PROGRAMIRANJE VEB APLIKACIJA

Nastavna jedinica: Predavanje 3

- Uvod u PHP
- Promenljive, opseg, operatori, petlje
- Funkcije
- $_POST, $_GET, $_REQUEST

PHP

- PHP – HyperText Preprocessor
- Personal Home Page
- Ima široku upotrebu, skript jezik otvorenog koda (Open Source)
- Izvršava se na serverskoj strani
- Besplatan je
- Jednostavan za korišćenje (C-like)
- Napredne mogućnosti za profesionalne programere
- Početna verzija napravljena 1994
- Trenutna verzija je 7.3.x

KAKO IZGLEDA PHP DATOTEKA

- Može da sadrži tekst, HTML, Java Script i PHP kod
- PHP se izvršava na serverskoj strani, a rezultat se vraća veb čitaču u HTML obliku
- Podrazumevana ekstenzija je .php

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>PVA - Predavanje 3</title>
</head>
<body>
<?php
echo "Moja prva PHP Skripta";
?>
</body>
</html>
```

ŠTA MOŽE PHP

- Generiše dinamičke veb stranice
- Kreira, otvara, čita i piše datoteke na serveru
- Prikuplja podatke
- Šalje i prima „kolačiće“ (cookies)
- Upisuje, menja ili briše podatke iz baze podataka
- Ograničava prava pristupa pojedinim veb stranicama
- Šifruje podatke
- Izlaz (odgovor servera) ne mora da bude u HTML formatu. Može biti PDF, XML, XHTML, tekst.....

ZAŠTO PHP

- Pokreće se na svim operativnim sistemima
- Kompatibilan je sa skoro svim serverskim programima
- Ima podršku za različite baze podataka
- Besplatan je
- Efikasno se izvršava na serverskoj strani
- Ima ugrađene biblioteke za sve što je potrebno
- Lak za učenje
- Od verzije 7 dobra podrška za OOP
- Izvorni kod dostupan svima

ŠTA JE POTREBNO

- Pronaći veb hosting koji podržava PHP i MySQL ili
- Instalirati jedan od paketa koji sadrži sve softvere (xampp, wampp)
- Kreirati .php datoteku u korenom (root) direktorijumu gde će je PHP interpreter automatski prevesti
  - XAMPP - htdocs
  - WAMPP - www
- Nije potrebno dodatno kompajlijanje ili instalacija
- Kreirana .php stranica se poziva iz veb čitača (Microsoft Edge, Google Chrome, ...)

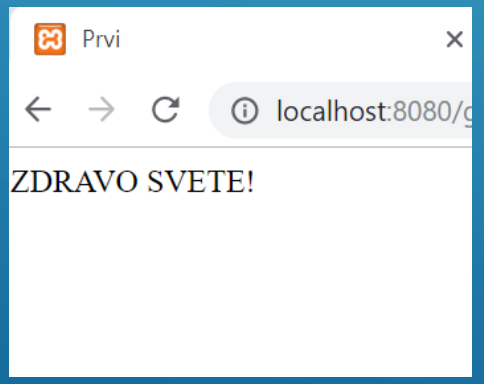
OSNOVNA PHP SINTAKSA

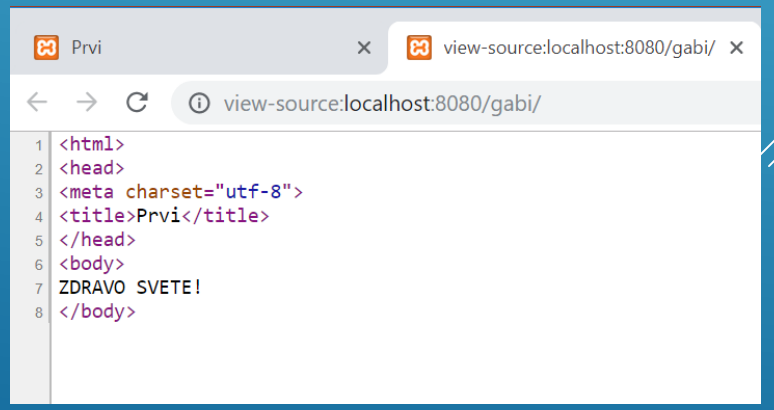
- PHP interpreter će se uključiti ukoliko stranica ima ekstenziju .php i ukoliko u okviru te stranice postoji PHP tag. Prevodi se samo kod napisan u PHP tagu

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>PVA - Predavanje 3</title>
</head>
<body>
<?php
    //Ovde ide PHP kôd
?>
</body>
</html>
```

- Sledeći kod vraća veb čitaču „Zdravo, svete!!!“

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>PVA - Predavanje 3</title>
</head>
<body>
<?php
    echo "Zdravo, svete!!!";
?>
</body>
</html>
```





SINTAKSA

- PHP skript može da bude postavljen bilo gde u dokumentu.
- PHP skript staruje sa <? a završava sa ?>
- Svaka linija koda se završava simbolom ;
- Imena varijabli su case-sensative
- Ključne reč (if, else, while, echo...) klase, funkcije nisu case - sensative
- Komentari u jednoj ili više linija
  - // - oznaka za jedan red komentara
  - /*.....*/ - oznaka za komentar u više redova

```php
<?php
// Možete da koristite komentare da izostavite delove kodne linije
$x = 5 /* + 15 */ + 5;
echo $x;
?>
```

ŠTAMPANJE

- Dve osnovne komande echo i print. Manje više rade isto – ispis podataka na izlaz.
  - echo – nema povratnu vrednost, može da primi više parametara, brži
  - print – vraća vrednost 1 koja može da se iskoristi u izrazima, prima samo jedan argumenat, sporiji.

```
<?php
    echo "zdravo";
    echo "zdravo", "pozdrav";
    echo ("zdravo");
    print "zdravo";
    print ("zdravo");
?>
```

PROMENLJIVE

- Promenljive su „skladišta“ za čuvanje informacija
- Imena promenljivih moraju početi simbolom $, a nakon toga mora biti slovo ili donja crta
- Ime promenljive može sadržati slova engleske abecede, brojeve i donju crtu
- PHP je osetljiv na veličinu slova (Case Sensitive). Promenljiva $x **nije ista** što i $X
- PHP nema komandu za deklarisanje varjabli. Varijabla se kreira u momentu dodele vrednosti.

```php
<?php
    $x=5;
    $y=3;
    $z=$x+$y;
    echo $z;
?>
```

```php
<?php
$txt = "Zdravo svete!";
$x = 5;
$y = 10.5;
?>
```

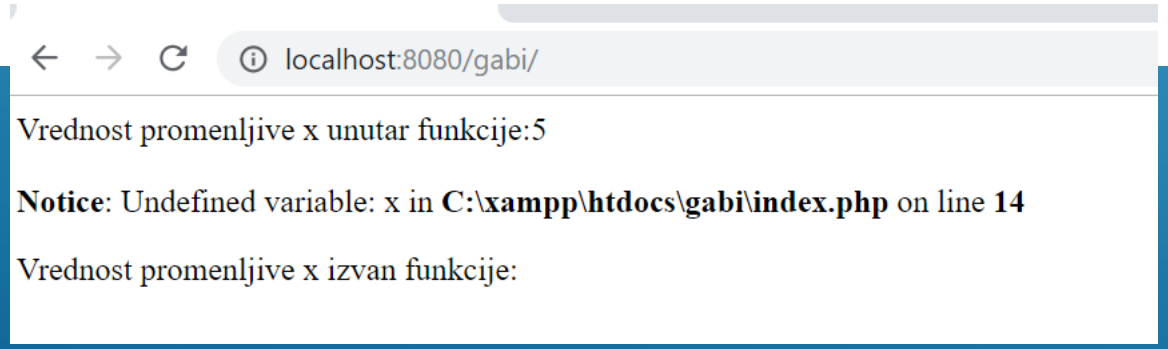
KREIRANJE I OPSEG PROMENLJIVIH

- PHP je **slabo tipiziran jezik**
- Promenljivoj pre dodele vrednosti nije potrebno fiksirati tip podatka. Jedna promenljiva može imati vrednosti različitih tipova podataka u okviru jednog skripta.
- U PHP 7, dodata deklaracija tipova promenljivih.
- Opseg vidljivosti je deo skripta u kome promenljiva može biti korišćena
  - Lokalni
  - Globalni
  - Statički
  - Parametarski

LOKALNI OPSEG

- Promenljive definisane u okviru jedne funkcije su vidljive samo u toj funkciji
- U različitim funkcijama različite promenljive mogu imati isto ime

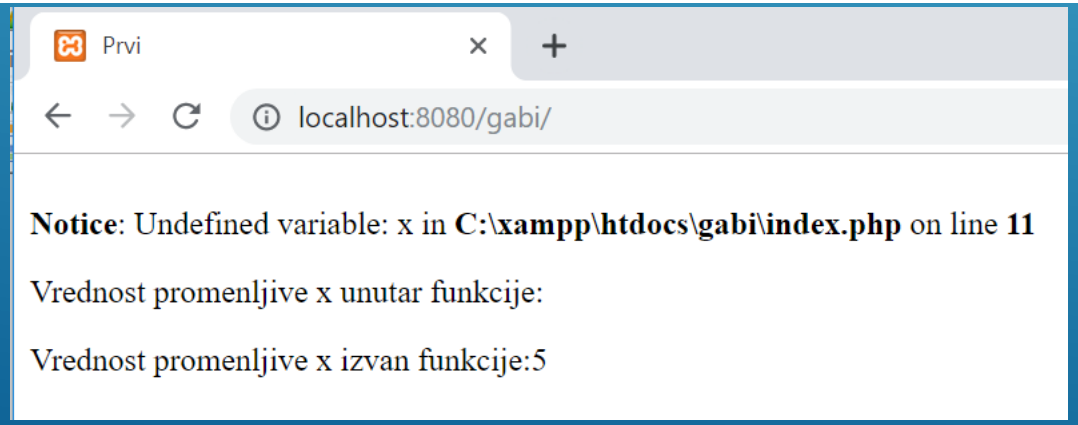
```php
<?php
function test() {
    $x = 5; // lokalni opseg varijable
    echo "<p>Varijabla x unutar funkcije je: $x</p>";
}
test();
// upotreba varijable x izvan funkcije
echo "<p> Varijabla x izvan funkcije je: $x</p>";
function test1(){
     $x="Zdravo";
     echo "<p>Ime varijable x u funkciji test2 kao i funkciji test1,ali vrednost je:$x</p>";
}
test1();
?>
```



GLOBALNI OPSEG

- Promenljive definisane van funkcija
- Može im se pristupiti svuda osim u funkciji
- Ako želimo da ih koristimo u funkciji moramo koristiti ključnu reč **global**

```php
<?php
$x = 5; //globalni opseg
function test() {
   echo "<p>Varijabla x unutar funkcije je: $x</p>";
}
test();
echo "<p>Varijabla x izvan funkcije je: $x</p>";
?>
```




```php
<?php
    $x=5;//globalni opseg
    $y=6;//globalni opseg
    function funkcija1()
    {
        global $x, $y;
        $y=$x+$y;
    }
    funkcija1();
    echo $y;//Izlaz je 11
?>
```

GLOBALNI OPSEG

- PHP takođe može da čuva globalne promenljive u nizu koji se zove $GLOBALS [indeks].
- Indeks sadrži ime promenljive. Ovaj niz je također dostupan iz funkcija i može se direktno koristiti za ažuriranje globalnih varijabli.

```php
<?php
$x = 5;
$y = 10;
function myTest() {
    $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

myTest();
echo $y; // izlaz je 15
?>
```

STATIČKI OPSEG

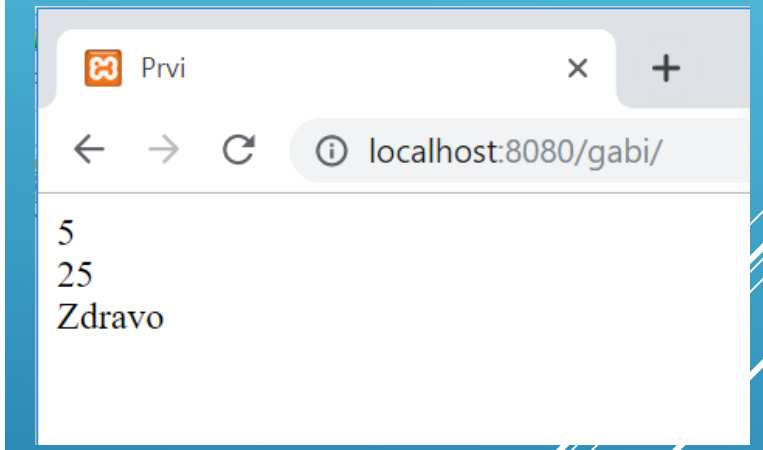
- Kada se funkcija izvrši, sve se njene varijable brišu. Međutim, ponekad želimo da se **lokalna** varijabla **NE briše**. Potreban nam je za dalji posao.
- Ako ne želimo da ih uništimo moramo koristiti ključnu reč **static**

```
<?php
    function funkcija1()
    {
        static $x=5;
        echo $x;
        $x++;
    }
    funkcija1();//Izlaz je 5
    funkcija1();//Izlaz je 6
    funkcija1();//Izlaz je 7
?>
```

PARAMETARSKI OPSEG

- Parametarske promenljive su promenljive koje se prosleđuju kao parametar funkciji (često ih zovemo argumentima funkcije)

```php
<?php
    function funkcija1($x)
    {
        echo $x;
    }
    funkcija1(5);//Izlaz je 5
    funkcija1(25);//Izlaz je 25
    funkcija1(„Zdravo“);//Izlaz je Zdravo
?>
```



PROMENLJIVE TIPA STRING

- Sadrže karaktere i tekst u njima se nalazi između navodnika (jednostrukih ili dvostrukih)
- Operator nadovezivanje (konkatenacije) je simbol „.“

```php
<?php
    $txt1="Dobar dan!";
    $txt2= "Divan dan danas.";
    echo $txt1;//Izlaz je: Dobar dan
    echo $txt2;//Izlaz je: Divan dan danas
    echo $txt1." ".$txt2;//Izlaz je: Dobar dan! Divan dan danas.
?>
```

ZNACI NAVODA U STRINGU

- Ako podatak tipa string počinje dvostrukim znacima navoda, mora se završiti sa tim znacima navoda
- Unutar takvog stringa slobodno se mogu stavljati jednostruki znaci navoda
- Važi i obrnuto

```php
<?php
    $txt1="Predmet 'PVA' je super!!!!";
    $txt2= 'Predmet "PVA" je super!!!!';
    echo $txt1;//Izlaz je: Predmet 'PVA' je super!!!!
    echo $txt2;//Izlaz je: Predmet "PVA" je super!!!!
?>
```

- Razlika je što, u okviru dvostrukih znakova navoda, možemo ubacivati promenljive, a u okviru jednostrukih ne

```php
<?php
    $txt1="Predmet 'PVA' je super!!!!";
    $txt2= 'Predmet "PVA" je super!!!!';
    echo "Vrednost prvog stringa je $txt1";//Izlaz je: Vrednost prvog stringa je Predmet 'PVA' je super!!!!
    echo 'Vrednost drugog stringa je $txt2';//Izlaz je: Vrednost drugog stringa je $txt2
?>
```

STRLEN()

- Funkcija ***strlen()*** vraća broj karaktera u stringu (dužinu stringa).
- Ulazni parametar funkcije je string čiju dužinu tražimo.
- Povratna vrednost je podatak tipa integer

```php
<?php
    $ime=„Sinisa";
    echo strlen($ime);//Izlaz je: 6
?>
```

```php
<?php
    $ime=„Siniša";
    echo strlen($ime);//Izlaz je: ?
?>
```

STRLEN()

- strlen(„Siniša“), ma kako to neverovatno zvučalo, neće vratiti 6, nego 7.
- Razlog za to je karakter „š“ koji je utf-8.
- Ima dva načina za rešenje problema sa enkodingom:
  - *mb_strlen() – multibyte strlen*
  - *utf8_decode()*

```
<?php
    echo mb_strlen(„Siniša");//Izlaz je: 6
    echo strlen(utf8_decode(„Siniša"));//Izlaz je: 6
?>
```

STRPOS()

- Funkcija ***strpos()*** traži određeni karakter ili string unutar stringa.
- Ima dva ulazna parametra.
- Prvi je string u kome tražimo (haystack), drugi je string/karakter koji tražimo (needle)
- Ako je string nađen, povratna vrednost je pozicija nađenog stringa (pozicije počinju od 0), ako nije nađen vraća se boolean **FALSE**

```
<?php
    echo strpos("Dobar dan", "dan");//Izlaz je: 6
?>
```

PHP OPERATORI

- Operatori se koriste za obavljanje operacija nad promenljivima i vrednostima.
- PHP operatore deli na sledeće grupe:
  - Aritmetički operatori
  - Operatori za dodelu
  - Operatori za poređenje
  - Operatori povećanja / smanjenja
  - Logični operatori
  - String operatori
  - Operatori niza
  - Operatori uslovne dodele

ARITMETIČKI OPERATORI

Operator	Ime	Opis	Primer	Rezultat
x+y	Sabiranje	Zbir	2+2	4
x-y	Oduzimanje	Razlika	5-2	3
x*y	Množenje	Proizvod	5*2	10
x/y	Deljenje	Celobrojno deljenje	15/5	3
x%y	Moduo	Ostatak pri deljenju	5%2	1
-x	Negacija	Negacija	-2	
a.b	Nadovezivanje	Nadovezivanje	„le“.„po“	„lepo“

DODELA VREDNOSTI

- Operatori za dodelu koriste se za upis vrednosti u promenjivu.
- Osnovni operater dodele u PHP-u je "=".
- To znači da se levi operand postavlja na vrednost izraza koji je s desne strane.

Dodela	Isto što i....	Opis
x=y	x=y	Dodela vrednosti levom operandu
x+=y	x=x+y	Sabiranje
x-=y	x=x-y	Oduzimanje
x*=y	x=x*y	Množenje
x/=y	x=x/y	Deljenje
x%=y	x=x%y	Moduo
a.=b	a=a.b	Nadovezivanje

OPERATOR DODELE

- Osnovni operator dodele „=“
- Čita se kao „dobija vrednost“ (npr. $x=3;//Promenljiva x dobija vrednost 3)
- Za sve operatore važi sledeće pravilo:
  vrednost celog izraza dodele je vrednost koja je dodeljena operandu na levoj strani npr. $x=5+($a=6);//Promeljiva x dobija vrednost 11
- Zagrade se upotrebljavaju zbog prioriteta izvršavanja operacija

OPERATORI INKREMENTIRANJA/DEKREMENTIRANJA

Operator	Ime	Opis
++x	Preinkrement	Povećanje x za 1 pa tek onda vraćanje vrednosti x
x++	Postinkrement	Vraćanje vrednosti x pa tek onda povećanje x za 1
--x	Predekrement	Smanjenje x za 1 pa tek onda vraćanje vrednosti x
x--	Postdekrement	Vraćanje vrednosti x pa tek onda smanjenje x za 1

OPERATORI POREĐENJA

Operator	Ime	Opis	Primer
x==y	jednakost	True ako su jednaki	5==8 vraća false
x===y	identičnost	True ako su jednaki i ako su istog tipa	5===„5“ vraća false
x!=y	nejednakost	True ako su različiti	5!=8 vraća true
x<>y	nejednakost	True ako su različiti	5<>8 vraća true
x!==y	neidentičnost	True ako su različiti ili ako nisu istog tipa	5!==„5“ vraća true
x>y	veće od	True ako je x veće od y	5>8 vraća false
x<y	manje od	True ako je x manje od y	5<8 vraća true
x>=y	Veće ili jednako od	True ako je x veće ili jednako od y	5>=8 vraća false
x<=y	Manje ili jednako od	True ako je x manje ili jednako od y	5<=8 vraća true

LOGIČKI OPERATORI

Operator	Ime	Opis
x and y	And	True ako su i x i y tačni
x or y	Or	True ako je barem jedan tačan
x xor y	Xor	True ako je samo jedan tačan
x && y	And	True ako su i x i y tačni
x || y	Or	True ako je barem jedan tačan
! x	Not	True ako je x netačan

OPERATORI ZA NIZOVE

Operator	Ime	Primer	Rezultat
+	unija	$x + $y	Unija $x i $y
==	jednakost	$x == $y	Vraća true ako $x i $y imaju iste parove kluč/vrednost
===	identičnost	$x === $y	Vraća true ako $x i $y imaju iste parove ključ/vrednost u istom redosledu i istog tipa
!=	nejednakost	$x != $y	Vraća true ako $x nije jednak $y
<>	nejednakost	$x <> $y	Vraća true ako $x nije jednak $y
!==	neidentičnost	$x !== $y	Vraća true ako $x nije identičan $y

DODATNI OPERATORI

- Ternarni operator

  Operatori za uslovno dodeljevanje koriste se za postavljanje vrednosti u zavisnosti od uslova:

  uslov ? uslov_true : uslov_false

  ($ocena > 5 ? „polozio“ : „pao“);

```php
<?php
  echo $ocena > 5 ? "polozio" : "pao" ."<br>" ;

  if ($ocena > 5)
  { echo "polozio";
  }
   else
   {
      echo "pao";
  }

?>
```

- Operator zanemarivanja greške

  $a = @(27/0)

  Bez operatora @, izvršno okruženje bi generisalo upozorenje „deljenje nulom“

FUNKCIJA VAR_DUMP()

- Za razliku od funkcija echo i print pomoću kojih smo štampali podatke ukoliko podatak ima složenu strukturu (niz, objekat ....) moramo koristiti var_dump() funkciju
- var_dump () funkcija vraća tip i vrednost podataka
- Funkcija pokazuje struktuirane informacije o jednom ili više izraza koje obuhvataju njegov tip i vrednost.

```php
<?php
    $a = array(1, 2, array("a", "b", "c"));
    var_dump($a);
?>
```

```
D:\services\www\PVA\predavanja\p3\index.php:10:
array (size=3)
  0 => int 1
  1 => int 2
  2 =>
    array (size=3)
      0 => string 'a' (length=1)
      1 => string 'b' (length=1)
      2 => string 'c' (length=1)
```

FUNKCIJE ZA RAD SA PROMENLJIVAMA

- **string** gettype(promenljiva) -**utvrđuje tip promenljive**
- **int** settype(promenljiva,tip) -**menja tip promenljive**
- **boolean** isset(promenljiva) - **true ako je promenljiva definisana**
- **void** unset(promenljiva) - **poništava definiciju promenljive**
- **boolean** empty(var) **-ispituje da li postoji promenljiva i da li ima vrednost koja nije nula**
- is_array
- is_double, is_float, is_real
- is_long, is_int, is_integer
- is_string
- is_object
- is_null()
- is_scalar()
- is_numeric()

USLOVNI ISKAZI - IF

- Struktura

  if*(uslov)*{*blok_naredbi*}

- Ako je *uslov* ispunjen izvršiće se *blok_naredbi*

```
<?php
    $x=5;
    if($x>3)
    {
        echo "Uslov je ispunjen";
    }
?>
```

- Ukoliko se *blok_naredbi* sastoji od samo jedne instrukcije, vitičaste zagrade nisu potrebne

IF....ELSE STRUKTURA

- Struktura

  if*(uslov)*

  {blok_naredbi1}

  else

  {blok_naredbi2}

- Ako je *uslov* ispunjen izvršiće se *blok_naredbi1* , a ako nije izvršiće se *blok_naredbi2*

```php
<?php
    $x=5;
    if($x>3)
    {
        echo "Uslov je ispunjen";
    }
    else
    {
        echo "Uslov nije ispunjen";
    }
?>
```

IF….ELSEIF…ELSE STRUKTURA

- Struktura

  if (uslov1) {

  blok_naredbi1 će se izvršiti ako je uslov1 ispunjen;

  } elseif (uslov2) {

  blok_naredbi2 će se izvršiti ako uslov1 nije ispunjen i uslov2 je ispunjen;

  } else {

  blok_naredbi3 će se izvršiti ako oba uslova (uslov1 i uslov2) nisu ispunjeni;

  }

- Izvršava različiti blok naredbi za više od dva uslova

```php
<?php
$t = date("H");
if ($t < "10") {
 echo „Dobro jutro!";
} elseif ($t < "20") {
    echo „Dobar dan!";
} else {
   echo „Laku noć!";
}
?>
```

SWITCH

- Struktura

```
switch(promenljiva)
{
    case vrednost1:
        blok_naredbi1;
        break;
    case vrednost2:
        blok_naredbi2;
        break;
    ....
    default:
        blok_naredbin;
}
```

```php
<?php
$omiljenaBoja =  "purple";

switch ($omiljenaBoja) {
    case "red":
        echo  "Omiljena boja je crvena!";
        break;
    case "blue":
        echo "Omiljena boja je plava!";
        break;
    case  "purple" :
        echo  "Omiljena boja je ljubičasta!";
        break;
    default:
        echo  "Nemate omiljenu boju!";
}
?>
```

- U zavisnosti koju *vrednost* ima *promenljiva* izvršiće se odgovarajući *blok_naredbi*

PETLJE

- **WHILE** – prolazak kroz blok koda dok je određeni uslov zadovoljen
- **DO...WHILE** – prolazak barem jednom kroz blok, a onda ponavljanje dok je uslov zadovoljen
- **FOR** – prolazak kroz blok zadati broj puta
- **FOREACH** – prolazak kroz blok za svaki elemenat niza

WHILE PETLJA

- Poznata i kao petlja sa izlazom na vrhu
- Nema svoj brojač prolazaka kroz petlju, mora se dodeliti programski i povećavati za svaki prolazak

```php
<?php
    $i=1;
    while($i<=5)
    {
        echo "Broj je " . $i . "<br>";
        $i++;
    }
?>
```

DO...WHILE PETLJA

- Poznata i kao petlja sa izlazom na dnu
- Nema svoj brojač prolazaka kroz petlju, mora se dodeliti programski i povećavati za svaki prolazak

```
<?php
     $i=1;
     do
     {
          $i++;
          echo "Broj je " . $i .
"<br>";
     }
     while ($i<=5);
?>
```

FOR PETLJA

- Petlja sa poznatim brojem prolazaka
- Ima svoj brojač prolazaka kroz petlju

```php
<?php
    for ($i=1; $i<=5; $i++)
    {
        echo „Broj je " . $i . "<br>";
    }
?>
```

FOREACH PETLJA

- Petlja koja prolazi kroz sve elemente niza, najčešće se koristi za asocijativne nizove gde ne možemo koristiti FOR petlju
- Nema svoj brojač prolazaka kroz petlju, mora se dodeliti programski i povećavati za svaki prolazak

```php
<?php
    $x=array("one", "two", "three");
    foreach ($x as $value)
    {
        echo $value . "<br>";
    }
?>
```

NIZOVI

- *Niz (array)* je promenljiva određenog imena koja sadrži skup vrednosti u nekom redosledu
- Vrednosti smeštene u nizu nazivaju se *elementi*
- Redni brojevi elemenata se nazivaju *indeksi* ili *ključevi*
- Jedan niz može imati više elemenata, a svaki element po jednu vrednost (tekst, broj, drugi niz)

NIZOVI

- Niz sadrži više promenljivih u jednoj koja je ime niza
- Kreira se naredbom array();

```
<?php
    $cars=array("Volvo", "BMW", "Toyota");
    echo "Ja volim" . $cars[0] . ", " . $cars[1] . " and " . $cars[2] .".";
?>
```

- Niz je efikasniji način za čuvanje podataka od pojedinačnih dodela:

TIPOVI NIZOVA

- Indeksirani nizovi
  - Nizovi sa numeričkim indeksima
- Asocijativni nizovi
  - Nizovi sa imenovanim ključevima kao indeksima
- Višedimenzionalni nizovi
  - Nizovi sastavljeni od više nizova

PREBROJAVANJE ELEMENATA U NIZU

- Funkcija *count()* vraća ukupan broj elemenata u nizu koji joj je prosleđen
- Funkcija *sizeof()* vraća ukupan broj elemenata u nizu koji joj je prosleđen

Napomena: razlika između *count()* i *sizeof()* je što count() vraća vrednost poslednjeg numeričkog indeksa kod numeričkih nizova, a sizeof() realnu veličinu niza

- Funkcija *array_count_values($niz)* broji koliko jedinstvenih vrednosti ima u nizu;
  - funkcija vraća asocijativni niz koji sadrži tabelu učestanosti (ključ je element niza, vrednost je broj ponavljanja)

INDEKSIRANI NIZOVI

- PHP: indeksi uvek počinju sa nulom
- Inicijalizovanje: niz se pravi jezičkom konstrukcijom *array*

```
$lekovi=array('Andol', 'Brufen', 'VitaminC');
```

- kopiranje jednog niza u drugi pomoću operatora =
- funkcija *range()* automatski pravi niz rastućih brojeva

```
$brojevi = range(1,10);
$neparni = range(1,10,2); //2 je korak za koji se povećava
$slova = range ('a', 'z');
```

PRISTUPANJE SADRŽAJU NIZA

- Sadržaju niza se pristupa pomoću imena promenljive i indeksa (ključa), koji se navodi u uglastim zagradama

```
$x=$cars[0];
```

- Isti sistem numerisanja kao u programskim jezicima C, C++, Java,... (indeks prvog člana u nizu je nula!)
- Sadržaj elementa niza menja se pomoću operatora =

```
$cars[0]='Ferari';
```

- Može se dodati i novi element

```
$cars[3]='Pežo';
```

PROLAZAK KROZ INDEKSIRANI NIZ

```php
<?php
    $cars=array("Volvo","BMW","Toyota");
    $arrlength=count($cars);
    for($x=0;$x<$arrlength;$x++)
    {
        echo $cars[$x];
        echo "<br>";
    }
?>
```

ASOCIJATIVNI NIZOVI

- Asocijativni nizovi su određeni parovima ključ=>vrednost

```php
<?php
    $age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
    echo $age['Peter'];//Izlaz je: 35
    echo $age['Ben'];//Izlaz je: 37
    echo $age['Joe'];//Izlaz je: 43
    echo "Peter ima " . $age['Peter'] . " godina.";//Izlaz je: Peter ima 35 godina
?>
```

PROLAZAK KROZ ASOCIJATIVNI NIZ

```php
<?php
    $age=array("Peter"=>"35","Ben"=>"37","Joe"=>"43");
    foreach($age as $kljuc=>$vrednost)
    {
        echo "Ključ = " . $kljuc . ", Vrednost = " . $vrednost;
        echo "<br>";
    }
?>
```

FUNKCIJE ZA SORTIRANJE NIZA

- **sort()** – sortira niz u rastućem poretku
- **rsort()** – sortira niz u opadajućem poretku
- **asort()** – sortira asocijativni niz u rastućem poretku po vrednostima
- **ksort()** - sortira asocijativni niz u rastućem poretku po ključevima
- **arsort()** - sortira asocijativni niz u opadajućem poretku po vrednostima
- **krsort()** - sortira asocijativni niz u opadajućem poretku po ključevima

PROMENA REDOSLEDA ELEMENATA NIZA

- Funkcija **shuffle()** nasumično menja redosled elemenata u nizu
- Funkcija **array_reverse()** pravi kopiju niza sa elementima u obrnutom redosledu
- Funkcija **array_push()** dodaje po jedan novi element na kraj niza
- Funkcija **array_pop()** uklanja poslednji element niza i vraća ga pozivajućem kodu

NAVIGACIJA UNUTAR NIZA

- Kada napravimo novi niz, pokazivač upućuje na prvi element niza
- Rezultat **current($niz)** je prvi element
- Funkcija **next($niz)** prvo pomera pokazivač unapred, a zatim vraća nov tekući element
- Funkcija **reset($niz)** vraća pokazivač na prvi element u nizu
- Funkcija **end($niz)** pomera pokazivač na kraj niza
- Funkcija **prev($niz)** pomera pokazivač unatrag za jedno mesto, a zatim vraća nov tekući element

ЗНАКОВНИ НИЗОВИ – СКРАЋИВАЊЕ ЗНАКОВНИХ ВРЕДНОСТИ

- Funkcija **trim()** briše beline s početka i kraja znakovnog podatka.
- Po definiciji briše znakove za povratak na početak reda(\r) i za prelazak u novi red (\n), horizontalne i vertikalne tabulatore (\t, \x0B), znakove za kraj znakovne vrednosti (\0) i razmake.
- Ovoj funkciji možete da prosledite i parametar koji sadrži eksplicitno zadat spisak znakova koje treba ukloniti umesto ovih navedenih

FORMATIRANJE ZNAKOVNIH VREDNOSTI U POGODAN OBLIK ZA PRIKAZIVANJE

- Funkcija **nl2br()** prihvata znakovnu vrednost kao ulazni parametar i sve znake za prelazak u novi red u njoj zamenjuje XHTML oznakama
 ili HTML oznakom
 (ovo se koristi za formatiranje teksta)
- Funkcija **printf()** prosleđuje formatiranu znakovnu vrednost čitaču Weba, a funkcija **sprintf()** vraća formatiranu znakovnu vrednost

PRETVARANJE MALIH I VELIKIH SLOVA

Upotreba	Opis	Vrednost
$ime		Programiranje na Vebu
strtoupper($ime)	sva slova menja u velika	PROGRAMIRANJE NA VEBU
strtolower($ime)	sva slova menja u mala	programiranje na vebu
ucfirst($ime)	menja u veliko slovo prvi znak ulaznog argumenta, ako je alfabetski	Programiranje na Vebu
ucwords($ime)	menja u veliko slovo prvi znak svake reči koja počinje alfabetskim znakom	Programiranje Na Vebu

SPAJANJE I RAZDVAJANJE ZNAKOVNIH VREDNOSTI

- Funkcija **explode(string granicnik, string text)** deli znakovnu vrednost text na mestima gde naiđe na znakovnu vrednost - graničnik. Funkcija vraća niz.
- Suprotne ovoj funkciji su **implode()** i **join()**

```php
<?php
    $email=„laki.lakicevic@gs.viser.edu.rs";
    $email_niz=explode("@", $email);
    echo $email_niz[0];//Izlaz je: laki.lakicevic
    echo $email_niz[1];//Izlaz je: gs.viser.edu.rs
?>
```

FUNKCIJE IMPLODE I JOIN

- Služe za spajanje niza u jedan string sa definisanim graničnikom

```php
<?php
    $tekst_niz=array("Ja", "idem", "u", "skolu", "i", "dobar", "sam", "đak");
    $tekst=implode(" ", $tekst_niz);
    echo $tekst;//Izlaz je: Ja idem u skolu i dobar sam đak
?>
```

KORISNIČKE FUNKCIJE

- Funkcija je samostalan modul koji propisuje način pozivanja funkcije, obavlja zadatak i eventualno vraća rezultat.
- Pozivanje funkcija:

```
moja_funkcija();
```

- Pozivanje nedefinisanih funkcija => greška!
- Nema razlike između malih i velikih slova kod funkcija u PHP (Funk, FUNK, funk... je ista funkcija)

OSNOVNA STRUKTURA FUNKCIJE

- Deklaracija počinje sa **function**, iza koje slede ime funkcije i parametri, a zatim i kod koji se izvršava kada je funkcija pozvana
- Imena funkcija:
  - Nova funkcija ne sme da ima isto ime kao neka postojeća
  - Ime funkcije može da bude sastavljeno od slova, cifara i donje crte _
  - Ime funkcije ne može počinjati cifrom
  - PHP ne dozvoljava preklapanje ugrađenih funkcija!!
- Može da postoji jedan ili više parametara, ali ne moraju svi biti obavezni
- Funkcija se NEĆE izvršiti ukoliko eksplicitno ne pozovemo njeno ime
- Funkcije mogu biti definisane u bilo kom delu programa

PRIMER

```php
<?php
    function imePredmeta()
    {
        echo "Programiranje veb aplikacija";
    }
    echo "Ime predmeta je ";
    imePredmeta();
?>
```

PRENOS PARAMETARA PO VREDNOSTI

- kada funkciji prosledite vrednost parametra, unutar funkcije se pravi nova promenljiva koja sadrži prosleđenu vrednost - kopija originala
- može slobodno da se menja, dok izvorna promenljiva ostaje nepromenjena

```php
<?php
    function dodatak($krajImena)
    {
        echo "Programiranje ".$krajImena;
    }
    dodatak("veb aplikacija");//Izlaz je: Programiranje veb aplikacija
    dodatak("aplikacija baza podataka");//Izlaz je: Programiranje aplikacija baza podataka
?>
```

PRENOS PARAMETARA PO VREDNOSTI

- Neki parametri koje prosleđujemo funkciji ne moraju biti obavezni
- To znači da za te parametre postoji predefinisana vrednost u okviru same funkcije

```php
<?php
    function ime($ime, $prezime="Nije definisano")
    {
        echo "$ime $prezime";
    }
    ime("Boško", "Bogojević");//Izlaz je: Boško Bogojević
    ime("Boško");//Izlaz je: Boško Nije definisano
?>
```

- Ukoliko postoje neobavezni parametri, mora se voditi računa o redosledu parametara

POVRATNA VREDNOST FUNKCIJE - RETURN

- Izvršenje funkcije se prekida naredbom **return()**
- Kao parametar naredbe **return()** može biti vrednost koju ćemo vratiti glavnom programu

```php
<?php
    function add($x,$y)
    {
        $total=$x+$y;
        return $total;
    }
    $zbir=add(1,16);
    echo "Zbir je: " . $zbir;
?>
```

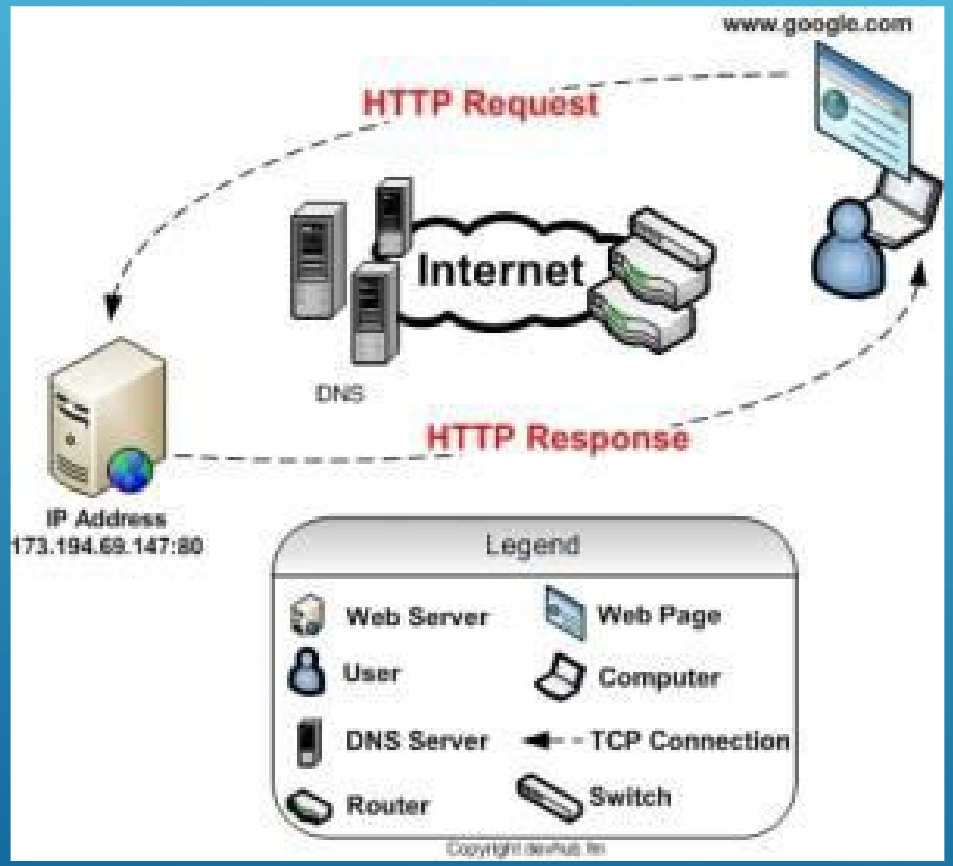
PRIMER FUNKCIJE – PRAVLJENJE TABELE

```php
<?php
    function napravi_tabelu($niz)
    {
        echo '<table border = 1>';
        reset($niz); // Pokazivac na pocetak niza
        $vrednost = current($niz);
        while ($vrednost) {
            echo "<tr><td>$vrednost</td></tr>\n";
            $vrednost = next($niz);
        }
        echo '</table>';
    }
    $moj_niz = array('Prvo polje.','Drugo polje.','Trece polje.');
    napravi_tabelu($moj_niz);
?>
```

HTTP ZAHTEV I ODGOVOR (HTTP REQUEST/RESPONSE) - OSNOVE

- Životni ciklus(za HTTP zahtev/odgovor)
- Anatomija HTTP zahteva i odgovora
- HTTP metodi i njihovo korišćenje

ŽIVOTNI CIKLUS HTTP ZAHTEVA I ODGOVORA



- Korisnik posećuje URL sajta
- Zatim kreira zahtev koji se prosleđuje web serveru putem HTTP protokola
- Web server vraća odgovor u obliku web stranice putem HTTP protokola

ŽIVOTNI CIKLUS HTTP ZAHTEVA I ODGOVORA

- Svaki put kada se klikne na neki link i poseti neka veb stranica, iza scene se odvija prethodna šema.
- HTTP zahtev se može inicirati ne samo preko veb čitača. Može biti realizovan preko TELNET protokola ili preko klijenta pisanog u programskom jeziku recimo Java ili C#.

ANATOMIJA METODA HTTP ZAHTEVA

- Neophodno je veb serveru proslediti informaciju o izboru metoda kojim se određuje vrsta zahteva od strane klijenta.
- Metoda pri HTTP zahtevu sadrži sledeće opcije:

Method = "OPTIONS" | "GET" | "HEAD" | "POST" | "PUT" | "DELETE" | "TRACE" | "CONNECT" | extension-method = token

OPTIONS

- Opcija „OPTIONS“ ukazuje koja HTTP metoda je dostupna klijentu.
- U zavisnosti od toga kako je veb server konfigurisan, klijentu su najčešće dostupne samo metode POST i GET.

GET ZAHTEV

- GET zahtev vraća podatke od veb servera specificirajući parametre putem URL-a. Na primer, u sledećem HTTP zahtevu, traži se stranica index.html pri čemu je specificiran parametar report_id:

http://www.test.com/index.html?raport_id=222333

KADA KORISTITI GET METODU

- U slučaju da je potrebno videti podatke bez njihovog ažuriranja (promene). Ovo u izvesnom smislu odgovara SQL SELECT upitu pri obraćanju bazi podataka.
- U slučaju da zahtev (stranica) treba da bude upamćen i ponovljen (bookmarkable URL)
- U slučaju da je nebitno ako je zahtev više puta ponovljen (recimo u vidu posete nekoj stranici)

KADA NE KORISTITI GET METODU

- Kada se koristi metoda get, podaci preuzeti iz forme, vidljivi su u URL putanji. Ne bi se trebao koristiti kada se prenosi lozinka ili osetljiva informacija
- Prilikom slanja velike količine podataka. Metoda get nije pogodna za veoma velike vrednosti promenljivih (preko 2000 karaktera)
- Kada je potrebno nešto ažurirati na serveru
- Samo ime metode upućuje da se radi o preuzimanju (dovlačenju) podataka

POST ZAHTEV

- Informacije se šalju putem hedera HTTP zahteva

```
POST /login.php HTTP/1.1
Host: www.awebsite.com
User-Agent: Safari/4.0
Content-Length: 27
Content-Type: application/x-www-form-urlencoded
userid=mojId&password=mojaLozinka
```

KADA KORISTITI POST METODU

- Prilikom slanja velike količine podataka
- Prilikom slanja osetljivih podataka kao što su korisnička imena, lozinke, sigurnosni brojevi itd.
- Pri potrebe za promenom podataka na veb serveru
- Samo ime metode upućuje na postavljanje (slanje) podataka

PUT, HEAD, TRACE, DELETE, CONNECT

- Ostale metode postoje u HTTP protokolu i uglavnom se koriste za mikroservise (RESTfull servise)

HTML FORME

- $_POST i $_GET promenljive se koriste za preuzimanje korisničkog unosa iz forme
- Svaki element iz forme je automatski vidljiv za PHP

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>PVA - Predavanje 3</title>
</head>
<body>
<form action="index.php" method="post">
    Ime: <input type="text" name="fname" id="fname"/><br>
    Godina: <input type="text" name="age" id="age"/><br>
    <input type="submit" value="Unesite podatke">
</form>
</body>
</html>
```

ПАРАМЕТРИ HTML ФОРМЕ

- Forma, kao i svi ostali HTML elementi, ima svoje parametre
  - action – na koju stranicu će otići podaci iz forme
  - method – kojom metodom se šalju podaci
  - enctype – nije obavezno ako se šalje tekst, ako se šalje datoteka mora da ima vrednost „**multipart/form-data**“
  - name, id – parametri koji opisuju formu ako ima više od jedne na stranici

PRIHVATANJE ELEMENATA IZ FORME

- Pri kliku na dugme submit podaci iz forme se šalju stanici index.php

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>PVA - Predavanje 3</title>
</head>
<body>
    Dobro došli <?php echo $_POST["fname"]; ?>!<br>
    Imate<?php echo $_POST["age"]; ?> godina.
</body>
</html>
```

VALIDACIJA UNOSA

- Validaciju je poželjno obaviti na klijentskoj strani (JavaScript)
- Validacija je potrebna i na serverskoj strani, pogotovo ako se podaci upisuju u bazu podataka.
- Dobra navika je da serverska validacija ne podrazumeva preusmeravanje na drugu stranicu, već ostajanje na istoj stranici gde se nalazi i forma.

$_GET PROMENLJIVA

- $_GET promenljiva se koristi za uzimanje podataka iz forme kada se koristi metod get (method=„get“)

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>PVA - Predavanje 3</title>
</head>
<body>
<form action="index.php" method="get">
    Ime: <input type="text" name="fname" id="fname"/><br>
    Godina: <input type="text" name="age" id="age"/><br>
    <input type="submit" value="Unesite podatke">
</form>
</body>
</html>
```

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>PVA - Predavanje 3</title>
</head>
<body>
    Dobro došli <?php echo $_GET["fname"]; ?>!<br>
    Imate<?php echo $_GET["age"]; ?> godina.
</body>
</html>
```

$_REQUEST PROMENLJIVA

- $_REQUEST promenljiva služi kao zamena promenljivama $_POST, $_GET, $_COOKIE u PHP skriptu
- Može preuzeti podatke kada se koristi bilo koja metoda (post ili get)

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>PVA - Predavanje 3</title>
</head>
<body>
    Dobro došli <?php echo $_REQUEST["fname"]; ?>!<br>
    Imate<?php echo $_REQUEST["age"]; ?> godina.
</body>
</html>
```