SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINARSKI RAD

PHP

Roko Končurat

Zagreb, siječanj 2005.

Sadržaj

1.Što je PHP

Skraćenica PHP dolazi od engl. Hypertext Preprocessor. PHP je skriptni jezik ugrađen u HTML (ugrađeni HTML kod ili *engl. HTML-embadded*), čiji se kod izvršava na strani poslužitelja (*engl. server*-side). Znači da se svi PHP programi izvode na poslužitelju, a klijentu se potom šalje samo rezultat izvođenja, odnosno odgovarajući HTML kod. Na strani klijenta se vidi samo HTML kod, dok je sam programski kod skriven od klijenta. Kako izgleda ugrađeni HTML kod pokazuje slijedeći primjer:

```
<html>
<head>
<title>Primjer ugradenog koda</title>
</head>
<body>

<?php
echo "Ja sam PHP skripta i izvršavam se na serveru !";
?>

</body>
</html>
```

Za razliku od Perl ili C++ programa koji se sastoje od puno dugačkih komandi za generiranje HTML-a, piše se obični HTML uz unutar HTML-a dodani PHP kod za obavljanje nekog određenog posla (u gornjem primjeru, ispisivanje teksta). Danas je PHP jedan od najpopularnijih skriptnih jezika za pisanje web aplikacija. Mogućnosti su velike, a uz to je PHP jako pogodan za početnike s obzirom na lakoću programiranja u njemu.

2.Osnove sintakse

2.1.Umetanje PHP koda u HTML

Postoje četiri načina umetanja PHP koda u HTML. Sva četiri primjera umetanja prikazana su, za primjer, slijedećim HTML kodom:

```
<html>
<head>
<title>Primjer ugradenog koda</title>
</head>
<body>

// 1. način

<?php echo ("Ovo je najjednostavniji nacin\n"); ?>

// 2. način

<?php echo("Ako zelite posluzivati XHTML ili HML dokumente, onda ovako...\n");
?>

// 3. način

<script language="php">
echo ("neki editori ne vole bas procesirati instrukcije (npr. Frontpage)");
</script>

// 4. način

<% echo ("Ako je omoguceno koristenje ASP tagova, onda mozete i ovako");
%>



</body>
</html>
```

Najčesće koristen je 2. način jer omogućuje lagano implementiranje novih generacija XHTML-a PHP kodom.

2.2.Separatori

Instrukcije se u PHP-u odvajaju točka-zarezom ";" kao u C-u ili perlu. Prije završnog taga (?>), točka-zarez(;) nije potreban pa su sljedeća dva primjera ekvivalentna:

```
<html>
<head>
<title>Primjer ugradenog koda</title>
</head>
<body>

<?php echo "test"; ?>

<?php echo "test" ?>


</body>
</html>
```

2.3.Komentari

PHP podržava 3 razlicita tipa komentiranja koda: C, C++ i Unix shell -- style komentari. Primjeri komentiranja u PHP-u dani su slijedećim ugrađenim HTML kodovima:

```
<html>
<head>
<title>Primjer ugradenog koda</title>
</head>
<body>

<?php

echo "This is a test"; // Ovo je C++ like komentar do kraja linije

/* Ovo je komentar koji se
proteže kroz više redaka */

echo "This is yet another test";

echo "One Final Test"; # Ovo je Unix shell style komentar

?>
```

```
</body>
</html>
```

Jednolinijski komentari idu do karja linije ili do kraja trenutnog bloka PHP koda, ovisno o tome što prije dođe.

```
<html>
<head>
<title>Primjer ugradenog koda</title>
</head>
<body>

<h1>This is an <?php # echo "simple";?> example.</h1>
<p>The header above will say 'This is an example'.


</body>
</html>
```

Treba paziti da ne gnijezdimo C-like komentare, što se lako može dogoditi prilikom komentiranja velikih blokova koda.

```
<html>
<head>
<title>Primjer ugradenog koda</title>
</head>
<body>

<?php
/*
echo "This is a test"; /* This comment will cause a problem */
*/
?>


</body>
</html>
```

3.Podržani tipovi

PHP podržava sljedeće tipove podataka:

- array (polje)
- floating-point numbers (brojevi s pomičnim zarezom)

- integer (cijeli brojevi)
- string (znakovni niz)
- object (objekt, iz ovog tipa se izvode svi tipovi)

Tip varijable najčešće ne postavlja programer, nego ga PHP implicitno postavlja ovisno o kontekstu u kojem se varijabla koristi. Upravo zbog te slobode programera dosta često dolazi do grešaka u definiranju tipova. Da se to ne bi dogodilo, možemo „*castati*“ varijablu uz pomoć settype() funkcije.

3.1.Integer tip podataka

Integer tip podataka se može definirati na bilo koji od sljedecih načina:

```
<html>
<head>
<title>Primjer ugradenog koda</title>
</head>
<body>

<?php
$a = 1234; # decimal number
$a = -123; # a negative number
$a = 0123; # octal number (equivalent to 83 decimal)
$a = 0x12; # hexadecimal number (equivalent to 18 decimal)
?>

</body>
</html>
```

Veličina integera ovisi o platformi na kojoj se PHP izvodi.

3.2.Brojevi s pomičnim zarezom

Brojevi s pomičnim zarezom mogu se definirati na sljedeće načine:

```
<html>
<head>
<title>Primjer ugradenog koda</title>
</head>
<body>

<?php

$a = 1.234;
$a = 1.2e3;
```

```
?>

</body>
</html>
```

Veličina brojeva s pomičnim zarezom ovisi o platformi, a najčešća maksimalna vrijednost je ~1.8e308 s preciznošću na 14 decimala (IEEE 64 bitni format).

3.3.Znakovni niz (string)

Stringovi se mogu specificirati koristeći jedan od dva seta delimitera. Ako se string nalazi unutar dvostrukih navodnika ("), varijable unutar stringa će se koristiti. Kao i u C-u ili Perlu, za specificiranje escape znakova koristi se backslash(\). Escape znakovi: \n, \r, \t, \\, \$, \", ... Drugi način odvajanja stringova su jednostruki navodnici ('). Kad je string unutar jednostrukih navodnika, jedini escape znakovi koje će PHP razumjeti su "\\" i "\'". Varijable navedene unutar jednostrukih navodnika se neće koristiti kao varijable, vec kao najobičniji niz znakova. Primjer sa stringom:

```
<html>
<head>
<title>Primjer ugradenog koda</title>
</head>
<body>

<?php

/* Assigning a string. */
$str = "This is a string";

/* Appending to it. */
$str = $str . " with some more text";

/* Another way to append, includes an escaped newline. */
$str .= " and a newline at the end.\n";

/* This string will end up being '<p>Number: 9</p>' */
$num = 9;
$str = "<p>Number: $num</p>";

/* This one will be '<p>Number: $num</p>' */
$num = 9;
$str = '<p>Number: $num</p>';
```

```
/* Get the first character of a string */
$str = 'This is a test.';
$first = $str[0];

/* Get the last character of a string. */
$str = 'This is still a test.';
$last = $str[strlen($str)-1];

?>

</body>
</html>
```

Znakovi unutar znakovnog niza se mogu dohvaćati kao da se radi o brojčano indeksiranom polju znakova, koristeći sintaksu sličnu C-u. Gore su navedena dva primjera gdje se koristi takav dohvat. Također, stringovi se mogu nadovezivati koristeći operator konkatenacije "." Primjer:

```
$str = 'Evo samo da se' . ' nadovezem';
```

3.4.Polja (arrays)

3.4.1.Jednodimenzionalna polja

PHP podržava i skalarna i asocijativna polja. Ustvari, uopce nema razlike izmedu ta dva tipa polja. Primjer:

```
<html>
<head>
<title>Primjer ugradenog koda</title>
</head>
<body>

<?php

$a[0] = "abc";
$a[1] = "def";
$b["foo"] = 13;

?>

</body>
</html>
```

Također, polje se može implicitno kreirati, jednostavno dodajući nove vrijednosti u polje. Kada se doda vrijednost u polje koristeći prazne uglate zagrade, ta vrijednost će se dodati na kraj polja. Primjer:

```
<html>
<head>
<title>Primjer ugradenog koda</title>
</head>
<body>

<?php

$a[] = "hello"; // $a[2] == "hello"
$a[] = "world"; // $a[3] == "world"

?>

</body>
</html>
```

Sortiranje polja se može napraviti na razlicite načine koristeći ugrađene funkcije: asort(), arsort(), ksort(), rsort(), sort(), uasort(), usort(), i uksort() ovisno o tome koji tip sortiranja nam je potreban.

3.4.2.Višedimenzionalna polja

Višedimenzionalna polja su također prilično jednostavna. Primjer sve govori:

```
<html>
<head>
<title>Primjer ugradenog koda</title>
</head>
<body>

<?php

$a[1] = $f; # one dimensional examples
$a["foo"] = $f;

$a[1][0] = $f; # two dimensional
$a["foo"][2] = $f; # (you can mix numeric and associative indices)
$a[3]["bar"] = $f; # (you can mix numeric and associative indices)

$a["foo"][4]["bar"][0] = $f; # four dimensional!
```

```
?>

</body>
</html>
```

3.5.Objekti

Za instanciranje objekta može se koristiti "new" da bi se instancirao objekt koji se kasnije pridružuje varijabli. Primjer:

```
<html>
<head>
<title>Primjer ugradenog koda</title>
</head>
<body>

<?php

class foo {
 function do_foo() {
   echo "Doing foo.";
 }
}

$bar = new foo;
$bar->do_foo();

?>

</body>
</html>
```

U gornjem primjeru prvo definiramo klasu foo i unutar klase funkciju do_foo() koja ispisuje tekst. Naredbom $bar=new foo; instanciramo objekt izveden iz klase foo i pridružujemo ga varijabli $bar. Sada preko varijable $bar možemo pokrenuti funkciju do_foo(). Ova problematika je pobliže objašnjena u poglavlju o klasama i objektima.

4.Varijable

4.1.Osnove

Varijable u PHP su označene dolar-znakom ($) iza čega slijedi ime varijable. Ime varijable je „*case-sensitive*“. Za imena varijabli vrijede ista pravila kao i za ostale labele u PHP. Ispravno ime varijable počinje s podvlakom(_) ili slovom, a iza toga se mogu nalaziti brojevi, slova i podvlake. Primjer:

```
<html>
<head>
<title>Primjer ugradenog koda</title>
</head>
<body>

<?php

class foo {
$var = "Bob";
$Var = "Joe";
echo "$var, $Var"; // outputs "Bob, Joe"

$4site = 'not yet'; // invalid; starts with a number
$_4site = 'not yet'; // valid; starts with an underscore
$täyte = 'mansikka'; // valid; 'ä' is ASCII 228.

?>

</body>
</html>
```

U PHP 3, varijablama su **uvijek** pridružene vrijednosti. To znači, npr. ako napišemo $var1=$var2 i poslije promijenimo vrijednost varijabli $var2, to neće utjecati na $var1. U PHP 4 je uveden „*assign-by-reference*“, što znači da neka nova varijabla jednostavno pokazuje na original varijablu. To se izvodi tako da se doda znak & ispred varijable na koju želimo da nova varijabla pokazuje. Primjer:

```
<html>
<head>
```

```
<title>Primjer ugradenog koda</title>
</head>
<body>

<?php

<?php
$foo = 'Bob'; // Pridjelimo vrijednost 'Bob' varijabli $foo
$bar = &$foo; // varijabli $bar pridruzujemo referencu na &foo
$bar = "My name is $bar"; // Varijabli $bar mijenjamo vrijednost usput koristeci
//staru vrijednost
echo $foo; // $foo se takoder promijenio, jer preko $bar mijenjamo i $foo
echo $bar;
?>

</body>
</html>
```

Ispis:

```
My name is Bob.
My name is Bob.
```

4.2.Predefinirane varijable

PHP pruža velik broj predefiniranih varijabli koje može koristiti svaka pokrenuta skripta. Ipak, mogućnost korištenja velikog broja od tih varijabli ovisi o tome koji je server pokrenut, koja verzija, te o podešavanju servera i drugim faktorima. Za listu svih predefiniranih varijabli koristi se funkcija phpinfo(). *Environment* varijable su unesene u PHP-ov globalni *namespace* iz okruženja u kojem je pokrenut PHP parser. Pošto okruženja ima puno, broj različitih *environment* varijabli je ogroman, tako da je teško naći definitivnu listu svih *environment* varijabli. Za popis odgovarajućih *environment* varijabli najbolje je pogledati dokumentaciju odgovarajuće ljuske u kojoj je pokrenut PHP parser.

4.3.Doseg varijabli

Doseg varijable u PHP je kontekst u kojem je varijabla definirana. Najcešće sve PHP varijable imaju jedan doseg koji uključuje i includane file-ove i cijelu .php skriptu. Primjer:

```
$a = 1;
include "b.inc";
```

U gornjem primjeru, varijabla $a se može koristiti i unutar b.inc skripte.

Kod korisnički definiranih funkcija situacija je malo drugačija. Doseg bilo koje varijable definirane unutar funkcije je ograničen samo na tu funkciju. Također, varijable izvan funkcije ne vrijede unutar funkcije. Primjer:

```
<html>
<head>
<title>Primjer ugradenog koda</title>
</head>
<body>

<?php

$a = 1; /* globalni doseg*/

Function Test () {
echo $a; /* ovdje se referenciramo na lokalnu varijablu $a koja ne postoji */
}

Test ();

?>

</body>
</html>
```

Gore navedena skripta neće ispisati ništa, jer varijabla $a je različita od globalne varijable $a. Naravno da i za to ima rješenje. Ako želimo koristiti globalne varijable unutar funkcije, moramo navesti ključnu rijec global i potom globalne varijable koje želimo koristiti. Primjer:

```
<html>
<head>
<title>Primjer ugradenog koda</title>
</head>
<body>

<?php

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

```
Function Sum () {
global $a, $b;

$b = $a + $b;
}

Sum ();
echo $b;

?>

</body>
</html>
```

Očekivano, ispis gornje skripte će biti 3.

4.4.Varijable izvan PHP-a

HTML forme (GET i POST varijable)

Kada se forma pošalje PHP skripti, sve varijable iz te forme će biti dostupne u PHP skripti kojoj je šaljemo. Ako je track_vars postavljen na TRUE, onda će varijable iz forme biti dostupne u asocijativnim poljima $HTTP_POST_VARS, $HTTP_GET_VARS i $HTTP_POST_FILES ovisno o kakvim je varijablama riječ. Primjer jednostavne forme:

```
<html>
<head>
<title>Primjer ugradenog koda</title>
</head>
<body>

<form action="foo.php" method="post">
    Name: <input type="text" name="username"><br>
    <input type="submit">
</form>

</body>
</html>
```

Kada se gornja forma ispuni i pošalje, vrijednost iz textbox-a će biti dostupna u polju HTTP_POST_VARS. Dakle, username u PHP skripti ćemo dohvaćati kao HTTP_POST_VARS['username']. Ako u konfiguraciju postavi register_globals na TRUE, onda ćemo username moći dohvaćati i kao

$username. GET varijable se nalaze u QUERY-STRINGU. To je onaj znakovni niz koji se pojavljuje u URL-u stranice iza upitnika...

Npr.: http://www.php.net/cal.php?id=1436

U gornjem URL-u prenesena je GET varijabla $id čija je vrijednost 1436. Moramo biti jako pažljivi pri prenošenju varijabli preko query-stringa ako se radi o povjerljivoj informaciji, a također i paziti da korisnik mijenjanjem query stringa ne bi mogao doći do neke sigurnosne rupe.

4.5.Konstante

U PHP-u se konstante definiraju uz pomoć funkcije define(). Slijedi primjer:

```
<html>
<head>
<title>Primjer ugradenog koda</title>
</head>
<body>

<?php

define("CONSTANT", "Hello world.");
echo CONSTANT; // outputs "Hello world."

?>

</body>
</html>
```

Također, postoje unaprijed definirane konstante.

5.Klase i objekti

Klasa je skup varijabli i funkcija koje rade s tim varijablama. Klasa se definira koristeći sljedeću sintaksu:

```
<html>
<head>
<title>Primjer ugradjenog koda</title>
</head>
<body>
```

```
<?php
class Kosarica {
var $artikli; // artikli u kosarici

// Dodaj $broj artikala $sifArtikla u kosaricu

function dodaj_artikl ($sifArtikla, $broj) {
   $this->artikli[$sifArtikla] += $broj;
}

// Izvadi$broj artikala $sifArtikla iz kosarice

function brisi_artikl ($sifArtikla, $broj) {
   if ($this->artikli[$sifArtikla] > $broj) {
      $this->artikli[$sifArtikla] -= $broj;
      return true;
   } else {
   return false;
   }
}


}
?>

</body>
</html>
```

Varijable unutar istog objekta se dohvaćaju sa $this->imeVarijable Gornji primjer definira klasu Kosarica koja se sastoji od polja artikala u košarici i metoda za dodavanje i brisanje artikala iz košarice. Klase su tipovi, odnosno, one su predložak za stvarne varijable. Da bi kreirali varijablu tipa neke klase, potrebno je koristiti operator new. Primjer:

```
$moja_kosara = new Kosarica;
$moja_kosara->dodaj_artikl("200342",5)
```

Gornjim primjerom smo kreirali objekt moja_kosara klase Kosarica, a potom smo u objekt moja_kosara dodali 5 artikala sifre 200342. Klase također mogu biti proširena verzija osnovne klase. To znači, da nova klasa ima sve varijable i funkcije kao i osnovna klasa, ali ima i dodatne funkcije i varijable

koje osnovna klasa nema. Ovo se radi uz pomoć ključne riječi extends. Primjer:

```
<html>
<head>
<title>Primjer ugradenog koda</title>
</head>
<body>

<?php
class Kosarica_s_imenom extends Kosarica {

var $vlasnik;

function postavi_vlasnika($ime) {


   $this->vlasnik=$ime;

}


}
?>

</body>
</html>
```

Dakle, gornja klasa ima sve metode i varijable klase Kosarica, ali uvodi i novu vlastitu varijablu $vlasnik i novu vlastitu metodu postavi_vlasnika. Primjer:

```
$imenovana_kosarica = new Kosarica_s_imenom;
$imenovana_kosarica->postavi_vlasnika("Djoni Bravo");
echo $imenovana_kosarica->vlasnik;
$imenovana_kosarica->dodajArtikl("200320",23);
```

Ne postoji mogućnost višestrukog nasljeđivanja u PHP-u, a to znači da proširena klasa može naslijediti svojstva i metode samo jedne osnovne klase.

6.Kontrolne strukture

6.1.IF

IF kontrolna struktura je jedna od najvažnijih u mnogim programskim jezicima pa tako i u PHP-u. Sintaksa je vrlo slična C-u.

```
<html>
<head>
<title>Primjer ugradjenog koda</title>
</head>
<body>

<?php


if ($a > $b) {
print "a je veci nego b";
$b = $a;
}


?>

</body>
</html>
```

Ako imamo samo jednu naredbu za izvršiti u IF bloku, možemo izostaviti vitičaste zagrade, iako zbog lakše čitljivosti programa to nije preporučljivo.

6.2.ELSE

Često bi htjeli izvršiti neku naredbu ili skup naredbi ako je uvjet ispunjen, a drugi skup naredbi ako uvjet nije ispunjen. Upravo zbog toga je ELSE kontrolna struktura.

```
<html>
<head>
<title>Primjer ugradjenog koda</title>
</head>
<body>

<?php
```

```
if ($a > $b) {
   print "a je veci b";
} else {
   print "a NIJE veci od b";
}

?>

</body>
</html>
```

6.3.ELSEIF

Elseif, kao sto i ime kaže je kombinacija if i else kontrolnih struktura. Kao i else, elseif proširuje if strukturu tako da se može provjeriti više uvjeta, ako prvi nije ispunjen, elseif za razliku od else će se izvršiti samo ako se njegov uvjet evaluira kao istinit.

```
<html>
<head>
<title>Primjer ugradjenog koda</title>
</head>
<body>

<?php

if ($a > $b) {
    print "a je veci od b";
} elseif ($a == $b) {
    print "a je jednak b";
} else {
    print "a je manji od b";
}

?>

</body>
</html>
```

6.4.WHILE

While petlje su najjednostavnije petlje u PHP-u. Tijelo petlje se izvršava dok god je uvjet while-a ispunjen.

```
<html>
<head>
<title>Primjer ugradjenog koda</title>
</head>
<body>

<?php

$i = 1;
while ($i <= 10) {
   print $i++; //uvecanje se izvodi tek nakon ispisa
}

?>

</body>
</html>
```

6.5.DO…WHILE

Do...while petlje su vrlo slične while petljama, jedina je razlika sto će se do..while petlja sigurno izvesti jedan puta, neovisno o tome da li je uvjet ispunjen ili nije.

```
<html>
<head>
<title>Primjer ugradjenog koda</title>
</head>
<body>

<?php

$i = 0;
do {
   print $i;
} while ($i>0);

?>

</body>
</html>
```

U gornjem primjeru će se petlja izvršiti bar jednom, odnosno ispisati će "0" iako uvjet u zagradama nije ispunjen.

6.6.FOR

For petlje su najsloženije petlje u PHP-u. Sintaksa je sljedeća:

```
for (expr1; expr2; expr3) {
statement1;
.
.
statementN;
}
```

expr1 se bezuvjetno evaluira jedan put na početku petlje. Na početku svake iteracije expr2 se evaluira. Ako se evaluira u TRUE, petlja se nastavlja i naredbe unutar for bloka se izvršavaju. Ako se evaluira u FALSE, petlja se prekida. Na kraju svake iteracije se evaluira expr3.

6.7.FOREACH

PHP 4 uključuje foreach strukturu. Ovo nam daje mogućnost jednostavnog iteriranja po svim stavkama polja. Primjer:

```
foreach(array_expression as $value) statement
```

Petlja se vrti kroz polje array_expression i u svakoj iteraciji vrijednost elementa polja stavlja u $value varijablu. Primjer:

```
foreach ($arr as $value) {
echo "Value: $value<br>\n";
}
```

6.8.BREAK

Break prekida izvođenje trenutne for, while ili switch structure.

6.9.CONTINUE

Continue se koristi u petljama da se preskoči ostatak trenutne iteracije petlje i da se izvođenje nastavi na početku nove iteracije petlje.

6.10.SWITCH

Switch je sličan seriji if-ova na istom izrazu. U puno slučajeva, želimo varijablu usporediti s puno vrijednosti i ovisno o odgovarajućoj vrijednosti, poduzeti određene akcije. Upravo tu nam je switch najpraktičniji.

```
<html>
<head>
<title>Primjer ugradjenog koda</title>
</head>
<body>

<?php

switch ($i) {
    case 0:
        print "i equals 0";
        break;
    case 1:
        print "i equals 1";
        break;
    case 2:
        print "i equals 2";
        break;
}

?>

</body>
</html>
```

Vrlo je bitno staviti break na kraju svakog case bloka, jer ako ne stavimo break, izvode se svi blokovi koji slijede. Npr.da je $i=0 i da nema breakova ispis bi bio sljedeći:

```
i equals 0
i equals 1
i equals 2
```

Poseban slučaj case-a je default. On hvata sve što ne odgovara prethodnim uvjetima i mora biti zadnji od svih case izraza.

```
<html>
<head>
<title>Primjer ugradjenog koda</title>
</head>
<body>

<?php

switch ($i) {
    case 0:
        print "i equals 0";
        break;
    case 1:
        print "i equals 1";
        break;
    case 2:
        print "i equals 2";
        break;
    default:
        print "i is not equal to 0, 1 or 2";
}

?>

</body>
</html>
```

7.Rad s bazama podataka (kratke upute)

7.1.Uvod

Pri izradi web aplikacija, najčešće je potrebno povezati aplikaciju s nekom bazom podataka. PHP podržava više vrsta baza, no ja ću samo kratko prikazati kako najjednostavnije povezati aplikaciju s mySQL bazom podataka.

7.2.MySQL baze podataka

Dolje navedene funkcije omogućavaju pristup MySQL serverima. Da bi ove funkcije bile dostupne, potrebno je kompajlirati php sa mysql podrškom koristeći --with-mysql.

7.2.1.mysql_connect

Otvara vezu prema mySQL serveru.

```
int mysql_connect ([string hostname [:port] [:/path/to/socket] [, string username [, string password]]])
```

Defaultni parametri koji se koriste su: host:port = 'localhost:3306', username = ime korisnika koji je vlasnik serverskog procesa i password =””. Primjer:

```
<html>
<head>
<title>Primjer ugradjenog koda</title>
</head>
<body>

<?php

$link = mysql_connect ("localhost", "username", "secret")
or die ("Could not connect");
print ("Connected successfully");
mysql_close ($link);

?>

</body>
</html>
```

Vrlo je bitno da se svaki otvoreni connection zatvori sa funkcijom mysql_close. Ako to ne navedemo, veza će se zatvoriti tek nakon izvršenog cijelog PHP programa.

7.2.2.mysql_select_db

Nakon što smo otvorili vezu s mySQL serverom, potrebno je odabrati bazu podataka s kojom ćemo raditi na serveru.

int mysql_select_db (string database_name [, int link_identifier])

Vraća cijeli broj veći od nule, ako je sve ok, inače vraća nulu. Mysql_select_db() postavlja trenutno aktivnu bazu na trenutno otvorenom linku na server. Ako nije naveden link u pozivu funkcije, koristi se posljednji otvoreni.

7.2.3.mysql_query

Sad kad smo se povezali na server i odabrali bazu s kojom ćemo raditi, potrebno je izvršiti prvi upit nad bazom. Za postavljanje upita treba detaljnije poznavanje mySQL. Ovdje nije bitan mySQL, pa se preko toga može proći.

int mysql_query (string query [, int link_identifier])

Funkcija vraća nulu ako je došlo do greške, a ako je sve prošlo ok, onda vraća cijeli broj različit od nule. Evo primjera:

```
<html>
<head>
<title>Primjer ugradjenog koda</title>
</head>
<body>

<?php

$result = mysql_query ("SELECT my_col FROM my_tbl")
or die ("Invalid query");

?>

</body>
</html>
```

U varijabli $result se nalaze svi rezultati našeg upita. Najčešće se $result varijabla obrađuje foreach petljom kako bi obavili potrebne radnje nad svim rezultatima. Nakon završenog rada s $result-om, potrebno je osloboditi resurs:

mysql_free_result($result);

Postoje još mnoge funkcije za rad s mySQL-om, tu su prikazane samo osnovne. Slijedi još jedan konkretan primjer koji koristi sve navedene funkcije i ispisuje podatke.

```
<html>
<head>
<title>Primjer ugradjenog koda</title>
</head>
<body>

<?php

$veza = mysql_connect('localhost','mladen','blablabla');
if (!$veza) die("Gre1ka: Ne mogu se spojiti na server !");

$ok = mysql_select_db('moja_baza');
if(!$ok) die("<BR> Gre1ka: Ne mogu otvoriti bazu ! <BR>");
if (mysql_errno()) die("<BR>".mysql_errno().": ".mysql_error()."<BR>");

$upit="SELECT * FROM tablica1 WHERE 1=1";

$rezultat_upita = mysql_query($upit);
if (mysql_errno()) die("<BR>".mysql_errno().": ".mysql_error()."<BR>");

while($redak=mysql_fetch_array($rezultat_upita)) {


   echo $redak['PrviPodatak'],' '.$redak['DrugiPodatak'];

   echo <BR>;

}

mysql_free_result($rezultat_upita);
mysql_close($veza);

?>

</body>
</html>
```

Dakle, prvo otvaramo vezu na server, potom odabiremo bazu na serveru, te izvršavamo upit i rezultati se nalaze u $rezultat_upita. Potom idemo redak po redak kroz $rezultat_upita i ispisujemo redak po redak iz tablice1 u bazi moja_baza. Na kraju je potrebno osloboditi resurse koje smo zauzeli pri dohvatu iz baze.

8.Zaključak

PHP programski jezik je jedan od najboljih izbora ako trebate izradu web aplikacije ili nekog web-orijentiranog sustava. Uči se vrlo brzo, jednostavan je za primjenu, jedini nedostatak je taj što ne postoje neki ozbiljniji alati za izradu php stranica, barem ja ne znam za taj alat. Inače, sve se svodi na notepad, a konkurencija (Microsoft ASP.NET) ima puno bolju razvojnu okolinu. Naravno, još jedna od velikih prednosti PHP-a pred konkurencijom je ta što je besplatan, a u kombinaciji s također besplatnim mySQL paketom može poslužiti za izradu vrlo moćnih web aplikacija.

9.Literatura

1. http://www.w3schools.com/php/default.asp

2. Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva (FER), Zavod za automatiku i procesno računarstvo (ZAPR), kolegij Otvoreno računarstvo, skripta za laboratorijske vježbe iz php-a.