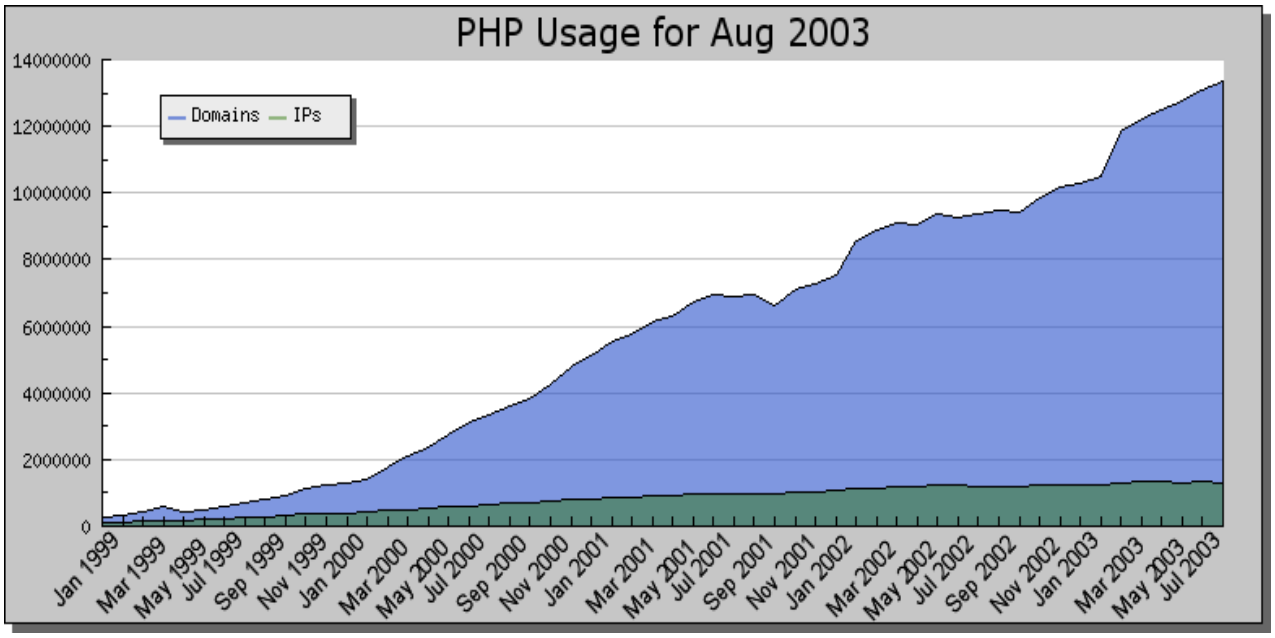
Internet programiranje PHP

Zašto PHP?

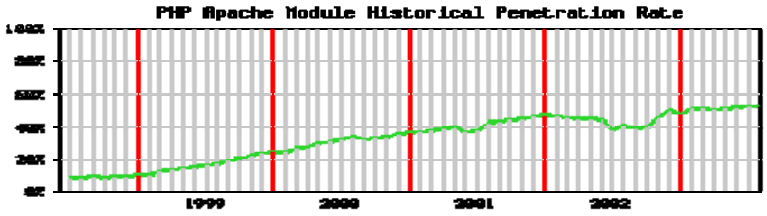
- *Web-based*, familijarni korisnički interfejs, jednostavna integracija sa drugim web-baziranim sistemima
- *Aktuelno*, dizajniran da se jednostavno prilagodi najsavremenijim dizajnima programskih jezika.
- *Portabl*, tako da može da se izvršava na bilo kom sistemu.
- *Objektno-orijentisan,* kao i većina popularnih jezika (Java, C++) koji se koriste za savremene aplikacije.
- *Konvencionalan*, uključuje mnoge moćne i korisne mogućnosti, podržava više različitih programerskih stilova
- *Korišćen*, PHP je u velikoj meri u upotrebi danas u Internet programiranju (uključujući i neke od najpopularnijih dinamičkih web sajtova), postoji veliki broj ne-trivijalnih primera i besplatnih resursa
- *Open-source*, (Microsoft)

PHP na Webu 1999-2003



Source: Netcraft

PHP u okviru Apache 1999-2003



Source: Security Space

Module	July 2003 Count	July 2003 %	June 2003 Count	June 2003 %	Growth %
PHP	3,894,353	52.26	3,685,550	52.24	0.03
OpenSSL	2,267,918	30.43	2,172,498	30.80	-1.18
mod_ssl	2,193,905	29.44	2,103,244	29.81	-1.25
FrontPage	1,633,599	21.92	1,555,986	22.06	-0.61
perl	1,481,305	19.88	1,457,669	20.66	-3.80

Preduslovi

- PHP je dizajniran tako da pomaže programerima da razvijaju dinamičke i podacima vođene Web stranice
- PHP na jednostavan način pristupa MySQL, kao i drugim open source bazama podataka, kao što je PostgreSQL.
- MySQL mora da bude instaliran zajedno sa funkcionalnim Web serverom, ali to je jednostavan korak u podešavanju PHP okruženja

Preduslovi

- PHP je skript jezik na serverskoj strani Web aplikacije koji se integriše u okviru HTML dokumenata
- Takođe, moguće je ubacivati HTMLkod u PHP skript
- PHP skript se parsira i interpretira na serverskoj strani Web aplikacije
- PHP budućnost je obećavajuća, jer je popularan kod Web programera i dizajnera, a moćan je i jednostavan za korišćenje

Malo istorije

- Rasmus Lerdorf
  - 1994 – Personal Home Page Tools
  - **PHP** (PHP: Hypertext Preprocessor)
- PHP2
  - Integracija sa bazom podataka
- PHP3
  - Novi parser
- PHP4
  - Zend engine
  - http://www.zend.com/
  - http://www.zend.com/zend/tut/
- PHP5
  - realizovan Aprila 2004

PHP

- Može se izvršavati bez Web servera
  - Ni malo zabavno
  - Ni malo praktično
- Mi ćemo koristiti Apache, PHP, MySQL kombinaciju
  - Veliki broj aplikacija se izvršava na ovoj platformi
  - Odlična Open Source kombinacija
  - Saznaćete više i o Open Source terminu

Windows OS

- Podešavanje Apache, MySQL, and PHP
  - http://httpd.apache.org/docs-2.0/platform/windows.html
  - http://www.mysql.com/doc/en/Windows_prepare_environment.html
  - http://www.php.net/downloads.php
- Gotove aplikacije koje obavljaju instaliranje
  - WAMP
    - http://www.wampserver.com/
  - Indigo
    - http://www.indigostar.com/indigoperl.htm
  - EasyPHP
    - http://easyphp.org/telechargements.php3

Open Source softver

- PHP je open-source softver, što znači da se može distribuirati bez ikakve nadoknade, a sopstveni kod je dat na korišćenje
- Linux operativni sistem, na primer, je takođe open-source softver koji je dostupan besplatno, jer je razvijen od grupe programera koji su zastupnici ovakve vrste softvera

Besplatan i Open Source Softver

- The Free Software Foundation
  - http://www.fsf.org
- SourceForge
  - http://www.sf.net
- Freshmeat
  - http://www.freshmeat.net
- SlashDot
  - http://www.slashdot.org

PHP proizvodi

- Zato što je PHP open-source proizvod, može se izvršiti download sa zvaničnog PHP Web sajta, *www.php.net*, u okviru koga postoji i deo FAQ (frequently asked questions), arhiva mailing liste, i različiti članci koji opisuju PHP proizvode

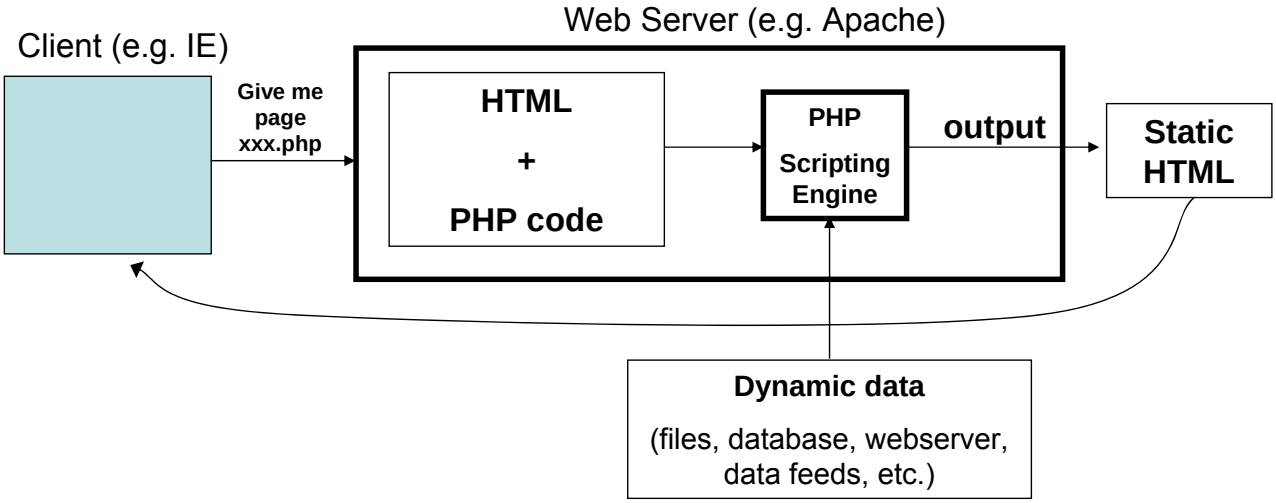
- PHP ima dosta built-in konekcija koje dozvoljavaju interakciju sa bilo kojom back-end bazom podataka, gde Web aplikacija smešta podataka

PHP proizvodi

- Prvi deo Web aplikacija, još se naziva i Prezentacioni nivo, je ono što korisnik vidi
- MySQL je najčešće korišćena back-end baza podataka za organizaciju Web sadržaja zato što je besplatna (open source), podržava veliki broj korisnika i platformi, i poseduje moćan i jednostavan SQL interfejs
- U velikom broju slučajeva ne koristi se samo PHP
- Uobičajeno je da je potrebna baza podataka koja sadrži informacije za dinamički Web sadržaj

Kako PHP funkcioniše?

- PHP je program koji izvršava dinamičke HTML skriptove
  - Interpreter jezika direktno izvršava komande koje generišu izlaz u formi HTML/HTTP (bolje nego korišćenje kompajlera koji kao rezultat prevodi komande jezika u mašinski kod)
  - Server zna da izvrši PHP kod kada zahtevani fajl ima .php ekstenziju



Primer PHP stranice

- U okviru tekstualnog editora otkucajte

```
<?php
echo '<big><center> Zdravo <center><BR>';
echo '1+2=';
echo 1+2;
?>
```

- PHP kod je uvek definisan u okviru <? … ?> tagova. Ostali deo koda (bilo unutar PHPa ili izvan) može takođe biti HTML
- Snimite fajl pod imenom “nekoIme.php”
- Snimite ga u direktorijum C:\Program Files\wamp\www\test,
- u čitaču ukucajte http://localhost/test/nekoIme.php

Primer PHP stranice

- Prethodni kod je moguće napisati i na sledeći način:

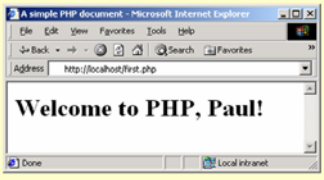
```
<big><center> Zdravo <center><BR>
1+2=
<?php
      echo 1+2;
?>
```

- Zašto se dobija isti rezultat?

Primer PHP stranice

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3
4   <!-- Fig. 29.1: first.php -->
5   <!-- Our first PHP script -->
6
7   <?php
8      $name = "Paul";   // declaration
9   ?>
10
11  <html xmlns = "http://www.w3.org/1999/xhtml">
12     <head>
13        <title>A simple PHP document</title>
14     </head>
15
16     <body style = "font-size: 2em">
17        <p>
18           <strong>
19
20              <!-- print variable name's value -->
21              Welcome to PHP, <?php print( "$name" ); ?>!
22           </strong>
23        </p>
24     </body>
25  </html>
```

First.php



Program Output

Definicija PHP promenljivih

- Kao i kod većine jezika u okviru PHPa postoje promenljive i operatori koji se izvršavaju nad njima
- Promenljive moraju početi znakom dolara ($) i mogu sadržati bilo koju kombinaciju slova i cifara
- PHP se naziva i loosely typed programski jezik, jer se ne mora unapred definisati tip promenljivih, već se mogu samo definisati i koristiti

Definicija PHP promenljivih

- Za imenovanje promenljivih postoji nekoliko pravila:
  - Ime promenljive mora početi znakom dolara ($)
  - Ime promenljive treba da bude definisano u skladu sa značenjem vrednosti koja se pamti u okviru promenljive
  - U okviru imena mogu se pojavljivati mal ili velika slova, cifre ili karakter _
  - Nije dozvoljeno da prvi karakter posle znaka dolara ($) bude cifra
  - PHP je case sensitive jezik
  - Inicijalna vrednost promenljive se dodeljuje pomoću znaka (=)

Tipovi podataka

Tip podataka	Opis
Integer	Celobrojne veličine
Double	Realni brojevi
String	Tekst koji se nalazi između jednostrukih ('') ili dvostrukih ("") navodnika.
Boolean	Može imati vrednost true ili false.
Array	Grupa elemenata istog tipa.
Object	Grupa povezanih podataka i metoda.
Resource	Eksterni izvor podataka.
Null	Bez vrednosti.

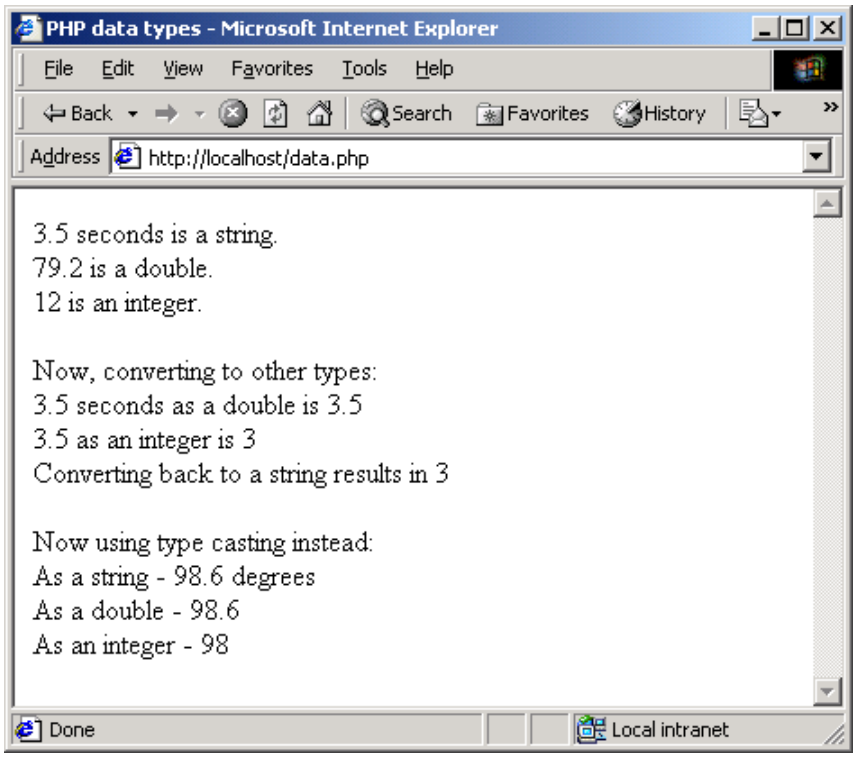
Data.php

```
1   <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4   <!-- Fig. 29.3: data.php             -->
5   <!-- Demonstration of PHP data types -->
6
7   <html xmlns = "http://www.w3.org/1999/xhtml">
8      <head>
9         <title>PHP data types</title>
10     </head>
11
12     <body>
13
14        <?php
15
16           // declare a string, double and integer
17           $testString = "3.5 seconds";
18           $testDouble = 79.2;
19           $testInteger = 12;
20        ?>
21
22        <!-- print each variable's value -->
23        <?php print( $testString ) ?> is a string.<br />
24        <?php print( $testDouble ) ?> is a double.<br />
25        <?php print( $testInteger ) ?> is an integer.<br />
26
27        <br />
28        Now, converting to other types:<br />
29        <?php
30
31           // call function settype to convert variable
32           // testString to different data types
33           print( "$testString" );
```

```
34            settype( $testString, "double" );
35            print( " as a double is $testString <br />" );
36            print( "$testString" );
37            settype( $testString, "integer" );
38            print( " as an integer is $testString <br />" );
39            settype( $testString, "string" );
40            print( "Converting back to a string results in
41               $testString <br /><br />" );
42
43            $value = "98.6 degrees";
44
45            // use type casting to cast variables to a
46            // different type
47            print( "Now using type casting instead: <br />
48               As a string - " . (string) $value .
49               "<br />As a double - " . (double) $value .
50               "<br />As an integer - " . (integer) $value );
51         ?>
52      </body>
53   </html>
```

Data.php

Rezultat primera



Podela tipova podataka

skalarni

- ili primitivni ili osnovni tipovi podataka
- Najjednostavniji tipovi
- Ne mogu se prebacivati u druge tipove
- Jedino pojedinačne vrednosti
- Primer:
  - Integer
  - floating point (real)
  - String
  - Boolean

grupni

Ili klasni tipovi

- kompleksniji
- Sastavljeni od drugih tipova (primitivnih ili klasnih tipova)
- Mogućnost više vrednosti
- Primer:
  - Array
  - Object

Tipovi podataka

- **`integer`**
  - Celobrojne veličine
  - Pozitivne ili negativne
  - Dodatni brojni sistemi
    - Oktalni: 0755 // počinju '0'
    - Hex: 0xFF // počinju '0x'
  - Referišu se sa **`int`**
- **`boolean`**
  - Dve vrednosti – `true` ili `false`
  - Koriste se kod raznih upita (`if`, `while`)
  - Referišu se sa **bool**

- **`floating point`**
  - Racionalni brojevi, pozitivni i negativni
  - Postoji decimalna tačka
  - dva formata
    - Brojevi sa decimalnom tačkom, 514.061
    - Eksponent i mantisa, 5.14061E2, što je 5.14061×10^2
  - Referišu se sa **`double`**
- **`null`**
  - Bez vrednosti

Tipovi podataka

- **`string`** je niz karaktera
  - Često korišćen tip podataka
    - Imena, šifre, adrese, istorija, ...
  - Koriste se bi se prikazali kompleksni podaci
    - Datumi, brojevi telefona, formatirani izlaz
  - Koristi se i za rad sa XML ili drugim tekstualnim fajlovima
- PHP poseduje složen i moćan skup funkcija za rad sa stringovima
  - Obrada, pretraga, upoređivanje, comparing, translacija, ...

Deklaracija stringova

- Vrednost stringa se može deklarisati na 3 načina
  - Apostrofa
    - `'Primer 1';`
  - Navodnika
    - `"Primer 2";`
  - Pomoćnih dokumenata
    - `<<< When_Will_It_End`

```
It was a dark and stormy night, my son said, "Dad,
  tell me a story" so I said, "It was a dark and
  stormy night, my son said, "Dad, tell me a story"
  so I said, "It was a dark and stormy night, my
  son said, "Dad, tell me a story" so I said,
When_Will_It_End
```

Echo

- Naredba `echo` se koristi za jednostavan izlaz
  - `echo 'hello';`
  - `echo 'hello', ' goodbye';`
  - `echo ('hello');`
- `print` je slična naredba
  - `print 'hello'`
- Mogu se koristiti i ()
  - `echo('hello');`
  - `print('hello');`

Printf()

- Naredba `printf()` se koristi za kompleksnije formatirane izlaze

```
printf('This prints 2 decimal places
  %.2f', 3.1415927);
```

This prints 2 decimal places 3.14

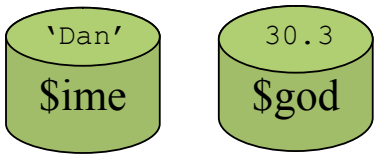
- `Printf()` je funkcija, čiji je prvi argument string koji opisuje željeni format, a preostali argumenti su vrednosti koje se unose u okviru specificiranog formata (svi članovi koji počinju sa %)

Izrazi

- Nad promenljivama se mogu izvršavati određene operacije

```
echo 1+2;
echo 3*2;
echo “Big“ . “ “ .“Fred”;
printf(“5/3 is about %3d”,
5/3);
 +,-,.,*,/,%
```

Promenljiva



- Imenovana lokacija za smeštanje podataka
  - Prostor za podatke (kao kutija za stvari)
- Promenljiva definiše samo jedan tip podatka
  - Na primer samo integer, samo floating point, ili samo character
  - Ako je promenljiva skalarnog tipa može sadržiti samo jednu skalarnu vrednost
  - Promenljiva složenog tipa može sadržati više skalarnih vrednosti
- Sintaksa za imena promenljivih je `$`<identifier>
  - Primer: `$ime, $god`
  - Case sensitive!

Dodeljivanje vrednosti promenljivoj

- Koristi se operator dodele vrednosti: “=“
  - Dodeljuje vrednost promenljivoj
  - Nema značenje “identično je”
  - Nije isto kao u algebri
- Pravo značenje -
  “Dodeljuje se vrednost izraza na desnoj strani promenljivoj na levoj strani.”
- Promenljive se mogu pojavljivati i na desnoj strani jednakosti:

```
$count = 10;//inicijalizacija promenljive count
   $count = $count - 1;// decrement count
```

  - Nova vrednost promenljivenew value of count je 9

Osnovne operacije

```
$add = $a + 1;
$mult = $a * 2;
$str = $s.” Fred”;
```

- PHP izvršava operacija sa promenljivama različitog tipa tako što konvertuje promenljive u isti tip po određenim pravilima
  - `$whatIsThis = 1.2 + “ Fred” . 0777 * true;`
- Ista promenljiva se može pojaviti na obe strane jednakodti
  - `$a = $a + 1;`

Kreiranje promenljive

- Promenljiva se deklariše prvi put kada joj se dodeljuje vrednost
- Pomoću deklaracije promenljive rezerviše se prostor u memoriji i specificira tip podatka koji će biti smešten u toj memorijskoj lokaciji:
  - $a = 1.1 // deklaracija i inicijalizacija realnog broja
  - $a = true // deklaracija i inicijalizacija boolean promenljve
  - $a = ‘fat stuff’ // deklaracija i inicijalizacija string promenljive

Default Vrednosti

- Promenljive imaju default vrednosti
  - `$a = $a + 1; // $a=0 po default`
  - `$s = $s."Fred"; // default $s=""`
  - Preporuka je da se ne koriste default vrednosti, već da se eksplicitno navode inicijalne vrednosti promenljivih
    - `$a = 0; $s = ""; $b = false;`

Imena promenljivih

Pravila

- **moraju** se poštovati

- Sva imena moraju koristiti ista pravila
- Ne smeju počinjati sa cifrom
- Mogu sadržati samo brojeve, slova, donju crtu (_) i neke specijalne karaktere
- Imena su case-sensitive (ThisName i thisName du imena dve različite promenljive)
- Bez blanko znakova!

Preporuka

- **trebalo** bi poštovati

- Koristiti imena koja opisuju tačno značenje promenljive
- Prvi simbol promenljive treba da bude malo slovo
- Svaka sledeća reč unutar imena počinje velikim slovom (`eggsPerBasket` umesto `eggsperbasket`)
- Koristii donju crtu (_) za blanko znake
- Konstante pisati velikim slovima

NULL

- Null je specijalan tip sa značenjem “bez vrednosti"
- Može se koristiti da bi se inicijalizovala promenljiva
  - $a = NULL;

String

- Niz karaktera, na primer abc123$%!
  - U PHP se naziva **`string`**
- Definiše se pomoću apostrofa ili navodnika
  - 'string', "string"
  - Bilo koja promenljiva unutar stringa sa navodnicima se pretvara u string i izvršava, dok se kod apostrofa ništa ne dešava

```
$a = 10; echo "I have $a more slides";
```

I have 10 more slides

```
$a = 10; echo 'I have $a more slides';
```

I have $a more slides

Specijalni karakteri

- Specijalni karakteri su karakteri koji nisu vidljivi većimaju specijalno značenje
  - \n nova linija (nema isto značenje kao u okviru *HTMLa
)*
  - \t tab
  - \$ karakter $
  - \\ karakter \
- Primer:
  - **echo** `"I need more` **`\$`** `but they` **`\\'`**`ed the budget`**`\n";`

    I need more $ but they \'ed the budget

Razlike

- Način na koji je deklarisan string određuje njegove osobine
  - Specijalni karakteri (n.p. \n) se prepoznaju
  - Izračuna se vrednost promenljivih
- Apostrofi posmatraju string kao literal
  - ```
    $variable = 10;
    echo 'This prints $variable\n';
    ```
    `This prints $variable\n`
  - Svi karakteri se prikazuju, specijalni karakteri se ne prepoznaju osim \' i \\
- Navodnici prepoznaju i specijalne karaktere i vrednosti promenljivih
  - ```
    $variable = 10;
    echo "This prints $variable\n";
    ```
    `This prints 10`

print_r(), var_dump()

- Tip i vrednost promenljivih se može prikazati pomoću funkcija `print_r(), var_dump()`
- Ove funkcije mogu veoma da pomognu pri proveri rada programa
- `print_r()` prikazuje samo vrednost
- `var_dump()` prikazuje i tip i vrednost

Izrazi

- Izrazi su PHP naredbe koje vraćaju određenu vrednost

```
$numberOfBaskets = 5;
$eggsPerBasket = 8;
$totalEggs = $numberOfBaskets * $eggsPerBasket;
$check = ($totalEggs = $numberOfBaskets *
  $eggsPerBasket) / 40;
```

- Takođe i funkcije mogu da kao rezultat vraćaju određenu vrednost

`is_string()`, vraća logičku vrednost `true` ili `false`,
`phpinfo()`, vraća string

Operatori

- Operatori koriste operandeda bi izvršili određene operacije
  - Postoje unarni, binarni i ternarni operatori, koji koriste 1, 2, ili 3 operanda respektivno.
    - Unarni: `$endOfFile++, !is_string()`
    - Binarni: `2 * 3, $a % 3, $a = 40, $a > 3, $a == 5, $a || $b`
  - Pored aritmetičkih mogu se izvršiti i druge operacije

Konkatenacija stringova

Operator '.' se koristi za sabiranje (konkatenaciju) stringova:

```
$name = "Mondo";
$greeting = "Hi, there!";
echo $greeting . $name . "Welcome";
```

Na ekranu će se pojaviti:

```
Hi, there!MondoWelcome
```

– Ili:

```
$greeting . " " . $name . " Welcome";
```

– Dobija se rezultat:

```
Hi, there! Mondo Welcome
```

Boolean tip podataka

- Primitivni tip
- Koristi se u okviru izraza, promenljivih, konstanti, ako i bilo koji drugi primitivni tip
- Samo dve vrednosti: **`true`** i **`false`**
- Svaki izraz se može prikazati kao `boolean`
  - `0, 0.0, '0', ''` su vrednosti za **`false`**
  - Sve ostale vrednosti se prepoznaju kao **`true`**
- Operatori poređenja kao rezultat dobijaju vrednost tipa `boolean`

```
$is_desired_grade = ($grade == 'A');
$is_drinking_age = ($age >= 21);
$not_graduating = ($year != 'senior');
```

Izrazi tipa boolean

- Izrazi tipa boolean se koriste za ispitivanje uslova kontrola toka
- Porede se dve vrednosti (brojevi, stringovi), kao rezultat se dobija vrednost tipa boolean (`true` ili `false`)
- Na primer:
  Da li je A veće od B?, Da li je A jednako B?, Da li je A manje ili jednako od B?
- U okviru izraza tipa boolean koriste se operatori poređenja
  `(<, >, <=, >=, ==, ===, !=, !==)`
- A i B mogu biti bilo kog tipa (ili klase), ali treba da generalno budu kompatibilni
  - Poređenje može biti numeričko ili leksičko, ali mora biti korisnički definisano preko objekata i funkcija.
  - Poređenje ne-kompatibilnih tipova je dozvoljeno, ali može dati neočekivane rezultate

Osnovni operatori poređenja

Mat.	Ime	PHP	Primer
=	jednako	==	balance == 0 answer == 'y'
≠	nejednako	!=	income != tax answer != 'y'
>	veće od	>	income > outgo
≥	veće ili jednako od	>=	points >= 60
<	manje od	<	pressure < max
≤	manje ili jednako od	<=	income <= outgo

Prikaz boolean vrednosti

- Komande za prikaz na ekranu konvertuju boolean vrednosti u string pre prikaza i to na sledeći način
  - `true` se konvertuje u ‘1’
  - `false` se konvertuje u ‘’ (prazan string)
- `echo true`

`1`

- `echo false`

<bez izlaza>

Poređenje bez konverzije tipova

- Ponekad je potrebno porediti dve vrednosti bez prethodne konverzije.
- Da li je 0.0 jednako sa 0?
- U tom slučaju se koriste operatori “===“ i “!==“ za poređenje bez konverzije tipova
  - `0.0 == 0` rezultat je `true`
  - `0.0 === 0` rezultat je `false`

Logički operatori

- `&&` ili `and` predstavlja logičko I
- `||` ili or predstavlja logičko ILI
- Da li je B jednako 0 vrednost između A i C :

```
(B == 0) || (A <= B && B < C)
(B == 0) or (A <= B) and B < C)
```

- U ovom primeru zagrade nisu neophodne ali su dodate zbog razumljivosti
  - Ovo je primer kada se prestaje sa izračunavanjem izraza čim se dobije rezultat
  - Mogu se koristiti `&` i `|` da se u svakom slučaju izračuna ceo izraz

Tabele istinitosti

&& (and)

Value of A	Value of B	A && B
true	true	**true**
true	false	**false**
false	true	**false**
false	false	**false**

|| (or)

Value of A	Value of B	A || B
true	true	**true**
true	false	**true**
false	true	**true**
false	false	**false**

! (not)

Value of A	!A
true	**false**
false	**true**

Prioritet operatora

- Izrazi se izvršavaju po određenom prioritetu
  - Da li je 2 + 3 * 4 jednako 5*4 ili 2 + 12?
- PHP operatori imaju prioritet sličan kod algebre realnih brojeva
  - Operatori sa većim prioriteom se prvo izvršavaju
    - 2 + 3*4, je 2 + 12, odnosno 14
- Zagrade se mogu koristiti da bi se promenio redosled izvršavanja
  - Ex. (2 + 3)*4 je 5*4
- Ne moraju se koristiti zagrade ako je samo pomoću prioriteta dobijeno korektno izračunavanje

Prioritet

Primer:

```
$score < $min/2 – 10 || $score > 90
```

- Operator deljenja ima najviši prioritet:

```
$score < ($min/2) – 10 || $score > 90
```

- Oduzimanje je sledeće po prioritetu:

```
$score < (($min/2) – 10) || $score > 90
```

- Operatori < i > imaju isti prioritet, pa će se izvršiti istovremeno:

```
($score < (($min/2) – 10)) || ($score > 90)
```

- Dobijeni izraz je potpuno popunjen zagradama, i identičan je originalu. On samo pokazuje redosled kako će se delovi izraza izračunati.

Prioritet

Najviši prioritet

- the unary operators: `++`, `--`, and `!`
- the binary arithmetic operators: `*`, `/`, `%`
- the binary arithmetic operators: `+`, `-`
- the boolean operators: `<`, `>`, `=<`, `>=`
- the boolean operators: `==`, `!=`
- the boolean operator `&`
- the boolean operator `|`
- the boolean operator `&&`
- the boolean operator `||`

Najniži prioritet

Aritmetički primeri

Ordinary Math Expression	PHP Expression (preferred form)	PHP Fully Parenthesized Expression
$rate^2 + delta$	$rate*$rate + $delta	($rate*$rate) + $delta
2(salary + bonus)	2 * ($salary + $bonus)	2 * ($salary + $bonus)
$\frac{1}{time+3mass}$	1/($time + 3 * $mass)	1/($time + (3 *$mass))
$\frac{a-7}{t+9v}$	($a - 7) / ($t + 9 * $v)	($a - 7) / ($t +(9 * $v))

Konverzija tipova

- Konverzija je proces prebacivanja vrednosti iz jednog tipa podatka u drugi. Nekada je korisno upotrebiti ovu funkciju.
- Konverzija menja samo dobijenu vrednost, a ne i tip promenljive nad kojom se primenjuje
- Na primer:

```
$n = 5.0;
$x = (int) $n;
```

- U primeru $n je realni broj, naredba (int) $n konvertuje njegovu vrednost u int, i sada je $x tipa integer.
- Treba voditi računa kada se složeniji tip konvertuje u prostiji

```
$n = (int) 5.2345; // rezultat je 5
```

Inkrement i dekrement

- Ovi operatori se koriste kao skraćena notacija za operacije sabiranja i oduzimanja
- Operacije inkrementa (i dekrementa) se mogu izvršiti pre ili posle korišćenja vrednosti promenljive

  `++$count` preinkrement

  `$count++` postinkrement

  `--$count` predekrement

  `$count-` postdekrement
- Ove operacije se mogu izvršiti i nad promenljivama koje nisu tipa integer

```
$var = 'a'; $var++; // $var je sada 'b'
```

- Šta se dešava ako je `$var = 'az'` ?

Inkrement/Dekrement primeri

Inicijalizacija:

```
$n = 3;
$m = 4;
```

Kolika je vrednost promenljive `m` u sledećim primerima?

```
(a) $result = $n * ++$m;
(b) $result = $n * $m++;
(c) $result = $n * --$m;
(d) $result = $n * $m--;
```

Specijalni operatori dodele

- Specijalni operatori dodele se takođe koriste kao skraćena notacija
- Opšta forma: *var <op>= expression;*
  - je identično kao: *var = var <op> (expression);*
  - <op> može biti +, -, *, /, %, . , |, ^, ~, &, <<, >>
- Primeri:

```
$amount += 5;
   //isto kao $amount = $amount + 5;
$amount *= 1 + $interestRate;
   //$amount = $amount * (1 + $interestRate);
```

Implicitna konverzija tipova

- Konverzija tipova se može izvršiti i automatski ako je prostiji tip dodeljuje složenijem u zavisnotsi od operatora
- Hijerarhija tipova podataka je (od najprostijeg n najsloženijem): bool -> int --> double --> string --> array --> object
- Primer:

```
$x = 0;
$n = 5 + 10.45;
$x = $n;
```

  - Vrednost 5 se konvertuje u double, i dodeljuje $n
  - `$x` je sada double
  - Naziva se implicitna konverzija, jer se obavlja automatski

Implicitna konverzija tipova

- U okviru izraza se može koristiti više tipova podataka
- Sve vrednosti se automatski konvertuju na najsloženiji tip, pre bilo kakvih izračunavanja
- Primer:

```
$n = 2;
$x = 5.1;
$y = 1.33;
$a = (n * x)/y;
```

  - `$n` se automatski konvertuje u tip double, pre nego što se izvrše operacije množenja i deljenja

`if` naredba

- Izvrši sledeću naredbu ako je uslov `true` ili je preskoči ako je uslov `false`
- Sintaksa:

```
if (Boolean_Izraz)
    Naredbe ako je true;
sledeca_naredba; uvek se izvršava
```

Primer

```
if ($eggsPerBasket < 12)
    echo "Less than a dozen eggs per basket";
$totalEggs = $numberOfEggs * $eggsPerBasket;
echo "You have a total of $totalEggs eggs.";
```

Blok izraza

Sve naredbe unutar zagrada su u okviru `if`

```
if ($eggsPerBasket < 12)
{
   echo “Less than a dozen ...”;
   $costPerBasket = 1.1 * $costPerBasket;
}

$totalEggs = $numberOfEggs * $eggsPerBasket;
echo “You have a total of $totalEggs eggs.”);
```

Blok izraza

Sve naredbe između ***:*** i ***endif;*** su u okviru naredbe `if`

```
if ($eggsPerBasket < 12) :
   echo “Less than a dozen ...”;
   $costPerBasket = 1.1 * $costPerBasket;
endif;

$totalEggs = $numberOfEggs * $eggsPerBasket;
echo “You have a total of $totalEggs eggs.”);
```

if-else

- Selektuje se jedna od dve opcije
- Izvršava se Action1 ili Action2, u zavisnosti od vrednosti uslova
- Sintaksa:

```
if (Boolean_Expression)
{
   Action1
}
else
{
  Action2
}
Action3 //uvek se izvrsava
```

if-else

- Primer sa jednom naredbom:

```
if ($time < $limit)
   echo "You made it.";
else
   echo "You missed the deadline.";
```

- Blok naredbi:

```
if ($time < $limit)
{
   echo "You made it.";
   $bonus = 100;
}
else
{
   echo "You missed the deadline.";
   $bonus = 0;
}
```

if-else if-elseif-…-else

- Situacije sa više od dva izbora
- Sintaksa:

```
if(Boolean_Expression_1)
   Action_1
elseif(Boolean_Expression_2)
   Action_2
      .
      .
      .
elseif(Boolean_Expression_n)
   Action_n
else
   Default_Action
```

if-elseif-elseif-…-else

```
if($score >= 90 && $score <= 100)
   $grade= 'A';
elseif ($score >= 80)
   $grade= 'B';
elseif ($score >= 70)
   $grade= 'C';
elseif ($score >= 60)
   $grade= 'D';
else
   $grade= 'E';
```

switch

```
switch($profRel) {
   case “colleague” :
        $greeting = “Thomas”;
        break;
   case “friend” :
        $greeting = “Tom”;
        break;
   case “grad student” :
        $greeting = “TC”;
        break;
   case “undergrad student” :
        $greeting = “professor”;
        break;
   default :
        $greeting = “Dr. Collins”;
        break;
}
```

Petlje

- Postoje sledeće petlje
  - while-do
  - for
  - do-while

Petlje

- Mogućnost ponavljanja određeni broj puta skupa naredbi

kontrola petlje **{**

}

Ponavljanje naredbi nula ili više puta, kontrolisano pomoću *kontrola petlje*

While

while (*uslov***)**
naredbe

Sadrži promenljive koje se menjaju u okviru petlje

Ponavlja se dok uslov ne postane false

```
$x = 5;
$s = 0;
while ($x > 0) {
  $s = $s + $x;
  $x = $x-1;
}
```

```
$x = 5;
while ($x>0) {
   echo $x;
   $x = $x-1;
}
```

while

```
$s = 0;
$x = 1;
while ($x <= 10) {
  $s = $s + $x;
  $x = $x + 1;
}
```

```
while (true)
      echo “I will do the reading assignments”;
```

```
while (false)
  echo “I will start my HW early”;
```

for

```
for (init; cond; incr)
    naredbe
```

≡

```
init;
while (cond) {
    naredbe
    incr;
}
```

```
$s = 0;
for ($x = 5; $x > 0;
       $x = $x-1)
  $s = $s + $x;
```

```
$s = 0;
for ($x = 1;
     $x <= 10;
     $x = $x+1)
  $s = $s + $x;
```

for

```
for ($i = 0; $i < 20; $i++)  {
     if ($i%2 == 1)
          echo “My robot loves me\n";
     else
          echo “My robot loves me not\n";
}
```

```
for ($i = 1; $i <= 20; $i++)
{
    echo “\nMy robot loves me";
    if ($i%2 == 0)
       echo " not";
}
```

Do-while

```
do naredbe
while (cond)
```

≡

```
naredbe;
while (cond)
  naredbe
```

`exit` funkcija

- Ako se pojavi situacija kada je nepotrebno da se nastavi izvršavanje programa, prekid se može izazvati korišćenjem funkcija *`exit()`*ili *`die()`*
- Može se koristiti argument tipa string da bi se utvrdilo da li je program normalno ili ne prekinuo rad

Nizovi

- Kolekcije podataka
- Osnove nizova
  - Deklaracija, alokacija, i inicijalizacija
- Rad sa nizovima pomoću iteracija i rekurzija
- Čitanje i upis u nizove

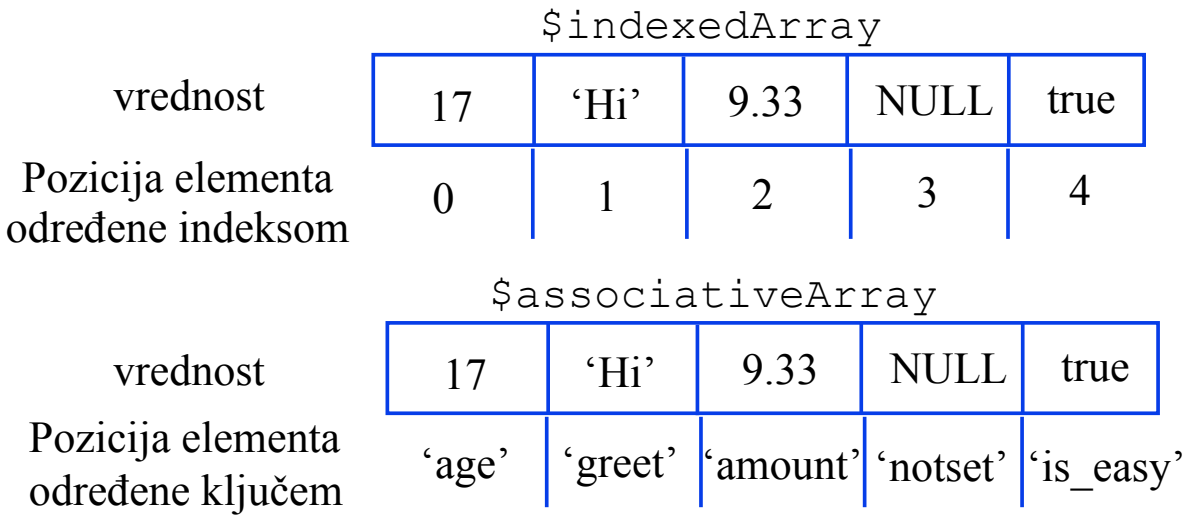
Kolekcije

- Nekada je potrebno izvršiti određene operacije sa više podataka kao sa jedinstvenom grupom
- Kolekcija je vrsta tipa podataka koje sadrži više vrednosti u jednom trenutku.
- Jedna vrsta kolekcije su i *stringovi* (koji čine kolekciju karaktera).

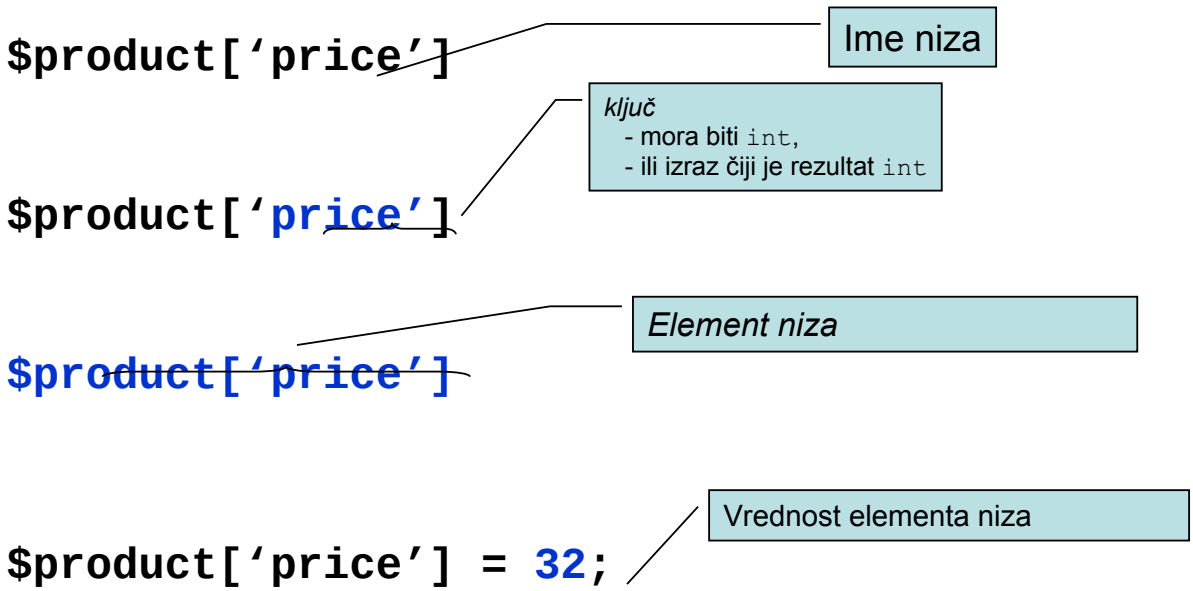
Nizovi

- Vrsta uređenih kolekcija.
- Specijalne mogućnosti:
  - Ugrađeni u okviru PHP, sa svojom sintaksom.
  - *Elementi* mogu biti bilo kog tipa podataka (uključujući i druge nizove!)
  - Kolekcija može menjati svoju dužinu bilo kad
  - Nizovi mogu biti indeksni ili asocijativni (ili oboje!)
  - Svi nizovi u okviru PHPa su asocijativni
    - Ključevi se automatski postavljaju na tip integer

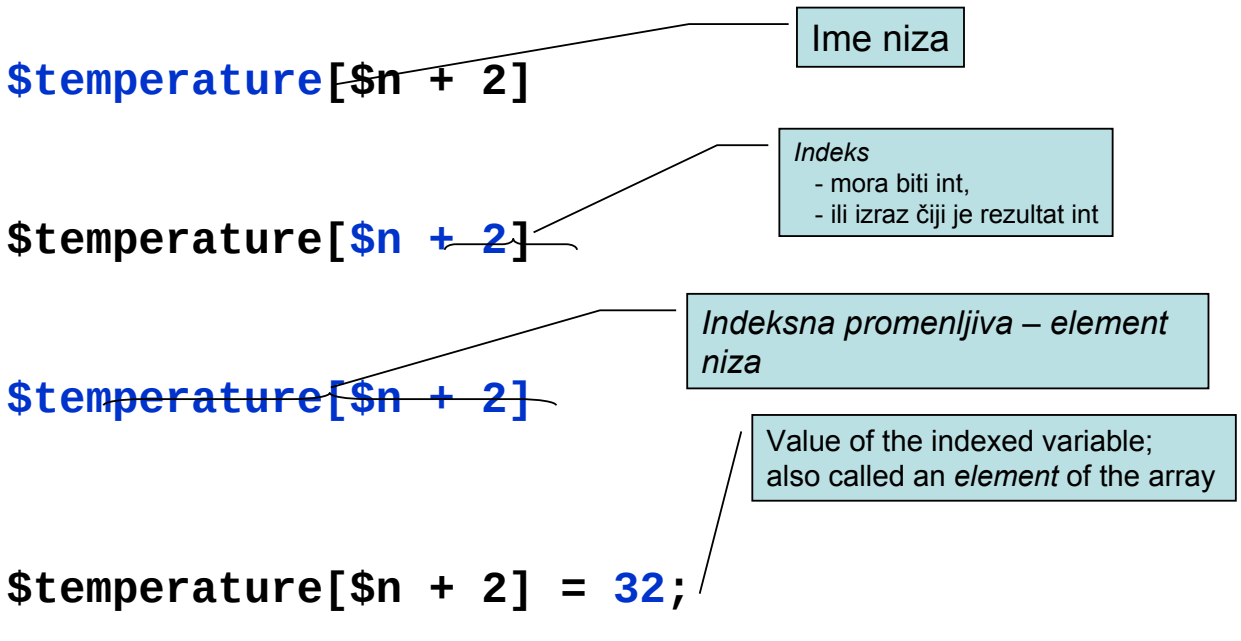
Arrays (cont.)


$indexedArray
vrednost
17
‘Hi’
9.33
NULL
true
Pozicija elementa određene indeksom
0
1
2
3
4
$associativeArray
vrednost
17
‘Hi’
9.33
NULL
true
Pozicija elementa određene ključem
‘age’
‘greet’
‘amount’
‘notset’
‘is_easy’

Terminologija - asocijativni nizovi



Terminologija - indeksni nizovi



Pristup

- Na jednostavan način se pristupa svakom elementu niza

array-name[*key-expression*]

  - Ako je key-*expression* celobrojna veličina *n*, dobija se *n*-ti element niza *array-name*, prvi je sa ključem 0
  - Ako je ključ string, dobija se vrednost koja je povezana sa tim ključem

Primeri

```
$grades[0] = 75;
echo $grades[0];

$message[0] = “Don’t”;

$i = 2;
$clocks[$i-1] = time();
printf(‘the %d’th time is %s‘, $i, $clocks[$i-
1]);

$name[‘first’] = “Rumple”;
$name[‘last’] = “Stilskin”;
echo “name is {$name[‘first’]} {$name[‘last’]}”;
```

Moguće greške

- Upotreba indeksa većeg od `duzina_niza-1` ili manjeg od `0` ili ključa koji ne postoji neće dovesti do prekida rada programa.
  - Neće se dobiti vrednost nijednog elementa
  - Treba voditi računa da se koriste korektni indeksi!

```
echo "The product is $product['name'];//nije
  korektno!
echo "The product is $product[name]";
echo "The product is {$product['name']}";
```

Deklaracija i inicijalizacija nizova

```
$angles = array(3.5, 9.7, 15.01);
```

- Jedan način: korišćenje liste elemenata i *array()*.
  U primeru dobijen je niz sa tri elementa, isti kao i sledeći:

```
$angles[0] = 3.5;
$angles[1] = 9.7;
$angles[2] = 15.01;
```

- Mogu se koristiti i vrednosti ključa

```
$product = array('name' => 'gumball',
                 'price' => 0.07,
                 'quantity' => 1500);
```

Dužina niza

- Dužina niza **`$anArray`** je rezultat funkcije **`count($anArray)`** ili **`sizeof($anArray)`.**
- Često se koristi u okviru petlji:

```
$counter=0;
while( $count < sizeof($anArray) ) {
     echo “element $counter is
{$anArray[$counter]}”;
     $counter++;
}
```

HTML i PHP

```
<?php
   echo '<b><i>Hello World!!!!</b></i><br>';
   echo '<hr>';
   echo '<table border = 1>
      <th>Column 1</th>
      <th>Column 2</th>
      <tr>
      <td>Hello Class</td>
      <td>Hello Again</td>
      </table>';
?>
```

Generisanje HTML koda pomoću petlji

```
<?php
    $users = array('Jason', 'Phil', 'Herbert', 'Anil');
    echo '<center>';
    echo '<table border = 2 >';
     echo '<th>Number</th><th>Username</th>';

    for ($i = 0; $i < count($users); $i++)
    {
      echo '<tr>';
      echo "<td>$i</td>";
      echo "<td>$users[$i]</td>";
      echo '</tr>';
    }
    echo '</table>';
?>
```

Obrada nizova

- Nizovi se mogu obrađivati pomoću petlji:

```
for ($i=0; $i < count($angles); $i++) {
   print “angle $i: “. $angles[$i];
}
```

```
foreach ($product as $key => $value) {
  print 'the ' . $key . ' is ' . $value;
  print '<br>';
}
```

Višedimenzionalni nizovi

- Element niza može biti podataka bilo kog tipa, uključujući i neki drugi niz
- Na taj način se definišu višedimenzionalni nizovi

```
$row1 = array('a','b','c');
$row2 = array('d','e','f');
$row3 = array('g','h','i');
// 2-dim niz
$matrix = array($row1, $row2, $row3);

echo $matrix[0][0];
echo $matrix[1][2];
echo $matrix[2][1];
$matrix[2][2] = 'I';
```

Višedimenzionalni nizovi

- Moguće je koristiti i indeksne i asocijativne nizove u okviru istog niza

```
$product1 = array('name' => 'small gumball', 'price' => 0.02);
$product2 = array('name' => 'med gumball', 'price' => 0.05);
$product3 = array('name' => 'large gumball', 'price' => 0.07);

// niz svih proizvoda
$products = array($product1, $product2, $product3);

for($i=0; $i<count($products); $i++)
    echo "Product: $i is {$products[$i]['name']} and
                   costs {$products[$i]['price']} each<br>";
```

Postojanje elementa

- Nekada je potrebno znati da li određeni element postoji u okviru niza. Tada se koristi funkcija **`array_key_exists()`**

```
$product = array(‘name’ => ‘small gumball’, ‘filling’ =>
   NULL, ‘price’ => 0.04);
if( array_key_exists(‘name’, $product) )
   echo “Product is $product[‘name’]”;
```

- Uporediti sa **`isset()`**?

```
array_key_exists(‘filling’, $product) // rezultat?

isset($product[‘filling’]) // rezultat?
```

Pretraga

- Testiranje da li je element u nizu može se izvršiti pomoću **`in_array()`**

```
if (in_array('small gumball', $product))
  print('we sell small gumballs');
```

- Može se izvršiti pretraga za vrednost ključa elementa pomoću funkcije **`array_search()`**

```
print "the key for small gumball is " .
  array_search('small gumball', $product);
```

Sortiranje

Postoji više načina da se izvrši sortiranje, na primer **`sort(),asort(),ksort()`**

```
$a = array('3', 'a', 'c', 'b', '2', '1');
sort($a);
```

Treba voditi računa o upotrebi ovih funkcija, jer se sortiranje obavlja direktno na samom nizu

Ključevi i vrednosti

```
$product = array(‘name’ => ‘small gumball’, ‘filling’ =>
   ‘solid’, ‘price’ => 0.04);
```

- Pomoću **`array_keys()`** dobija se niz ključeva nizova

```
$prodKeys = array_keys($product);
```

- Pomoću **`array_values()`** dobija se niz vrednosti niza

```
$prodValues = array_values($product);
```

Brisanje i umetanje elemenata

```
$product = array('small gumball','solid', 0.04);
```

- Moguće je kreirati novi niz

```
$prodNoFilling = array($product[0], $product[2]);
```

- Moguće je koristiti funkciju **`array_splice()`** da bi se obrisao ili umetnuo element direktno

```
array_splice($product, 1, 1); // obrisan element 1
array_splice($product, 1, 1, 'hollow'); //zamena elementa 1
array_splice($product, 1, 0, 'logo'); // umetanje na
   poziciju 1
```

Atribut Method HTML forme

- Vrednost atributa method može biti ‘POST’ ili ‘GET’.
- $_GET i $_POST su nizovi koji su ugrađeni u okviru PHPa i olakšavaju rad sa formama.
- Elementi forme se pojavljuju kao ključevi u okviru nizova, navedeni po imenu
- Da bi se video sadržaj niza koristi se var_dump (na primer var_dump($_GET);)

Login

```
<?php
print "logged in " . $_POST['username'];
?>
<form action = 'process_login.php' method =
  'post'>
<input type = 'text' name = 'username'>
<input type = 'password' name = 'password'>
<input type = 'submit' name = 'submit_button'
  value = 'login'>
</form>
```

Obrada na istoj stranici

- Ponekad je potrebno ostati na istoj stranici nakon obrade forme

```
<form action = ‘<?= $_SERVER[‘PHP_SELF’]?>’ method = ‘post’>
</form>
```

ili:

```
<form action = ‘<?php $_SERVER['PHP_SELF'] ?>’method = ‘post’>
</form>
```

- <?= je skraćena notacija za echo kada se PHP uključuje u HTML

Login

```
if( array_key_exists('submit_button', $_POST) ) {
     print "logged in " . $_POST['username'];
}
else {
?>
<form action = '<?= $_SERVER['PHP_SELF'] ?>' method =
   'post'>
<input type = 'text' name = 'username'>
<input type = 'password' name = 'password'>
<input type = 'submit' name = 'submit_button' value =
   'login'>
</form>

<?php } ?>
```

Login

```
if( array_key_exists('submit_button', $_POST) ) {
     print "logged in " . $_POST['username'];
}
else {
?>
<form action = '<?= $_SERVER['PHP_SELF'] ?>'
       method = 'post'>
<select name='username'>
<option>Moe</option>
<option>Larry</option>
<option>Curly</option>
</select>
<input type = 'password' name = 'password'>
<input type = 'submit' name = 'submit_button'
       value = 'login'>
</form>
<?php } ?>
```

Funkcije

- Funkcija je blok programa (unutar {}) koja izvršava određeni skup naredbi
  - Može da prihvata ulazne parametre
  - Može da vraća rezultat obrade
- Funkcije su korisne za
  - Iste operacije koje se pojavljuju na više lokacija
  - Smanjuju vreme izvršavanja
  - Modularizacija

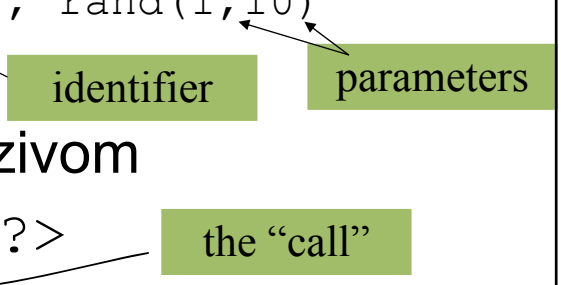
Poziv funkcije

- Funkcije u PHPu su definisane sa <identifier> (<parameters>)
  - Na primer. `phpinfo(), rand(1,10)`
  - – ***Nisu*** case sensitive
- Funkcije se koriste pozivom

```
<? phpinfo(); ?>
```

- Sintaksa:

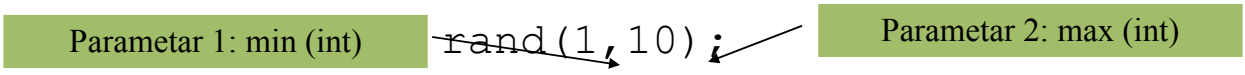
```
<return type> <identifier> ( <parameters> )
```



Parametri

- Funkcije su fleksibilne, jer mogu da prihvataju ulazne podatke
- Ulazni podaci koji se šalju funkciji se nazivaju ***parametri***

```
rand(int min, int max)
```



Promenljive kao parametri

- Bilo koji regularan izraz može se koristiti kao parametar

```
rand(44%3, 2*$max*$max);
```

- I promenljiva se može koristiti kao parametar

```
rand($minGuess,$maxGuess);
substr($name,1,$i);
```

Rezultat funkcije

- Neke funkcije samo obavljaju određenu operaciju (n.p. Čitanje vrednosti sa tastature, prelazak na web stranicu) i ne generišu rezultat
- Mnoge funkcije izvrše operaciju i generišu određenu vrednost
- Tip podataka koji generiše funkcija može biti:
  - prost tip podataka: `int`, `float`, `boolean`, ...
  - Složeni tip podataka `array`, `object`, ...
  - `void` ako se ne generiše rezultat

Rezultat funkcije

- Rezultat funkcije se može koristiti u okviru regularnih izraza
  - `$guess = rand(1,10);`
  - `3.14159*rand(1,10);`
  - `echo phpinfo();`
  - `$yummy = “piece of ” . substr(“piece”,0,3);`
  - `if( is_int($guess) ) …`
- Može se vratiti i vrednost boolean tipa
  - `true` ako je funkcija operacije izvršila korektno
  - `false` ako funkcija nije obavila željene operacije

```
if( !writeFile() ) echo “file not written”;
```

Definicija funkcija

- Prednost korišćenja PHPa je i mogućnost da se kreiraju i sopstvene funkcije.
- Funkcija se deklariše pomoću reči `function` ispred identifikatora funkcije sa skupom parametara :

```
function <identifier> ( <parameters> ) {
// kod funkcije

return <expression>
}
```

- Način na koji se definiše funkcije je interfejs funkcije

Sintaksa i primer

```
function funcname($var1, $var
    2...)
{
   statement 1;
   statement 2;
   statement 3;
   ...
   return $retval;
}
```

```
function square($num)
{
   $answer = $num * $num;
   return $answer;
}
```

Poziv funkcije:

```
$quantity = 3;
$myvar = square
   ($quantity);
```

$myvar ima vrednost 9

Globalne i lokalne promenljive

- Promenljive definisane u okviru funkcije nazivaju se “lokalne promenljive” i dostupne su samo u okviru funkcije
  - Nakon napuštanja funkcije promenljive ne postoje!!!
- Promenljive definisane izvan funkcije u opštem slučaju nisu dostupne unutar funkcije (razlog je da bi se izbegli konflikti i da bi promenljive bile dostupne funkciji kao argumenti)
- Globalne promenljive su dostupne i unutar i izvan funkcija
- Oblast važenja je deo programa gde je promenljiva dostupna
  - Treba koristiti specifikatore `global` and `static`

Primer

```
<?php
    function deposit($amount)
    {
      $balance += $amount;
      echo “New balance is $balance <BR>";
    }
    $balance = 600;
    deposit(50);
    echo "Balance is $balance";
?>
```

Šta je rezultat rada programa?

Primer

```
function swapTwoValues($a, $b) {
  $tmp = $a;
  $a = $b;
  $b = $tmp;
}

$var1 = 1; $var2 = 2;
echo “\$var1 is $var1 and \$var2 is $var2<BR>”;

swapTwoValues($var1, $var2);
echo “\$var1 is $var1 and \$var2 is $var2”;
```

Dodela po vrednosti i po referenci

- Pri pozivu funkcije, vrednost svakog argumenta se kopira i koristi lokalno sa funkcijom
- ***Promenljive koje se koriste kao argumenti ne mogu se menjati pomoću funkcije!!!!!!!!!***
- Jedna mogućnost promene vrednosti argumenta je korišćenje rezultata funkcije:

```
function doubler($value) {
   return 2*$value;
}
echo doubler($doubleUp);
$doubleUp = doubler($doubleUp);
```

Dodela po vrednosti i po referenci

- Nekada je potrebno direktno raditi nad promenljivom, i tada se izvršava dodela po referenci (adresi):

```
function swapTwoValues(&$a, &$b) {
  $tmp = $a;
  $a = $b;
  $b = $tmp;
}
$var1 = 1; $var2 = 2;
echo “\$var1 is $var1 and \$var2 is $var2<BR>”;
swapTwoValues($var1, $var2);
echo “\$var1 is $var1 and \$var2 is $var2”;
```

Validacija podataka

- Prvo je potrebno proveriti da li je forma poslata, moguće je koristiti funkciju `array_key_exists()` u okviru dugmeta pomoću koga je forma submit-ovana
- Korišćenje funkcije je korisno za validaciju, zbog mogućnosti upotrebe u okviru više različitih formi:

```
<?php
function validate_price($value)
{
   if(!isset($errors)) $errors = array(); // init empty array if not defined already
   if( !is_numeric($value) ) $errors['not_number'] = "not numeric";
   if( $value - round($value, 2) != 0 ) $errors['not_dollar'] = "not a dollar amount";
   if( $value < 0 ) $errors['not_non-negative'] = "price cannot be negative";

   return $errors();
}
?>
```

Validacija podataka

- Uobičajeno da je da se koristi globalna promenljiva koja bi sadržala greške, tako da bude dostupna unutar i izvan funkcije.

```
function validate_price($value)
{
global $errors;
   if(!isset($errors)) $errors = array(); // init empty array if not
      defined already
   if( !is_numeric($value) ) $errors['not_number'] = "not numeric";
   if( $value - round($value, 2) != 0 ) $errors['not_dollar'] = "not a
      dollar amount";
   if( $value < 0 ) $errors['not_non-negative'] = "price cannot be
      negative";
}
```

- Pri validaciji treba obratiti pažnju da se podaci dobijaju u formi stringa. Primer testiranja da li je uneti string u formatu broja

```
is_int($a - (int) $a)
```

Hidden vrednosti

- Da bi se određene vrednosti razmenjivale između stranica moguće je koristiti *hidden* element forme
- Samo vrednosti tipa string se mogu slati i primati
  - Pre slanja je potrebno konvertovati u string
  - Funkcije `urlencode(), serialize()` se mogu koristiti pri konverziji složenijih tipova, kao što su nizovi. Ako se koriste ove funkcije onda se pomoću funkcija `urldecode(), unserialize()` dobijaju originalne vrednosti koje su kao sting poslate u okviru `$_POST` ili `$_GET` nizova

Hidden vrednosti

```
<? $things = array('one thing', 'two thing'); ?>
<form action= "<?= $_SERVER['PHP_SELF'] ?>" method='POST'>
<input type='hidden' name='secret' value=96>
<input type='hidden' name='stuff' value='<?=
  urlencode(serialize($things)) ?> ' >
<input type='submit'>
</form>
```

Nakon slanja forme

```
$_POST['secret'] = ???
$_POST['stuff'] = ??
$things = unserialize(urldecode($_POST['stuff'] ));
```

Promenljive

- Informacije sa web servera su dostupne pomoću EGPCS
  - Environment, GET, POST, Cookies, Server
- PHP kreira nizove sa EGPCS informacijama
  - `$HTTP_COOKIE_VARS, $HTTP_GET_VARS, $HTTP_POST_VARS, ...`
  - Ovi nizovi su globalni. Čak i ako se koriste u okviru funkcija
- PHP definiše i `$_SERVER['PHP_SELF']` pomoću koga se referiše na trenutni skript fajl, što je korisno kod formi koje sadrže i obradu sopstvenih podataka

Server Info

- Veliki broj informacija o serveru i korišćenom čitaču je dostupno pomoću `$_SERVER` niza
  - SERVER_NAME
  - REQUEST_METHOD
  - QUERY_STRING
  - REMOTE_ADDR
  - PHP_SELF
  - ....

Primer: Header prosleđivanje

- Moguće je preusmeriti korisnika na drugu stranicu pomoću header() funkcije.

```
header('Location: http://mysite.com/myfile.php');
```

- Na ovaj način će se u okviru header-a upisati `'Location: http://mysite.com/myfile.php'`
  - Dobijeni efekat je da se stranica myfile.php učita
  - *Napomena*: uvek koristiti protokol kao što je http:// ili file:// da bi bili sigurni da će se izvršiti pravi dokument

Još header primera

- Mogu se proslediti podaci tokom preusmeravanja pomoću `$_GET` niza

```
header('Location:myfile.php?name=Dan&s
  core=98&grade=A');
```

- Može se zabraniti pristup stranici

```
header('WWW-Authenticate:Basic
  realm="My Website"');
header('HTTP/1.0 401 Unauthorized');
```

Implementacija Back opcije

- Back opciju moguće je realizovati na više načina
  - Hyperlink

```
<A href="<?= $_SERVER['HTTP_REFERER'] ?>">BACK</A>
```

  - Submit Button

```
<form action='<?= $_SERVER['HTTP_REFERER'] ?>'>
<INPUT TYPE="SUBMIT"  value="back">
</form>
```

  - Java script history action u okviru dugmeta

```
 <FORM>
<INPUT TYPE="button" VALUE="Back!" onClick="history.go(-1)">
</FORM>
```

Novi prozor

- Nekada je potrebno otvoriti novi prozor umesto zamene dosadašnjeg

```
<FORM  action="./action_process.php" method="POST" target="_blank">
<INPUT TYPE="TEXT"  name="stuff_input_field">
<INPUT TYPE="SUBMIT"  value="open new window">
</FORM>
```

./action_process.php

```
<?php echo 'You entered ' . $_POST['stuff_input_field']; ?>
```

- Šta se dešava ako se koristi

<FORM action="<?= $_SERVER['PHP_SELF'] ?>" method="POST" target="_blank">

Nizovi u okviru HTML formi

- Ako se u okviru iste forme koriste ista imena i znakovi [], realizovaće se niz (od bilo kog ulaznog tipa). Elementi niza su samo one vrednosti koje sadrže neke podatke.

```
<FORM  action="<?= $_SERVER['PHP_SELF'] ?>" method='post'>
        <INPUT TYPE="TEXT"  name="a[]">
        <INPUT TYPE="TEXT"  name="a[]">
        <INPUT TYPE="TEXT"  name="a[]">
        <INPUT TYPE="SUBMIT">
</FORM>

<?
        var_dump($_POST);
?>
```

Asocijativni nizovi

- Mogu se kao indeksi koristiti i stringovi (korisno pri direktnoj asocijaciji podatka niza)

```
<FORM  action="<?= $_SERVER['PHP_SELF'] ?>"
      method='post'>
      <INPUT TYPE="TEXT"  name="a[name]">
      <INPUT TYPE="TEXT"  name="a[price]">
      <INPUT TYPE="TEXT"  name="a[description]">
      <INPUT TYPE="SUBMIT">
</FORM>

<?
      var_dump($_POST);
?>
```

Indeksni nizovi i kreiranje elemenata forme

- Upotreba integer vrednosti kao indeksa omogućava automatsko generisanje elemenata forme

```
<FORM  action='<?= $_SERVER['PHP_SELF'] ?>' method='post'>
<?php
var_dump($_POST);
$size = 10;
for($i=0; $i<$size; $i++){
      echo "<br>checkbox $i <INPUT TYPE='CHECKBOX'
      name='a[$i]'>";
}
?>
<INPUT TYPE="SUBMIT">
</FORM>
```

- Korisno kada je potrebno tačno znati koji element je neprazan (u gornjem primeru koje polje za potvrdu je potvrđeno)

Provera niza checkbox elemenata

```
<? if(array_key_exists('a', $_POST)) {
    $selections = $_POST['a'];
    foreach($selections as $key => $value)
       if ($selections[$key] == 'on')
          echo "<br>you selected box $key";
    exit;
   }
?>
   <FORM  action='<?= $_SERVER['PHP_SELF'] ?>'
   method='post'>
   <?php
   $size = 10;
   for($i=0; $i<$size; $i++){
       echo "<br>checkbox $i <INPUT TYPE='CHECKBOX'
   name='a[$i]'>";
   } ?>
   <br><INPUT TYPE="SUBMIT">
   </FORM>
```

Rad sa greškama

```php
<?php
define('MIN_PASS_LEN', 3);
define('MAX_PASS_LEN', 10);

function check_pass($pword) {
    global $errors;
    if (strlen($pword) < MIN_PASS_LEN ) $errors['password_short'] = 'Enter a longer password';
    if (strlen($pword) > MAX_PASS_LEN ) $errors['password_long'] = 'Enter a shorter password';
}

$username = 'user';
$password = 'pass';
$errors = array();

if(array_key_exists('form_data', $_POST)) {
    check_pass($_POST['password']);
    if(count($errors) == 0 && $_POST['username'] == $username && $_POST['password'] == $password) {
        die('correct!!');
    } else {
        echo 'wrong';
    }
}
?>
<form action = '<?= $_SERVER['PHP_SELF'] ?>' method= 'POST'>
Username: <br>
<INPUT TYPE="TEXT"  name="username" value = "<? if(isset($_POST['username'])) echo $_POST['username']
?>"><br>
Password: <br>
<INPUT TYPE="password" name = 'password'>
<?
if(isset($errors['password_short'])) echo " <font color='red'>{$errors['password_short']}</font>";
if(isset($errors['password_long'])) echo " <font color='red'>{$errors['password_long']}</font>";
?>
<br><br>
<INPUT TYPE="HIDDEN" name = 'form_data' value='submitted'>
<INPUT TYPE="SUBMIT" name = 'submit'>
</form>
```

Baze podataka

- Baze podatka su skupovi podataka organizovani na određeni način
- Najviše su korišćene relacione baze podataka.
- Ali ne moraju sve baze podataka da budu i relacione.

Relacione baze podataka

- Relacione baze podataka sadrže skup tabela, između kojih mogu postojati određene realcije.
- Svaka tabela sadrži određeni broj zapisa (record). Svaki zapis ima određeni broj polja (fields).
- Pri kreiranju baze podatka, definiše se
  - Veličina svakog polja
  - Relacije između tabela

Baza filmova

ID	*title*	*director*	*date*
M1	Gone with the wind	F. Ford Coppola	1963
M2	Room with a view	Coppola, F Ford	1985
M3	High Noon	Woody Allan	1974
M4	Star Wars	Steve Spielberg	1993
M5	Alien	Allen, Woody	1987
M6	Blowing in the Wind	Spielberg, Steven	1962

- Jedna tabela
- Naravno, ne postoje relacije između tabela

Problem sa prethodnim primerom

- Svi podaci su pogrešni, ali služe samo za ilustraciju.
- Korišćenje imena nije konzistentno. Nije moguće pronaći sve filmove Woody Allan-a bez provere svih varijacija.
- Greške je teško otkloniti. Potrebno je proći kroz sve zapise, veoma naporno.

Bolja verzija

ID	*title*	*director*	*year*
M1	Gone with the wind	D1	1963
M2	Room with a view	D1	1985
M3	High Noon	D2	1974
M4	Star Wars	D3	1993
M5	Alien	D2	1987
M6	Blowing in the Wind	D3	1962

ID	*director name*	*birth year*
D1	Ford Coppola, Francis	1942
D2	Allan, Woody	1957
D3	Spielberg, Steven	1942

Relaciona baza podataka

- Postoji relacija jedan ka više (one to many) između režisera i filma
  - Svaki film ima jednog režisera
  - Svaki režiser je režirao više filmova
- Sada je moguće da računar odredi
  - Koje filmove je režirao Woody Allen
  - Koje filmove je režirao režiser koji je rođen 1942

više-ka-više relacija

- Svaki film ima jednog režisera, ali postoji više glumaca. Relacija između glumca i filma je više-ka-više relacija (many-to-many).

ID	*sex*	*actor name*	*birth year*
A1	f	Brigitte Bardot	1972
A2	m	George Clooney	1927
A3	f	Marilyn Monroe	1934

Glumac/Film tabela

actor id	*movie id*
A1	M4
A2	M3
A3	M2
A1	M5
A1	M3
A2	M6
A3	M4
...	

Upotreba baza podatka

- Relacione baze podataka su dominantne kada se koriste strukturirani podaci
- Nisu toliko popularne kod poslova kao što je biblioteka
  - Sporije kod velike količine podataka
  - Biblioteke imaju nestrukturirane relacije.
    - Prevod prve edicije knjige
    - CD dodatak koji se dobija uz knjigu

  Dobro se ponašaju u situacijama kada se polja i relacije mogu definisati na početku rada, kasnije je teško promeniti početnu strukturu.

Baze podataka i web sajt

- Većina web aplikacija koriste relacione baze podataka
- Dizajn same aplikacije zavisi od dizajna baze.
- Primer prodavnice
  - kupci
  - proizvodi
  - narudzbine
  - narudzbine _proizvodi

    Relacija više-ka-više između narudzbina i proizvoda.

mySQL

- Najpoznatija i najviše korišćena open-source baza podataka zasnovana na SQL jeziku.
- Korišćenje ove baze podataka je besplatno.
- U daljem delu nastave koristiće se mySQL upotreba SQL naredbi.

phpmyadmin

- phpmyadmin je skup PHP scriptova koji kreiraju opšti interfejs za rad sa mySQL bazom podataka.
- Napisan je pomoću PHP jezika.
- http://wotan.liu.edu/phpmyadmin.
- Potreban je nalog. To nije wotan nalog, već dozvola za korišćenje baze podataka na mySQL serveru koji se nalazai na sajtu wotan.

Kreiranje mySQL baze

- Moguće je direktno u mySQL kreirati nalog. Nakon pristupa mySQL-u kao root korisnik, potrebno je sledeće

  GRANT ALL *ON user_name.** TO *user_name* IDENTIFIED BY *'secret_word'* WITH GRANT OPTION;

- *user_name* je vaše korisničko ime. To je ime baze, ali i ime pomoću koga možete pristupati bazi.

mySQL

- mySQL je instalirana na wotan sajtu.
- Uobičajeno je da se korsinik uloguje na wotan nalog i upotrebi komandu za korišćenje karakter interfejsa.
- Komanda je
  - mysql -u *user -p*

Velika i mala slova

- Praksa je da se SQL komande pišu velikim slovima.
- mySQL komande su case-insensitive
- Ali promenljive u okviru komandi su case-sensitive. U primerima će se koristiti mala slova za komande.

Kreiranje baze

- CREATE DATABASE je mySQL komanda za kreiranje nove baze podataka.
- Primer

  CREATE DATABASE primer;
- Kreira bazu podataka pod imenom primer

GRANT

- Ova komada kreira korisnika i dodeljuje mu određene privilegije.

  GRANT *privileges* ON *item* TO *user_name*

  [IDENTIFIED BY '*password*'] [WITH GRANT OPTION]
- Ako se koristi WITH GRANT OPTION, dozvoljava se korisniku da dodeljuje drugim korisnicima privilegije koje i sam ima.

Korisničke privilegije

- SELECT dozvoljava korisnicima da selektuju (čitaju) zapise iz tabela.
- INSERT dozvoljava korisnicima da upisuju nove zapise u tabelu.
- UPDATE dozvoljava korisnicima da menjaju postojeće zapise u tabelama.
- DELETE dozvoljava korisnicima da brišu zapise iz tabela
- INDEX dozvoljava korisnicima da indeksiraju tabele

Korisničke privilegije

- ALTER dozvoljava korisnicima da menjaju strukturu baze podataka.
  - Dodavanje novih kolona
  - Promena imena kolona ili tabela
  - Promena tipa podataka u okviru tabela
- DROP dozvoljava brisanje baza podataka ili tabela.

Korisničke privilegije

- CREATE dozvoljava korisnicima da kreiraju nove baze podataka ili tabele. Ako se određena tabela ili baza pominje u GRANT naredbi, korisnik može da kreira samo tu bazu ili tabelu, što znači da mora prvo da je obriše.
- USAGE daje minimalne privilegije. Ova opcija je korisna ako se želi samo kreirati novi korisnik.

REVOKE

- Ova naredba je suprotnog dejstva od naredbe GRANT.

Primer

CREATE DATABASE *user_name;*

GRANT * ON *user_name* TO *user_name* IDENTIFIED BY '*secret_word'* WITH GRANT OPTION;

- Gde je
  - user_name korisničko ime
  - secret_word šifra
  - * označava sva prava

Kreiranje tabele

```
CREATE TABLE customers (customer_id
INT UNSIGNED
 NOT NULL AUTO_INCREMENT PRIMARY KEY,
   name
 CHAR(50) NOT NULL, address CHAR(100) NOT
   NULL, email CHAR(40), state CHAR(2) NOT
   NULL);
```

Tipovi podataka kolona

- TINYINT označava broj između -128 i 127 ili između 0 i 255. BIT ili BOOL su sinonimi za TINYINT.
- SMALLINT označava broj između -32768 i +32767 ili 0 i 65535
- INT označava broj između -2**31 i 2**31-1 ili između 0 i 2**32-1. INTEGER je sinonim za INT.
- BIGINT označava broj između -2**63 i 2**61-1 ili između 0 i 2**64-1.

Tipovi podataka kolona : float

- FLOAT je racionalan broj prikazan sa 4 B
- DOUBLE je racionalan broj prikazan sa 8 B
- DECIMAL(*x*,*y*) gde je *x* broj cifara pre decimalne tačke, a *y* broj cifara posle decimalne tačke.

Tipovi podataka kolona: datumi

- DATE je dan između 1000-01-01 i 9999-12-31.
- TIME je vreme između -838:59:59 i 838:59:59
- DATETIME je datum i vreme, sa prikazom YYYY-MM-DD HH:MM:SS
- TIMESTAMP je broj sekundi od 1970-01-01 u 0 časova. Ovaj broj se prikazuje za datume do 2037.

Opcije polja

- PRIMARY KEY označava da je data kolona primarni ključ tabele. Može postojati samo jedna takva kolona. Vrednosti u koloni moraju biti jedinstvene.
- AUTO_INCREMENT se može koristiti kod kolona koje sadrže celobrojne vrednosti.
- NOT NULL definiše da polje ne može bez upisane vrednosti.

USE

- USE *database* definiše mySQL da počinje rad sa bazom *database.*
- Ako se ne koristi USE komanda, i dalje se može pristupati tabeli pomoću naredbe *database.table*, gde je *datebase* ime baze, a *table* ime tabele. Tačka se koristi da bi se povezala tabela sa bazom.

Pristup kolonama

- Ako postoji baza *database* sa tabelom *table* i kolonom *column.* Tada se koloni pristupa sa *database.table.column*
- Ako se pre ove naredbe koristi naredba USE *database* before, tada se može izostaviti *database* deo.

INSERT

- INSERT upisuje novi zapis u tabelu.

  INSERT INTO *table* VALUES (*value1, value2,* ..);

- Primer:

  INSERT INTO products VALUES ('',‘Banane',1.23);

- U primeru za vrednost prve kolone je korišćen prazan string jer je kolona definisana kao auto_increment.

SELECT

- SQL naredba pomoću koje se slektuju zapisi iz tabele.

SELECT [*options*] *columns* [INTO *file_details*]
FROM *table* [WHERE *conditions*]
[GROUP BY *group_type*]
[HAVING *where_definitions*]
[ORDER BY *order_type*] [LIMIT *limit_criteria*]
[PROCEDURE *proc_name*(*arguments*)]
[*lock_options*]

SELECT

- Može se selektovati više kolona, tada se moraju razdvojiti zarezom

  SELECT name, price FROM products;
- Mogu se selektovati i sve kolone

  SELECT * FROM products;

WHERE *uslov*

- = označava jednakost

  WHERE id = 3
- >, <, >=, <= i != imaju uobičajeno značenje
- IS NULL ispituje da li je vrednost NULL
- IS NOT NULL
- IN dozvoljava da se definiše skup vrednosti

  WHERE state IN ("NY","NJ","CT")

SELECT i više tabela

- *table1,table2* se može koristiti da bi se spojile obe tabele u jednu veću koja se može obrađivati

```
SELECT orders.id FROM customers, orders
WHERE customers.id= 3
```

- Ovako definisan join je primer potpunog spajanja. Na svaki zapis iz prve tabele dodaje se zapis iz druge.

ORDER

- Moguće je sortirati podatke pomoću ORDER BY.
- Može se koristiti opcije ASC ili DESC da bi se definisao način sortiranja.

```
SELECT name, address FROM customers
  ORDER BY name ASC
```

Funkcije koje koriste kolone

- AVG(*column*) daje prosečnu vrednost kolona
- COUNT(*column*) daje broj ne NULL vrednosti u okviru kolone
- COUNT(DISTINCT *column*) daje broj različitih vrednosti u okviru kolone
- MIN(*column*), MAX(*column*)
- STD(*column*) daje standardnu devijaciju
- SUM(*column*) zbir vrednosti kolona

Funkcije koje koriste kolone

- Funkcije se mogu koristiti

  SELECT AVG(amount) FROM orders;
- Slekecija se može i grupisati. Na primer, pronaći minimum za svakog potrošača

  SELECT MIN(amount) FROM orders

  GROUP BY customerid;
- Mogu se koristiti i u okviru opcije HAVING:

  SELECT customerid FROM orders

  HAVING AVG(amount) > 10;

LIMIT

- Može se koristiti da bi se limitirao broj zapisa.

  LIMIT 10 19
- Korisno je za web sajtove gde se pokazuje selekcija rezultata

Promena vrednosti

- Opšta sintaksa je
  UPDATE [LOW_PRIORITY] [IGNORE] *table* SET *column1=expression1, column2=expression2...* [WHERE *condition*] [ORDER BY *order_criteria*] [LIMIT *number*] ;
- Primer
  UPDATE students SET email= ‘nbosko@etf.bg.ac.yu'
  WHERE name=‘Bosko Nikolic';
- IGNORE se koristi da bi se ignorisale greške.
- LOW_PRIORITY definiše rad sa manjim prioritetom, ako je server zauzet – čeka se.

Brisanje zapisa

- Opšta sintaksa je

  DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM *table* [WHERE *condition*] [ORDER BY *order_criteria*] [LIMIT *number*]
- Loš primer

  DELETE FROM customers;
- Dobar primer

  DELETE FROM customers WHERE

  customer.name=‘Bosko Nikolic'

PHP mySQL funkcije

- Koristiće se nova verzija PHP mySQL funkcija, počev sa mysql_
- Interfejs je objektno-orijentisan, ali se može pristupai i na ne-objektno-orijentisani način. Ovakav pristup se još naziva i proceduralni pristup.
- Online dokumentacija http://php.net/mysql

mysql_connect()

- Koristi se da bi se uspostavila konekcija sa mySQL serverom. mysql_connect('*host*', '*user*', '*password*');
- Primer

  $link= mysql_connect('localhost',‘nbosko',‘bbb888');
- Može se koristiti localhost da bi se pristupilo lokalnoj bazi podataka, ali se uz odgovarajuću dozvolu može pristupiti bilo kom Internet hostu.
- Rezultat funkcije je promenljiva tipa “resource”. Ako je došlo do greške, rezultat je false.

mysql_error()

- Rezultat ove funkcije je greška koja se dogodila tokom izvršavanja poslednje mySQL naredbe. Rezultat je false ako se greška nije dogovodila.

```
$error=mysql_error();
if($error) {
  print "mySQL error: $error<br/>";
}
```

- Vrednost dobijena kao rezultat funkcije je string.
- Ovo je dobar način za proveru grešaka.

mysql_select_db()

- Sintaksa ove naredbe je mysql_select_db('*database*')

gde je *database* ime baze*.*

- Rezultat je tipa Boolean.
- Ova naredba obaveštava mySQL da se želi pristup bazi *database.*

  mysql_select_db(‘primer');
- Ova naredba ima isti efekat kao naredba

  USE beer_shop*;*

  u okviru mySQL*.*

mysql_query()

- mysql_query(*query*) šalje upit *query* ka mySQL.

```
$link = mysql_connect("localhost", "shop_owner",
   "bruch");
$query="SELECT * FROM beer_shop.customers";
$result=mysql_query($query);
```

- Upit se ne mora završavati simbolom ;
- Rezultat upita je smešten u okviru promenljive $result.

Rezultat mysql_query()

- Za upite tipa SELECT, SHOW, DESCRIBE ili EXPLAIN, rezultat mysql_query() je resource kome se može pristupiti sa mysql_fetch_array().
- Za upite tipa UPDATE, DELETE, DROP i druge, rezultat mysql_query() je vrednost logičkog tipa.

Rezultat

- Vrednost dobijena sa mysql_fetch_array(*result*)

je niz sa zapisima koji su dobijeni kao rezultat upita. Rezultat je u formi niza koji sadrži kolone kojima se može pristupa i pomoću broj ai pomoću imena:

```
while($columns=mysql_fetch_array($result)) {
  print 'name: '.$columns['name'];
  print 'first column: $columns[0];
}
```

Rezultat

- mysql_data_seek(*result, number*) postavlja niz koji je dobijen samysql_fetch_array u broj *number.*

```
while($row=mysql_fetch_array($result)) {
  print 'first column: '.$row[0];
}
mysql_data_seek($result,0);
```

mysql_real_escape_string()

- Rezultat mysql_real_escape_string(*string) je string*.

```
$name="John O'Guiness";
$s_name=mysql_real_escape_string($name);
print $s_name; // rezultat: John O\'Guiness
```

- Ova funkcija pristupa mySQL, pa veza sa bazom mora biti uspostavljena.

mysql_close()

- Ovo je komanda za zatvaranje konekcije. Ako se ne navodi argument, zatvara se trenutna konekcija.
- Ova komanda govori da je rad sa bazom završen.
- Takođe, ne koristi se često, zato što se mySQL konekcija zatvara automatski kada se završi izvršavanje skripta.

extra: sha1()

- Ova funkcija pravi različite kombinacije 40 karaktera iz stringa.
- Rezultat izvršavanja sha1() ne može se kasnije prevesti u originalni string.
- Ova funkcija je korisna za kreiranje šifri.
  - $s_password=sha1($password);

Složeni upiti

```
SELECT customer.id from customers, orders,
orders_items, products WHERE
customers.id=orders.customer_id AND
orders.id=orders_items.order_id AND
orders_items.item_id=products_id AND
products.name='Bruch Landbock'
```

left join

```
SELECT customers.name FROM customers LEFT
  JOIN
orders ON customers.id = orders.customerid AND
  orders.id
IS NULL
```

- U tabelu dobijenu sa joint upisuju se vrednosti NULL za one costumers koji još nisu sortirani.

Primer

- Table A

A1	A2
1	4
4	5
6	3

Table B

B1	B2	B3
2	3	4
6	7	3
1	1	4

- Left outer join na A2 i B3

A1	A2	B1	B2	B3
1	4	2	3	4
1	4	1	1	4
4	5			
6	3	6	7	3

aliasi

- Službena reč AS se može koristiti za kreiranje aliasa. Ako se želi pronaći koji kupci žive u istom gradu kao neki drugi kupci

```
select c1.name, c2.name, c1.city
FROM customers AS c1, customers AS c2
WHERE c1.city = c2.city AND c1.name != c2.name
```

utility funkcija iz php.net

```
function mysql_fetch_all($query) {
  $r=@mysql_query($query);
  if($err=mysql_error()) { return $err;}
  if(mysql_num_rows($r)) {
    while($row=mysql_fetch_array($r)) {$result[]=$row; }
    return $result;}}
// usage
if(is_array($rows=mysql_fetch_all($query)) {
 // do something
}
else { if (! is_null($rows)) {
        die("Query failed!");}
}
```

Sesije

- HTTP je “stateless” protokol.
- Na ovaj način korisnik mora da izvršu identifikaciju svaki put kada pristupa novoj stranici.
- Jedan način da se implementira ova mogućnost je i korišćenje cookie-ja.
- PHP koristi cookie-je da bi realizovao koncept svojih sopstvenih, sesija, koje olakšavaju ove operacije.

cookie

- Cookie je atribut/vrednost određenih podataka. Server može da pošalje cookie-je kao vrednost HTTP header-a: Set-Cookie:.
- Kada korisnik ponovo poseti sajt, cookie se pošalje natrag serveru sa HTTP header Cookie

Kreiranje Cookie-ija

- Sintaksa funkcije `setcookie()` je:

  `setcookie(`*`name`* `[`*`,value ,expires, path, domain, secure`*`])`
- Mora se proslediti svaki od argumenata u specificiranom redosledu
- Može se izbeći prosleđivanje argumenata `value`, `path`, i `domain`, specificiranjem praznog stringa kao vrednosti
- Može se izbeći prosleđivanje argumenata `expires` i `secure`, specificiranjem 0 kao vrednosti

setcookie();

- Treba pozvati funkciju `setcookie()` pre slanja Web browser-u bilo kakvih izlaznih informacija
- Korisnik može da bira da li će prihvatiti da se cookie upiše na njegov sistem
- Rezultat true funkcije se vraća čak i ako korisnik odbije cookie

`name` i `value` argumenti

- Cookie-iji kreirani sa argumentima `name` i `value` u okviru `setcookie()` funkcije su temporalni cookie-iji zato što su dostupni samo tokom trenutne browser sesije

```
<?php
setcookie(“firstName”, “Don”);
?>
<!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 Strict//EN”
    “http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd”>
<html xmlns=”http://www.w3.org/1999/xhtml”>
<head>
<title>Skyward Aviation</title>
...
```

Upotreba Cookie-ija

- Funkcija `setcookie()` se može pozivati više puta da bi se kreirali novi cookie-iji – svaki od njih se mora pojaviti pre bilo kakvog izlaza na Web stranici

```
setcookie("firstName", "Don");
setcookie("lastName", "Gosselin");
setcookie("occupation", "writer");
```

`expires` argument

- Argument `expires` definiše koliko dugo će cookie ostati na klijentskoj mašini pre nego što se obriše
- Cookie-iji koji se kreiraju sa argumentom `expires` su dostupni samo tokom trenutne browser sesije
- Da bi se specificiralo vreme brisanja cookie-ija, koristi se funkcija `time()`

```
setcookie("firstName", "Don", time()+3600);
```

`path` Argument

- Argument `path` dozvoljava pristup cookie-iju i drugim Web stranama na serveru
- Pomoću `path` argumenata dozvoljava se deljenje cookie-ija između stranica na serveru

```
setcookie(“firstName”, “Don”, time()+3600, “/marketing/”);
setcookie(“firstName”, “Don”, time()+3600, “/”);
```

`domain` Argument

- Argument `domain` se koristi da bi se delili cookie-iji između više servera na istom domenu
- Cookie-iji se ne mogu deliti izvan domena

```
setcookie("firstName", "Don", time()+3600, "/",
   ".gosselin.com");
```

`secure` Argument

- Argument `secure` definiše slanje cookie samo pomoću bezbedne Internet konekcije
- Vrednosti ovog argumenta su 1 (true) ili 0 (false)

```
setcookie("firstName", "Don", time()+3600, "/",
   ".gosselin.com", 1);
```

Čitanje cookie-ija

- Cookie-iji su dostupni trenutnoj Web stranici preko `$_COOKIE` promenljive
- Svakom cookie-iju se pristupa pomoću imena kao ključa u okviru asocijativnog `$_COOKIE[]` niza

```
echo $_COOKIE['firstName'];
```

Provera cookie-ija

- Da bi se utvrdilo da je cookie postavljen, koristi se `isset()` funkcija

```
setcookie("firstName", "Don");
setcookie("lastName", "Gosselin");
setcookie("occupation", "writer");
if (isset($_COOKIE['firstName'])
     && isset($_COOKIE['lastName'])
     && isset($_COOKIE['occupation']))
     echo "{$_COOKIE['firstName']}
  {$_COOKIE['lastName']}
          is a {$_COOKIE['occupation']}.";
```

Rad sa nizovima

- Može se koristiti i sintaksa višedimenzionalnih nizova da bi se pročitala pojedinačna vrednost cookie

```
setcookie("professional[0]", "Don");
setcookie("professional[1]", "Gosselin");
setcookie("professional[2]", "writer");
if (isset($_COOKIE['professional']))
     echo "{$_COOKIE['professional'][0]}
          {$_COOKIE['professional'][1]} is a
          {$_COOKIE['professional'][2]}.";
```

Brisanje cookie-ija

- Da bi se obrisali stalni cookie-iji pre nego što istekne vreme definisano u okviru `expires` argumenta, postavi se nova vrednost koja specificira datum iz prošlosti

```
setcookie("firstName", "", time()-3600);
setcookie("lastName", "", time()-3600);
setcookie("occupation", "", time()-3600);
```

Startovanje sesije

- Funkcija `session_start()` startuje novu sesiju ili nastavlja postojeću
- Funkcija `session_start()` generiše jedinstveni ID sesije radi identifikacije
- **session ID** je slučajan string:

  `7f39d7dd020773f115d753c71290e11f`
- Funkcija `session_start()` kreira tekst fajl na Web server koji se zove kao i session ID, sa prefiksom `sess_`

Startovanje sesije

- **Funkcija `session_start()` mora se poslati Web browseru pre bilo kakvog izlaza**
- Ako klijent prihvata cookie-ije, session ID je pridružen trenutnom cookie-iju koji se zove `PHPSESSID`

Startovanje sesije

```
<?php
session_start();
...
?>
<p><a href='<?php echo
  "Occupation.php?PHPSESSID="
      . session_id() ?>'>Occupation</a></p>
```

Rad se promenljivama sesije

- Informacije o sesiji su smeštene u okviru promenljive `$_SESSION`
- Kada se pozove funkcija `session_start()` PHP inicijalizuje novu promenljivu `$_SESSION` ili vraća promenljivu za trenutnu sesiju (baziranu na session ID) u okviru promenljive `$_SESSION`

Rad se promenljivama sesije

```
<?php
session_start();
session_set_cookie_params(3600);
$_SESSION['firstName'] = "Don";
$_SESSION['lastName'] = "Gosselin";
$_SESSION['occupation'] = "writer";
?>
<p><a href='<?php echo "Occupation.php?"
 . session_id() ?>'>Occupation</a></p>
```

Provera sesije

```
<?php
session_start();
if (isset($_SESSION['firstName']) &&
  isset($_SESSION['lastName'])
    && isset($_SESSION['occupation']))
   echo "<p>" . $_SESSION['firstName'] . " "
        . $_SESSION['lastName'] . " is a "
        . $_SESSION['occupation'] . "</p>";
?>
```

Brisanje sesije

- Ručno brisanje sesije:

  1. Izvrši se funkcija `session_start()`

  2. Koristi se `array()` da bi se reinicijalizovala promenljiva `$_SESSION`

  3. Koristi se funkcija`session_destroy()` da bi se obrisala sesija

Brisanje svih sesija!

```
<?php
session_start();
$_SESSION = array();
session_destroy();
?>
```

Sesije

- Sesije su mogućnosti PHP-a. PHP pamti sesiju pomoću specijalnog cookie-ija PHPSESSID.
- Da bi se aktivirala sesiji, mora se navesti session_start(); na početku skripta, pre bilo koje naredbe za prikaz na ekranu.
- Kada se sesija aktivira, može se pristupiti pomoću specijalne super-global promenljive $_SESSION.

$_SESSION

- Ovo je niz kome se može pristupiti i postvaljati vrednosti koje se čuvaju tokom sesije.

```
if($_SESSION[user_name]) {
  print "welcome $_SESSION[user_name]";
}
else {
  // show users login form
  print login_form();
}
```

Kraj sesije

- Da bi se uklonila sesija potrebno je pozvati naredbu

  session_destroy()

U okviru skripta.

visit.php

```
<?php
$top='<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html><head><title></title><meta http-equiv="content-type"
content="text/html; charset=UTF-8"/>
</head><body><div>';
$bottom='</div><p>
<a href="http://validator.w3.org/check?uri=referer">
<img style="border: 0pt" src="/valid-xhtml10.png"
    alt="Valid XHTML 1.0!" height="31" width="88" />
</a></p></body></html>';
```

visit.php

```
session_start();
$current=mktime();  // look at the current time
if($_SESSION[last_click]) {
  $passed=$current-$_SESSION[last_click];
  $to_print.="$passed seconds have passed since your last
    visit.\n";
  $_SESSION[last_click]=$current;
} else {
  $to_print="This is your first visit.\n";
  $_SESSION[last_click]=$current;
  }
print "$top\n$to_print\n$bottom";
?>
```

Kreiranje klasa

- Da bi se kreirala klasa u okviru PHP-a, koristi se naredba `class` u okviru definicije klase
- Definicija klase sadrži promenljive i metode klase

```
class ClassName {
data member and member function
  definitions
}
```

Imena klasa

- ***ClassName*** definiše ime nove klase
- Uobičajeno je da imena klasa počinju sa velikim slovom da bi se razlikovali od ostalih identifikatora

```
class BankAccount {
      data member and member function definitions
      }
      $Checking = new BankAccount();

echo ('The $Checking object is instantiated from the '
        . get_class($Checking) . " class.</p>");
```

Smeštanje klasa u spoljašnje fajlove

- PHP sadrži sledeće funkcije koje omogućavaju korišćenje spoljašnjih fajlovs u okviru PHP skripta:
  - `include()`
  - `require()`
  - `include_once()`
  - `require_once()`
- Svakoj funkciji se prosleđuje ime i putanja do spoljašnjih fajlova koji se žele upotrebiti

Smeštanje klasa u spoljašnje fajlove

- `include()` i `require()` smeštaju sadržaj spoljašnjih fajlova u PHP skript
- `include_once()` i `require_once()` smeštaju spolajšnji fajl jednom tokom procesiranja skripta

Smeštanje klasa u spoljašnje fajlove

- Koristiti `include()` i `include_once()` za HTML kod
- Koristiti `require()` ili `require_once()` za PHP kod
- Spoljašnji fajlovi se mogu koristiti za klase i za bilo koji tip PHP ili HTML koda koji se želi koristiti više puta u okviru Web stranice
- Može se koristiti proizvoljna ekstenzija fajlova

Specifikatori pristupa

- Specifikatori pristupa kontrolišu klijentski pristup pojedinačnim promenljivama i funkcijama klasa
- Postoje tri nivoa pristupa: `public`, `private`, i `protected`
- **`public`** dozvoljava da bilo ko poziva metode klase ili menja njene članice

private

- **`private`** specifikator sprečava klijente da pozivaju metode klase ili menja njene članice

public

```
class BankAccount {
     public $Balance = 0;
}

class BankAccount {
     public $Balance = 1 + 2;
}
```

Serijalizacija objekata

- **Serijalizacija** označava proces konvertovanja objekta u string koji se kasnije može ponovo koristiti
- Ovim postupkom se smeštaju i članovi i metode klase u stringove
- Potrebno je poslati ime objekta funkciji `serialize()`

```
$SavedAccount = serialize($Checking);
```

Obrnut postupak

- Ta bi se serijalizovani podaci konvertovali u početni objekat, koristi se `unserialize()` funkcija

```
$Checking = unserialize($SavedAccount);
```

- Da bi se serijalizovani objekti koristili između skriptova, dodeljuju se promenljivoj koja označava sesiju

```
session_start();
$_SESSION('SavedAccount') = serialize($Checking);
```

Rad sa metodama

- Kreira se **public** metoda za sve funkcije kojima treba klijent da pristupa
- Kreira se **private** metoda za sve funkcije kojima klijent ne treba da pristupa
- Definišu se specifikatori pristupa yza sve pojedinačne podatake

Primer

```
class BankAccount {
     public $Balance = 958.20;
     public function withdrawal($Amount) {
          $this->Balance -= $Amount;
     }
}
if (class_exists("BankAccount"))
     $Checking = new BankAccount();
else
     exit("<p>The BankAccount class is not available!</p>");
printf("<p>Your checking account balance is $%.2f.</p>",
     $Checking->Balance);
     $Cash = 200;
     $Checking->withdrawal(200);
     printf("<p>After withdrawing $%.2f, your checking account balance
          is $%.2f.</p>", $Cash, $Checking->Balance);
```

Konstruktor

- Konstrukor je funkcija koja se poziva pri inicijalizaciji

```
class BankAccount {
     private $AccountNumber;
     private $CustomerName;
     private $Balance;
     function __construct() {
           $this->AccountNumber = 0;
           $this->Balance = 0;
           $this->CustomerName = "";
     }
```

Destructor funkcija

- **destructor** funkcija se poziva u trenutku kada se objekat briše
- Ova funkcija oslobađa sve resurse dodeljene objektu nakon njegovog brisanja

_destruct()

- destructor funkcija se poziva u dva slučaja:
  - Kada se skript završava
  - Kada se objekat ručno briše pomoću funkcije `unset()`
- Da bi se dodala destructor funkcija PHP klasi, kreira se funkcija pod imenom `__destruct()`

Primer

```
function __construct() {
      $DBConnect = new mysqli("localhost",
  "dongosselin",
           "rosebud", "real_estate")
}
function __destruct() {
      $DBConnect->close();
}
```

Primer Accessor funkcije

```
class BankAccount {
     private $Balance = 0;
     public function setBalance($NewValue) {
          $this->Balance = $NewValue;
     }
     public function getBalance() {
          return $this->Balance;
     }
}
if (class_exists("BankAccount"))
     $Checking = new BankAccount();
else
     exit("<p>The BankAccount class is not available!</p>");
$Checking->setBalance(100);
echo "<p>Your checking account balance is "
     . $Checking->getBalance() . "</p>";
```

Serijalizacija

- Pri pozivu `serialize()` funkcije, PHP traži u okviru klase objekta specijalnu funkciju pod imenom `__sleep()`
- Osnovni cilj ove funkcije je da se specificira koji se članovi klase serijalizuju

Serijalizacija

- Ako se ne uključi funkcija `__sleep()` u okviru klase, tada se u okviru funkcije `serialize()` koriste svi članovi klase

```
function __sleep() {
     $SerialVars = array('Balance');
     return $SerialVars;
}
```

- Kada se izvršava `unserialize()` funkcija, PHP u okviru klase traži funkciju `__wakeup()`