Serverske tehnologije

Web hosting, PHP

Ak. god. 2011/2012

Web hosting

- Web hosting servis je tip Internet hosting servisa koji pojedincima i organizacijama omogućava postavljanje web stranica.
- Web host-ovi tj. web host pružatelji usluga su kompanije koje pružaju svojim korisnicima prostor na web serveru za postavljanje njihovih stranica.
- Često ISP (*Internet Service Provider*) pružatelji usluga nude i web hosting. ISP pružatelji usluga omogućavaju korisnicima spajanje na Internet (HT, VIP, Iskon, Carnet,….).

Web hosting

- Najosnoviji web hosting omogućava postavljanje statičkih web stranica koje se obično na server prebacuju FTP-om.
- Ovakva web hosting usluga je obično dostatna samo za osobne web stranice (*personal web pages*).
- Kompleksni web *site*-ovi zahtijevaju od pružatelja usluga podršku za baze podataka i platforme za razvoja web aplikacija poput PHP-a, ASP.NET-a i Jave.
- Izvor: www.wikipedia.org

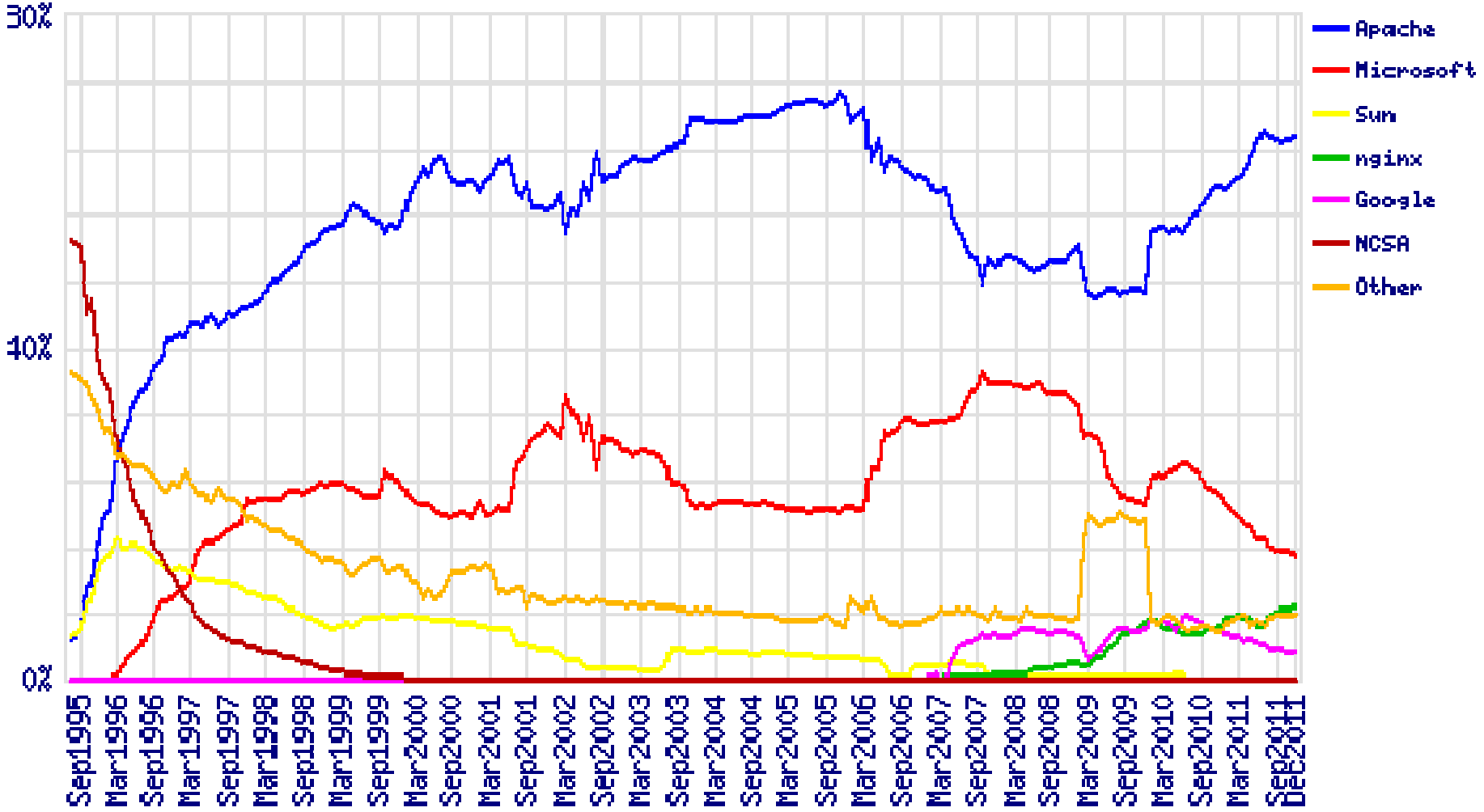
Web hosting

- Često je potrebna i podrška za SSL odnosno noviju verziju SSL-a TLS (*Transport Layer Security*) ako web aplikacija zahtijeva zaštitu podataka (npr. on-line kupovina).
- Znači složeniji web hosting uključuje i dodatne funkcionalnosti pored postavljanja web stranica.
- Često se klijentu omogućava pristup i administriranje svih tih funkcionalnosti (npr. dodavanje baze, dodavanje poddomena i sl.) preko Web sučelja.

Web hosting

- Odabir web hostinga ovisi o zahtjevima klijenta (baza, operacijski sustav, aplikacijski softver).
- Češće se nudi Linux-based web hosting sa tipičnom tzv. LAMP platformom **L**inux, **A**pache, **M**ySQL i **P**HP/**P**erl/**P**ython.
- Na Windows os-u imamo WAMP (Windows, **A**pache, **M**ySQL i **P**HP/**P**erl/**P**ython).
- Neki hrvatski web hosting pružatelji usluga:
  www.orbis.hr
  www.avalon.hr
  www.inside.hr
  www.posluh.hr

Web hosting



Izvor: http://news.netcraft.com/archives/web_server_survey.html

Web hosting

- **Koliko prostora na disku trebate?**
- Mali i srednje veliki *site* treba između 10 i 100MB diskovnog prostora.
- Veličina obične HTML stranice može biti i ispod 1KB, ali je veličina slika koje se nalaze gotovo na svakoj stranici znatno veća.
- Stoga HTML stranica zauzima između 5 i 50KB, ovisno o korištenju slika i drugih elemenata (video, zvuk, i sl.) koji zauzimaju diskovni prostor.
- Izvor: http://www.w3schools.com/hosting/host_capacity.asp

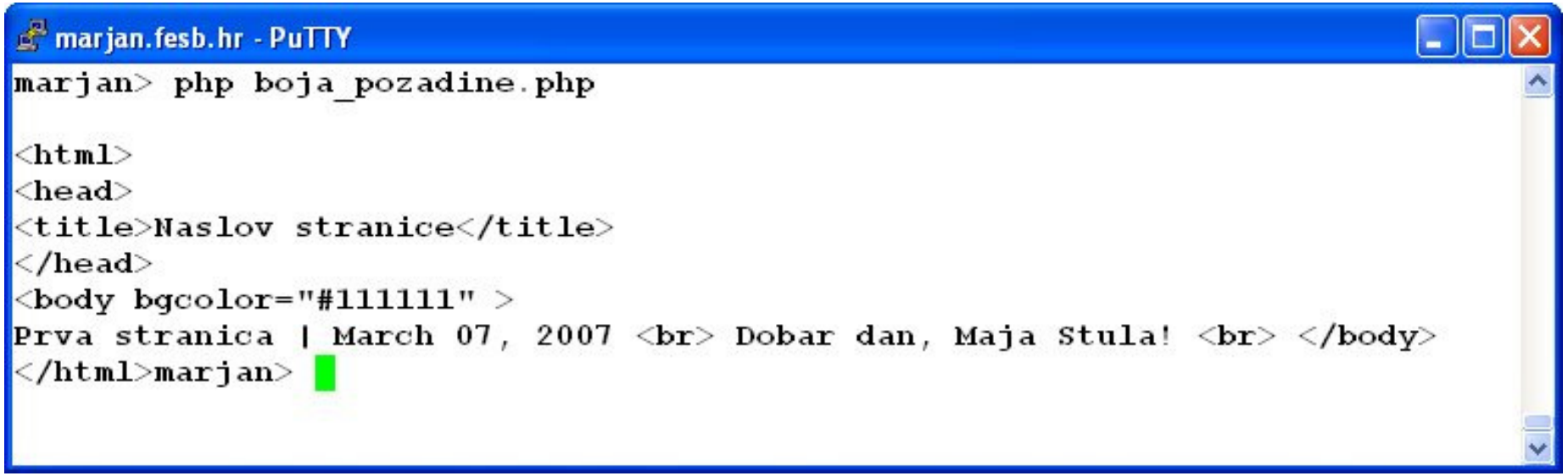
Web hosting

- **Koliki mjesečni promet trebate zakupiti?**
- Mali i srednje veliki *site* ima mjesečni promet od 1GB do 5GB.
- Promet se može dobiti množenjem prosječne veličine stranice s očekivanim brojem pristupa stranicama. Npr. ako je prosječna veličina stranice 30KB i očekivani mjesečni broj posjeta je 50,000 promet će biti 0.03MB x 50,000 = 1.5GB.

- Pri izboru web hostinga treba voditi računa o:
  - Pored već navedenog (izbor platforme)
  - O ograničenju mjesečnog prometa
  - O troškovima prekoračenja mjesečnog prometa i diskovnog prostora
  - Podršci ….

PHP

- PHP je skriptni jezik koji se uglavnom koristi na web serveru za generiranje dinamičkih web stranica, mada se PHP interpreter može koristiti i u ljuski (*shellu*).
- Putem ljuske se ostvaruje interakcija korisnika s kernelom *Unix-like* (Unix sličnih) operacijskih sustava.
- Poziv php interpretera u shellu:

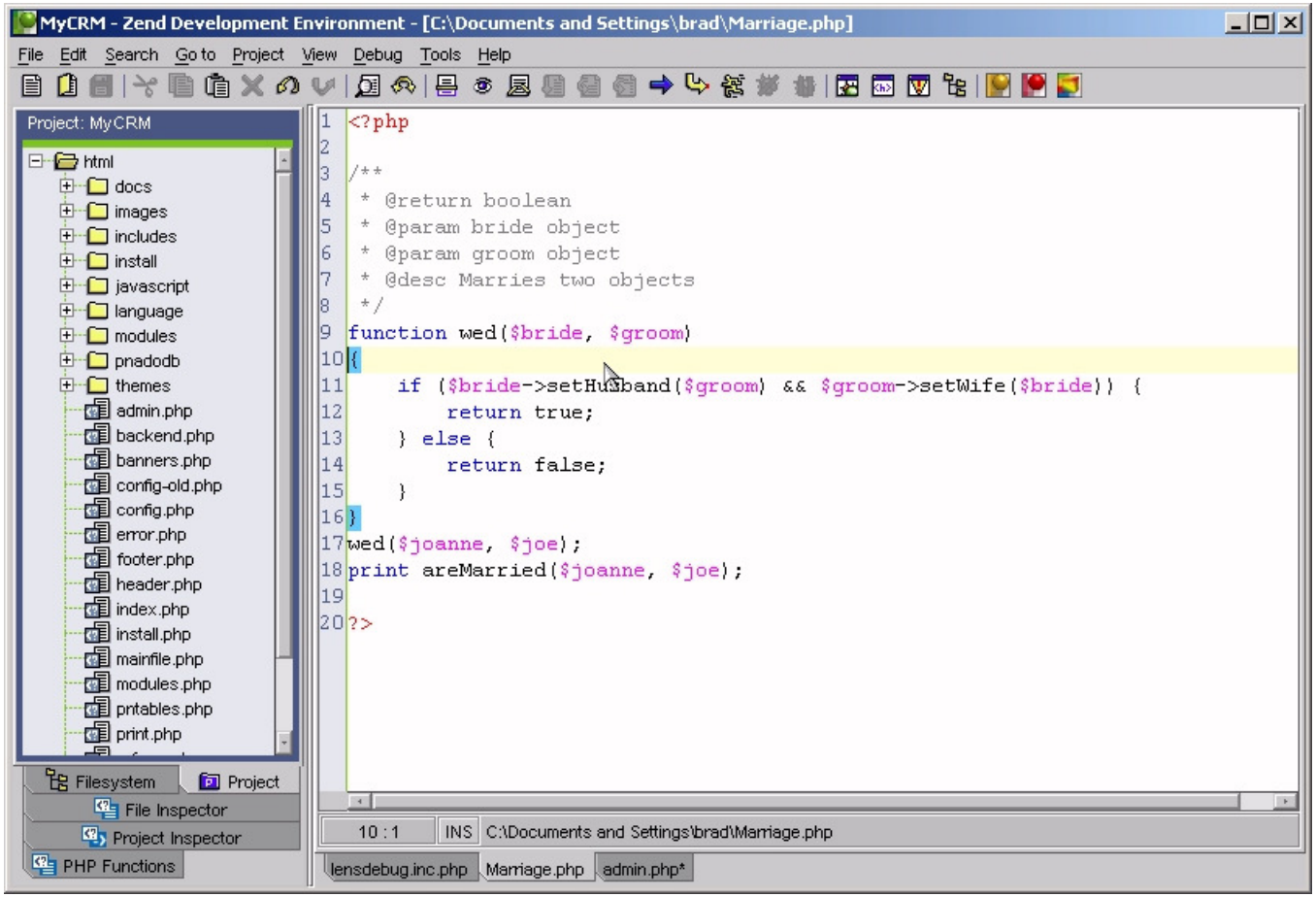


- Poziv php interpretera preko stranice

http://www.fesb.hr/~kiki/primjeri_internet_2/boja_pozadine.php

PHP

- PHP stranice se mogu pisati u bilo kojem tekstualnom editoru jer su to po svom formatu obične tekstualne datoteke. Npr. program Edit plus je dobar editor jer ima ugrađenu podršku za PHP.
- Postoji i komercijalni IDE za PHP koji se zove Zend Studio.
- PHP kôd se kombinira s HTML kodom tj. to je tzv. umetnuti jezik (*embedded language*) kao i ASP (*Active Server Pages*), SSI (*Server-Side Includes*), JSP (*Java Server Pages*).



Izgled prozora Zend studija

Interpreter ↔ Kompajler

- Interpreter prevodi i izvodi jednu po jednu liniju koda.
- Kompajler prevede cijeli kôd koji se nakon prevođenja može izvesti.
- Granični slučajevi su Java i .NET (CLR) kôd.
- Java kôd se prevodi u Java bajt kôd (Java *byte code*) koji onda interpretira JVM (Java Virtual Machine) ili kompajlira JIT kompajler.
- .NET kôd se prevodi u kôd koji se onda izvodi opet unutar .NET virtualnog stroja (CLR) koji interpretira kôd u strojni.

PHP

- Nakon izrade PHP stranice ta se stranica pohranjuje na web serveru koji ima podršku za izvršavanje PHP aplikacija (u direktoriju kojem server može pristupiti npr. na adriji /home/korisnik/public_html).
- Prava na PHP skripti trebaju biti ista kao i na običnoj html stranici (tj. treba biti dopušteno čitanje za sve procese – chmod 644 stranica.php).
- Web server zna da to nije obična HTML stranica nego PHP skripta obično prema ekstenziji datoteke, zato PHP skripte obično moraju imati ekstenziju .php (ili .php3 ili .php4 ili nešto treće ako je dozvoljeno na web serveru).

PHP

- Za izvršenje PHP stranice potrebno je da web server ima instaliran PHP interpreter, koji će pozvati kada klijenti dohvaća PHP skriptu, predati će interpreteru skriptu na izvršavanja i klijentu nazad vratiti rezultat izvršavanja PHP skripte.
- Većina web *hosting* pružatelja usluga (*provider*) pruža podršku za PHP.

PHP

- PHP se može integrirati s različitim web serverima npr. Apache, IIS, ….
- PHP interpreter će izvršiti sve stranice čije ekstenzije su u konfiguracijskim datotekama web servera postavljene kao PHP ekstenzije (obično php, php3, ali to se može postaviti u konfiguraciji web servera).
- Neovisan je o platformi (Linux, Windows, …).
- Može se vezivati na vanjske komponente kao što su Java Beans i Win32 COM.
- PHP je *open source*. Instalacijske verzije i izvorni kod možete dobaviti sa adrese http://www.php.net/downloads.php.
- Trenutno aktualne verzije su 5.3.8 i 5.2.17

PHP tagovi

- PHP kôd se umeće (*embedded language*) između početne i krajnje specijalne oznake tj. taga.
- Kada PHP parser (analizator, gramatički analizator, raščlanjivač sintakse) parsira PHP datoteku, traži početne i krajnje tagove i koristi samo dio koji se nalazi između tih tagova.
- Zbog toga je moguće PHP kôd ubacivati u različite tipove dokumenata jer se sve što je izvan PHP tagova jednostavno ignorira.
- Najčešće se PHP kôd umeće u HTML dokument.

PHP tagovi

1.	<?php	?>
2.	<script language="php">	</script>
3.	<?	?>
4.	<%	%>

- Prvi i drugi tagovi su uvijek na raspolaganju. Skraćeni tagovi (3) su na raspolaganju samo ako je PHP konfiguriran s tom opcijom (--enable-short-tags). ASP stil tagova (4) je na raspolaganju ako je PHP konfiguriran na taj način.
- Ako niste sigurni u opcije instaliranog PHP-a najbolje je koristiti prvi stil tagova.

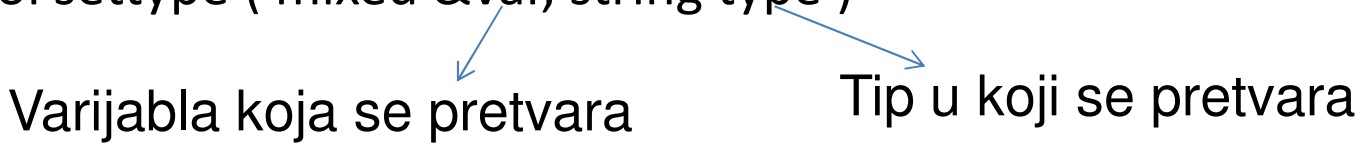
Komentari

- PHP podržava komentare u stilu C-a, C++ i u stilu Unix ljuske (Perl stil). :

```
<?php
echo 'This is a test'; // Svaka linija koda završava sa ;
      /* Ovo je isto
       kao u C programskom jeziku */
echo 'This is yet another test';
echo 'One Final Test'; # This is a one-line shell-style comment
?>
```

Tipovi podataka

- PHP podržava četiri skalarna tipa podataka (boolean ,integer, float, string); dva složena tipa (array ,object) i dva specijalna tipa (resource, NULL).
- Pretvaranje tipova podataka moguće je korištenjem *cast* operatora:
  - (int), (integer) - cast to integer
  - (bool), (boolean) - cast to boolean
  - (float), (double), (real) - cast to float
  - (string) - cast to string
  - (array) - cast to array
  - (object) - cast to object
- I korištenjem funkcije settype:
  - bool settype (mixed &var, string type)



Tipovi podataka

- ```
  <?php
  $a_bool = TRUE;   // a boolean
  $a_str  = "foo";  // a string
  $a_str2 = 'foo';  // a string
  $an_int = 12;     // an integer

  echo gettype($a_bool); // prints out:  boolean
  echo gettype($a_str);  // prints out:  string

  // If this is an integer, increment it by four
  if (is_int($an_int)) {
    $an_int += 4;
  }

  // If $bool is a string, print it out
  // (does not print out anything)
  if (is_string($a_bool)) {
    echo "String: $a_bool";
  }
  ?>
  ```
  ```

Varijabla

- Varijabla u PHP-u mora imati prefiks $.
- Što se tiče samih tipova varijabli kao i kod drugih skriptnih jezika, tip varijable se ustvari određuje prilikom dodjeljivanja vrijednosti varijabli ($varijabla = 4 (varijabla će biti integer)).
- To ne znači da ne moramo voditi računa o tipovima varijabli prilikom uspoređivanja vrijednosti varijabli, prilikom prenošenja vrijednosti u funkciju i sl.

Tipovi podataka

- Konverzija varijable u boolean može dati rezultat na koji niste navikli u drugim programskim jezicima:

- ```
  <?php
  var_dump((bool) "");        // bool(false)
  var_dump((bool) 1);         // bool(true)
  var_dump((bool) -2);        // bool(true)
  var_dump((bool) "foo");     // bool(true)
  var_dump((bool) 2.3e5);     // bool(true)
  var_dump((bool) array(12)); // bool(true)
  var_dump((bool) array());   // bool(false)
  var_dump((bool) "false");   // bool(true)
  ?>
  ```
  ```

Tipovi podataka

- Negativni broj u PHP-u se tretira kao TRUE!
- Funkcija var_dump prikazuje podatke o izrazu uključujući tip i vrijednost.

  void var_dump (mixed expression [, mixed expression [, ...]])

- Funkcija print_r prikazuje podatke o izrazu (vrijednost) u obliku prilagođenom za čitanje (npr. ispisati će čitav niz vrijednosti iz niza)
  bool print_r (mixed $expression [, bool $return])

- http://www.fesb.hr/~kiki/primjeri_internet_2/tipovi_podataka.php

Tipovi podataka

- String je niz karaktera. Karakter je u PHP-u bajt tj. PHP nema izvornu podršku za Unicode.
- Ali imate funkcije (string utf8_encode (string data)) koje vam omogućavaju pretvaranje stringa bajtova u Unicode string.

- String se može definirati na tri načina:
  1. $a='Ovo je string u jednostrukim navodnicima';
  2. $a="Ovo je string u dvostrukim navodnicima";
  3. Korištenjem *heredoc* sintakse. Nakon znaka <<< treba staviti nekakav identifikator, nakon kojeg ide novi red pa string pa na kraju stringa novi red pa onda isti identifikator kao na početku. Identifikator definirate sami uz poštivanje PHP konvencija imenovanja (može sadržavati samo alfanumeričke karaktere i *underscores*, i mora počinjati sa slovom ili *underscore-om*):

Tipovi podataka

- Primjer stringa definiranog *heredoc* sintaksom:

```
$a= <<<EOD
Ovo je string
napisan korištenjem
heredoc sintakse u više linija.
EOD;
```

- http://www.fesb.hr/~kiki/primjeri_internet_2/stringovi.php

Tipovi podataka

- PHP niz je uređena mapa (*ordered map*). Mapa je struktura podataka koja mapira vrijednosti prema ključevima.
- Hash tablica ili hash mapa je jedna implementacija uređene mape kod koje se ključ korištenjem hash funkcije pretvara u hash vrijednost tj. indeks na kojem se nalazi tražena vrijednost.
- PHP niz se može koristiti za pohranu "običnog" niza, liste, hash tablice, kolekcije, stoga, reda i sl.
- Niz se kreira korištenjem naredbe array(). Argumenti su parovi ključ=>vrijednost međusobno odvojeni zarezom.

```
array( key => value , ... )
```

- Ključ može biti integer ili string, a vrijednost može biti bilo što.

Tipovi podataka

```
<?
    $arr = array("foo" => "bar", 12 => true);
    echo $arr["foo"]; // bar
    echo $arr[12];    // 1
    // Create a simple array.
    $array = array(1, 2, 3, 4, 5);
    print_r($array);
    // Now delete every item, but leave the array itself intact:
    foreach ($array as $i => $value) {
      unset($array[$i]);
    }
    print_r($array);
    // Append an item (note that the new key is 5, instead of 0 as you might expect).
    $array[] = 6;
    print_r($array);
    // Re-index:
    $array = array_values($array);
    $array[] = 7;
    print_r($array);
?>
```

http://www.fesb.hr/~kiki/primjeri_internet_2/nizovi.php

Foreach naredba

- Naredba se koristi isključivo za dohvaćanje elemenata niza. Podržana su dva tipa sintakse:

  foreach (array_expression as $value) statement

  foreach (array_expression as $key => $value) statement

- Prvi oblik će proći kroz niz definiran u *array_expression*. U svakom koraku vrijednost elementa niza na kojem se petlja trenutno nalazi smješta se u *$value*, a petlja se pomiče na sljedeći element niza.
- Drugi oblik ima istu funkcionalnost osim što se još u varijablu *$key* dodaje i vrijednost ključa.

http://www.fesb.hr/~kiki/primjeri_internet_2/prforeache.php

Kontrola toka

- if ; else; elseif;
  alternativna sintaksa:
  if(): ... endif;
  while; do..while; for; break; continue; switch; require; include
- PRIMJER:
- if($s>1)
  printf(“Vrijednost je veća od jedan\n”);
  else
  printf(“Vrijednost nije veća od jedan\n”);

http://www.fesb.hr/~kiki/primjeri_internet_2/kontrolatoka.php

Uključivanje drugih datoteka

- Uključivanje druge datoteke u skriptu koja se izvodi:

  include(ime datoteke),

  require(ime datoteke),

  include_once(ime datoteke) i require_once(ime datoteke) – osiguravaju ukoliko je datoteka već uključena da se ne uključi dva puta

- Funkcije *require* i *include* se razlikuju u obradi greške. Ukoliko navedena datoteka ne postoji *require* će zaustaviti daljnje izvođenje skripte, a *include* neće.

Operatori

- Aritmetički:
- $a + $b , $a - $b, $a * $b, $a / $b, $a % $b

- Logički:
- $a and $b, $a or $b, $a xor $b, ! $a, $a && $b, $a || $b

- String:
- .operator konkatenacije dva stringa
- $a = "Hello "; $b = $a . "World!"; // rezultat $b = "Hello World!"

- Uspoređivanja:
- $a == $b, $a != $b, $a < $b, $a > $b, $a <= $b, $a >= $b

- Na razini bita:
- $a & $b, $a | $b, $a ^ $b, ~ $a, $a << $b, $a >> $b

Operator @

- PHP podržava operator kontrole greške (*error control operator*) koji se označava se @. Kada se taj znak doda ispred izraza sve greške koji bi taj izraz generirao prilikom izvođenja se ignoriraju tj. php interpreter ih ne generira što je zgodno ukoliko želite da se greške ne prikažu korisniku kada pristupi skripti.
- Operator kontrole greške "@" trenutno onemogućava i kritične greške koje bi dovele do prestanka izvođenja skripte. Npr. ako se ovaj operator koristi uz poziv funkcije sve greške koje je generirala funkcija će se izostaviti.

  $a = @(4/0); // Skripta neće generirati grešku u ovoj liniji
- Ukoliko je u interpreteru postavljena opcija track_errors greške će se pohranjivati u varijabli $php_errormsg tako da ih možete obraditi.

Funkcije

- Funkcija se definira ključnom riječi *function*. Nema definiranje povratnog tipa funkcije ni tipa parametara koje se prenose u funkciju. Povratna vrijednost iz funkcije se vraća sa ključnom riječi *return*.

```
function moja($arg1,$arg2,….,$argn)
{
    echo “Primjer funkcije.\n";
    return $retval;
}
```

Funkcije

- Funkcija treba biti definirana prije no što je se može koristiti. Varijable se u funkciju mogu prenositi i preko reference ako je potrebno u funkciji mijenjati vrijednost prenesene varijable.

```
function add_some_extra(&$string)
{
   $string .= ‘dodaj ovo.';
}
$str = ‘Ovo je string, ';
add_some_extra($str);
echo $str; // izlaz ‘Ovo je string, dodaj ovo.'
```

PHP predefinirane varijable

- Za razvoj web aplikacije u PHP-u izuzetno su važne predefinirane varijable. Predefinirane varijable imaju različiti sadržaj od postavki servera (npr. $SERVER_NAME) do prenesenih podataka (npr. $_POST) koji se dinamički mijenja.
- Predefinirane varijable su:
- Superglobals – ugrađene varijable dostupne svuda. To su varijable:
  - $GLOBALS – niz svih globalnih varijabli
  - $_SERVER – niz varijabli sa podacima o serveru
  - $_GET - HTTP GET varijable
  - $_POST - HTTP POST varijable
  - $_FILES - HTTP File Upload varijable
  - $_REQUEST - HTTP Request varijable (POST + GET + Cookie)
  - $_SESSION - Sesijske varijable
  - $_ENV - Okolinske varijable
  - $_COOKIE - HTTP Cookie varijable

PHP predefinirane varijable

- $php_errormsg – Prethodna poruka o grešci
- $HTTP_RAW_POST_DATA – Neformatirani POST podaci
- $http_response_header - HTTP zaglavlja odgovora
- $argc — Broj argumenata predanih skripti
- $argv — Niz predanih argumenata

PHP predefinirane varijable

http://www.fesb.hr/~kiki/primjeri_internet_2/predefinirane_varijable.php

```
<html>
<body>
<?
print_r($GLOBALS);
print_r($_SERVER);
print_r($_GET);
print_r($_POST);
print_r($_FILES);
print_r($_COOKIE);
print_r($_SESSION);
print_r($_REQUEST);
print_r($_ENV);
</body>
</html>
```

Obrada podataka sa forme

- Najbolje je za dohvaćanje podataka sa forme u PHP-u koristiti globalni niz koji se zove $_REQUEST. To je jedna od predefiniranih varijabli PHP interpretera.
- Taj niz uvijek sadrži sve varijable koje su sa forme poslane PHP skripti POST ili GET metodom.
- Niz je asocijativni, tj. vrijednostima sa forme se pristupa preko ključa koji je ime polja na formi čiju vrijednost želimo dohvatiti.

Obrada podataka sa forme

Vrijednost polja *polje* sa forme

http://www.fesb.hr/~kiki/primjeri_internet_2/stranica.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
      Transitional//EN">
<HTML><HEAD>
<TITLE> New Document </TITLE>
<META NAME="Generator" CONTENT="EditPlus">
<META NAME="Author" CONTENT="">
<META NAME="Keywords" CONTENT="">
<META NAME="Description" CONTENT="">
</HEAD><BODY>
<BR>Stranica za pokretanje programa<BR>
<form METHOD="POST" ACTION="obradi.php">
<INPUT TEXT NAME="polje"><br>
<INPUT TEXT NAME="polje2"><br>
<INPUT TEXT NAME="polje3"><br>
<input type="submit" value="POKRENI PROGRAM"
      size="1" >
</form></BODY></HTML>
```

http://www.fesb.hr/~kiki/primjeri_internet_2/obradi.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
      Transitional//EN">
<HTML>
<HEAD>
<TITLE> <? echo $_REQUEST["polje"];?></TITLE>
<META NAME="Generator" CONTENT="EditPlus">
<META NAME="Author" CONTENT="">
<META NAME="Keywords" CONTENT="">
<META NAME="Description" CONTENT="">
</HEAD>
<BODY>
<H1>
<?
echo $_REQUEST["polje2"];
?>
</H1>
</BODY>
</HTML>
```

Vrijednost polja *polje2* sa forme

Obrada podataka sa forme

- Često se forma smješta unutar same PHP skripte u kojoj se vrši obrada podataka.
- Tada se obično koristi predefinirana varijabla $_SERVER["PHP_SELF"] sa vrijednošću elementa identificiranog ključem PHP_SELF. To je uvijek ime skripte u kojoj je varijabla pozvana pa se može koristiti za definiranje ACTION atributa kao u skripti sa desne strane.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML><HEAD>
<TITLE>
<? if($_REQUEST["botun"])
echo $_REQUEST["polje"];?>
</TITLE>
</HEAD><BODY>
<H1>
<?
if($_REQUEST["botun"])
{
echo "Forma je pozvana i drugom polju je vrijednost ";
echo $_REQUEST["polje2"];
}
else echo "Nije pozvana forma!";
?>
</H1><BR>Stranica za pokretanje programa<BR>
<form METHOD="POST" ACTION="<?echo $_SERVER["PHP_SELF"];?>">
<INPUT TEXT NAME="polje"><br>
<INPUT TEXT NAME="polje2"><br>
<INPUT TEXT NAME="polje3"><br>
<input type="submit" name="botun" value="POKRENI PROGRAM" size="1" >
</form></BODY></HTML>
```

obradi2.php

Klase

- Definicija klase (tek od verzije PHP 3.0) počinje sa ključnom riječi *class* nakon koje ide ime klase.
- Ime klase može biti bilo koji string koji nije ključna riječ u PHP-u.
- Klasa je omeđena vitičastim zagradama unutar kojih se nalaze definicije metoda i članova klase.
- Ključna riječ *$this* sadrži referencu na instancu objekta s kojom se u tom trenutku radi.

Klase

```
class foo
{
    function do_foo ()
    {
        echo "Doing foo.";
    }
    var $varijabla = “vrijednost”;   //kod verzija PHP 4.x.x
                                     // treba se koristiti ključna
                                     // riječ var za deklaraciju
                                     //podatkovnih članova klase
    function koristithis()
    {
        echo $this-> $varijabla;
    }
}
```

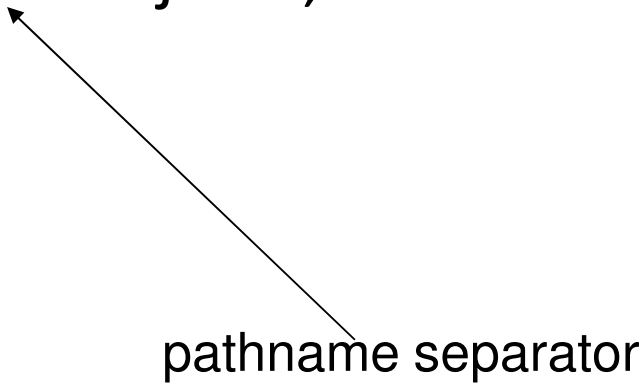
Klase

- Mogućnost pristupa članovima klase određuje se ključnim riječima public, protected ili private (samo kod PHP 5.x.x, kod ostalih verzija pristup članovima klase je isključivo *public* i ne mogu se navoditi modifikatori pristupa (istog značenja kao u C++) – najjednostavnije je provjeriti postavke PHP pomoću funkcije phpinfo()). Ukoliko nije definiran način pristupa metodi klase PHP podrazumijeva public način.
- Za kreiranje objekta koristi se naredba new.

Klase

- Prije instanciranja objekta klasa treba biti definirana.

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



Klase (samo kod verzija PHP 5.x.x)

- Ako je član klase deklariran sa ključnom riječi *static* može mu se pristupiti bez instanciranja klase.
- Statičkom podatkovnom članu ne može se pristupiti preko instance klase (objektu), statičkoj metodi može.
- Unutar statičke metode ne može se koristiti *$this*.
- Ključna riječ self omogućava pristup statičkim članovima klase unutar klase.

Klase

- Klasa može naslijediti metode i članove druge klase koristeći ključnu riječ *extends* u deklaraciji klase.
- Višestruko nasljeđivanje nije dozvoljeno.
- Naslijeđene metode i članovi mogu se prepisati (*override*), ukoliko nisu definirane s ključnom riječi *final*.Ukoliko je cijela klasa definirana s ključnom riječi *final* ta se klasa ne može naslijediti.
- Pristup prepisanim elementima osnovne klase radi se preko operatora dosega (::) i ključne riječi *parent* tj. parent::

Klase

```
class SimpleClass
    {
      // member declaration
      var $var = 'a default value';
      // method declaration
      function displayVar() {
         echo $this->var;
      }
    }
class ExtendClass extends SimpleClass
    {
      // Redefine the parent method
      function displayVar()
      {
         echo "Extending class\n";
         parent::displayVar();
      }
    }
```

http://www.fesb.hr/~kiki/primjeri_internet_2/klasa_nasljed.php

Klase

- Apstraktna klasa definira se ključnom riječi *abstract*. Apstraktna klase se ne može instancirati.
- Klasa koja sadrži barem jednu apstraktnu metodu je apstraktna. Apstraktne metode određuju samo signaturu (potpis) metode (ime metode, povratni tip i argumente) dok implementacija metode mora biti negdje drugdje (npr. u izvedenoj klasi).
- Prilikom nasljeđivanja izvedena klasa mora implementirati sve apstraktne metode. Implementacija metoda mora zadržati vidljivost metoda iz naslijeđene klase ili je može povećati, ali je ne može smanjiti. (Npr. ako je apstraktna metoda definirana kao *protected*, implementacije metode mora biti ili *protected* ili *public*).

```
abstract class AbstractClass
{
  // Force Extending class to define this method
  abstract protected function getValue();
  abstract protected function prefixValue($prefix);

  // Common method
  public function printOut() {
    print $this->getValue() . "\n";
  }
}

class ConcreteClass1 extends AbstractClass
{
  protected function getValue() {
    return "ConcreteClass1";
  }

  public function prefixValue($prefix) {
    return "{$prefix}ConcreteClass1";
  }
}
```

Sučelje

- Sučelje se definira ključnom riječi *interface*.
- Isto kao i drugim objektno orijentiranim jezicima sučelje deklarira metodu, ali implementacija metoda sučelja nalazi se isključivo u klasama koje implementiraju sučelje.
- Metode sučelja trebaju biti *public*.
- Za implementaciju sučelja koristi se operator *implements*. Sve metode iz sučelja trebaju biti implementirane u klasi.
- Klasa može implementirati više od jednog sučelja, ali implementirana sučelja ne smiju imati ista imena metoda.

```
// Declare the interface 'iTemplate'
interface iTemplate
{
  public function setVariable($name, $var);
  public function getHtml();
}

// Implement the interface
// This will work
class Template implements iTemplate
{
  private $vars = array();

  public function setVariable($name, $var)
  {
     $this->vars[$name] = $var;
  }
   public function getHtml()
  {
     foreach($this->vars as $name => $value) {
        $template =$name.", ".$value;
     }
     return $template;
  }
}
```