sensitive:
razlikuju se velika i mala slova!!!

Konstante

Konstante sadržavati vrijednosti kao i varijable, ali se ta vrijednost ne može mijenjati

```
define(name, value, case-insensitive)
```

Vrijednost se konstantama dodjeljuje preko funkcije `define` koja sadrži tri parametra:

- ime konstante,
- njezinu vrijednost i
- da li je case-insensitive (podrazumijevana vrijednost je 'false')

Dodjela vrijednosti za tekstualnu vrijednost izgleda ovako:

```
<?php
 define("skola", "PIM");
 echo skola
?>
```

Imena konstanti su case-insensitive ako se drugačije ne definišu.

Pridruživanje vrijednosti varijabli

Vrijednost varijabli `$varijabla` dodjeljuje se naredbom pridruživanja:

```
<?php
     $varijabla = vrijednost;
?>
```

Gornja naredba je naredba pridruživanja,

Znak = je operator pridruživanja.

S **lijeve** strane operatora pridruživanja **mora** se nalaziti varijabla

Sa desne strane može biti konstantna, rezultat nekog izraza (npr. aritmetičkog) ili pak poziv funkcije.

Pridruživanje **tekstualne** vrijednosti varijabli:

```
<?php
    $Varijabla1 = "Ana";
?>
```

Pridruživanje **cjelobrojne** vrijednosti varijabli:

```
<?php
    $Varijabla2 = 2023;
?>
```

Dodjela vrijednosti **decimalnog** tipa:

```
<?php
    $Varijabla3 = 12.34;
?>
```

Pridruživanje pisane u **ekponencijalnom** zapisu:

```
<?php
    $Varijabla4 = 1.2e12;
?>
```

Pridruživanje **logičke** vrijednosti varijabli (ne razlikuje mala i velika slova):

```
<?php
    $Varijabla5 = True;
?>
```

Izjednačavanje vrijednosti dvije varijable:

```
<?php
    $Varijabla6 = $Varijabla1;
?>
```

Aritmetičke operacije - operatori

Osnovni aritmetički operatori

Operator	Značenje	Primjer
-	negativan predznak	`- $v1`
+	sabiranje	`$v1 + $v2`
-	oduzimanje	`$v1 - $v2`
*	množenje	`$v1 * $v2`
/	dijeljenje	`$v1 / $v2`
%	moduo (ostatak dijeljenja)	`$v1 % $v2`

Prioritet izvršavanja aritmetičkih operacija

Operacije se ne izvršavaju redoslijedom kojim su napisane već prema prioritetu aritmetičkih operacija (zadan u tabeli).

Prioritet	Operator	Značenje
1	-	negativan predznak
2	*, /, %	množenje, dijeljenje, moduo
3	+ , -	sabiranje, oduzimanje

Tako da će nakon pridruživanja

```
$v1 = 2 + 3 * 2;
```

Varijabla `$v1` imati vrijednost 8.

Redoslijed izvršavanja se može promijeniti upoterbom zagrada!

Aritmetičke operacije - operatori

Operatori inkrementacije/dekrementacije*

Operator	Značenje	Primjer
++	povećanje za 1	`$v1 = $v1 + 1;`
--	umanjivanje za 1	`$v1 = $v1 - 1;`

```
<?php
    $broj = $broj + 1;
?>
```

je isto što i

```
<?php
    $broj++;
?>
```

* povećavanje/umanjivanje za 1

```
$v1 = $v1 + 1;    inkrementacija
$v1 = $v1 - 1;    dekrementacija
```

Aritmetičke operacije - operatori

Ostali operatori inkrementacije/dekrementacije su:

Operator	Naziv	Značenje
++$v1	pre-inkrement	povećaj $v1 za 1 pa vrati $v1
$v1++	post-inkrement	vrati $v1 pa povećaj $v1 za 1
--$v1	pre-dekrement	smanji $v1 za 1 pa vrati $v1
$v1--	post-dekremen	vrati $v1 pa smanji $v1 za 1

* povećavanje/umanjivanje za 1

```
$v1 = $v1 + 1;    inkrementacija
$v1 = $v1 - 1;    dekrementacija
```

Kod **prefiksnog oblika** pisanja operatora (`++$a`) kada se operator navodi prije varijable, povećavanje vrijednosti varijable za 1 izvršava se **prije** ostatka izraza.

```
$a = 2;
$b = 1 + ++$a;
```

što je ekvivalent sa:

```
$a = 2;
$a = $a + 1;
$b = 1 + $a;
```

Kod **postfiksnog oblika** pisanja operatora (`$a++`) kada se operator navodi poslije varijable, povećavanje vrijednosti varijable za 1 izvršava se **nakon** ostatka što je izračunat izraz te ne utiče na rezultat.

```
$a = 2;
$b = 1 + $a++;
```

što je ekvivalent sa:

```
$a = 2;
$b = 1 + $a;
$a = $a + 1;
```

Složeni operatori pridruživanja (prvo izvrši aritmetičku operaciju a onda rezultat dodijeli varijabli sa lijeve strane!:

Operator	Značenje	Primjer	Ekvivalent
+=	sabiranje	`$a += vrijednost`	`$a = $a + vrijednost`
-=	oduzimanje	`$a -= vrijednost`	`$a = $a - vrijednost`
*=	množenje	`$a *= vrijednost`	`$a = $a * vrijednost`
/=	dijeljenje	`$a /= vrijednost`	`$a = $a / vrijednost`
%=	moduo	`$a %= vrijednost`	`$a = $a % vrijednost`

Konkatenacija

Operator spajanja (konkatenacije) može više nizova znakova spojiti u jedan niz. Kao operator spajanja koristi se znak **.** (tačka).

```
<?php
    $niz1 = "Dobar";
    $niz2 = "dan";
    echo $niz1 . $niz2;
?>
```

Dobardan

Moguće je koristiti i skraćeni način pridruživanja:

```
<?php
    $niz = "Dobar";
    $niz .= "dan";
    echo $niz;
?>
```

Dobardan

Ispis vrijednosti

Koristi se naredba `echo`:

```
<?php
      $broj = 1255;
      echo $broj;
 ?>
```

Moguće je kombinovati ispis teksta i brojeva:

```
<?php
   $broj = 1255;
   echo "Traženi broj je " . $broj . ". Hvala!";
 ?>
```

Traženi broj je 1255. Hvala!

PHP dozvoljava navođenje imena varijable direktno u tekstu pod navodnicima.

```
<?php
   $broj = 1255;
   echo "Traženi broj je $broj. Hvala!";
?>
```

Traženi broj je 1255. Hvala!

Ako se želi ispisati tekst koji počinje znakom $ potrebno je koristiti jednostruke navodnike:

```
<?php
   $broj = 1255;
   echo 'Traženi broj je $broj. Hvala!';
?>
```

Traženi broj je $broj. Hvala!

Moguće je spojiti dvije varijable unutar stringa za ispis:

```
<?php
      $broj1 = 12;
      $broj2 = 55;
      echo "Spojeni brojevi:
{$broj1}{$broj2}";
?>
```

Spojeni brojevi: 1255

Specijalni karakteri

Ispisuju se unutar dvostrukih navodnika:

niz znakova	značenje
`\n`	novi red
`\r`	carriage return (za prelazak u novi red)
`\t`	tabulator
`\$`	ispisuje znak `$`
`\"`	ispisuje znak "
`\\`	ispisuje znak `\`

Operatori poređenja, logički operatori i kontrola toka

Operatori poređenja

Operatori poređenja se koriste za poređenje vrijednosti. Rezultat poređenja je logička vrijednost `TRUE` ili `FALSE` i zavisi od istinitosti poređenja.

Voditi računa da postoji razlika između operatora dodjeljivanja **=**, operatora jednakosti **==** i operatora identičnosti **===**.

operator	značenje	primjer	rezultat
==	jednako	`$a == $b`	*TRUE* ako je *$a* jednako *$b*
===	identično	`$a === $b`	*TRUE* ako je *$a* jednako *$b* **i** ako su *$a* i *$b* istog tipa podataka
!=	nije jednako	`$a != $b`	*TRUE* ako *$a* nije jednako *$b*
<>	nije jednako	`$a <> $b`	*TRUE* ako *$a* nije jednako *$b*
!==	nije identično	`$a !== $b`	*TRUE* ako *$a* nije jednako *$b* **ili** ako nisu istog tipa podataka
<	manje od	`$a < $b`	*TRUE* ako je *$a* manje od *$b*
>	veće od	`$a > $b`	*TRUE* ako je *$a* veće od *$b*
<=	manje ili jednako od	`$a <= $b`	*TRUE* ako je *$a* manje ili jednako *$b*
>=	veće ili jednako od	`$a >= $b`	*TRUE* ako je *$a* veće ili jednako *$b*

PHP dozvoljava poređenje različitih tipova podataka pri čemu važi:

- ako se porede brojčana i tekstualna vrijednost, prvo će se tekstualna vrijednost pretvoriti u brojčanu pa će se porediti dvije brojčane vrijednosti
- ako je pretvaranje u brojčanu vrijednost nije moguće ili varijable nemaju istu vrijednost, rezultat poređenja je `FALSE`
- operator identičnosti `===` provjerava da li dvije varijable imaju istu vrijednost i isti tip podataka

Primjeri upotrebe operatora poređenja

```php
<?php
   $a = 1;
   $b = 2;
   $a_tekst = "1";
   echo "(".($a == $b).")";        // FALSE (ispisuje "")
   echo "(".($a < $b).")";         // TRUE (ispisuje "1")
   echo "(".($a == $a_tekst).")"; // TRUE (ispisuje "1")
   echo "(".($a === $a_tekst).")";// FALSE (ispisuje "")
   echo "(".($a_tekst < $b).")";  // TRUE (ispisuje "1")
?>
```

Naredba `echo` za svaku vrijednost `TRUE` ispisuje "1", a za svaku vrijednost `FALSE` ispisuje prazan niz, tj. "".
Razlog je to što naredba `echo` pretvara dobijene argumente u tekstualni tip podataka, a logički tip podatka pretvara se u tekstualni (string) tip po navedenom principu.

Logički operatori

Uz aritmetičke operacije i operacije poređenja, u programskim jezicima postoje i logičke operacije.

Operandi i rezultati logičkih operacija su logičke vrijednosti (*TRUE* ili *FALSE*). Za pisanje logičkih izraza koriste se ovi logički operatori:

operator	značenje	prioritet	primjer	rezultat
`!`	negacija	1	`!$a`	`TRUE` ako je `!$a` `FALSE`
`&&`	logičko AND	2	`$a && $b`	`TRUE` ako su obe vrijednosti `TRUE`
`and`	logičko AND	2	`$a and $b`	`TRUE` ako su obe vrijednosti `TRUE`
`||`	logičko OR	4	`$a || $b`	`TRUE` ako je bar jedna vrijednost `TRUE`
`or`	logičko OR	4	`$a or $b`	`TRUE` ako je bar jedna vrijednost `TRUE`
`xor`	logičko XOR	5	`$a xor $b`	`TRUE` ako je samo jedna vrijednost `TRUE`

Primjeri upotrebe logičkih operatora

```
<?php
  $a = TRUE;
  $b = FALSE;
  echo !$a;                    // FALSE (ispisuje "")
  echo $a and $b;              // FALSE (ispisuje "")
  echo $a or $b;               // TRUE  (ispisuje "1")
  echo $a xor $b;              // TRUE  (ispisuje "1")
  $c = TRUE;
  echo $a and ($b or !$c);     // FALSE (ispisuje "")
?>
```

Naredba `echo` za svaku vrijednost `TRUE` ispisuje "1", a za svaku vrijednost `FALSE` ispisuje prazan niz, tj. "".
Kod slaganja složenijih logičkih izraza mogu se koristiti zagrade za promjenu redoslijeda izvršavanja.

Kontrola toka programa - `If` struktura

I u PHP jeziku postoje strukture koje se koriste za donošenje odluke o daljnjem toku izvođenja programa. Ako je zadani uslov zadovoljen (tj. Ako je vrijednost uslova *TRUE* ili *FALSE*), obavit će se ili se neće obaviti neki blok naredbi.

- **`If` struktura**

```
if (uslov)
{
    naredba1;
    naredba2;
}
```

Ako je uslov zadovoljen (vrijednost je TRUE), izvršiće se blok naredbi (naredbe 1 i 2).

Ako uslov nije zadovoljen (vrijednost je FALSE), izvršiće se prva naredba van bloka naredbi.

Ako vrijednost ili varijabla unutar zagrada nije logičkog tipa, izvršiće se konverzija u logičku vrijednost.

If – else struktura

- **If – else struktura**

```
if (uslov) {
   naredba1;
}
else {
   naredba2;
}
```

Ako je uslov zadovoljen (vrijednost je TRUE), izvršiće se blok naredbi (naredba 1).

Ako uslov nije zadovoljen (vrijednost je FALSE), izvršiće se blok naredbi (naredba 2).

If – elseif struktura

moguće je dodati proizvoljan broj `elseif` blokova

izvršava se samo onaj blok naredbi uz prvi uslov koji je `TRUE`, a ako nije ni jedan, izvršava se blok naredbi unutar `else`.

If – elseif struktura

```
if (uslov1) {
   naredba1;
}
elseif (uslov2) {
   naredba2;
}
else {
   naredba3;
}
```

Ako je `uslov1` zadovoljen izvršiće se blok naredbi 1 i program se nastavlja u liniji poslije `if` strukture.

Ako `uslov1` nije zadovoljen ispitaće se da li je uslov2 zadovoljen: ako `uslov2` jeste zadovoljen, izvršiće se blok naredbi 2 i program se nastavlja u liniji poslije `if` strukture, a ako `uslov2` nije zadovoljen, izvršiće se blok naredbi 3 i program se nastavlja u liniji poslije `if` strukture.

`Switch` struktura

U situacijama kada je potrebno ispitati vrijednost jedne varijable, umjesto `if-elseif` strukture preporučeno je koristiti `switch` strukturu (brže od `if-elseif` strukture). U zavisnosti od vrijednosti varijable koja se testira, izvršiće se odgovarajući blok naredbi:

```
<?php
  switch (izraz) {
    case vrijednost1:
      naredba1;
      break;
    case vrijednost2:
      naredba2;
      break;
    default:
      naredba3;
  }
?>
```

Veoma je važno navesti naredbu **`break`**.
Ako se ona izostavi, program neće nakon bloka naredbi iskočiti iz `switch` strukture, već će se izvršiti sve preostale naredbe u nastavku strukture.

Primjer `switch` strukture

```
 $a = 1;
switch ($a) {
      case 0:
            echo "<BR>a je jednako 0.";
            break;
      case 1:
            echo "<BR>a je jednako 1.";
            //break;
      case 2:
            echo "<BR>a je jednako 2.";
            break;
      default:
            echo "<BR>a nije jednako 0, 1 ni 2.";
}
```

Primjer switch strukture:
a je jednako 1.
a je jednako 2.

kod ispitivanja druge vrijednosti (`case 1`) izostavljena je naredba break tako da program nije na tom mjestu izašao iz strukture već je nastavio sa izvršavanjem ostalih naredbi

Primjer `switch` strukture

```
switch ($a) {
      case 0:
      case 1:
            echo "<BR>a je jednako 0 ili 1";
            break;
      case 2:
            echo "<BR>a je jednako 2";
            break;
      default:
            echo "<BR>a nije jednako 0, 1 ni 2";
}
```

Moguće je za više ispitanih vrijednosti izvršiti isti blok naredbi.

Ugniježđene `if` strukture

```
<?php
 $pada_kisa = TRUE;
 $trava_raste = FALSE;
 if ($pada_kisa) {
   if ($trava_raste) {
     echo "Pada kiša, trava raste.";
   }
   else {
     echo "Pada kiša, ali trava ne raste!?";
   }
 }
 else  {
   echo "Ne pada kiša.";
 }
?>
```

Kod ispitivanja složenih uslova moguće je unutar jedne `if` strukture ubaciti drugu `if` strukturu.

Ugniježđene `if` strukture

```
<?php
 if ($pada_kisa && $trava_raste) {
   echo "Pada kiša, trava raste.";
 }
 else if ($pada_kisa && !$trava_raste) {
   echo "Pada kiša, ali trava ne raste!?";
 }
 else {
   echo "Ne pada kiša.";
 }
?>
```

Ugniježđene strukture se uvijek mogu napisati kao jedna `if-elseif` struktura pri čemu se može izgubiti preglednost.

Nizovi

Rad sa više varijabli koje se mogu grupisati pod istim nazivom postiže se upotrebom nizova.

Niz je varijabla koja može uskladištiti više vrijednosti odjednom, sadrži niz vrijednosti a svakoj vrijednosti (elementu niza) pridružen je ključ.

Varijable mogu biti bilo koji postojeći PHP tip podataka.

Određenom elementu niza se pristupa navođenjem ključa (može biti broj ili niz znakova).

Nizovi mogu imati jednu, dvije ili više dimenzija, tj. postoje jednodimenzionalni, dvodimenzionalni ili višedimenzionalni nizovi.

Numerički nizovi

Numerički ključ je obično redni broj elementa u nizu i naziva se **indeks**.

Uobičajeno je da brojanje elemenata počinje od nule*, tako da je indeks prvog člana niza 0, drugog 1, trećeg 2 i tako dalje.

primjer niza koji sadrži imena gradova:

Indeks	Vrijednost
0	Banja Luka
1	Trebinje
2	Tuzla

* zero based (indeks se broji od 0)

Kreiranje niza

Za kreiranje se koristi ključna riječ `array`:

```
<?php
      $gradovi = array("Banja Luka", "Trebinje", "Tuzla");
?>
```

Za pristup elementu polja navodi se indeks unutar uglastih zagrada:

```
<?php
      echo $gradovi[1];
?>
```

Ispisaće:

```
Trebinje
```

Dodjeljivanje vrijednosti je moguće i poslije kreiranja niza:

```
<?php
      $gradovi[0] = "Banja Luka";
      $gradovi[1] = "Trebinje";
      $gradovi[2] = "Tuzla";
?>
```

Ako prethodno nije kreiran (upotrebom ključne riječi `array`), na ovaj način će biti kreiran niz.

Kreirani niz nema fiksnu veličinu (moguće je kasnije dodati proizvoljan broj elemenata).

Pristup nepostojećem elementu niza izazvaće grešku.

Asocijativni niz

Osim broja, ključ niza može biti i niz znakova - string.

Ideja je da takav ključ bude naziv koji je smisleno povezan s vrijednošću člana niza.

Niz sa znakovnim ključem se zato naziva **asocijativni niz**. Elementi ovog niza se pohranjuju u obliku para ključ-vrijednost.

Za niz koji treba da sadrži poštanske brojeve gradova, ključ može biti naziv grada:

Ključ	Vrijednost
Banja Luka	78000
Trebinje	89000
Tuzla	75000

SS1

SS2

Pri stvaranju polja sa znakovnim ključem, za svaki član potrebno je navesti i ključ i vrijednost:

```
<?php
   $post_br = array ("Banja Luka" => 78000, "Trebinje" => 89000,
                     "Tuzla" => 75000);
?>
```

Za pristup članu polja koristi se njegov ključ:

```
<?php
   echo $post_br["Trebinje"];
?>
89000
```

Pridruživanje vrijednosti elementu polja obavlja se preko njegovog ključa:

```
<?php
      $post_br[„Mostar"] = 88000;
?>
```

Slide 64

SS1 Saša Salapura; 27. 4. 2022.

SS2 Saša Salapura; 27. 4. 2022.

Dvodimenzionalni nizovi

Jednodimenzionalni niz je niz vrijednosti a dvodimenzionalni niz je tabela, matrica.

Svaki element je određen sa dva ključa, za svaku dimenziju po jedan.

Primjer pokazuje tabelu 3x3, indeksi elemenata za svaku dimenziju su od 0 do 2.

Deklaracija polja iz primjera je moguće sljedećom komandom:

```
<?php
   $igra = array( array ("○", "○", ""),
   array ("○", "X", "○"),
   array ("X", "○", "X") );
?>
```

	0	1	2
0	○	○	
1	○	X	○
2	X	○	X

Definiše se polje čiji su elementi jednodimenzionalna polja!

Za pristup elementima 2D polja potrebno je navesti oba indeksa, pri čemu prvi indeks označava red, a drugi indeks kolonu.

Za ispis elementa (red 1, kolona 0) u koristi se komanda:

```
<?php
    echo $igra[1][0];
?>
```

a za dodavanje nepostojećeg elementa (red 0, kolona 2)

```
<?php
    $igra[0][2] = "X";
?>
```

	0	1	2
0	○	○	X
1	○	X	○
2	X	○	X

Ako se jedan ključ zamijeni slovima A, B i C, deklaracija polja iz primjera je moguće sljedećom komandom:

```
<?php
    $igra = array( "A" => array ("○", "○", ""),
                   "B" => array ("○", "X", "○"),
                   "C" => array ("X", "○", "X") );
?>
```

a dodavanje vrijednosti elementa koji nedostaje na sljedeći način:

```
<?php
     $igra["A"][2] = "X";
?>
```

	0	1	2
A	○	○	X
B	○	X	○
C	X	○	X

Petlje

Petlje sa uslovom

Petlje su strukture kojima je moguće višestruko izvršavanja određenog bloka naredbi.

Blok naredbi se može izvršiti tačno definisan broj puta (poznat broj izvršavanja, tzv. brojačke petlje) ili je broj izvršavanja uslovljen (petlje sa uslovom)

`while` petlja

Ovo je najjednostavnija petlja.

Opšti oblik je:

```
while (uslov)  {
      tijelo petlje;
  }
```

Petlja će se petlja izvršavati sve dok je uslov zadovoljen.

Istinitost uslova se ispituje na početku svake iteracije, a prije tijela petlje:

- Ako je uslov zadovoljen, izvršiće se naredbe u tijelu petlje.
- Ako uslov nije zadovoljen, naredbe u tijelu petlje se neće izvršiti, već se izvršava prva naredba van tijela petlje.
- primjer petlje koja će se izvršiti 10 puta

```
<?php
    $i = 0;
    while($i < 10) {
      echo $i . " ";
      $i++;
    }
?>
```

```
0 1 2 3 4 5 6 7 8 9
```

Beskonačne petlje

Beskonačne petlje su petlje kod kojih izvršavanje petlje neće nikada završiti jer uslov koji je postavljen će uvijek biti zadovoljen.

```
while (TRUE)
{
    echo "Petlja se izvršava";
}
```

Brojačke petlje

`do...while` petlja

Petlja će se petlja izvršavati sve dok je uslov zadovoljen. Za ovu petlju je karakteristično da će biti izvršena **bar jednom**, čak i ako uslov nije zadovoljen.

Istinitost uslova se ispituje na kraju svake iteracije, nakon što su izvršene sve naredbe u tijelu petlje:

- Ako je uslov zadovoljen, ide se na izvršavanje prve naredbe u tijelu petlje.
- Ako uslov nije zadovoljen, izvršava se prva naredba van tijela petlje.

Zato će tijelo petlje biti izvršeno bar jednom.

Opšti oblik je:

```
do {
      tijelo petlje;
} while (uslov)
```

primjer petlje koja će se izvršiti 10 puta

```
<?php
    $i = 0;
    do {
      echo $i . " ";
      $i++;
    } while($i < 10)
?>
```

0 1 2 3 4 5 6 7 8 9

Ako se obrne uslov koji se testira `($i >= 10)` izvršiće se štampa prvog broja: 0

`for` petlja

`foreach` petlja

Ugnježđivanje petlji

Izlaz iz petlje

PHP i MySQL

Komunikacija sa MySQL bazom iz PHP skritpe

Postupak:

- Uspostavljanje konekcije sa MySQL
- Odabir DB
- Izvršavanje SQL upita
- Obrada rezultata (za SELECT)
- Zatvaranje veze

Konekcija prema bazi podataka

- Prema MySQL je moguće otvoriti 100 istovremenih konekcija – iz tog razloga se konekcija zatvara čim su podaci obrađeni
- Primjer komunikacije sa DB:

```
<?php
        $veza = mysql_connect('server', 'korisnik', 'lozinka');
        mysql_select_db('imebaze', $veza);
        /*
                izvršavanje upita i dohvacanje rezultata
        */
        mysql_close($veza);
?>
```

U primjerima se konekcija često navodi kao:

```
$veza = mysql_connect("localhost", "root", "");
```

Parametri konekcije `mysql_connect`

```
mysql_connect(
  string $server =
ini_get("mysql.default_host")
, string $username =
ini_get("mysql.default_user")
, string $password =
ini_get("mysql.default_passwD
")
, bool $new_link = false
, int $client_flags = 0
): resource|false
```

```
mysql_connect(hostname,username,password,databasename);
```

Server - MySQL server; može da sadrži i broj porta, npr. "hostname:port" ili putanju to lokalnog soketa npr. ':/tmp/mysql'

Defaultna vrijednost je 'localhost:3306'. U SQL safe mode, ovaj parametar se ignoriše i uvijek se koristi 'localhost:3306'

username - Korisničko ime; Defaultna vrijednost je opisana sa mysql.default_user. U SQL safe mode, ovaj parametar se ignoriše i uvijek se koristi ime korisnika koji je vlasnik procesa

password - Lozinka; The password. Defaultna vrijednost je opisana sa mysql.default_password. U SQL safe mode, ovaj parametar se ignoriše i koristi se prazna lozinka

new_link - Ako drugi poziv mysql_connect() ima iste argumente, neće se kreirati novi link već će se koristiti identifikator prethodno otvorene konekcije. Parametar new_link čini da mysql_connect() uvijek otvara novi link čak i kada je prethodno otvaran sa istim argumentima. U SQL safe mode, ovaj parametar se ignoriše.

client_flags - Parameter je kombinacija konstanti: 128 (enable LOAD DATA LOCAL handling), MYSQL_CLIENT_SSL, MYSQL_CLIENT_COMPRESS, MYSQL_CLIENT_IGNORE_SPACE or MYSQL_CLIENT_INTERACTIVE.

Return Values – Povratna vrijednost je link identifikator (ako je konekcija OK) ili `False` (za neuspješnu koneciju)

Konekcija prema bazi podataka `MySQLI_Connect()`

`MySQLI_Connect()` funkcija je poboljšana verzija funkcije `MySQL_Connect()` (*i* for improved).

`MySQLI_Connect` je sigurnija i sa više mogućnosti:

- podržava pripremljene komande
- omogućava proceduralna i objektno-orijentisani intefejs
- transakcije se izvode pomoću API-ja
- podržava stored procedure
- Povećana sigurnost i lakše otklanjanje grešaka

Preporuka PHP.NET je da se koristi PDO_MySQL ili MySQLI.
MySQL ekstenzije su zastarile u PHP 5.5.0 i uklonjene iz PHP 7

Ako se umjesto `MySQL` koristi `MySQLI`

1. mysql_connect se mijenja u `mysqli_connect`
2. `mysql_query` se mijenja u `mysqli_query`
3. `mysql_error` se mijenja u `mysqli_error`

…

Izbor baze podataka na serveru `mysql_select_db`

Za izbor željene baze podataka sa DB servera koristi se funkcija `mysql_select_db`

Zadaje se ime baze podataka kao niz znakova i identifikator na uspostavljenu konekciju iz funkcije `mysql_connect`.

U slučaju greške funkcija vraća `false`

Zatvaranje konekcije `mysql_close`

Zatvaranje veze zadaje se funkcijom `mysql_close`

kao argument se navodi identifikator na vezu:

```
mysql_close($veza);
```

Ako konekcija nije uspjela?

Funkcije `mysql_connect` i `mysql_select_db` vraćaju `false` u slučaju pogreške.

Preporuka je da se provjerava da li je spajanje na bazu podataka uspjelo.

Ako nije, treba ispisati poruku i prekinuti s izvođenjem daljnjih naredbi.

U tu svrhu se koristi funkcija `exit` koja ispisuje tekst i prekida izvođenje skripte.

```
<?php
$veza = mysql_connect("localhost", "root", "");
if ($veza)  {
  if (!mysql_select_db("ime_baze", $veza))
  {
       exit("Baza s tim imenom ne postoji!");
  }
}
else {
   exit(„Greška! Ne mogu se spojiti na MySql server!");
}
?>
```

Ako konekcija nije uspjela?

Ako funkcija `mysql_connect` vraća grešku:

```
<?php
   $veza = mysql_connect(’localhost’, ’root’, ’’);
   if (!$veza)  {
       die('Greška, ne mogu se spojiti. Razlog: ' . mysql_error());
   }
   echo ’uspješna konekcija’;
   mysql_close($link);
?>
```

Izvršavanje SQL upita i obrada rezultata

Izvršavanjem upita tipa `SELECT` iz baze podataka se uzimaju svi slogovi (redovi/records) koji zadovoljavaju zadani kriterij SQL upita.

Izvršavanje SQL upita vrši se pozivom funkcije `mysql_query`

```
mysql_query(string $query, resource $link_identifier = NULL
```

Izvršava se jedan upit na ternutno aktivnoj bazi podataka na serveru koja je određena identifikatorom na uspostavljenu konekciju

Ako je upit uspješno izvršen, funkcija vraća identifikator rez,ultata, a u slučaju neuspješnog izvršavanja dobije se vrijednost `FALSE`

Izvršavanje SQL upita i obrada rezultata

Najčešće SELECT upit vraća više od jednu reda rezultata.

Zato je potrebno čitati i obrađivati jedan po jedan red unutar petlje pri čemu se koristi funkcija `mysql_fetch_array`

```
<?php
    $rezultat = mysql_query("SELECT * FROM kontakti");
    while ($redak = mysql_fetch_array($rezultat))
    {
            echo "$redak['Ime'] $redak['Adresa']"
            echo "$redak['Grad'] $redak['Email'] <br/>";
    }
?>
```

Izvršavanje SQL upita i obrada rezultata

Svaki uzastopni poziv funkcije `mysql_fetch_array` vraća sljedeći red iz skupa rezultata

```
while ($redak = mysql_fetch_array($rezultat))
```

i prestaje sa izvršavanjem petlje kada funkcija vrati vrijednost `false`, tj. došla je do kraja.

Vrijednostima kolona se pristupa navođenjem atributa imena kolone, npr. `$redak['Ime']`

Koliko redova je zadovoljilo SQL upit, tj. koliko redova je sadržano u rezultatu se može saznati pozivom funkcije `mysql_num_rows`.

Kao argument se navodi identifikator SQL upita:

```
mysql_num_rows($rezultat)
```

Nakon prestanka obrade rezultata potrebno je zatvoriti konekciju prema bazi.

Izvršavanje SQL upita i obrada rezultata

Dobra praksa je na kraju obrade osloboditi memoriju koja je zauzeta dobijenim odgovorom na SELECT upit.

To se obavlja pozivom funkcije `mysql_free_result` kojoj predajemo identifikator rezultata.

```
<?php
   $rezultat = mysql_query("SELECT * FROM kontakti");
   while ($redak = mysql_fetch_array($rezultat))
   {
      echo "$redak['Ime'] $redak['Adresa']";
      echo "$redak['Grad'] $redak['Email'] <br/>";
   }
   echo "Dohvaćeno " . mysql_num_rows($rezultat) . " redaka";
   mysql_free_result($rezultat);
?>
```

Izvršavanje SQL upita za upis u bazu - INSERT

Upisivanje reda u bazu podataka se vrši pomoću INSERT upita i funkcijom `mysql_query`

```
<?php
   $sql = "INSERT INTO kontakti (Id, Ime, Adresa, Grad,
           Email, pol, Prijatelj) VALUES ( 2, Petar Petrović’,
           'Aleja jorgovana 5', 2, ’ppeter@gmail.com', 'M', 'Da')";
   mysql_query($sql);
?>
```

Izvršavanje SQL upita za izmjenu - UPDATE

Izmjena postojećeg reda u bazi podataka se vrši pomoću UPDATE upita i funkcijom `mysql_query`

```
<?php
    $sql = "UPDATE kontakti SET pol = 'M' WHERE Id = 2");
    mysql_query($sql);
?>
```

Izvršavanje SQL upita za brisanje - DELETE

Brisanje postojećeg reda u bazi podataka se vrši pomoću DELETE upita i funkcijom `mysql_query`

```
<?php
   $sql = „DELETE kontakti WHERE Id = 2");
   mysql_query($sql);
?>
```