FAKULTET STROJARSTVA I BRODOGRADNJE
ZAGREB

OSNOVE PROGRAMSKOG JEZIKA PHP

- semestralni rad -

Prof. Mario Essert

Damir Robeli
35951063

Zagreb, 26. 10. 2002

Sadržaj

1. Uvod

Kao pripremu za diplomski rad, kojeg sam odlučio napraviti u programskom jeziku PHP, dobio sam zadatak sastaviti kratak tutorijal o PHP-u. Cilj ovog rada je stjecanje potrebnog predznanja koje će mi pomoći pri izradi diplomskog rada.

Zašto odabrati baš PHP? PHP je jedna od najnaprednijih i najkorištenijih server-side skriptnih tehnologija danas u upotrebi. On je po svojoj sintaksi poput mnogih drugih sličnih jezika, čak i koristi funkcije nekih drugih jezika kao što su C ili Perl. To znači da jednu radnju možete izvesti korištenjem više različitih funkcija. Npr. ova dva primjera rezultiraju istim prikazom:

Primjer:

```
<?
echo 'Pozdrav svima';
?>

<?
Print ( 'Pozdrav svima');
?>
```

Još jedna važna stvar, PHP je bogat funkcijama za manipuliranje mnogo različitih tipova sadržaja. Npr. možemo kreirati slike ili flasheve u trenutku kada ih korisnik zatraži na vašoj web stranici ali ostavimo to za sad naprednim korisnicima.

2. Što je PHP?

PHP je open-source server-side skriptni jezik za dinamičko generiranje HTML koda, zapravo to je skraćenica od "*Hypertext Preprocesor*". Drugim riječima, PHP je skriptni jezik pomoću kojeg možete kreirati HTML stranicu na serveru prije nego što se ona, popunjena dinamičkim sadržajem, pošalje klijentu. Ovim načinom generiranja sadržaja klijent ne može vidjeti kod (skriptu) koji je generirao sadržaj koji gleda, već ima pristup čistom HTML kodu.

Vrlo jednostavno, PHP je jedan od najpopularnijih i najmoćnijih skriptnih jezika trenutno na tržištu. Broj siteova koji koriste PHP raste iz dana u dan, a broj tvrtki koje žele primijeniti PHP na svojim siteovima je još veći. PHP je izvrstan jer pomoću njega s nevjerojatnom lakoćom možete stvoriti opširnu web aplikaciju s velikim količinama podataka.

Zamislite samo koliko biste se namučili u HTML-u kada biste na webu trebali prezentirati paletu proizvoda neke firme. Uzmimo da dotična firma ima u svom asortimanu oko 1000 proizvoda. To znači da biste morali napraviti 1000 stranica (za svaki proizvod posebnu) i paziti da svaka stranica izgleda isto (da su tablice poravnane, da su naslovi iste veličine i boje…). Ovo bi bio vrlo mukotrpan i stresan posao. Ako biste željeli omogućiti svojim posjetiteljima vrlo jednostavno pretraživanje asortimana proizvoda, ovaj zadatak bi postao praktički neizvediv! Isti ovakav site možete napraviti u duplo kraćem roku sa trostruko većom funkcionalnošću, s potpuno besplatnim alatom u vašem omiljenom tekstualnom editoru.

Open-source u gornjoj definiciji znači da svatko tko želi može skinuti izvorne PHP kodove pisane u C-u i, ukoliko ih razumije, može ih mijenjati po svojoj volji te dodavati nove funkcije PHP-u. Štoviše, svi su pozvani da sudjeluju u razvoju novih verzija PHP-a. Izvorne kodove i instalacijske datoteke možete skinuti sa službenog PHP sitea.

Ono što PHP stavlja još više ispred ostalih web skriptnih tehnologija je njegova podrška za baratanje širokom paletom baza podataka. Podržava sve popularnije baze podataka kao što su MySQL, PostgreSQL, dBase, Oracle, ODBC…

3. Sintaksa PHP-a

Već smo u uvodu mogli vidjeti neke bitne stvari, npr. da se sav PHP kod nalazi između **<?** i **?>** kvačica. Mali dodatak ovom pravilu bilo bi korištenje **<?php … ?>** kvačica radi razlikovanja između PHP i XML koda (XML koristi iste ove kvačice). Druga stvar koja je očita iz prijašnjih primjera je da varijable prije svog imena imaju znak $. To je ujedno i prva tema koje ćemo se dotaknuti.

3.1 Varijable

Već smo spomenuli, varijable prije svog imena obavezno moraju sadržavati znak $. Tako PHP govori prevoditelju da se radi o varijabli, a ne o tekstu. Ukoliko izostavite znak $, aplikacija će javiti grešku (u najboljem slučaju) ili će prijeći preko nje (u najgorem slučaju) i umjesto sadržaja varijable ispisati samo njeno ime.

Još jedna vrlo bitna stvar kod varijabli u PHP-u je da su **imena varijabli case-sensitive**. To znači da program razlikuje velika i mala slova, evo jednog malog primjera za ilustraciju:

"$mojeime"** nije isto što i **"$MojeIme"

Isto tako, u imenima varijabli ne smijete koristiti razmake niti bilo kakve znakove osim [i] (koji se koriste u nizovima i kod nekih metoda rada sa stringovima) te znaka '_'. Svi ostali znakovi su zabranjeni u imenima varijabli. Isto tako, ime varijable ne smije početi s brojem, ali ga može sadržavati na bilo kojoj drugoj poziciji u imenu. Kod **imenovanje varijabli** preporuka je držati se sljedećeg nepisanog pravila. Na prvom mjestu imena se nalazi opis tipa varijable od tri znaka. Nakon njega odvojeno sa '_' slijedi ime varijable koje bi trebalo pobliže opisati ime sadržaj same varijable. Ime se obično sastoji od jedne ili dvije riječi koje pobliže opisuju sadržaj varijable. Riječi možete odvajati sa znakom '_' ili svako početno slovo riječi možete napisati velikim slovom.

Pridržavanjem ovih pravila činite uslugu sebi jednako koliko i ostalima koji će jednog dana pokušati pročitati i razumjeti vaše kodove.

Pokušajte si zamisliti ovaj primjer : Otvorite vašu aplikaciju na kojoj ste radili prije godinu dana. Pred vama se sada nalazi hrpa slova i znakova, ako niste vodili računa o označavanju i izboru imena varijabli (npr. $prva, $druga, $post, $mail, $ime) koja vam više nemaju nekakvo značenje. Sada morate cijelu aplikaciju ponovo proraditi da bi mogli napraviti tek malu promjenu. Vrlo jednostavnom upotrebom standarda pri imenovanju varijabli i čestim komentiranjem koda ovih problema ne bi bilo. Zamislite samo da je netko drugi otvorio takve kodove. Trebalo bi mu par dana da shvati što koji red izvršava i što se nalazi u kojoj varijabli u određenom trenutku. Stoga, pomognite sebi i drugima i pišite kodove sa komentarima i standardiziranim imenima varijabli. Vrijedno je truda.

Prebacivanje tipova varijabli

Sadržaj bilo koje varijable podložan je izmjeni svog tipa. Znači da nekakav broj može vrlo lako postati string i obratno. Tipove možete mijenjati implicitno i eksplicitno (u slijedećim primjerima nećemo se pridržavati gore navedenih pravila imenovanja varijabli).

Primjer:

```
<?
// eksplicitno
$int_neki_broj=10;
$str_neki_broj= (string) $int_neki_broj ;
// prebacili smo broj u string (ASCII znakove)
?>
```

Eksplicitna izmjena tipa varijable vrši se tako da u neku novu varijablu pridružite neku već postojeću varijablu i ispred nje navedete u zagradama tip u koji želimo prebaciti varijablu koju pridružujete. Isto tako možete u varijablu pridružiti nju samu s eksplicitnom izmjenom tipa podatka.

Primjer:

```
<?
$int_var=10;
$int_var=(double) $int_var;
echo gettype($int_var);
echo "<br>"; //Prijelaz u novi red
echo $int_var;
// gettype($int_var) vraća tip dane varijable
?>
```

Moguće konverzije su:
(int), (integer) - prebaci u integer
(real), (double), (float) - prebaci u double (realni broj)
(string) - prebaci u string
(array) - prebaci u niz
(object) - prebaci u objekt

3.2 Komentari

PHP podržava više tipova komentara. U dosadašnjem tekstu koristili smo inline komentare (// komentar). Oni preskaču sav tekst koji se nalazi iza njih sve do početka novog reda. Komentirati možemo kad na početak reda stavimo znak "#"Želite li komentirati više redova koristite se multiline komentarima.

Primjer:

```
<?
/* Ovo je komentar
koje se proteže kroz
čaktri reda */

echo "gornji tekst se nece izvrsavati";

# ovo je isto komentar
?>
```

3.3 Tipovi podataka

U PHP-u ne postoje fiksni tipovi podataka varijabli. Naime, ne morate definirati tip varijable prije njenog korištenja - varijablu možete deklarirati bilo kada unutar skripte i pridruživati joj različite tipove podataka tokom izvođenja skripte. Isto tako možete mijenjati tip podataka neke varijable jednog te istog sadržaja, ali o tome par redaka kasnije.

Tipovi podataka koje podržava PHP su:

- Cijeli brojevi (integer)
- Realni brojevi (floating-point numbers)
- Tekstualni podaci (String)
- Nizovi
- Objekti

U sljedećim primjerima koristit ću neke funkcije koje će vam možda biti nepoznate i neće vam odmah biti jasno čemu one služe. Zanemarite ih i pokušajte shvatiti primjere što bolje možete. Sve funkcije će biti detaljnije objašnjene malo kasnije kad ćemo pričati o kontrolnim strukturama i sličnim stvarima.

Cijeli brojevi

U ovaj tip varijable možemo pohraniti pozitivne i negativne brojeve u rasponu od -2147483648 do 2147483647 tj. 32 bita podataka. Možemo ih zapisati u decimalnom, oktalnom ili heksadecimalnom zapisu. Par primjera:

Primjer:

```
$int_var=123; //pozitivan decimalni broj
$int_var=-123; //negativni decimalni broj
$int_var=0123; //oktalni broj
$int_var=0#123; //heksadecimalni broj
```

-probajte ispisati varijable dodavši naredbu **"echo"**

Realni brojevi

Postoje dva načina spremanja realnih brojeva:

Primjer:

```
$dbl_var=0.123;
// ili
$dbl_var=1.123e8;
```

Budite pažljivi kada koristite realne brojeve. Naime, njihova točnost nije garantirana (zbog pretvaranja ovog broja u njegov binarni ekvivalent - recimo 0.33333 nikada neće biti točno prebačen u binarni ekvivalent). Stoga, nemojte ih uspoređivati za jednakost i vjerovati im do posljednje decimale.

Tekstualni podaci

Sadržaj string tipa varijable nalazi se između navodnika. Možete koristiti duple i jednostruke navodnike. Postoje razlike u ispisu sadržaja ovisno o tipu navodnika koje koristite.

Korištenjem duplih navodnika možete koristiti 'special characters'. To su posebni znakovi koji govore PHP-u da izvrši određene radnje pri ispisu sadržaja varijable. Ako ste ikada radili u C-u ili Perlu, već ste upoznati s ovim znakovima. To su znakovi koji slijede iza znaka backslash (\ ). On se ujedno koristi za preskakanje određenog znaka unutar stringa.

Lista posebnih znakova

Znak	**Značenje**
\n	Novi red(LF ili 0x0A u ASCIIu)
\t	Tab razmak (HT ili 0x09 u ASCIIu)
\\	Backslash
\$	Dolar znak
\"	Dupli navodnik

Ovi znakovi neće imati utjecaja na izgled same stranice u prozoru browsera, već će njihov utjecaj biti vidljiv tek pri pregledu sourcea dokumenta. Ovime možete sasvim sakriti činjenicu da je stranica stvorena putem PHP-a i pomoću njih je lakše pronaći grešku u generiranoj stranici. U protivnom bi se sav sadržaj ispisao u jedan red bez razmaka. Uviđate da bi bilo vrlo teško u tom neredu naći bilo što, a kamoli grešku u ispisu, ako ju tražite u source viewu.

Ukoliko želite da se neki tekst prebaci u novi red, pri gledanju stranice u prozoru browsera morat ćete se poslužiti
 i sličnim tagovima. Znači, ako se želite koristiti PHP-om, morate se jako dobro znati služiti HTML-om.

Još jedan bitna razlika između duplih i jednostrukih navodnika je ta da će se pri korištenju duplih navodnika sadržaj varijable ispisati, a pri korištenju jednostrukih navodnika ispisat će se ime varijable skupa sa znakom $. Mali primjer će ovo dobro ilustrirati:

Primjer:

```
<?
$str_ime="Kreso";
echo ("moje ime je $str_ime");
?>
// Ispisuje: moje ime je Kreso
<?
$str_ime="Kreso";
echo ('moje ime je $str_ime');
// Ispisuje: ispisati: moje ime je $ime
?>
```

U PHP-u je također moguće spajanje više stringova u jedan ispis. To radimo pomoću '.' . Evo primjera:

Primjer:

```
<?
$str_var1='Student';
$str_var2='ide'; //nema razmaka prije ili
                 // poslije riječi

echo $str_var1 . ' ' . $str_var2 . ' na fax.';
// rezultira sa
// Student ide na fax.
?>
```

Uočite da su u gornjem primjeru korišteni jednostruki navodnici, ali varijable se ne nalaze u njima, pa će se njihov sadržaj ispisati. Također su izostavljene zagrade, što je dozvoljeno.

Kao što sam već spomenuo, pri radu sa stringovima možemo u njihovom imenu koristiti znakove [i]. Njih koristimo kada želimo izdvojiti određeni znak iz stringa. Drugim riječima, string zamislimo kao jednodimenzionalni numerički niz indexiran na taj način da se na svakom broju, počevši od 0, nalazi jedan znak stringa. Raspon indexa je od 0 do n-1; gdje je n broj znakova niza.

Primjer:

```
<?
$str_tekst='Dijete ide u školu';
$str_znak=$str_tekst[0];
echo $str_znak; //ispisuje 'D'
echo $str_tekst[3]; //ispisuje 'e'
echo $str_tekst[strlen($str_tekst)-1]
//ispisuje zadnji znak 'u'
// strlen($str_tekst) vraća broj znakova u stringu
?>
```

Kao što vidite, u njih možete smjestiti bilo koji izraz koji će na kraju rezultirati cijelim brojem (integerom).

Nizovi

PHP podržava više vrsta nizova - tekstualne (associative) i cjelobrojne (vectors / indexed). Mogu biti jednodimenzionalni ili multidimenzionalni.
Primjer cjelobrojnog jednodimenzionalnog niza:

Primjer:

```
<?
$arr_boje=array('plavo','žuto','zeleno');
echo $arr_boje[0]; // ispisat će 'plavo'
echo $arr_boje[2]; // ispisat će 'zeleno'
$arr_boje[3]='crveno'; // dodaje novi element u niz
$arr_boje[2]='ljubičasto';
// mijenja staru vrijednost na indexu 2
// - zeleno prelazi u ljubičasto
$arr_boje[7]='roza'
// indexi ne moraju slijediti kronološki redoslijed
// želite li ispisati sve elemente niza možete
// se služiti ovom metodom
foreach
($arr_boje as $int_kljuc => $str_vrijednost){
echo $int_kljuc . " => " . $str_vrijednost . "<br>\n";
```

```
}
// što će rezultirati sa
// 0 => plavo
// 1 => žuto
// 2 => ljubičasto
// 3 => crveno
// 7 => roza
?>
```

Primjer:

```
<?
// ako želite petljom stvoriti niz od n elemenata
// gdje će svakom elementu biti pridodan
// faktorijel njegovog indeksa učinite to ovako
$n=10; // niz će imati 10 elemenata
$int_faktorijel=1; // inicijalizacija faktorijela
for ($i=1;$i<=$n;$i++){
$int_faktorijel*=$i;
// ovdje smo mogli komotno koristiti i
//$int_faktorijel=$int_faktorijel * $i;
$arr_niz[$i]=$int_faktorijel;
// elementu pridružujemo njegov faktorijel
// mogli smo koristiti i
// $arr_niz[]=$int_faktorijel;
// ali onda indeks ne bi odgovarao faktorijelu
// jer bi indeksi počeli s 0 a završili sa 9
}
foreach
($arr_niz as $int_kljuc => $int_vrijednost){
echo $int_kljuc . " => " . $int_vrijednost . "<br>\n";
}
// što će rezultirati sa
// 1 => 1
// 2 => 2
// 3 => 6
// 4 => 24
// 5 => 120
// 6 => 720
// 7 => 5040
// 8 => 40320
// 9 => 362880
// 10 => 3628800
?>
```

Primjer tekstualnog (associative) niza:

Primjer:

```
<?
// recimo da želite reproducirati sadržaj
// svoje torbe u niz
$arr_torba=array(
"olovka"=>4,
"gumica"=>1,
"knjiga"=>3,
"index"=>"0",
"disketa"=>"5"
);

echo $arr_torba["knjiga"]."<br>";
// ispisali ste koliko knjga imate u torbi
// ili multidimezionalni
$arr_boje=array(
"tople"=>array("žuta","crvena"),
"hladne"=>array("plava","zelena")
);
// ako želite ispisati npr žuta
echo $arr_boje["tople"][0]."<br>";
?>
```

Možemo i kombinirati ova dva tipa niza:

Primjer:

```
<?
$arr_kontakti = array(
"kreso"=>array("visina"=>182,"tezina"=>70,
0=>"01/9876-543",1=>"091/3432-876"),
"mirta"=>array("visina"=>164,"tezina"=>63,
0=>"01/3256-937",1=>"098/435-556")
);
echo $arr_kontakti["kreso"][0]."<br>\n";
echo $arr_kontakti["mirta"]["tezina"]."<br>\n";
?>
```

Objekti

Istina je - PHP podržava objektno programiranje. Doduše, ne onako kako ga podržava C++ ili slični jezici, ali svoje funkcije možete grupirati u klase te stvarati instance tog objekta kroz cijelu aplikaciju. Sve u svemu, da biste koristili objekte, prvo morate stvoriti klasu (class) s nekim funkcijama u njoj te ju pozivati unutar koda.

Primjer:

```
<?
class class_proba{
function ispis_probne_klase(){
echo "Ispisujem probnu funkciju objekta";
}
}
$obj_probni_objekt=new class_proba;
$obj_probni_objekt->ispis_probne_klase();
// rezultira ispisom
// Ispisujem probnu funkciju objekta
?>
```

3.4 Operatori

Aritmetički operatori:

Primjer	Ime	Rezultat
$a + $b	Zbrajanje	Zbroj od $a i $b
$a - $b	Oduzimanje	Razlika od $a i $b
$a * $b	Množenje	Produkt od $a i $b
$a / $b	Dijeljenje	Kvocijent od $a i $b
$a % $b	Modul	Ostatak dijeljenja od $a i $b

Mala napomena : Ukoliko dijelite dva broja koja su oba cjelobrojna, i kvocijent će biti cjelobrojan. Ukoliko je jedna od varijabli realnog tipa i kvocijent će biti realnog tipa.

Operatori pridruživanja:

Primjer:

```
<?
$int_var=5;
$int_var+=5;  // sada je vrijednost varijable 10
?>
- isto kao da smo napisali
<?
$int_var=$int_var+5; // opet je vrijednost varijable 10
$int_var*=5; // vrijednost varijable je 50
$int_var/=10 // vrijednost je 5
?>
kod stringova imamo
<?
$str_tekst='Moje ime je ';
$str_tekst.='Kreso';
// sada je sadržaj varijable Moje ime je Kreso
//možete se igrati malo složenijim izrazima poput
$a = ($b = 4) + 5; // rezultat je 9
?>
```

Logički operatori:

$a and $b	I	True ako su oboje $a i $b true
$a or $b	Ili	True ako je $a true ili ako je $b true
$a xor $b	Xor	ako je $a true ili ako je $b true, ali ne i ako su oba true
! $a	Ne	True ako je $a false i obrnuto
$a && $b	I	True ako su oboje $a i $b true
$a || $b	Ili	True ako je $a true ili ako je $b true

U gornjoj tablici $a ili $b mogu biti bilo koji izrazi koji vraćaju true ili false kao ishod svoje operacije. Zato će sljedeći primjer biti ne samo dozvoljen, već i prijeko potreban.

Primjer:

```
<?
If ( ( ($int_var%2)==0) and ($int_var>10)){
// kod koji se izvršava samo ako je broj paran
// i veći od 10

} else {
// kod koji se izvršava ako je broj neparan ili
// ako je manji ili jednak 10 ili oboje
}
?>
```

Operatori uspoređivanja:

Primjer	Ime	Rezultat
$a == $b	Jednako	True ako je $a jednako $b
$a === $b	Identično	True ako je $a jednako $b, i ako su istog tipa
$a != $b	Nije jednako	True ako $a nije jednako $b
$a !== $b	Nije identično	True ako $a nije jednako $b, i ako nisu istog tipa
$a < $b	Manje	True ako je $a izričito manje od $b
$a > $b	Veće	True ako je $a izričito veće od $b
$a <= $b	Manje jednako	True ako je $a manje ili jednako $b
$a >= $b	Veće jednako	True ako je $a veće ili jednako $b

Budite pažljivi pri traženju jednakosti dvije varijable da ne upišete $a=$b umjesto $a ==$b . Naime, u prvom slučaju izraz će vratiti true ako uspješno pridruži sadržaj varijable $b varijabli $a, a u drugom slučaju će vratiti true ako su jednake.

Operatori uvećavanja i smanjivanja:

Primjer	Ime	Efekt
++$a	Preduvećavanje	Uveća $a za jedan, i onda vrati $a
$a++	Naknadno uvećanje	Vrati $a, i onda ga uveća za jedan
--$a	Predsmanjenje	Umanji $a za jedan, i onda vrati $a
$a--	Naknadno smanjenje	Vrati $a, i onda ga umanji za jedan

3.5 Kontrolne strukture

Pomoću kontrolnih struktura određujemo tok skripti, odlučujemo i računamo. One su zadužene za logiku aplikacija.

If.. else

If.. else je najčešće korištena kontrolna struktura. Njoj dajemo logički izraz koji se provjerava i ovisno o njegovom ishodu koji može biti true ili false izvršava se blok naredbi.

Primjer:

```
<?
If ( uvjet ) {
// naredbe koje se izvršavaju ukoliko je uvjet == true
} else {
// naredbe koje se izvršavaju ukoliko je uvjet == false
}
?>
```

Vitičaste zagrade ({}) označavaju blok naredbi. Njih možete izostaviti ukoliko grana ima samo jednu naredbu. Npr.:

Primjer:

```
<?
$str_ime='Matija';
if ($str_ime=='Matija')
echo 'Bok matija';
else
die ('Ti nisi Matija. Ajde bok');
?>
```

die() je funkcija koja prekida izvršavanje skripte.
Ako joj u argument date neki tekst ili broj, ispisat će ga. Također prima neku funkciju kao argument

Elseif

Umjesto else ključne riječi može se koristiti i elseif ključna riječ. Ona se izvršava ako je uvjet u if-u rezultirao false. Ona također ispituje logički izraz.

Primjer:

```
<?
If ( uvjet ) {
// naredbe koje se izvršavaju ako je uvjet == true

} elseif ( uvjet2 ) {
// naredbe koje se izvršavaju ako je uvjet == false
// i uvjet2==true

} elseif ( uvjet3 ) {
// naredbe koje se izvršavaju ako je uvjet == false
// i uvjet2 == false i uvjet3 == true

} else {
// naredbe koje se izvršavaju ako su svi uvjeti ==
//false
}
?>
```

Switch

Ukoliko pokušavamo riješiti situaciju s mnogo mogućih ishoda, nije praktično koristiti if … elseif tip grananja. U tom slučaju koristite se switch strukturom. Switch uzima za argument nekakav izraz i onda gleda da li je on jednak jednom od zadanih slučaja. Ukoliko nije jednak niti jednom od njih, izvršava default akciju ili ne izvršava ništa. Switch je idealan alat za izradu višenamjenskih stranica.

Primjer:

```
<?
switch ( uvjet ){
case < slučaj1 >:
// naredbe koje se izvršavaju ukoliko je uvjet
// jednak slučaju 1

break;
case < slučaj2 >:
// naredbe koje se izvršavaju ukoliko je uvjet
// jednak slučaju 2

break;
case < slučaj3 >:
// naredbe koje se izvršavaju ukoliko je uvjet
// jednak slučaju 3

break;
default:
// naredbe koje se izvršavaju ukoliko uvjet
// nije jednak niti jednom slučaju. Njega
// se može izostaviti ukoliko se niti
// jedna naredba ne treba izvršiti u tom slučaju
}
?>
```

Ključna riječ break označava završetak grane. Ukoliko ga izostavite između dvije grane, naredbe obje grane će se izvršiti ukoliko je viša (ona koja slijedi prije) aktivirana. Naredbe će se izvršavati sve dok se ne pojavi break ili završetak switcha.

Primjer:

```
<?
                $int_var=5;
switch ($int_var){
                case 0:
                echo 'Broj je nula';
                break;
                case 5:
                echo 'Broj je pet';
                case 6:
                echo 'Broj je 6';
                break;
                default:
                echo 'Broj nije poznat';
                }
?>
```

While petlja

While petlja izvršava svoj blok naredbi dokle god je izraz u uvjetu istinit (true). Uvjet se ispituje prije izvođenja bloka naredbi. Zbog toga je moguće da se blok ne izvrši niti jednom ukoliko je uvjet na početku false.

Primjer:

```
<?
while ( uvjet ) {
// naredbe koje se izvršavaju dok je uvjet true
}
?>
Evo konkretnog primjera:
<?
$int_var=10;
while ($int_var<=20){
echo '$int_var = '. ++$int_var. "<br>";
}
// rezultira sa
                // $int_var = 11
                // $int_var = 12
                // $int_var = 13
                // $int_var = 14
                // $int_var = 15
                // $int_var = 16
                // $int_var = 17
                // $int_var = 18
                // $int_var = 19
                // $int_var = 20
                // $int_var = 21
?>
```

Do … while

Za razliku od normalne while petlje, kod Do … while petlje uvjet se ispituje tek nakon izvršavanja bloka naredbi. Tako je uvijek zagarantirano barem jedno izvršavanje bloka naredbi iako je uvjet odmah na početku false.

Primjer:

```
<?
do {
// naredbe koje se izvršavaju dok je uvjet true
} while ( uvjet )
?>
```

For petlja

For petlja koristi brojač petlje koji se prije svakog izvršavanja bloka naredbi petlje uveća ili smanji. For petlju koristite kada znate točan broj potrebnog ponavljanja bloka petlje. Brojač petlje može biti bilo koja već postojeća varijabla ili možmo stvoriti novu varijablu za potrebe petlje. Ukoliko rabite drugi tip, uobičajena imena takvih varijabli su $i, $j, $k i njih ćete sresti u gotovim svim aplikacijama diljem svijeta.

Primjer:

```
<?
for ($i=0;$i (operator usporedivanja) (vrijednost sa
kojom uspoređujete); (operator uvećanja ili smanjenja){
// naredbe koje se izvrašavaju svaki put dok je uvijet
// jednak true
}
// ili na konkretnom primjeru
for ($i=10;$i>=0;$i--){
echo '$i = ' . $i . '<br>';
}
// što rezultira
                // $i = 10
                // $i = 9
                // $i = 8
                // $i = 7
                // $i = 6
                // $i = 5
                // $i = 4
                // $i = 3
                // $i = 2
                // $i = 1
                // $i = 0
?>
```

Foreach petlja

Foreach petlja se koristi za rad s nizovima. Ona prolazi kroz svaki element danog niza i obavlja blok naredbi. Može spremiti ključ i vrijednost svakog elementa niza u posebne varijable. U tim se varijablama za svako ponavljanje petlje nalaze ključ i vrijednost elementa niza na kojem se trenutno nalazi nutarnji pokazivač. Nutarnji pokazivač se prije ulaska u petlju nalazi na 0 i svakim novim krugom u petlji povećava se za 1. Novim zvanjem foreach petlje nutarnji pokazivač se resetira. Petlja se vrti sve dok ne ostane bez elemenata niza.

Primjer:

```
<?
// općenito
foreach ($neki_niz as $vrijednost){
// naredbe koje se izvršavaju za svaki element //niza
}
// ili
foreach ($neki niz as $kljuc => $vrijednost){
// naredbe koje se izvršavaju za svaki element niza
}
// evo jedan primjer
$arr_torba=array(
                "bilježnica"=>4,
                "index"=>1,
                "knjiga"=>2,
                "gumica"=>1,
                "sokova"=>"0.5 l - u bočici",
                );

foreach ($arr_torba as $kljuc => $vrijednost){
echo "$kljuc => $vrijednost <br>";
}
?>
```

3.6 Forme i prihvat informacija iz forme

Kao mali dodatak ovom dijelu nužno je spomenuti forme i prihvat podataka iz njih. Da bismo demonstrirali način prihvata informacija iz forme, možda bi bilo potrebno reći par stvari prije samog primjera. Postoje tri vrste formi.
Promotrimo sam izgled HTML forme:

Primjer:
```
<form  name="form1" method="post" action="">

</form>
```

Form tag ima tri parametra. Prvi je ime. Drugi parametar je metoda slanja forme. O njemu ovisi hoće li podaci poslani formom biti vidljivi korisniku pri odlasku na stranicu koja obrađuje formu ili ne. Naime, podaci iz forme šalju se skupa sa zahtjevom za stranicu koja obrađuje formu.

Ukoliko koristimo method="post", informacije će biti nevidljive korisniku i do njihovih vrijednosti možete doći samo putem skripte. Ukoliko koristite method="get", informacije iz forme nalazit će se u URL-u pri otvaranju stranice koja obrađuje formu.

Primjer: http://www.fsb.hr/obrada.php?ime=Dražen&prezime=Petrović (Ovaj link nije stvaran i ne postoji. Nemojte ga kliknuti.)

Isti ste učinak mogli dobiti ako bi negdje na stranici imali link u kojem se uz adresu stranice nalazi znak ? i ime=vrijednost. Ukoliko ih želite imati više, odvojite dva ime=vrijednost para sa znakom &. Ovo je query string metoda. U action="" upisujete adresu stranice koja obrađuje formu.
Tako bi forma mogla izgledati otprilike ovako:

Primjer:
```
<form
 name="form1" method="post" action="obrada.php">
 Ime
<input type="text" name="ime">
<br>
 Prezime
<input type="text" name="prezime">
<br>
<input
 type="submit" name="slanje" value="Po&#154;alji">
</form>
Spremite ovu formu u file koji nazovite forma.htm.
Sada stvorite novi file i nazovite ga obrada.php.
Spremite ga u isti folder kao i forma.htm.
Primijetite da file koji sadrži formu ne mora biti php
file.
<?
// prihvat i ispis podataka forme
echo 'Dobar dan'. $ime.' ' . $prezime;
// i to je to.
?>
```

Istom ovom metodom prihvaćate podatke poslane GET metodom ili pomoću query string metode.

4. Rad s bazama podataka

Mogućnost pristupa bazama podataka nesumnjivo je najvažnije svojstvo svakog modernog programskog jezika. Značaj pristupa bazama podataka je leži u činjenici da sistem za upravljanje bazama podataka osigurava jako puno moćnih funkcija za rad s bazama podataka. Jezik PHP sadrži bogat skup funkcija za pristup različitim bazama podataka. Autori jezika PHP smatraju mogućnost povezivanja s bazama podataka njegovim najjačim i najznačajnijim svojstvom.

Podržava sljedeće sisteme za upravljanje bazama podataka:

Abadas D	InteBaes	Solid
dBase	mSQL	Sybase
Empress	MySQL	Velocis
FilePro	Oracle	Unix dbm
Informix	PostgreSQL	Microsoft SQL Server
ODBC		

4.1 MySQL

MySQL je odličan DBMS sistem koji ima široku primjenu, ujedno je i baza podataka koja se najviše koristi pri radu u PHP-u. Razvijen od Švedske firme TcX. Višenitni je sistem za rukovanje relacionim bazama podataka, kontrolira tko smije koristiti baze, vodi evidenciju o procesima. Baza koja se sastoji od tablica sa stupcima koji su međusobno povezani. Međusobne veze su definirane ključnim vrijednostima u stupcima.

Odnosi u bazi

- One-to-One
- One-to-Many
- Many-to-Many

Kao što smo već spomenuli u uvodu prednost PHP-a pred drugim web skriptnim tehnologijama je rad sa bazama podataka, jedna od najčešće korištenih je MySQL baza podataka.

5. PHP funkcije

U ovom djelu obraditi ćemo neke od funkcija PHP-a. Funkcije su podjeljene u podgrupe i uz većinu će biti primjer da bi lakše shvatili kako se određena funkcija koristi.

Grupe opisanih funkcija:

- Array Funkcije
- Datum i Vrijeme
- Direktorijske Funkcije
- Direktorijske Funkcije
- File Funkcije
- FTP Funkcije
- MySQL Funkcije
- Regularni Izrazi
- Sessions
- String Funkcije
- Varijable

5.1 Array Funkcije

Grupe funkcija za rad sa nizovima *"array"*, kreiranje nizova, uspoređivanje, brisanje i dr.

- **array()** - kreira numeričke i asocijativne arraye
  Format: array $array (...)
  Opis: Kreira numeričke i asociativne arraye. Elementi su razdvojeni zarezom. Da bi jednom elementu dodali odgovarajući index i napravili asocijativni array, koristite =>. Također je moguće praviti arraye u arrayu, tako da je taj novi array jedan element glavnog arraya = multidimenzionalni array.

Primjer:

```
<?php

$adrese = array(
    "fsb"=>"http://www.fsb.hr",
    "email"=>"http://www.yahoo.com/");

$imena = array("Marko","Vedran","Stipe");

echo $adrese["fsb"]." - ".$imena[1];
// Rezultat je: http://www.fsb.hr - Vedran
?>
```

Važno: Elementi u arrayu počinju s rednim brojem 0.

- **array_diff() -** izračunava razlike između arraya
  Format: array_diff($array1,$array2...)
  Opis: Izračunava razlike između arraya. Kao rezultat vraća array sa elementima iz array1 koji se ne ponavljaju u drugim arrayima.

Primjer:

```
<?php

$array1 = array ("zeleno", "crveno", "plavo","zuto");
$array2 = array ("zeleno", "crveno", "crno");

$result = array_diff ($array1, $array2);

echo implode(" | ",$result);

// Rezultat je: plavo | zuto

?>
```

- **array_intersect()** – izračunava razliku između arraya
  Format: array_intersect($array1,$array2...)
  Opis: Izračunava razlike između arraya. Kao rezultat vraća array sa elementima iz array1 koji se ne ponavljaju u drugim arrayima.

Primjer:

```
 <?php

$array1 = array ("zeleno", "crveno", "plavo","zuto");
$array2 = array ("zeleno", "crveno", "crno");

$result = array_diff ($array1, $array2);

echo implode(" | ",$result);

// Rezultat je: plavo | zuto

?>
```

- **array_keys()** – pokazuje sve keys iz arraya
  Format: rray_keys($array[trazeni value])
  Opis: Pokazuje sve keys (numeričke i stringove) od datog arraya. Ako je zadana i tražena vrijednost (value), onda će biti pokazani samo keys sa tim value (vrijednostima).

Primjer:

```
 <?php

$array = array (0 => 100, "boja" => "crvena");
print_r(array_keys ($array));

// Rezultat je: Array ( [0] => 0 [1] => boja )


$array = array ("plava", "crvena", "zelena", "plava",
"plava");
print_r(array_keys ($array, "plava"));

// Rezultat je: Array ( [0] => 0 [1] => 3 [2] => 4 )

$array = array ("color" => array("plava", "crvena",
"zelena"), "size" => array("mala", "srednja",
"velika"));
print_r(array_keys ($array));

// Rezultat je: Array ( [0] => color [1] => size )

?>
```

5.2 Datum i Vrijeme

Funkcije za rad sa datumima i vremenom. PHP podržava jako puno ovih funkcija, vrijeme se može prikazivati u danima, satim, sekundama. Može biti globalno ili lokalno i dr. Evo nekih od funkcija.

- **checkdate()** – provjerava da li je datum važeći
  Format: checkdate(mjesec,dan,godina)
  Opis: Provjerava da li je datum važeci. Ako je, daje true kao odgovor a inače false.
  Sljedeći uvjeti moraju biti ispunjeni:
  - godina mora biti između 1 i 32767
  - mjesec mora biti između 1 i 12
  - dan mora biti važeći (npr. mjesec ne moze imati 40 dana)
  Prijestupne godine također se kontroliraju.

Primjer:

```
<?php
if(checkdate(2,29,2000))
{
     echo "2000. je prestupna godina";
}
else
{
     echo "Pogresan Datum.";
}
// Rezultat je: 2000. je prestupna godina
?>
```

- **date()** – formatira datum i pokazuje lokalno vrijeme
  Format: date(datum_format[,"timestamp"])
  Opis: Pokazuje lokalno vrijeme. Ako ne napišete timestamp, pokazati će trenutno vrijeme. Timestamp pokazuje vrijeme od 01.01.1970. godine.

  Sljedeći formati smiiju se koristiti:
  a - "am" ili "pm" ("Prijeponde" ili "Poslijepodne")
  A - "AM" ili "PM"
  B - swatch internet vrijeme(1000 Beats u 24 h; Vrijeme u gradu Biel = 000 Beats)
  d - dan mjeseca sa nulom (npr. "09")
  j - dan mjeseca bez nule (npr. "9")
  t - broj dana u mjesecu (npr. "30")
  z - dan od početka godine (npr. "156")
  w - numerički dan tjedna (od "0" za nedelju do "6" za subotu)
  D - skraćeno ime za dan u tjednu (npr. "Thu")
  l - puno ime dana u tjednu (npr. "Thursday")
  F - puno ime mjeseca (npr. "December")
  M - skraćeno ime mjeseca (npr. "Dec")
  m - numeričko ime mjeseca sa nulom (npr. "05")
  n - numeričko ime mjeseca sa nulom (npr. "5")
  h - sat u 12-satnom formatiranju (npr. "05")
  H - sat u 24-satnom formatiranju (npr. "18")

g - sat u 12-satnom formatiranju bez nule (npr. "5")
G - sat u 24-satnom formatiranju bez nule (npr. "7")
i - minute (npr. "07")
U - protekle sekunde od 01.01.1970 (npr. "4890729")
s - sekunde (npr. "02")
S - engleski redni broj (npr. "th", "nd")
L - prijestupna godina. rezultat "0" ili "1"
Y - 4-cifreni godina (npr. "1999")
y - 2-cifrena godina (npr. "99")
Z - razlika u vremenu u odnosu na GMT (od "-43200" do "43200")

Primjer:

```
<?php

echo date("Y-F-l - H:i:s");

// Rezultat je: 2002-March-Thursday - 18:07:02

?>
```

- **getdate()**–pokazuje datum/vrijeme
  Format: getdate(timestamp)
  Opis: Vraća datum/vrijeme u obliku arraya, i to u sekundama od 01.01.1970. godine. Ako ne koristite timestamp, bit će pokazano trenutno vrijeme.

Sljedeći elementi se nalaze u array-u:
"seconds" - sekunde
"minutes" - minute
"hours" - sati
"mday" - dan u mjesecu (npr. "13")
"wday" - numerički dan u tjrdnu (npr. "2" za utorak)
"mon" - numerički mjesec (npr. "10")
"year" - godina (npr. "2002")
"yday" - dan od početka godine (npr. "225")
"weekday" - puno ime dana u tjrdnu (npr. "Saturday")
"month" - puno ime mjeseca (npr. "February")

Primjer:

```
 <?php

$today = getdate();
$month = $today[month];
$mday = $today[mday];
$year = $today[year];
echo "$month $mday, $year";

// Rezultat je: March 23, 2002

?>
```

- **gmdate()** – date(), ali u GMT zoni
  Format: gmdate()
  Opis: Isto kao date(), samo što se odnosi na GMT (Greenwich Mean Time). Na primjeru ispod se vidi razlika.

Primjer:

```
 <?php
echo date ("M d Y H:i:s", mktime (0,0,0,1,1,1998));

// Rezultat je: Jan 01 1998 00:00:00

echo gmdate ("M d Y H:i:s", mktime (0,0,0,1,1,1998));

// Rezultat je: Dec 31 1997 23:00:00
?>
```

5.3 Direktorijske Funkcije

Funkcije za rad sa direktorijima na serveru. Za otvaranje direktorija, ulaz u direktorij, prelazi u drugi direktorij i dr. Evo nekih od funkcija.

- **chdir()** – mijenja direktorij
  Format: chdir($dir)
  Opis: Mijenja trenutni PHP direktorij.

  Vraća true ako komanda uspije, inače false

- **dir()** – čita direktorij
  Format: dir($dir)
  Opis: Čita direktorij koji je prethodno otvoren.

Primjer:

```
<?php

$d = dir(".");
echo "Handle: ".$d->handle." | ";
echo "Path: ".$d->path." | ";

while($entry=$d->read())
{
     echo $entry." | ";
}

$d->close();

// Rezultat su fileovi iz direktorija "C:/":
//Handle: Resource id #1 | Path: C:/ | //AUTOEXEC.BAT |
BOOT.DOS | …
?>
```

- **getcwd()** – pokazuje trenutni aktivni direktorij
  Format: getcwd()
  Opis: Pokazuje trenutni aktivni direktorij.

Primjer:
```
<?php

echo getcwd();

// Rezultat je: C:\apache\htdocs\PHP_ucionica

?>
```

5.4 File Funkcije

Pomoću ovih funkcija radimo sa datotekama na serveru. Čitamo ih, kreiramo pišemo u njih, mjenjamo im vlasnika (na *nix sustavima) i dr.. Evo nekih od funkcija.

- **copy()** – kopira file
  Format: copy($file,$novi_file)
  Opis: Kopira file. $novi_file može biti samo novo ime file-a, ali može i novi direktorij, npr.: slike/index.html ili ../slikestare/index.html.bak . U svakom slučaju direktorij mora postojati, inače komanda neće biti uspješno izvršena.

Primjer:
```
<?php

$file = "index.html";
$novi_file = "index.html.bak";

if (!copy($file, $novi_file))
{
     echo "Kopiranje nije izvršeno";
}

?>
```

- **readfile()** – čita file i pokazuje sadržaj
  Format: readfile($file)
  Opis: Čita file i pokazuje sadržaj file-a.

Primjer:
```
<?php

readfile("neki_file.txt")

// Rezultat je: citav file bez \n

?>
```

- **file_exists()** – provjerava da li file postoji
  Format: file_exists($file)
  Opis: Provjerava da li file postoji i vraća true ako postoji, odnosno false ako ne postoji.

Primjer:
```
<?php

echo file_exists("data.txt");

// Rezultat je: 1

?>
```

- **fopen()** – otvara file ili URL
  Format: fopen($file,$modus)
  Opis: Otvara file lokalno ili na nekom drugom serveru (http:// ili ftp://). Funkcija vraća tzv. Index file-a koji drugim komandama govori na koji se od otvorenih file-ova misli. $modus govori u kojem modusu treba da se otvori file. Na izboru imate:
  'r' - otvara file samo za čitanje
  'r+' - otvara file za čitanje i pisanje
  'w' - otvara file samo za pisanje
  'w+' - otvara file za čitanje i pisanje i briše dosadašnji sadrzaj file-a, odnosno stvara novi file, ako ne postoji
  'a' - otvara file samo za pisanje i stavlja kursor na kraj file-a i stvara novi file, ako ne postoji
  'a+' - otvara file za pisanje i čitanje i stavlja kursor na kraj file-a i stvara novi file, ako ne postoji

  $modus može sadržati i slovo "b", koje govori komandi da se radi o binarnom file-u. Od značaja je samo na Windows serverima.

Primjer:
```
<?php

$fp = fopen ("/home/rasmus/file.txt", "r");
$fp = fopen ("/home/rasmus/file.gif", "wb");
$fp = fopen ("http://www.php.net/", "r");
$fp = fopen ("ftp://user:password@example.com/", "w");

// na Windowsu pazite na backslashes
$fp = fopen ("c:\\data\\info.txt", "r");

// Rezultat je: Indexi file-ova

?>
```

- **fwrite()** – piše u file pointer
  Format: fwrite($file_pointer)
  Opis: Piše u file pointer. Možete zadati koliko želite da upišete u $file_pointer pomoću $dužina. Ako ništa ne zadate, biti će upisan čitav string.

Primjer:
```
<?php

fwrite($file_pointer,"Ovaj tekst se upisuje u file");

// Rezultat je: u file pointer je upisan tekst

?>
```

5.5 FTP Funkcije

Koriste se za rad sa udaljenim računalom preko FTP protokola. Uspostavljanje FTP veze, skidanje(download) i slanje (upload) datoteka na udaljenom računalu i dr. Evo nekih od funkcija.

- **ftp_connect()** – uspostavlja vezu sa FTP serverom
  Format: ftp_connect($host[,$port])
  Opis: Uspostavlja/spaja se na FTP server. Obično koristi port 21, ako neki drugi $port nije posebno specificiran. Vraća 1 ako komanda uspije.

Primjer:
```
<?php

$ftp = ftp_connect("alpha1.fsb.hr",21);
echo $ftp;
// Rezultat je: 1

?>
```

- **ftp_fget()** – skida file sa FTP servera i snima ga u otvoreni file
  Format: ftp_fget($ftp,$lokalni_file_pointer,$ftp_file,$modus)
  Opis: Downloada file sa FTP servera i snima ga u otvoreni file pointer. Morate odrediti $modus prenosa file-a: FTP_ASCII ili FTP_BINARY. Vraća 1 ako komanda uspije.

Primjer:
```
<?php

$lokalni_file_pointer = fopen($lokal,"a+");
$ftp_file = "mirza.html";
echo
ftp_fget($ftp,$lokalni_file_pointer,$ftp_file,FTP_ASCII);

// Rezultat je: 1

?>
```

- **ftp_rename()** – preimenuje file na serveru
  Format: ftp_rename($ftp,$staro_ime,$novo_ime)
  Opis: Mjenja ime file-a na FTP serveru. Ako funkcija uspije vraća true, inače false.

Primjer:
```
<?php

if (!@ftp_rename($ftp,"stari.html","novi.html"))
{
     echo "File stari.html nije moguće preimenovati.";
}

else
{
     echo "Mjenjanje imena uspješno.";
}

?>
```

5.6 MySQL Funkcije

Služe za upravljanje sa MySQL bazom podataka. Evo nekih od funkcija.

- **mysql_connect()** – upostavlja vezu sa MySQL serverom
  Format: mysql_connect($host[:$port][$socket_path],$user[,$pass])
  Opis: Uspostavlja vezu sa MySQL serverom. Obično je dovoljno navesti samo $host, $user i $pass (ako $user ima definisan password). Dosta rijetko je potrebno upisati $port odnosno $socket_path. $host je najčešće definiran kao "localhost".

Primjer:
```
<?php

$host = "localhost";
$user = "fsb_stud";
$pass = "passwd";

$x = mysql_connect($host, $user, $pass);

// ili

$x = mysql_connect($host, $user, $pass) or die("Vezu
nije moguće uspostaviti.");

?>
```

- **mysql_create_db()** – kreira MySQL bazu
  Format: mysql_create_db($db[,$veza])
  Opis: Kreira bazu na MySQL serveru. Definiranje $veze (mysql_connect ili mysql_pconnect) nije obavezno. Moguće je koristiti i mysql_createdb (starija funkcija).

Primjer:
```
<?php

$x = mysql_connect($host,$user,$pass);
$y = mysql_create_db("user_fsb",$x]);

// Rezultat je: baza sa imenom user_fsb je kreirana.

?>
```

- **mysql_drop_db()** – Brise MySQL bazu
  Format: mysql_drop_db($db[,$veza])
  Opis: Briše bazu na MySQL serveru. Ako $veza nije definirana, onda se koristi trenutno akitvna veza.

Primjer:
```
<?php
$x = mysql_drop_db("user_fsb ");

if ($x)
{
     echo "Baza user_fsb je obrisana";
}
// Rezultat je: Baza user_fsb je obrisana (ako je
//komanda uspjela)

?>
```

5.7 Regularni Izrazi

Služe za rad sa stringovima. Evo nekih od funkcija.

- **ereg_replace()** – zamjenjuje dio stringa kroz drugi string
  Format: ereg_replace($stari,$novi,$string)
  Opis: Zamjenjuje dio stringa kroz drugi string.

Primjer:
```
<?php

$string = "www.fsb.hr";
echo ereg_replace ("www", "zrno", $string);

// Rezultat je: zrno.fsb.hr

?>
```

- **preg_quote()** - stavlja backslash ispred regexp znakova
  Format: preg_quote($string[,$granica])
  Opis: Stavlja backslash ispred znakova u regularnom izrazu, da specijalni znakovi kao što su . \\ + * ? [^] $ ( ) { } = ! < > | : ne bi bili prepoznati kao dio regularnog izraza. Ako $granica definirana, i isped nje će biti stavljen backslash.

Primjer:

```
<?php

$tekst = "www.fsb.hr je jedna *dobra* stranica.";
$rijec = "*dobra*";
$tekst = preg_replace
("/".preg_quote($rijec)."/","<i><b>".$rijec."</b></i>",$tekst);

echo $tekst;

/* Rezultat je: www.fsb.hr je jedna *dobra* stranica. ( rijec
*dobra* napisana debelim i kurzivnim slovima) */

?>
```

5.8 String Funkcije

Služe za rad sa stringovima. Evo nekih od funkcija.

- **print()** – prikazuje string
  Format: print($string)
  Opis: Prikazuje string.

Primjer:

```
 <?php

print ("www.fsb.hr");

// Rezultat je: www.fsb.hr

?>
```

- **strlen()** – otkriva duzinu stringa
  Format: strlen($string)
  Opis: Otkriva dužinu string - broji koliko ima znakova u stringu uključujući i space (prazan prostor).

Primjer:

```
<?php

echo strlen("koliko ima znakova ovaj tekst?");

// Rezultat je: 30 (sa praznim mjestima)

?>
```

- **strcspn()** – uspoređuje dva stringa po njihovim znakovima
  Format: strcspn($string1,$string2)
  Opis: Uspoređuje dva stringa po njihovim znakovima i vraća broj znakova u $string1 koji se nalaze prije nego što je nađen bilo koji znak iz $string2 u njemu -> bilo koji znak iz $string2 se traži u $string1, i kao rezultat se vraća broj mjesta s lijeve strane nađenog znaka. Komparacija je case sensitive. Pogledajte primjer za jasnije objašnjenje.

Primjer:

```
 <?php

echo strcspn("www.fsb.hr je super","ri");

/* Rezultat je: 9 (jer se slovo "r" iz "ri" nalazi  na
devetom mjestu u prvom stringu */

?>
```

- **strtr()** – zamjenjuje određene znakove kroz druge znakove
  Format: strtr($string,$znakovi,$zamjena)
  Opis: Zamjenjuje određene znakove ($znakovi) u stringu kroz druge znakove ($zamjena). Redoslijed i broj znakova u $znakovi i $zamjena moraju biti isti - prvi znak iz $znakovi se odnosi na prvi znak iz $zamjena.

Primjer:

```
<?php

echo strtr("www.fsb.hr", "wfsb", "Whrt");

// Rezultat je: WWW.hrt.hr


?>
```

5.9 Varijable

Služe za rad sa varijablama. Evo nekih od funkcija.

- **isset()** – provjerava da li varijabla postoji
  Format: isset($varijabla)
  Opis: Provjerava da li varijabla postoji. Ako postoji vraća 0, inače 1.

  Vazno: Kada formulari šalju varijablu, ona ima uvijek vrijednost 1. I kada se u varijabli ne nalazi ništa, formular dodaje prazan string u varijablu.

Primjer:

```
<?php

$test1 = 3;

echo isset($test1);

// Rezultat je: 1

?>
```

- **is_integer()** – kratki opis funkcije
  Format: is_integer($varijabla)
  Opis: Ispituje da li je varijabla prirodni broj (integer). Ako je vraća 1, inače 0.

Primjer:

```
<?php

$test1 = 12;
$test2 = 12.55;

echo is_integer($test1)." : ".is_integer($test2);

// Rezultat je: 1 : 0

?>
```

- **var_dump()** – daje informacije o varijabli
  Format: var_dump($varijabla)
  Opis: Daje informacije o varijabli.

Primjer:

```
<?php

$test1 = "Pero";
$test2 = 5872;

var_dump($test1);
echo " - ";
var_dump($test2);

// Rezultat je: string(5) "Pero" - int(5872)

?>
```

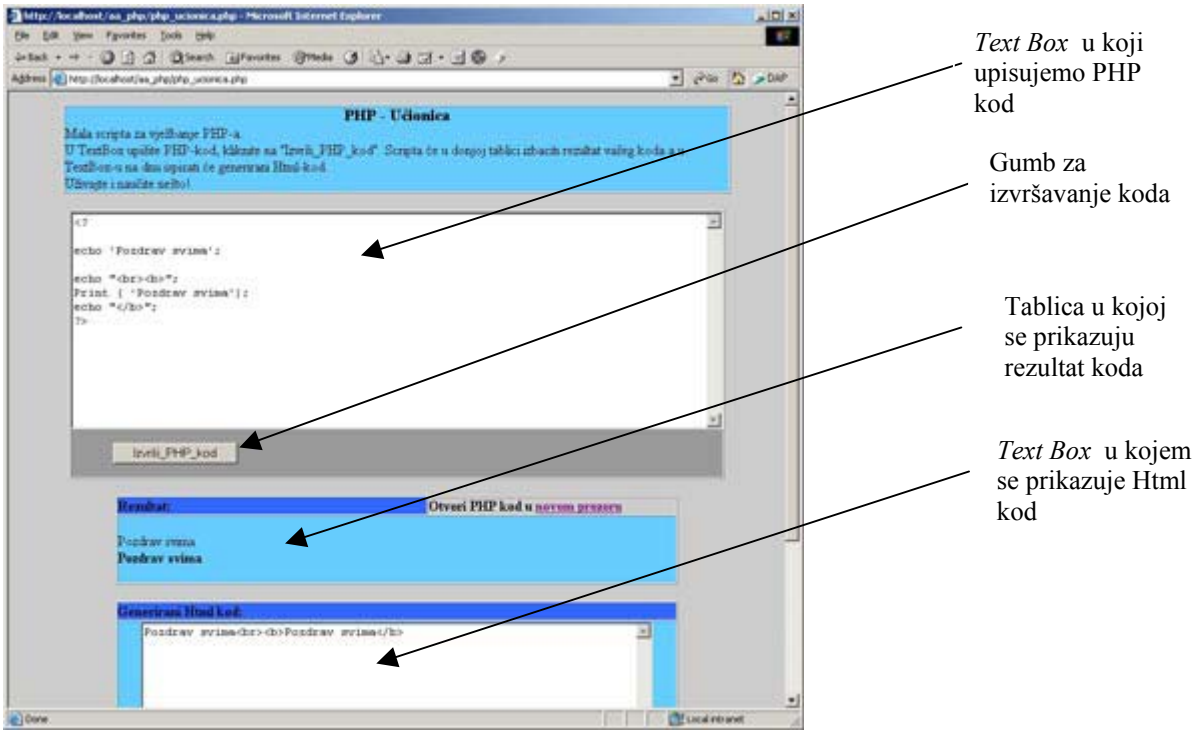
6. Prilog – (PHP_učionica)

Kao prilog ovom radu prilažem PHP scriptu koja sliži kao pripomoć pri savladavanju osnova u programskom jeziku PHP. Napisao sam je prije svega da meni olakša prolaz kroz početke programiranja a nadam se da će pomoći još nekom.

6.1 Način korištenje

Prije svega moram napomenuti da na vašem računalu morate imati instaliran PHP da bi mogli korirtiti ovu scriptu, što se toga tiče preporučam da instalirate "*phptriad*" ili "*nusphere*". Jedna i druga instalacija imaju sve što vam treba za početak (Apache-server, PHP i MySQL). Nakon toga potrebno je pokrenuti aplikacije. Kad smo to obavili sve fileove koji idu sa scriptom moramo smjestiti u "apache/htdosc/" direktorij ili neki poddrirektorij (u mom slučaju C:\apache\htdocs\php_ucionica). Onda u IE (ili u vašem omiljenom pregledniku) upišemo link scripte (u mom slučaju http://localhost/php_ucionica/php_ucionica.php) i naša scripta je pokrenuta i spremni smo za rad.

Sad dalje je sve jednostavno! U prvi "*Texbox*" upišemo php kod, i kliknemo na gumb "*Izvrši_PHP_kod*", scripta će preuzeti kod, izvršiti ga i ispisati rezultat u donoj tablici. U slučaju nekakvih grešaka, PHP vam automatski ljavlja u kojem redu je greška i karakter greške, vi pogledate, razmislite i otklonite grešku. Kad u prvom "*Texbox-u*" upisujete PHP kod koji treba generirati Html kod, puni prikaz tog Html koda biti će prikazan u drugom "*Texbox-u*".



Slika 1 - prikaz izgleda scripte

Kratka napomena: Ako želite primjere iz ovog word dokumenata prebacivati sa "*Copy*", "*Paste*" u ovu scriptu doći će do problema jer kopirani tekst iz tablice gubi svoj format (prikazati će se sve u istom redu). Rješenje je da ovaj dokument iz Worda snimite kao html file i onda iz njega kopirate u scriptu.

6.2 Način rada

Način rada scripte je vrlo jednostavan. Sav tekst koji upišemo u "*Tex Box-u*" šaljemo preko forme, tu operaciju izvodimo kliknuvši na gimb "*Izvrši_PHP_kod*". Evo prikaza koda koji to radi.

Primjer:

```
<form name="form1" method="post" action="">

<textarea name="textfield" rows="15" cols="90">
<?
$textfield = stripslashes($textfield); echo $textfield;
?>
</textarea>

<input type="submit" name="Submit" value="Izvrsi_PHP_kod">
</form>
```

Primjećujete da u parametru *action=" "* ne postoji adresa na koju se šalju podaci iz forme na obradu, to znači da se forma obrađuje na istoj stranici odakle je i poslana. PHP kod unutar forme služi da se podaci koje smo poslali opet ispišu u formi.

Nakon toga text iz "*Tex Box-a*" preuzima PHP, prihvaća ga kao "*string*" i upisuje u eksternu datoteku. Ovaj dio se izvršava sljedećim kodom.

Primjer:

```
<?
$myFile = fopen("data.txt","w");

$textfield = stripslashes($textfield);

fwrite($myFile, "$textfield");

fclose($myFile);
?>
```

Da bi upisali u eksternu datoteku moramo je prije toga otvoriti i nakon upisa zatvoriti. Sa funkcijom "*stripslashes*" obrisali smo beckslasheve koje nam je upisao PHP uz posebne znakove (npr. " - navodnici).

Sljedi izvršavanje tog istog teksta iz forme na način što ga pozivamo iz eksterne datoteke s naredbom "*include*" i to u tablici gdje se automatski izvršava i u "*Tex Box-u*" gdje se izvršava ali se Html kod prikazuje u potpunosti. Evo i tog dijela koda.

Primjer:

```
<?
    include "data.txt"  // kod se izvrsava
?>

<textarea name="textarea_2" rows="15" cols="70" ><?
include "data.txt" ?></textarea>
```

7. Zaključak

Sastavljanjem ovog tutorijala upoznao sam se sa osnovnim značajkama programskog jezika PHP. Uvidio sam njegove mogućnosti i spoznao mogućnosti koje nam pruža pri izradi Web stranica.

8. Literatura

[1] Webmajstori – Internet stranica (link: http://www.webmajstori.net)
[2] PHP.co.ba – Internet stranica (link: http://www.php.co.ba)