Интернет програмирање

vladaf@matf.bg.ac.rs

PHP

vladaf@matf.bg.ac.rs

PHP

- PHP је специјализовани скриптни језик, првенствено намењен за израду динамичног веб садржаја, који се извршава на серверу
- Стекао популарност због своје једноставности и синтаксе наслеђене из програмског језика C
- Језик се проширивао и стицао могућности за објектно оријентисано програмирање, нарочито од верзије 5.0
- Дозвољава процедурално програмирање, али истовремено омогућава и коришћење класа и других концепата објектно-оријентисаног програмирања
- PHP користи неколико стотина хиљада програмера и неколико милиона сајтова, укључујући неке од најпосећенијих сајтова на свету

vladaf@matf.bg.ac.rs

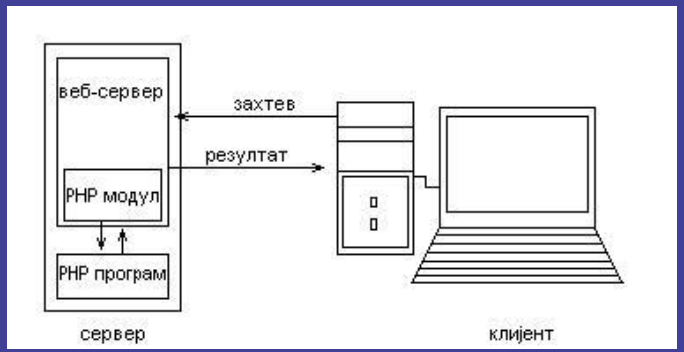
Извршавање PHP-а

- Веб прегледач не извршава PHP када је PHP уграђен у веб страну. Стога, за разлику од JavaScript-а, не мора да се води рачуна о томе да ли прегледач подржава PHP.
- За креирање веб стране са PHP-ом, потребно је убацити PHP наредбе које треба да изврши у HTML страну и поставити да HTML документ има наставак .php – да би се документ повезао са PHP процесором.
- Када прегледач захтјева веб страну с наставком .php, тада веб сервер шаље тражени документ PHP процесору, па потом веб страну коју врати PHP процесор шаље као одговор прегледачу да је прикаже.
- PHP процесор извршава PHP наредбе и обезбеђује да веб страна послата серверу (а затим прослеђена веб прегледачу) буде исправно форматирана и са ознакама које прегледач разуме.

vladaf@matf.bg.ac.rs

Извршавање PHP-а (2)

- Програм који се напише у PHP-у не захтева превођење, већ се интерпретира при сваком извршавању
- PHP интерпретатор може радити на следећи начин
  1. По PHP CGI принципу
     - Интерпретатор ће постојати као екстерна апликација, која се позива да изврши дату скрипту сваки пут кад буде захтевана од неког корисника
  2. Као модул веб сервера
     - Интерпретатор је увек учитан у меморију, те се не мора позивати спољашњи програм
     - Ова варијанта је данас у највећој употреби јер пружа знатно већу брзину извршавања



vladaf@matf.bg.ac.rs

Структура PHP програма

- За разлику од већине програмских језика који поседују почетну функцију, PHP датотека, налик на већину скрипт језика, једноставно садржи секвенцу наредби, које се извршавају једна за другом, од прве до последње, а последња се уједно сматра и крајем PHP програма.
- PHP датотеке могу да садрже следеће врсте елемената
  1. HTML елементи
  2. Елементи за PHP
  3. PHP наредбе
  4. белине
  5. PHP коментари

vladaf@matf.bg.ac.rs

Структура PHP програма (2)

1. HTML елементи су претходно детаљно описани
2. Стилови за елементе за PHP
   1. XML стил уз помоћ ознака <?php и ?>
      - Овај стил се препоручује, а обавезан је када је PHP код уметнут у HTML код
   2. Скраћени стил уз помоћ ознака <? и ?>
      - Његова употреба не препоручује јер у неким окружењима није подржан.
      - Од верзије PHP 5.4, скраћени стил је увек омогућен
   3. скрипт стил између ознака <script language="php"> и </script>
      - Препоручује када се истовремено користе и други скрипт језици
   4. ASP стил између ознака <% и %>
      - Најчешће се користи у ASP и ASP.NET окружењу
      - Неопходно је претходно активирати опцију short_open_tag у конфигурацискоj датотеци php.ini
      - Такође се не препоручује.

vladaf@matf.bg.ac.rs

Структура PHP програма (2)

3. PHP наредбе ће бити описане у наставку текста
4. Под белинама се подразумевају знакови за раздвајање, као што су ознаке за повратак на почетак реда, размаци и табулатори
   - Интерпретатори за HTML код и PHP занемарују ове знакове
5. PHP коментари ће бити описани у наставку текста

- У опису структуре PHP програма користиће се следећи мета-симболи:
  - [a] - означава да се елеменат a може али не мора јавити у конструкцији
  - [a…] - означава да се елеменат a не мора јавити, али и да се може поновити једном или више пута
  - a | b - јавља се тачно један од израза: или a или b

vladaf@matf.bg.ac.rs

Структура PHP програма (3)

- PHP наредбе угнеждавају се у заглавље и/или тело веб стране на следећи начин:

```
<?php
    PHP наредбе
?>
```

- Уколико нека веб страна садржи PHP скрипту, требало би да датотека на серверу има наставак .php, како би веб сервер био информисан
- У PHP датотеци, блок који је окружен PHP елементима (нпр. блок између <?php и ?>), се сматра PHP кодом и извршава се помоћу PHP интерпретатора, док се остатак датотеке сматра обичним текстом који се једноставно испише на излаз, без додатног интерпретирања

vladaf@matf.bg.ac.rs

Структура PHP програма (4)

- Пример једног PHP програма:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
</head>
<body>
  <h1>Моја друга PHP веб страна</h1>
  <?php
    echo "Здраво, живо!";
  ?>
</body>
</html>
```

- PHP код може бити организован у функције и класе, и може се организовати у више датотека
- Као почетна датотека се узима она датотека која се проследи интерпретатору на извршавање. Уколико датотека није наведена, сервер ће аутоматски покушати да покрене датотеку index.php

vladaf@matf.bg.ac.rs

Основна синтакса PHP

vladaf@matf.bg.ac.rs

Коментари

- Коментари су текст иза две косе црте // или симбола # (ако су једноредни)
- Коментари су текст између /* и */ (ако су једноредни или вишередни)

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
</head>
<body>
<?php
// Ovo je komentar u jednoj liniji

# Ovo je takođe komentar u jednoj liniji

/*
Ovo je blokovski komentar, koji
se proteže kroz
više linija
*/

// Komentari se mogu koristiti da bi se isključili delovi linije sa naredbom
$x = 5 /* + 15 */ + 5;
echo $x;
?>
</body>
</html>
```

vladaf@matf.bg.ac.rs

Кључне речи

Кључне речи	
кључна реч	**опис**
__halt_compiler()	Зауставља извршење компајлера
abstract	дефинише апстрактне класе и методе
and	логички оператор нижег приоритета
array()	креира низ
as	користи се у петљи foreach за нумеричке или асоцијативне низове
break	прекида извршавање актуелне структуре у петљама for, foreach, while, do-while или switch
callable	
case	користи се као услов у структури switch
catch	користи се као клаузула блока try
class	дефинише класу
clone	креира копију објекта
const	дефинише константу на локалном нивоу
constant()	враћа вредност претходно дефинисане константе
continue	наставља извршавање на крај петље
declare	
default	подразумевана клаузула блока структури switch
define	дефинише константе на глобалном нивоу
defined()	проверава да ли је дефинисана константа
die()	исто што и exit()
do	клаузула петље do-while
echo	шаље променљиве на стандардни излаз
else	клаузула контролне структуре if
elseif	клаузула контролне структуре if
empty()	утврђује да је променљива без додељене врености (null), обрнуто од isset()
enddeclare	
endfor	завршетак петље for у другој синтакси
endforeach	завршетак петље foreach у другој синтакси
endif	завршетак контролне структуре if у другој синтакси
endswitch	завршетак контролне структуре switch у другој синтакси
endwhile	завршетак петље while у другој синтакси
eval()	процењује да ли променљива знаковног типа представља PHP кода
exit()	зауставља извршавање скрипте, исто што и die()
extends	дефинише родитељску класу актуелној класи
final	у декларацији да означи као класу или методу која не може да се наследи
for	петља
foreach	петља
function	дефинише корисничку функцију
global	глобална променљива
goto	скаче на други део програма, циљна тачка је наведена лабелом праћеном двотачком
if	контролна структура
implements	употреба интерфејса у класи
include	коришћење спољашње датотеке
include_once	коришћење спољашње датотеке са једном декларацијом
instanceof	оператор који утврђује да ли је неки објекат примерак одређене класе или наслеђене класе
insteadof	
interface	декларише интерфјес
isset()	проверава да ли променљива има додељену вредност, супротно од isempty()
list()	додељује вредности низа променљивама из листе аргумената
namespace	декларише именски простор
new	креира нови објекат према моделу класе
or	клаузула наредбе if
print	шаље променљиву на излаз
private	декларише атрибут или методу доступну унутар једне класе
protected	декларише атрибут или методу доступну унутар једне класе и свих класа изведених од ње
public	декларише јавни атрибут или класу
require	коришћење спољашње датотеке
require_once	коришћење спољашње датотеке са једном деклларацијом
return	повратак из корисничке функције
static	декларише променљиве и функције које не мењају вредност између позива
switch	структура за вишесмерно гранање
throw	испаљује изузетак
trait	
try	блок кода за хватање изузетака
unset()	поништава претходно додељену вредност променљивој или класи
use	коришћене именског простора
var	декларише јавни атрибут (променљиву) класе
while	петља
xor	логички оператор нижег приоритета

vladaf@matf.bg.ac.rs

Променљиве

- Променљиве у PHP језику се представљају помоћу знака $, иза кога следи име променљиве.
- Име почиње словом или подвлаком _ иза којег следе слова, цифре или подвлаке, при чему се словом сматра слово латинице [A-Z], [a-z] или било који знак са ASCII кодом већим од 127.
- У језику PHP се разликују велика и мала слова, па променљива $A није исто што и променљива $a.
- Променљиве се не морају декларисати
- Променљиве нису типизиране - њихов тип се одређује према вредностима које им се у изразима додељују током програма и постоји могућност да се тип променљиве промени
- Није неопходна додела почетних вредности за променљиве, мада се препоручује, ради прегледности, безбедности и бољег разумевања кода

vladaf@matf.bg.ac.rs

Променљиве (2)

- PHP садржи следеће типове променљивих:
  - Основни типови:
    - boolean - логичке вредности: тачно (true) и нетачно (false)
    - integer - цели бројеви
    - float - бројеви у покретном зарезу
    - string - ниска знакова, односно текст
  - Сложени типови променљивих:
    - array - низ
    - object – објекат
  - Посебни типови променљивих:
    - resource - ресурс, садржи референцу на спољашњи ресурс, нпр. датотеку или веза са базом података
    - NULL – променљива нема вредност, може садржати једино вредност null

vladaf@matf.bg.ac.rs

Променљиве (3)

- Додела се реализује помоћу знака једнакости (=)
- Додела променљиве може бити по вредности или по референци (за шта се користи симбол &)
  1. Када се додела врши по вредности, тада се променљивој (лево од знака једнакости) даје вредност израза који се налази десно од знака једнакости
     Након доделе, променљива је независна од других – промена вредности једне променљиве не утиче на другу
  2. Када се додела врши по референци, тада променљива лево од знака једнакости указује (реферише) на променљиву која се налази десно од знака једнакости
     Након доделе променљиве су «везане» - промена једне од променљивих утиче на ону другу

vladaf@matf.bg.ac.rs

Променљиве (3)

- Пример доделе променљиве по вредности

zad02-01-deklaracije-promenljivih.php pog02-php-promenljive

```php
<?php
$txt = "Hello world!";
$x = 5;
$y = 10.5;
?>
```

- Пример доделе променљиве по референци

zad02-10-dodela-po-vrednosti-i-referenci.php pog02-php-promenljive

```php
<?php
$x = 5;
$y = $x;
echo $x;
echo $y;
echo "<br>";
$x = 6;
echo $x;
echo $y;
echo "<br>";
$y = &$x;
echo $x;
echo $y;
echo "<br>";
$x = 7;
echo $x;
echo $y;
echo "<br>";
$y = 9;
echo $x;
echo $y;
echo "<br>";
?>
```

vladaf@matf.bg.ac.rs

Испис променљивих

- За исписивање вредности променљивих (прецизније, израза) користе се две основне функције

  print ($arg)

  echo ($arg1 [, $arg2 ...])

- При позивању ове две функције се не морају користити заграде
- Приликом извршавања ових функција, аргументи ма ког типа ће пре исписа бити претворени у тип string
- Функцији print може да се проследи само један аргумент, док echo може да има више аргумената међусобно раздвојених зарезом.

vladaf@matf.bg.ac.rs

Испис променљивих (2)

- Примери исписа променљивих

zad02-03-prikaz-promenljivih.php pog02-php-promenljive

```php
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
</head>

<body>
  <?php
  $txt = "Матф Београд";
  echo "Ја сам студент $txt!";
  ?>
</body>
</html>
```

zad02-04-prikaz-promenljivih.php pog02-php-promenljive

```php
<?php
$x = 5;
$y = 4;
echo $x + $y;
?>
```

vladaf@matf.bg.ac.rs

Испис променљивих (3)

- Примери исписа променљивих

zad03-03-prikaz-promenljivih.php pog03-php-naredbe-echo-print

```php
<?php
$txt1 = "Ucim PHP";
$txt2 = "Internet programiranje";
$x = 5;
$y = 4;

echo "<h2>$txt1</h2>";
echo "Proucavam PHP u okviru predmeta $txt2<br/>";
echo $x + $y;
?>
```

zad03-05-prikaz-promenljivih.php pog03-php-naredbe-echo-print

```php
<?php
$txt1 = "Ucim PHP";
$txt2 = "Internet programiranje";
$x = 5;
$y = 4;

print "<h2>$txt1</h2>";
print "Proucavam PHP u okviru predmeta $txt2<br/>";
print $x + $y;
?>
```

vladaf@matf.bg.ac.rs

Испитивање променљивих

- За испитивање да ли је нека променљива празна користи се функција empty
- За испитивање да ли нека променљива постављена на вредност која није NULL, користи се функција isset
  - Ако се функцији проследи више од једне променљиве, враћа вредност тачно само у случају када је свака од променљивих постављена
  - Испитивање променљивих извршава се с лева на десно и завршава се у случају да наиђе на прву променљиву која није постављена
- За испитивање да ли нека променљива постављена на вредност NULL, користи се функција is_null

vladaf@matf.bg.ac.rs

Испитивање променљивих (2)

- Поређење функција empty, isset, is_null

Value of variable ($var)	isset($var)	empty($var)	is_null($var)
“” (an empty string)	bool(true)	bool(true)	
” ” (space)	bool(true)		
FALSE	bool(true)	bool(true)	
TRUE	bool(true)		
array() (an empty array)	bool(true)	bool(true)	
NULL		bool(true)	bool(true)
“0” (0 as a string)	bool(true)	bool(true)	
0 (0 as an integer)	bool(true)	bool(true)	
0.0 (0 as a float)	bool(true)	bool(true)	
var $var; (a variable declared, but without a value)		bool(true)	bool(true)
NULL byte (“\ 0”)	bool(true)		

vladaf@matf.bg.ac.rs

Испитивање променљивих (2)

- Елиминисање постављња вредности променљиве врши се помоћу функције unset
  - Понашање ове функције у великој мери зависи од опсега променљиве и места одакле је позвана
  - Нпр. уколико се unset позове унутар блока неке корисничке функције, биће елиминисана вредност само локалне променљиве
  - За промене вредности глобалних променљивих препоручује се употреба низа $GLOBALS
- За структурно приказивање типа и вредности променљиве на стандардни излаз користи се функција var_dump
  - Ова функција је посебно корисна у току развоја програма

vladaf@matf.bg.ac.rs

Испитивање променљивих (3)

- Примери коришћња функције var_dump

zad04-02-ceo-broj.php pog04-php-tipovi-podataka

```php
<?php
$x = 5985;
var_dump($x);
?>
```

ad04-03-broj-u-pokretnom-zarezu.php pog04-php-tipovi-podataka

```php
<?php
$x = 10.365;
var_dump($x);
?>
```

zad04-04-logicke-vrednosti.php pog04-php-tipovi-podataka

```php
<?php
$x = true;
$y = false;
echo "\$x= $x<br/>";
var_dump($x);
echo "<br/>";
echo "\$y= $y<br/>";
var_dump($y);
echo "<br/>";
?>
```

vladaf@matf.bg.ac.rs

Испитивање променљивих (4)

- Примери коришћња функције var_dump

zad04-05-niska.php pog04-php-tipovi-podataka

```php
<?php
$x = "Zdravo, zivo!";
$y = 'Zdravo, zivo!';
var_dump($x);
echo "<br/>";
var_dump($y);
echo $y;
?>
```

zad04-06-null-vrednost.php pog04-php-tipovi-podataka

```php
<?php
$x = "Zdravo, zivo!";
var_dump($x);
echo "<br/>";
$x = null;
var_dump($x);
?>
```

zad04-07-nizovi.php pog04-php-tipovi-podataka

```php
<?php
$markeVozila = array("Volvo","BMW","Toyota");
echo "\$markeVozila = $markeVozila<br/>";
var_dump($markeVozila);
?>
```

vladaf@matf.bg.ac.rs

Испитивање променљивих (5)

- Примери коришћња функције var_dump

zad04-08-klase-objekti.php pog04-php-tipovi-podataka

```php
<?php
class Auto
{
    function Auto()
    {
        $this->model = "Folksvagen";
    }
}
// kreiraj objekat
$buba = new Auto();
// prikazi osobine objekta
echo $buba->model;
echo "<br/>";
// izvrsi prikaz objekta
var_dump( $buba );
?>
```

vladaf@matf.bg.ac.rs

Испитивање променљивих (5)

- За испитивање типа променљивих могу се користити следеће функције:
  - is_array - променљива је низ
  - is_double, is_float, is_real - променљива у формату са покретном зарезом
  - is_long, is_int, is_integer - промељнива је цеолобројна
  - is_string - променљива је знаковна ниска
  - is_object - променљива је објекат
  - is_resource - променљива је ресурс
  - is_scalar - променљива је скаларног типа: нумерички, логички или знаковни податак
  - is_numeric - променљива број или знаковна ниска која се може претворити у број
  - is_callable - вредност променљиве име постојеће функције

vladaf@matf.bg.ac.rs

Константе

- За разлику од променљивих, имена константи се пишу без симбола $ на почетку.
- Правила за именовање константи су иста као и за именовање променљивих
  - Мада није неопходно, имена константи се обично пишу свим великим словима
- Вредности за константе се додељују нердбом define, на следећи начин:

```
define("IME_KONSTANTE", vrednost [, razlika_velika_mala ]);
```

- Једном дефинисана константа не може да мења вредност.
- Константама је могуће доделити искључиво скаларне вредности, тј. вредности основних типова података (boolean, integer, float и string)

vladaf@matf.bg.ac.rs

Константе (2)

- Примери коришћења константи

zad06-01-definicija-konst.php pog06-php-konstante

```php
<?php
define("POZDRAV", "Ja volim programiranje!");
echo POZDRAV;
?>
```

zad06-02-konstante-i-slova.php pog06-php-konstante

```php
<?php
define("POZDRAV", "Ja volim programiranje!" /*, true */);
echo POZDRAV;
echo "<br/><br/>";
echo pozdrav;
echo "<br/><br/>";
echo Pozdrav;
echo "<br/><br/>";
echo poZdraV;
?>
```

vladaf@matf.bg.ac.rs

Константе (3)

- Пример коришћења константи

zad06-03-opseg-vazenja-konstante.php pog06-php-konstante

```php
<?php
define("POZDRAV", "Ja volim programiranje!");

function malaProba()
{
    echo POZDRAV;
}

malaProba();
?>
```

- За добијање вредности одређене константе користи се функција constant
  - Њој се прослеђује име константе, а као резултат враћа вредност константе или null ако константа није дефинисана
- За добијање вредности свих дефинисаних константи може се користити функција get_defined_constants

vladaf@matf.bg.ac.rs

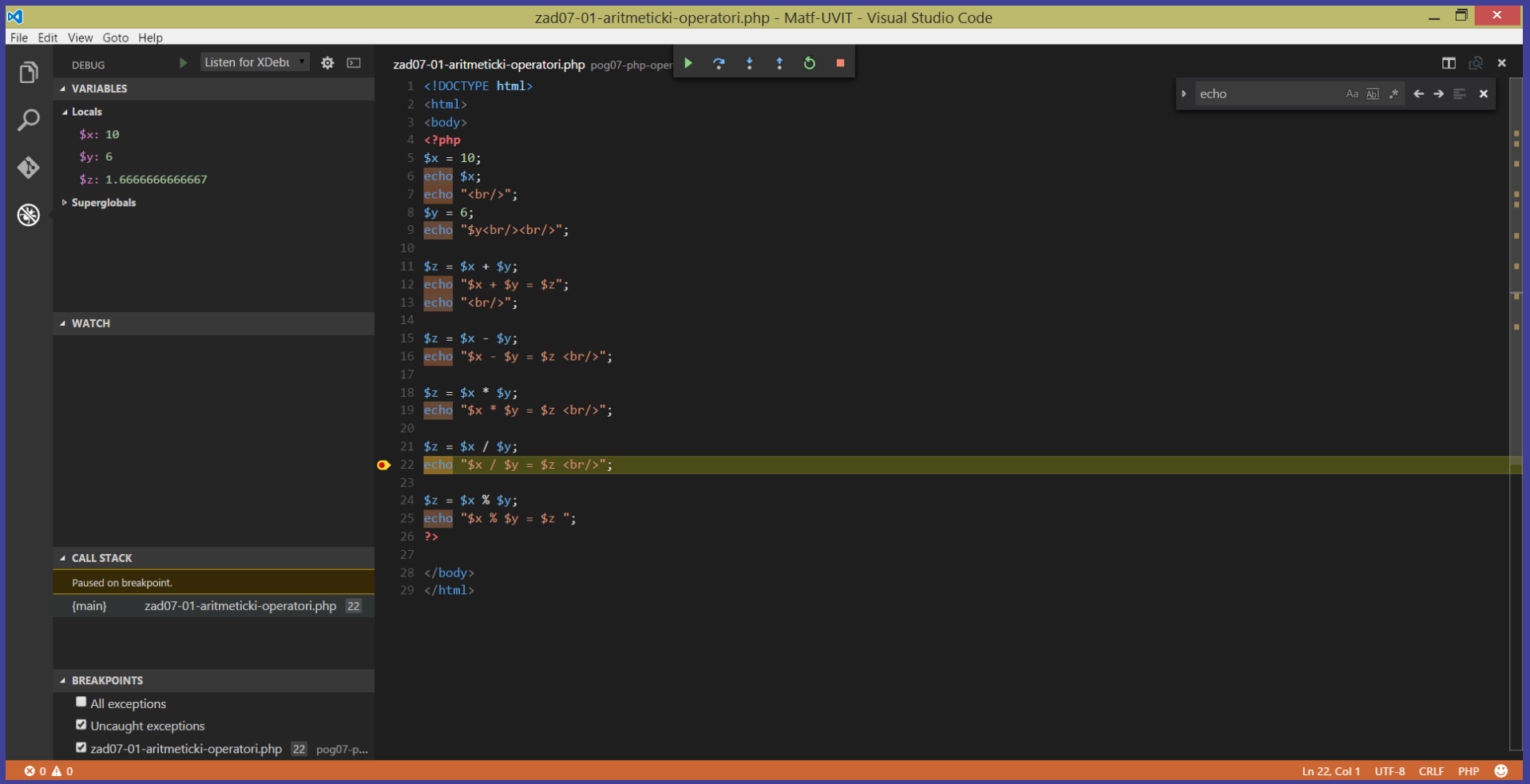
Изрази

- Изрази су један од основних појмова сваког вишег програмског језика, па и PHP
  - Изразом се рачуна и даје резултат. Вредност неког израза одређује се на основу синтаксе израза, односно правила првенства сваког од оператора у изразу и асоцијативности
- Оператори су симболи који омогућавају извршавање операција над операндима. Могу имати један, два или више аргумената.
- Врсте оператора:
  1. аритметички оператори: сабирање (+), одузимање (-), множење (*), дељење (/), модуло (%)
  2. оператор за надовезивање ниски тј. тачка-оператор (.)
  3. оператор доделе (=)

vladaf@matf.bg.ac.rs

Изрази (2)

- Примери израза са аритметичким операторима



vladaf@matf.bg.ac.rs

Изрази (3)

4. комбиновани оператори доделе: извршавају операцију и резултат додељују променљивој

Комбиновани оператори доделе		
оператор	**употреба**	**значење**
++	$a++	$a = $a + 1
--	$a--	$a = $a - 1
+=	$a += $b	$a = $a + $b
-=	$a -= $b	$a = $a - $b
*=	$a *= $b	$a = $a * $b
/=	$a /= $b	$a = $a / $b
%=	$a %= b	$a = $a % $b
.=	$a .= b	$a = $a . $b

Оператори увећања (++), и умањења (--), могу се применити префиксно или суфиксно

Резулатат зависи од положаја оператора у односу на променљиву

vladaf@matf.bg.ac.rs

Изрази (4)

- Примери израза са оператором надовезивања

zad-01-05-promenljive-slova.php pog01-php-osnovna-sintaksa

```php
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
</head>
<body>
  <?php
  $boja = "црвена";
  echo "Боја мог аута је " . $boja . "<br>";
  echo "Боја моје куће је " . $BOJA . "<br>";
  echo "Боја мог чамца је " . $boJA . "<br>";
  ?>
</body>
</html>
```

zad07-06-operatori-sa-niskama.php pog07-php-operatori

```php
<?php

$txt1 = "Zdravo,";
$txt2 = " zivo";
$s = $txt1 . $txt2; //nadovezivanje dve niske
echo $s;
echo "<br/>";

$r = " Radovane!";
$s .= $r; // dodela uz nadovezivanje niske
echo $s;

?>
```

vladaf@matf.bg.ac.rs

Изрази (5)

- Примери израза са комбинованим оператором доделе

zad07-02-operatori-dodele.php pog07-php-operatori

```php
<?php
$x = 10;
echo "\$x = $x <br/>";
$x += 30;
echo "Posle naredbe uvecanja za 30, \$x = $x <br/><br/>";

$x = 30;
echo "x = $x <br/>";
$x -= 50;
echo "Posle naredbe umanjenja za 50, \$x = $x <br/><br/>";

$x = 10;
echo "x = $x <br/>";
$x *= 6;
echo "Posle naredbe umnozavanja sa 6, \$x = $x <br/><br/>";

echo "x = $x <br/>";
$x /= 7;
echo "Posle naredbe deljenja sa 7, \$x = $x <br/><br/>";
?>
```

vladaf@matf.bg.ac.rs

Изрази (6)

- Примери израза са увећањем и умањењем

zad07-04-operatori-inkrementiranja-dekrementiranja.php pog07-php-operatori

```php
<!DOCTYPE html>
<html>
<body>

<?php
$x = 10;
echo $x;
echo "<br/><br/>";

echo ++$x;  // prefiksno inkrementiranje - prvo se inkrementira, a onda se vrati vrednost
echo "   ";
echo $x;
echo "<br/><br/>";

$x = 10;
echo $x++;  // postfiksno inkrementiranje - prvo se vrati vrednost, a onda se inkrementira
echo "   ";
echo $x;
echo "<br/><br/>";

$x = 10;
echo --$x;  // prefiksno dekrementiranje - prvo se dekrementira, a onda se vrati vrednost
echo "   ";
echo $x;
echo "<br/><br/>";

$x = 10;
echo $x--;  // postfiksno dekrementiranje - prvo se vrati vrednost, a onda se dekrementira
echo "   ";
echo $x;
echo "<br/><br/>";
?>

</body>
</html>
```

vladaf@matf.bg.ac.rs

Изрази (7)

5. оператори поређења: као резултат поређења израза враћају вредности тачно (true) или нетачно (false)
6. логички оператори: такође враћају вредности тачно (true) или нетачно (false). Могу да се користе и логички оператори NOT (уместо !), AND (уместо &&), OR (уместо ||), као и оператор XOR за ексклузивну дисјункцију

Оператори поређења		
оператор	**назив**	**употреба**
==	једнако	$a == $b
===	идентично (тип и вредност)	$a === $b
!=	није једнако (различито)	$a != $b
<>	није једнако (различито)	$a <> $b
!==	није идентично	$a !== $b
<	мање од	$a < $b
>	веће од	$a > $b
<=	мање или једнако	$a <= $b
>=	веће или једнако	$a >= $b

Логички оператори			
оператор	**назив**	**употреба**	**резултат**
!	негација	!$a	израз тачан ако је $a нетачно
&&	конјункција	$a && $b	тачан ако су $a и $b тачни
||	дисјункција	$a || $b	тачан ако је $a или $b тачно

vladaf@matf.bg.ac.rs

Изрази (8)

- Примери коришћења логичких оператора

zad07-05-logicki-operatori.php pog07-php-operatori

```php
<?php

$x = 100;
echo "\$x ima vrednost $x <br/>";
$y = 50;
echo "\$y ima vrednost $y <br/><br/>";

if ($x == 100 and $y == 50)
{
  echo "1: \$x ima vrednost 100 i \$y ima vrednost 50! <br/>";
}
if ($x == 100 && $y == 50)
{
  echo "2: \$x ima vrednost 100 i \$y ima vrednost 50! <br/>";
}
if ($x == 100 or $y == 80)
{
  echo "3: \$x ima vrednost 100 ili \$y ima vrednost 80! <br/>";
}
if ($x == 100 || $y == 80)
{
  echo "4: \$x ima vrednost 100 ili \$y ima vrednost 80! <br/>";
}
if ($x == 100 xor $y == 80)
{
  echo "5: ili \$x ima vrednost 100 ili \$y ima vrednost 80! <br/>";
}
if ($x == 100 xor $y == 50)
{
  echo "6: ili \$x ima vrednost 100 ili \$y ima vrednost 50! <br/>";
}
if( !($x == 100 and $y == 80) )
{
  echo "7: \$x ima vrednost 100 i \$y ima vrednost 80! <br/>";
}
if ( !($x == 100 xor $y == 50) )
{
  echo "8: ne vazi da ili \$x ima vrednost 100 ili \$y ima vrednost 50! <br/>";
}

?>
```

vladaf@matf.bg.ac.rs

Изрази (9)

7. оператори над битовима: омогућавају да се цео број обрађује као група битова којим је тај број представљен у меморији

оператор	назив	употреба	резултат
&	конјункција	$a & $b	битови активни у $a и $b активни су и у резултату
|	дисјункција	$a | $b	битови активни у $a или у $b активни су и у резултату
~	негација	~$a	битови активни у $a нису активни у резултату и обрнуто
^	искључива дисјункција	$a ^ $b	битови активни или у $a или у $b, али не у оба, активни су и у резултату
<<	померање битова улево	$a << $b	помера битове у $a улево за $b места
>>	померање битова удесно	$a >> $b	помера битове у $a удесно за $b места

8. условни оператор има следећи облик:
   (uslov) **?** (izraz1) **:** (izraz2)
   ако је uslov испуњен (тачан) враћа се вредност izraz1, а ако није тачан, враћа се вредност izraz2
9. остали оператори: зарез (**,**) за раздвајање аргумената или параметара функције и за раздвајање елемената листе, оператор за прављење објеката (new), оператор за приступање члановима објекта (->) итд.

vladaf@matf.bg.ac.rs

Изрази (10)

- Приоритет и асоцијативност оператора:

Приоритет	Оператори	Асоцијативност
1.	Оператори clone и new	неасоцијативни
2.	Оператор низа: [	слева
3.	Оператори инкрементирања и декрементирања: ++ --	неасоцијативни
4.	Битско не: ~ Унарно минус: - Оператори промене типа података: (int) (float) (string) (array) (object) (bool) Оператор неисписивања грешке: @	здесна
5.	Оператор instanceof	неасоцијативни
6.	Логичко не: !	здесна
7.	Аритметички оператори: * / %	слева
8.	Аритметички оператори: + - Оператор за придруживање ниски: .	слева
9.	Оператори померања битова: << >>	слева
10.	Оператори провере неједнакости: < <= > >= <>	неасоцијативни
11.	Оператори провере једнакости: == != === !==	неасоцијативни
12.	Битско и: &	слева
13.	Битско ексклузивно или: ^	слева
14.	Битско или: |	слева
15.	Логичко и: &&	слева
16.	Логичко или: ||	слева
17.	Условни оператор: ? :	слева
18.	Оператори доделе: = += -= *= /= .= %= &= |= ^= <<= >>= =>	здесна
19.	Логичко и: and	слева
20.	Логичко ексклузивно или: xor	слева
21.	Логичко или: or	слева
22.	Запета: ,	слева

редослед извршења оператора се може променити помоћу малих заграда, са редсоледом од унутрашњих, ка спољашњим заградама

vladaf@matf.bg.ac.rs

Наредбе

- Наредбе или искази одређују неку акцију
- Свака PHP скрипта је састављена од секвенци наредби
  - По правилу, наредбе се извршавају оним редом којим су наведени у програму
  - Редослед извршења наредби може да се промени неком од наредби за пренос тока програма, при чему пренос може ићи на било који други део програма или ван њега
- Основни облик наредбе је израз иза кога следи знак тачка-зарез (;), који представља завршни знак наредбе
- Сложена наредба или блок је механизам који групише скуп наредби у једну семантичку целину
  - Блокови могу да буду угнеждени један у други
  - Сваки од блокова може да садржи своје сопствене декларације, иницијализације и извршне наредбе

vladaf@matf.bg.ac.rs

Наредбе гранања

- Наредбе гранања су наредбе којима се одређује ток извршавања под одређеним условима и које се групишу у једну засебну структуру
  - У оквиру овакве структуре се свака од грана извршавања, ако садржи више од једне наредбе, поставља између витичастих заграда: { и }.
  - Цела управљачка структура такође представља једну наредбу у облику блока
  - Код наредби гранања кристе се различити облици управљачких структура, које се међусобно разликују по начину испитивања услова
- Наредбе гранања су:
  1. Наредба if
  2. Наредба switch

vladaf@matf.bg.ac.rs

Наредбе гранања (2)

1. Наредба if може имати различите форме, а једна од њих је:

```
if (uslov1)
{
    blok1;
}
elseif (uslov2)
{
    blok2;
}
/*... */
elseif (uslovN)
{
    blokN;
}
else
{
    blokN1;
}
```

- У случају да се блок наредби састоји од само једне наредбе, није неопходна употреба витичастих заграда
- Када није потребно настављати даље гранање, већ се оно завршава са проверавањем првог израза - тада се неће писати ни elseif, а ни else.

vladaf@matf.bg.ac.rs

Наредбе гранања (3)

- Пример коришћења наредбе if без elseif и без else

zad08-03-if-naredba.php pog08-php-naredbe-grananja

```php
<?php

$t = date("H");
if ($t < "12")
{
    echo "Dobro jutro!";
}
if ($t >= "12" && $t<"20")
{
    echo "Dobar dan!";
}
if ($t >= "20")
{
    echo "Dobro vece!";
}

?>
```

vladaf@matf.bg.ac.rs

Наредбе гранања (4)

- Пример коришћења наредбе if са elseif и са else

zad08-04-if-elseif-else-naredba.php pog08-php-naredbe-grananja

```php
<?php

$t = date("H");
if ($t < "12")
{
    echo "Dobro jutro!";
}
elseif ($t<"20")
{
    echo "Dobar dan!";
}
else
{
    echo "Dobro vece!";
}

?>
```

vladaf@matf.bg.ac.rs

Наредбе гранања (5)

Понекад је неопходно један израз упоредити са различитим вредностима и у зависности од резултата поређења извршити различите акције

2. Тада је, уместо наредбе if, једноставније употребити наредбу switch, која има следећи облик:

```
switch (izraz)
{
    case vrednost1:
        blok1;
    case vrednost2:
        blok2;
    /*... */
    case vrednost2:
        blokN;
    default:
        blokN1;
}
```

- Блокови могу да садрже групу од једне или више наредби, од којих неке, такође могу бити сложене
- Наредба switch се извршава тако што се вредност израза, цео број или ниска, редом пореди са различитим случајевима могућих очекиваних вредности, све док се не наиђе на случај када су те две вредности једнаке
- Одатле, па на даље, извршавају се сви блокови наредби, по реду, све до последњег, или се прекида извршавање ако у блоку постоји наредба за излазак из структуре break
- Последњи блок наредби је подразумевани, није га обавезно навести, и он се односи се на вредности које нису биле обухваћене ни једним од претходних случајева

vladaf@matf.bg.ac.rs

Наредбе гранања (6)

- Пример коришћења наредбе switch

zad08-05-switch-naredba.php pog08-php-naredbe-grananja

```php
<?php

$omiljenaBoja = "crvena";
switch ($omiljenaBoja)
{
    case "crvena":
        echo "Crveno je boja ljubavi...";
        break;
    case "crna":
        echo "Crno vino, crne oci...";
        break;
    case "zelena":
        echo "Zelene su bile oci te...";
        break;
    default:
        echo "Nema pesme za boju koja nije crvena, crna ili plava!";
}

?>
```

vladaf@matf.bg.ac.rs

Наредбе циклуса

- Циклуси (петље) су управљачке структуре за циклично понављање групе наредби, под одређеним условом
  - Разликује се тело циклуса и услов циклуса
  - Извршавање циклуса могуће је прекинути у било ком делу тела циклуса употребом наредбе break, док се прелазак на следећу итерацију циклуса врши наредбом continue
- У зависности од начина испитивања услова циклуса и места где је он постављен разликује се неколико врста петљи:
  1. циклус while
  2. циклус do-while
  3. циклус for
  4. циклус foreach

vladaf@matf.bg.ac.rs

Наредбе циклуса (2)

1. Код циклуса while услов се налази на почетку. Представља се на следећи начин:

```
while (uslov)
{
    blok;
}
```

- Циклус се извршава тако што се blok поновљено извршава све док је испуњен uslov
- Приликом извршавања се прво проверава да ли важи uslov, па ако важи онда се изврши blok, па се у новој итерацији поново иде на проверу
- Дакле, ако је вредност логичког израза uslov нетачна од самог почетка, blok се никад неће извршити

- Пример за циклус while

zad09-01-naredba-while.php pog09-php-naredbe-ciklusa

```php
<?php

$x = 1;
while($x <= 5)
{
    echo "Broj je: $x <br/>";
    $x++;
}

?>
```

vladaf@matf.bg.ac.rs

Наредбе циклуса (3)

2. Код циклуса do-while услов за понављање налази се на крају. Овај циклус се представља на следећи начин:

```
do
{
    blok;
}
while (uslov);
```

- Циклус се извршава тако што се blok поновљено извршава све док је испуњен uslov
- Приликом извршавања се прво изврши blok, а потом се проверава да ли важи uslov, па ако важи иде се у нову итерацију
- Дакле, blok ће се извршити макар једном, чак и када uslov није испуњен ни на почетку

- Пример за циклус do-while

zad09-02-naredba-do-while.php pog09-php-naredbe-ciklusa

```php
<?php

$x = 1;
do
{
    echo "\$x = $x <br/>";
    $x++;
} while ($x <= 5);

?>
```

vladaf@matf.bg.ac.rs

Наредбе циклуса (4)

3. Циклус for може да се користи када је број итерација унапред познат. Он има следећи облик:

```
for (izraz1; izraz2; izraz3)
{
    blok;
}
```

- Први израз тј. izraz1 се користи да би се поставила почетна вредност бројача. У изразу izraz2 се поставља услов за понављање, а помоћу izraz3 се мења вредност бројача
- Дакле, кораци у извршавању су:
  1. прво се изврши izraz1
  2. потом се провери да ли је izraz2 тачан
  3. ако није, излази се из петље и прелази се на следећу наредбу
  4. ако јесте, изврши се blok, па izraz3, а потом се прелази на корак 2

- Пример за циклус for

zad09-04-naredba-for.php pog09-php-naredbe-ciklusa

```php
<?php
for ($x = -2; $x <= 10; $x++) {
    echo "\$x = $x <br/>";
}
?>
```

vladaf@matf.bg.ac.rs

Наредбе циклуса (5)

4. Циклус foreach је варијација циклуса for, специјално направљена за рад са низовима и другим објектима.
   О наредби циклуса ће бити више речи у делу који се односи на низове

Облик који се обично користи за низ:

```
foreach (niz as promenljiva)
{
    blok;
}
```

- Пролази се кроз цео низ niz и у свакој итерацији се вредност елемента низа смешта у promenljiva. Ова вредност касније се користи у блоку наредби blok

zad09-05-naredba-foreach.php pog09-php-naredbe-ciklusa

```php
<?php

$boje = array("crvena", "zelena", "plava", "zuta");

foreach ($boje as $boja) {
    echo "$boja <br/>";
}

?>
```

vladaf@matf.bg.ac.rs

Низови

- У PHP-у низ се може посматрати као група променљивих на које се реферише преко истог имена
  - PHP допушта да елементи низа могу бити различитог типа.
- У употреби су две врсте низова:
  1. индексни низови, код којих се за приступ елеменатима низа (тј. индексирање) користе ненегативни цели бројеви (почевши од индекса 0)
  2. асоцијативни низови, који за приступ (тј. кључеве) могу да користе ниске и целобројне податке

vladaf@matf.bg.ac.rs

Низови (2)

- Пример рада са индексним низовима

zad11-01-indeksni-niz.php pog11-php-nizovi

```php
<?php

$auta = array("Volvo", "BMW", "Tojota");

echo "Svidja mi se " . $auta[0] . ", " . $auta[1] . " i " . $auta[2] . ".";

?>
```

- Пример рада са асоцијативним низовима

zad11-04-asocijativni-niz.php pog11-php-nizovi

```php
<?php

$uzrast = array("Petar"=>"35", "Marko"=>"37", "Janko"=>"43");

echo "Petar ima " . $uzrast['Petar'] . " godina.";

?>
```

vladaf@matf.bg.ac.rs

Низови (3)

- Пример цикуса for код нумеричког низа

zad11-03-petlja-indeksni-niz.php pog11-php-nizovi

```php
<?php

$auta = array("Volvo", "BMW", "Tojota");
$brojAuta = count($auta);

for($x = 0; $x < $brojAuta; $x++)
{
    echo $auta[$x];
    echo "<br/>";
}

?>
```

vladaf@matf.bg.ac.rs

Низови (4)

- Циклус foreach је варијација циклуса for, специјално направљена за рад са низовима и другим објектима. Облик који се обично користи код индексних низова:

```
foreach (niz as promenljiva)
{
    blok;
}
```

- Пролази се кроз цео низ niz и у свакој итерацији се вредност елемента низа смешта у promenljiva. Ова вредност касније се користи у блоку наредби blok

- Пример за циклус foreach код индексног низа

zad09-05-naredba-foreach.php pog09-php-naredbe-ciklusa

```php
<?php

$boje = array("crvena", "zelena", "plava", "zuta");

foreach ($boje as $boja) {
    echo "$boja <br/>";
}

?>
```

vladaf@matf.bg.ac.rs

Низови (5)

- Облик foreach циклуса који се обично користи код асоцијативних низова:

```
foreach (niz as promenljiva_kljuc => promenljiva_vrednost)
{
    blok;
}
```

  - Променљива promenljiva_kljuc итерира кроз секвенцу кључева низа niz
  - У телу циклуса blok се могу користити и кључ (тј. promenljiva_kljuc) и вредност која одговара том кључу (тј. promenljiva_vrednost)

- Пример за циклус foreach код асоцијативног низа

zad11-05-petlja-asocijativni-niz.php pog11-php-nizovi

```php
<?php

$uzrast = array("Petar"=>"35", "Marko"=>"37", "Janko"=>"43");

foreach($uzrast as $x => $x_value)
{
    echo "Kljuc=" . $x . ", Vrednost=" . $x_value;
    echo "<br/>";
}

?>
```

vladaf@matf.bg.ac.rs

Низови (6)

Оператори за рад са низовима[37][4]			
оператор	**назив**	**употреба**	**значење**
+	унија	$a + $б	враћа низ који се састоји од свих елемента $a и $б
==	једнако	$a == $б	израз је тачан ако су низови $a и $б имају исте парове индекс/вредност
===	идентично	$a === $б	израз је тачан ако су низови $a и $б имају исте парове индекс/вредност, по истом редоследу и типу података
!=	различито	$a != $б	израз је тачан ако је $a различит од $б
<>	различито	$a <> $б	израз је тачан ако је $a различит од $б
!==	није идентично	$a !== $б	израз је тачан ако $a није идентичан $б

vladaf@matf.bg.ac.rs

Низови (7)

Функције низова	
назив функције	**опис функције**
unset()	елиминисање елемената из низа
array_key_exists()	провера за постојање неког индекса (кључа) у низу
array_search()	претрага елемената на основу индекса (кључа)
in_array()	провера за постојање неке вредности у низу
array_keys()	враћа низ кључева наведеног низа
array_values()	враћа низ вредности наведеног низа
array_splice()	брисање елемената низа или њихова замена
count()	враћа број елемената наведеног низа *(синоним: sizeof())*
sort()	уређује низ по њиховим вредностима у растући редослед
rsort()	уређује низ по у опадајући редослед
asort()	уређује низове по њиховом вредностима у растући абецедни редослед, при чему се врши њихова реиндексација
ksort()	уређује индексе низа по вредностима индекса у растући абецедни редослед
krsort()	уређује индексе низа по вредностима индекса у опадајући редослед
shuffle()	меша елементе низа (насумично уређивање)
array_sum()	сабира вредности свих елемената наведеног низа
array_merge()	спаја два или више низова у један. Важно је водити рачуна о индексима, јер се приликом спајања елиминишу дупликати (важи само за знаковне индексе, док се бројчани аутоматски уређују)
explode()	формира низ из наведене знаковне променљиве (текста). Осим имена низа и променљиве, захтева и знак за раздвајање
implode()	формира текст из наведеног низа. Осим имена низа и променљиве, захтева и знак за раздвајање
array_diff()	упоређује два или више низова, и креира нови на основу елемената по којима се они разликују
array_unique()	уклања дупликате из наведеног низа

vladaf@matf.bg.ac.rs

Низови (8)

- Пример за одређивање броја елеманата у низу

zad11-02-duzina-niza.php pog11-php-nizovi

```php
<?php

$auta = array("Volvo", "BMW", "Tojota");

echo count($auta);

?>
```

vladaf@matf.bg.ac.rs

Низови (9)

- Напредне функције за рад са низовима
  - list() - додељује променљивама (аргументима функције, десно од знака једнакости) вредности елемената низа (лево од знака једнскости), респективно
  - print_r() – приказује информацију о аргументу у читљивом формату; ако је аргумент низ, пролази кроз цео низ приказујући све његове парове кључева и вредности
  - array_walk() – пролази цео низ и над сваким елементом низа (паром кључ-вредност) изврши функцију чије се име прослеђује као аргумент ове функције
  - array_flip() - разамењује вредности кључева и одговарајућих вредности оригиналног низа.

vladaf@matf.bg.ac.rs

Низови (10)

- Пример сортирања индексног низа

zad11-06-sort-indeksni-niz.php pog11-php-nizovi

```php
<?php

$auta = array("Volvo", "BMW", "Tojota", "Fiat", "Mercedes");
echo "Pre sortiranja: ";
print_r($auta);
echo "<br/>";
sort($auta);  // sortira niz u rastuci poredak
echo "Posle sortiranja: ";
print_r($auta);
echo "<br/><br/>";

$brojevi = array(4, 6, 2, 22, 11, -5, 17);
echo "Pre sortiranja: ";
print_r($brojevi);
echo "<br/>";
sort($brojevi);  // sortira niz u rastuci poredak
echo "Posle sortiranja: ";
print_r($brojevi);
echo "<br/><br/>";


$auta = array("Volvo", "BMW", "Tojota", "Fiat", "Mercedes");
echo "Pre sortiranja: ";
print_r($auta);
echo "<br/>";
rsort($auta);  // sortira niz u opadajuci poredak
echo "Posle sortiranja: ";
print_r($auta);
echo "<br/><br/>";

$brojevi = array(4, 6, 2, 22, 11, -5, 17);
echo "Pre sortiranja: ";
print_r($brojevi);
echo "<br/>";
rsort($brojevi);  // sortira niz u opadajuci poredak
echo "Posle sortiranja: ";
print_r($brojevi);
echo "<br/><br/>";

?>
```

vladaf@matf.bg.ac.rs

Низови (11)

- Пример сортирања асоцијативног низа

zad11-07-sort-asocijativni-niz.php pog11-php-nizovi

```php
<?php
$uzrast = array("Petar"=>35, "Marko"=>37, "Janko"=>43, "Zarko"=> 7, "Milica" => 50 );
echo "Pre sortiranja: ";
print_r($uzrast);
echo "<br/>";
asort($uzrast); // Sortira asocijativni niz u rastuci poredak po vrednostima
echo "Posle sortiranja: ";
print_r($uzrast);
echo "<br/><br/>";

$uzrast = array("Petar"=>35, "Marko"=>37, "Janko"=>43, "Zarko"=> 7, "Milica" => 50 );
echo "Pre sortiranja: ";
print_r($uzrast);
echo "<br/>";
ksort($uzrast); // Sortira asocijativni niz u rastuci poredak po kljucevima
echo "Posle sortiranja: ";
print_r($uzrast);
echo "<br/><br/>";

$uzrast = array("Petar"=>35, "Marko"=>37, "Janko"=>43, "Zarko"=> 7, "Milica" => 50 );
echo "Pre sortiranja: ";
print_r($uzrast);
echo "<br/>";
arsort($uzrast); // Sortira asocijativni niz u opadajuci poredak po vrednostima
echo "Posle sortiranja: ";
print_r($uzrast);
echo "<br/><br/>";

$uzrast = array("Petar"=>35, "Marko"=>37, "Janko"=>43, "Zarko"=> 7, "Milica" => 50 );
echo "Pre sortiranja: ";
print_r($uzrast);
echo "<br/>";
krsort($uzrast); // Sortira asocijativni niz u opadajuci poredak po kljucevima
echo "Posle sortiranja: ";
print_r($uzrast);
echo "<br/><br/>";
?>
```

vladaf@matf.bg.ac.rs

Низови (12)

- Вишедимензионални низови садрже елементе који могу такође бити низови
  - Код њих сваки елемент има више од једног индекса
  - Ако сваки елемент има два индекса, онда се ради о дводимензионалном низу
- Вишедимензионални низови омогућавају обраду врло сложених структура података, као што су оне који се користе у табелама или базама података
- Ови низови такође могу бити нумерички и асоцијативни, те сваки елемент може бити податак различитог типа, што омогућава велику флексибилност и функционалност

vladaf@matf.bg.ac.rs

Низови (13)

- Пример рада са дводимензионалним низом

zad12-01-dvodimenzionalni-niz.php pog12-php-visedimenzionalni-nizovi

```php
<!DOCTYPE html>
<html>
<body>
<?php
$auta = array
  (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );
?>
Prikaz dvodimenzionalnog niza: <br>
<?php

echo $auta[0][0].": Na zalihama".$auta[0][1].", prodato: ".$auta[0][2].".<br>";
echo $auta[1][0].": Na zalihama".$auta[1][1].", prodato: ".$auta[1][2].".<br>";
echo $auta[2][0].": Na zalihama".$auta[2][1].", prodato: ".$auta[2][2].".<br>";
echo $auta[3][0].": Na zalihama".$auta[3][1].", prodato: ".$auta[3][2].".<br>";
?>
</body>
</html>
```

vladaf@matf.bg.ac.rs

Низови (14)

- Пример угнежђених петљи код дводимензионалног низа

zad12-02-dvodimenzionalni-petlja..php pog12-php-visedimenzionalni-nizovi

```php
<!DOCTYPE html>
<html>
<body>
<?php
$auta = array
  (
  array("Volvo",22,18),
  array("BMW",15,13),
  array("Saab",5,2),
  array("Land Rover",17,15)
  );
?>

<br><br> Prikaz dvodimenzionalnog niza: <br>
<?php
for ($i = 0; $i < count($auta); $i++)
{
  echo "<p><b>VrstAuto $i</b></p>";
  echo "<ul>";
  for ($j = 0; $j < count($auta[0]); $j++)
  {
    echo "<li>".$auta[$i][$j]."</li>";
  }
  echo "</ul>";
}
?>

</body>
</html>
```

vladaf@matf.bg.ac.rs

Функције

- Функције омогућавају извршавање одређеног блока наредби, које се јављају на разним местима
- PHP има доста већ уграђених функција, али постоји могућност дефинисиња сопствених функција
- Функција може да се позове из неког другог дела скрипте, прослеђујући јој аргументе, а она затим, када се изврши, може да врати неку вредност
- Разлози због којег се користе функције су:
  - избегавање поновног писања већ написаног дела кода
  - лакше разумевање кода

vladaf@matf.bg.ac.rs

Функције (2)

- Функција се дефинише помоћу кључне речи function, на следећи начин:

```
function ime_funkcije(lista_parametara)
{
    blok;
}
```

- За име функције важе иста правила као за име променљивие, с том разликом да се код функције мала и велика слова не разликују и што се не пише се знак $.
  - Препоручује се да име функције асоцира на акцију коју она треба да раеализује
  - За имена функција не могу искористи имена већ постојећих функција
- Листа параметара функције је низ променљивих, међусобно раздвојених зарезом. Чак и када функција нема параметара обавезно је писање малих заграда у њеној дефиницији.

vladaf@matf.bg.ac.rs

Функције (3)

- Да би се функција извршила, није довољно да само буде дефинисана - она мора да буде позвана
- Препоручује се да пре позивања функција претходно буде дефинисана
- Приликом позивања функције, наводи се листа аргумената – изрази раздвојени зарезима
  - Редослед аргумената, који се наводе у позиву, треба да одговара редоследу параметара у дефиницији функције
  - Није неопходно да број аргумената буде једнак броју параметара
    - Ако је број аргумената већи од броја параметара, вишак аргумената се игнорише
    - Ако је број аргумената мањи, при позиву функције се недостајућим параметрима додељује вредност null, при чему се неће зауставити извршавање функције

vladaf@matf.bg.ac.rs

Функције (4)

- Примери декларације и позива функције

zad10-01-funkcija-poziv.php pog10-php-funkcije

```php
<?php

// definicija funkcije
function prikaziPoruku()
{
    echo "Zdravo, zivo!";
}

prikaziPoruku(); // poziv funkcije
?>
```

zad10-03-funkcija-argumenti.php pog10-php-funkcije

```php
<?php

function opis($ime, $godina)
{
    echo "$ime Markovic, godiste $godina.<br/>";
}

opis("Mirko", 1975);
opis("Slavko", "1972");
opis("Milena", "1939");
opis("Zivana", "1953");
opis("Petar", "1960");
opis("Mitar", 1948);

?>
```

vladaf@matf.bg.ac.rs

Функције (5)

- Ако функција враћа неку вредност, она се у позиву може доделити некој променљивој или употребити у неком изразу
- Када се извршавање функције заврши, ток програма се наставља, тачно од места одакле је позвана
- Извршавање функције може да се прекине на два начина:
  1. када се се изврше све наредбе у телу функције (блока)
  2. када се (у блоку), изврши наредби за повратак return
- Пример функције која враћа вредност

zad10-04-povratna-vrednost.php pog10-php-funkcije

```php
<?php
function suma($x, $y) {
    $z = $x + $y;
    return $z;
}

echo "5 + 10 = " . suma(5, 10) . "<br>";
echo "7 + 13 = " . suma(7, 13) . "<br>";
echo "2 + 4 = " . suma(2, 4);
?>
```

vladaf@matf.bg.ac.rs

Функције (6)

- Навођењем променљивих у листи аргумената, променљиве се могу проследити на два начина:
  1. по вредности, када нема потребе да евентуална промена вредности параметра током извршавања функције буде трајна тј. да се одлика на аргумент
  2. по референци (помоћу симбола &), када је неопходно да се вредност аргумента промени током извршавања функције
- Пример функције са позивањем по референци

zad10-05-prenos-po-referenci.php pog10-php-funkcije

```php
<?php
function uvecaj_prvi_argument(&$x, $y)
{
    $x += $y;
}

$broj = 10;
echo "broj = " . $broj . "<br>";
uvecaj_prvi_argument( $broj, 4 );
echo "broj = " . $broj . "<br>";
uvecaj_prvi_argument( $broj, 6 );
echo "broj = " . $broj . "<br>";
?>
```

vladaf@matf.bg.ac.rs

Функције (7)

- Параметри могу да имају и подразумеване вредности
- Ако при позивању функције нису наведене вредности аргумената, тада ће параметри узети подразумевану вредност
- У декларацији функције се параметри са подразумеваним вредностима наводе на крају листе параметара
- Пример функције са подразумеваним вредностима параметара

zad10-06-podrazumevani-argument.php pog10-php-funkcije

```php
<?php

function podesiVisinu($visina = 50)
{
    echo "Visina je : $visina <br>";
}

podesiVisinu(350);
podesiVisinu(); // bice iskoriscena podrazumevana visina 50
podesiVisinu(135);
podesiVisinu(80);

?>
```

vladaf@matf.bg.ac.rs

Функције и опсези важења

- Променљива декларисана унутар функције има свој локални опсег важења – тело функције у ком је декларисана
- Променљива декларисана у оквиру скрипте (ван свих функција) има свој опсег глобални важења
- Променљива са глобалним опсегом важења се не може дитектно користити унутар функције
- Пример покушаја директног коришћења променљиве са глобалним опсегом важења у функцији

zad02-05-opseg-vazenja-promenljivih.php pog02-php-promenljive

```php
<?php
$x = 5; // globalni opseg vazenja

function myTest()
{
    // koriscenje x unutar ove funkcije ce dovesti do greske
    echo "<p>Vrednost promenljive x unutar funkcije je: $x</p>";
}
myTest();

echo "<p>Vrednost promenljive x van funkcije je: $x</p>";
?>
```

vladaf@matf.bg.ac.rs

Функције и опсези важења (2)

- Пример покушаја коришћења променљиве са локалним опсегом важења ван функције

zad02-06-opseg-vazenja-promenljivih.php pog02-php-promenljive

```php
<?php
function myTest()
{
    $x = 5; // lokalni opseg vazenja
    echo "<p>Vrednost promenljive x unutar funkcije je: $x</p>";
}
myTest();

// koriscenje x van funkcije ce dovesti do greske
echo "<p>Vrednost promenljive x van funkcije je: $x</p>";
?>
```

vladaf@matf.bg.ac.rs

Функције и опсези важења (3)

- Ако треба обезбедити успешно коришћење променљиве са глобалним опсегом важења у некој функцији, то се може постићи коришћењем кључне речи global:

  global $ime_promenljive [, $ime_promenljive ...];

- Пример успешног коришћења променљиве са глобалним опсегом важења у функцији

zad02-07-kljucna-rec-global.php pog02-php-promenljive

```php
<?php
$x = 5;
$y = 10;

function myTest() {
    global $x, $y;
    $y = $x + $y;
}

myTest();
echo $y; // prikazace 15, nece biti greske
?>
```

vladaf@matf.bg.ac.rs

Функције и опсези важења (4)

- Ако треба обезбедити успешно коришћење променљиве са глобалним опсегом важења у некој функцији, то се може постићи коришћењем асоцијативног низа $GLOBALS
  - Асоцијативни низ $GLOBALS је расположив увек и свуда, у било ком окружењу, због чега се назива супер-глобалним
  - Елементе овог низа чине вредности, док индексе низа чине имена супер-глобалних променљивих
- Пример коришћења супер-глобалне променљиве у функцији

zad02-08-niz-globalnih-promenljivih.php pog02-php-promenljive

```php
<?php
$x = 5;
$y = 10;

function myTest()
{
  $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

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

vladaf@matf.bg.ac.rs

Функције и опсези важења (5)

- Локалне променљиве функције између различитих позива губе своје вредности, па се локалним променљивима поново додељују вредности приликом сваког извршавања функције
- Постоји могућност да се вредности локланих променљивих функције из претходног позива сачувају - тако што се променљиве унутар функције декларишу као статичке
- Декларисање статичке променљиве се реализује помоћу кључне речи static
- Приликом декларисања статичке променљиве, тој променљивој се може и доделити почетна вредност

  static $ime_promenljive [= izraz];
- Почетна вредност ће статичкој променљивој бити додељена само при првом позиву функције у којој је та променљива декларисана

vladaf@matf.bg.ac.rs

Функције и опсези важења (6)

- Пример коришћења статичке променљиве у функцији

zad02-09-staticke-promenljive.php pog02-php-promenljive

```php
<?php

function myTest()
{
  static $x = 0;
  echo $x;
  $x++;
}

myTest(); // prikazuje 0
myTest(); // prikazuje 1
myTest(); // prikazuje 2

?>
```

vladaf@matf.bg.ac.rs

Захвалница

Делови материјала ове презентације су преузети из:

- Скрипте из предмета Увод у веб и интернет програмирање на Математичком факултету, аутор проф. др Филип Марић
- Скрипте из предмета Информатика на Универзитету Milano Bicocca, аутор Mirko Cesarini
- Књиге Head First PHP & MySQL, аутори Lynn Beighley и Michael Morrison, издавач O'Reilly, 2009.
- Књиге Head First Servlets and JSP, аутори Bryan Basham, Kathy Sierra и Bert Bates, издавач O'Reilly, 2008.
- Веб сајта „W3Schools", на адреси http://www.w3schools.com