Fakultet elektrotehnike, računarstva
i informacijskih tehnologija Osijek

Web programiranje

LV4: PHP - Uvod

ver 1.1 | Autor: Luka Omrčen

Contents

1 Uvod

PHP (rekurzivna akronim za PHP: Hypertext Preprocessor) široko je korišten dinamički jezik otvorenog koda opće namjene koji je prikladan za razvoj web aplikacija koji se može integrirati unutar HTML-a. Ono što razlikuje PHP od jezika kao JavaScript-a je to što se JavaScript izvršava na klijentovoj strani, dok PHP kod se izvršava na poslužitelju, generirajući HTML koji se šalje na klijentovu stranu. Klijent dobiva rezultate izvođenja php skripte sa servera, ali ne i izvorni kod koji se nalazi na serveru (Slika 1).

Figure 1: Prikaz izvođenja php skripte

2 Osnovna sintaksa

Kad PHP analizira datoteku, traži otvaranje i zatvaranje oznaka (**<?php ?>**). Interpretiran je sav kod koji se nalazi unutar PHP oznaka. PHP parser ignorira sve izvan navedenih oznaka. Postoje dva načina definiranja PHP oznaka:

Ako PHP datoteke ima samo PHP kod unutar sebe - u navedenom slučaju moguće je izostaviti oznake zatvaranja. Time se sprječava dodavanje slučajnih razmaka ili novih redaka nakon PHP završne oznake.

```
<?php
echo 'Hello World';
```

Kod 1: Primjer php kod bez oznake zatvaranja

Ako se PHP kod koristi u kombinaciji s HTML-om potrebno je zatvoriti oznake kako ne bi došlo do greške prilikom interpretiranja koda. PHP oznake mogu se koristiti unutar bilo kojeg strukturiranog koda kao što su HTML, JavaScript, JSON itd. PHP vraća definirani tekst koji nije unutar PHP oznaka bez izvršavanja. Kod koji se izvršava unutar HTML-a bit će vraćen na istu lokaciju na kojoj je definiran (Kod 2).

```
<!DOCTYPE html>
<html>
<head>
  <title>Hello World</title>
</head>
<body>
  <h1>My PHP page</h1>
  <p><?php echo 'Hello World'; ?></p>
</body>
</html>
```

Kod 2: Primjer php unutar HTML-a

PHP jezik je **semi-case-sensitive**, što znači da neke značajke razlikuju velika i mala slova, dok druge ne. Unutar PHP-a case-insensitive su:

- sve ključne riječi (if,else,while,for,foreach)
- funkcije
- izjave
- klase

PHP varijable su case-sensitive (Kod 3).

```
<?php
$name = 'Laki';

echo $name;// Laki
echo $Name;// Vraca error
```

Kod 3: Primjer PHP varijabla kao case sensitive

3 Varijable, konstante i okolina

U PHP-u postoje određena pravila koja se treba pridržavati prilikom imenovanja varijable:

- Ispred naziva varijable treba dodati identifikator **$**.
- Naziv varijable mora započeti slovom ili doljnom crtom.
- Ime varijable ne može započeti brojem.
- Naziv varijable ne može biti **$this**.
- Imena varijabli razlikuju velika i mala slova.

```
<?php
$name = 'Luka'; // ispravno
$_name = 'Luka'; // ispravno
$1name = 'Luka'; // error
$this = 'luka';  // error
```

Kod 4: Primjer deklaracija varijabli

Prilikom deklariranja nizova potrebno je definirati niz unutar " (dvostrukih) ili ' (jednostrukih) navodnika.
U PHP-u postoje tri vrste okoline:

- Globalna okolina
- Lokalna okolina
- Statička okolina

Varijabla deklarirana unutar glavnog toka programa (koja se ne nalazi unutar funkcije) nalazi se unutar globalne okoline. Ove varijable ne mogu se koristiti unutar funkcija. Kako bi pristupili varijablama unutar lokalne okoline potrebno je koristiti ključnu riječ **global** ili **$globals** polje.

```
<?php
$number = 10; // globalna varijabla
echo $number; // 10
function num() {
  echo $number;
}
num(); // undefined
//////////////////////////////////////
<?php
$x = 'Ding Dong';
$y = 'Master';
function sayMyName() {
  global $x, $y;
  echo $x, $y;
}
websiteName(); // Ding Dong Master
//////////////////////////////////////
<?php
$x = 'Ding Dong';
$y = 'Master';
function sayMyName() {
  echo $GLOBALS['x'],$GLOBALS['y'];
}
websiteName(); // Ding Dong Master
```

Kod 5: Definiranje globalne i lokalne okoline

Varijable lokalnog okvira brišu se nakon završetka izvršenja funkcije. Ponekad postoje situacije u kojima je potrebno sačuvati vrijednost promjene koja se nalazi unutar funkcije. U tu svrhu potrebno je koristiti ključnu riječ **static** unutar lokalnog okvira kako bi definirali statičku varijablu (Kod 6).

```
<?php
function test() {
  static $number = 0; // deklariranje staticke varijable
  echo $number . '<br>'; // ispis
  $number = $number + 5; // add five to $number
}
test(); // 0
test(); // 5
test(); // 10
```

Kod 6: Definiranje statičke varijable unutar funkcije

PHP nema ključnu riječ za deklariranje konstante. U tu svrhu potrebno je koristiti funkciju **define**(). Konstante sadržavaju sljedeće karakteristike:

- Jednom kada je definirana konstanta, ona se ne može promijeniti.

- Sve konstante imaju globalnu okolinu, čak i kada je definirana unutar funkcije.
- Za razliku od varijabli, konstante nemaju $ prije imena.
- Naziv konstante treba započeti slovom ili doljnom crtom.

```
<?php
define("GREETING", "Hello World");
define("GREETING2", "Hello You");
define("GREETING_3", "Hello everyone");
echo GREETING, '<br>';
echo GREETING2, '<br>';
echo GREETING_3, '<br>';
////////////////////////////////////
<?php
define('GREETING', 'Hello World');
function hello() {
  echo GREETING;
}
hello();
////////////////////////////////////
<?php
function constantInsideFunction() {
  define('GREETING', 'Hello World');
}
constantInsideFunction();
echo GREETING;
```

Kod 7: Definiranje konstanti

4 Znakovi niz

U PHP-u postoje tri načina definiranja znakovnog niza:

- Definiranje niza jednostrukim navodnicima
- Definiranje niza dvostrukim navodnicima
- Heredoc

Najlakši način za deklariranje znakovnog niza je zatvarajući ga jednostrukim navodnikom. Ako je potrebno koristiti jednostruki navodnik unutar teksta koristi se \'. Prilikom korištenja dvostrukih navodnika, PHP ima mogućnost interpretirati varijable koje se nalaze unutar dvostrukih navodnika (Kod 8).

```
$string = 'Hello world';
$string = 'Hello world\'s';
////////////////////////////////////
<?php
$name = "Luka";
```

```
echo "My name is $name";
//////////////////////////////////////
$fruit = 'Apple';
echo "$fruits"; // Error
echo "{$fruit}s"; // Apples
```

Kod 8: Definiranje znakovnog niza

Heredoc pruža alternativni način definiranja znakovnih nizova u PHP-u. Posebno su korisni kada je potrebno definirati niz u više redova. Glavna karakteristika heredoc-a je ta što automatski izbjegava specijalne znakove unutar niza. To olakšava pisanje HTML, XML, SQL itd. unutar PHP znakovnih nizova.

```
<?php
$name = 'Luka';
$string = <<<STR
I'm a heredoc
I parse variables. (You see $name)
\t \\t adds additional tab space. That means I accept escape sequences
STR;

echo '<pre>' . $string . '</pre>';
```

Kod 9: Znakovni niz definiran Heredoc sintaksom

5 Tipovi podataka

U PHP-u, za razliku od drugih programskih jezika, nije potrebno deifnirati odgovarajući tip podataka ispred deklaracije varijable. PHP automatski bira odgovarajući tip podatka. Postoje više podataka:

- Boolean
- Integer
- Float ili Double
- String
- Array
- Object
- Null

Korištenje **var_dump()** funkcije dobivamo podatke o varijabli. Važnost ove funkcije osim što vraća vrijednost, vraća i tip podatka varijable.

```
<?php
$x = 12;
var_dump($x); // int(12)
//////////////////////////////
<?php
```

```
$a = true;
$b = false;
var_dump($a); //
bool(true)
var_dump($b); // bool(false)
```

Kod 10: Korištenje metode var_dump za prikaz podatkovnog tipa varijable

U PHP-u cijelobrojne tipove podatke moguće je definirati na četiri načina:

- Definiranje cijelih brojeva po bazi 10
- Definiranje cijelih brojeva u heksa zapisu - baza 16 (početni zapis unutar php-a: **0x**)
- Definiranje cijelih brojeva u oktalnom zapisu - baza 8 (početni zapis unutar php-a: **0**)
- Definiranje integriranih vrijednosti binarno - baza 2 (početni zapis unutar php-a: **0b**)

```
<?php
$a = 128;        // decimal zapis
var_dump($a);
$a = -128;       // negativni broj
var_dump($a);
$a = 0x80;       // heksa zapis
var_dump($a);
$a = 0200;       // oktalni zapis
var_dump($a);
$a = 0b10000000;  // binarni zapis
var_dump($a);
```

Kod 11: Korištenje metode var_dump za prikaz podatkovnog tipa varijable

U PHP-u postoji dva načina definiranje polja:

- Polje se može deklarirati pomoću funkcije **array**().
- Deklaracijom sa uglatim zagradama [].

Elementi nizova trebaju biti odvojeni zarezom. Tip pojedinog elementa niza može biti svaki navedeni tip koji je definiran u gornjem tekstu.

```
<?php
$array = array('Apple', 'Banana', 'Orange', 'Mango');
var_dump($array);
$array = ['Apple', 'Banana', 'Orange', 'Mango'];
var_dump($array);
```

Kod 12: Definiranje polja

6 Implicitna i eksplicitna pretvorba podataka

PHP programski jezik ima mogućnost automatske ili ručne pretvorbe jedne vrste podatka u drugu i obrnuto. Popularan programerski naziv za ovakvu vrstu akcije naziva se "Cast".

Postoje dvije vrste pretvorbe podatka:

- Implicitno castanje (Automatsko)
- Eksplicitno castanje (Ručno)

Implicitno castanje (pretvorba) događa se u situacijama kada programski jezik automatski odradi pretvorbu podataka, bez mogućnosti programera da ga promjeni u danom trenutku. Najčešći primjer implicitnog castanja događa se prilikom dijeljenja dva cijela broja, a rezultat dijeljenja je decimalni broj odnosno broj s pomičnom točkom (Kod 13).

```
<?php
$x = 2;
$y = 4;

var_dump($x / $y); // 2/4 = 0.5 (Float)
var_dump($y / $x); // 4/2 = 2 (Int)
```

Kod 13: Primjer implicitnog castanja (pretvorba)

Eksplicitno castanje definira programer prije same interpretacije PHP programa s jasnom naznakom u koji tip podatka želi promijeniti navedenu varijablu.

```
<?php
$x = 5.35;
$y = (int) $x; // castanje varijable X u cijelobrojni tip podatka
var_dump($y);
```

Kod 14: Primjer eksplicitnog castanja (pretvorba)

Prilikom castanja dopušteni su sljedeći izrazi:

- (int) ili (integer) - castanje u cijelobrojni tip podatka
- (float), (real) ili (real) - castanje u broj sa pomičnom točkom
- (string) - castanje u znakovni niz
- (array) - castanje u polje
- (object) - castanje u objekt

7 Operatori

Operator unutar programskog jezika predstavlja određenu radnju nad elementima. PHP ima 11 vrsta operatora:

- Aritmetički operatori

- Operatori dodjele
- Operateri usporedbe
- Logički operatori
- Operatori za povećanje / smanjivanje
- Operatori znakovnih nizova
- Operatori polja
- Operatori tipova podataka
- Operatori nad bitovima
- Operatori za kontrolu pogrešaka
- Izvršni operatori

Popis svih operatora i primjeri unutar koda nalaze se na poveznici [2].

8 Uvjeti

U PHP-u, postoje ukupno šest vrsta uvjetnih grana:

- If uvjet
- If-Else uvjet
- If-Elseif-Else uvjet
- Switch uvjet
- Ternarni operator
- Null operator koaliranja

PHP programski jezik podržava korištenje **elseif** i **else if** sintakse prilikom definiranja uvjeta.

```
<?php
$day = date('j'); // dan u mjesecu
if ($day >= 21) {
  $quarter = 'zadnji tjedan';
} else if ($day >= 14) {
  $quarter = 'treci';
} elseif ($day >= 7) {
  $quarter = 'drugi';
} else {
  $quarter = 'prvi';
}
echo $quarter;
```

Kod 15: Primjer elseif uvjetnog grananja

Switch case uvjet ima standardnu sintaksu kao i u drugim standardnim programskim jezicima.

```
switch(expression) {
  case value1:
    // kod uvjeta ukoliko je value1 == expression
    break;
  case value2:
    // kod uvjeta ukoliko je value1 == expression
    break;
  ...
  default:
  // kod uvjeta
}
```

Kod 16: Primjer elseif uvjetnog grananja

Ternarni operator predstavlja kraću sintaksu if uvjetne grane (Kod 17). Od PHP-a 5.3, moguće je izostaviti dio istinskog uvjeta - **(uvjet) ?: (on false)** vraća vrijednost (uvjet) ako je on istinit. Ako uvjet nije ispunjen, vraća se **false** vrijednost iz neispunjenog uvjeta. Ternarni operator obično se koristi s funkcijom **isset()** u PHP-u.

```
<?php
$username = null;
echo 'Hello ' . ($username ?: 'Guest') . '<br>'; // Hello Guest
////////////////////////////////////
$firstname = 'Luka';
$lastname = 'Junior Developer';
$username = (isset($firstname, $lastname)) ? $firstname . ' ' . $lastname : "
    Guest";
echo $username;
```

Kod 17: Primjer elseif uvjetnog grananja

PHP-ov null operator koaliranja korisna je nova značajka koja je uvedena u PHP 7. Nulti operator koaliranja može se koristiti za dodjeljivanje zadanih vrijednosti varijabli ako im vrijednost nije prethodno dodijeljena (Kod 18).

```
<?php
$x = $a ?? 2;
var_dump($x);
// Null operator radi isto sto i sljedece naredba
if (isset($a)) {
  return $a;
} else {
  return 2;
}
////////////////////////////////////
<?php
$a = null;
$b = null;
$c = 5;
```

```
$d = 3;
var_dump($a ?? $b ?? $c ?? $d); // 5
```

Kod 18: Primjer korištenja operatora koaliranja

9 Petlje

U PHP-u postoje 4 vrste petlje:

- while
- do-while
- for
- foreach - petlja koja iterira kroz polje

Više o petljama i njihovo deklariranje nalazi se na poveznici [1]. Kontrolu iteriranja unutar petlji moguće je izvesti pomoću:

- **break** - prekida izvršenje trenutne petlje.
- **continue** - preskače sljedeći dio trenutne iteracije petlje i skoči na uvjet provjere sljedeće iteracije petlje.

```
<?php
$a = 0;
while ($a < 10) {
  echo $a . '<br>';
  if ($a === 5) {
  break;
}
$a++;
}
//////////////////////////////////////
<?php
$a = 0;
while ($a < 10) {
if ($a === 5) {
  $a++;
  continue; // 5 je preskoceno
}
echo $a . '<br>';
$a++;
}
```

Kod 19: Korištenje break i continue kao kontrolu toka petlje

10 Funkcije

Imenovanje funkcija u PHP-u gotovo je isto kao i imenovanje varijabli, osim znaka $ na početku. Postoje određena pravila prilikom definiranja funkcija:

- Naziv funkcije treba započeti slovom ili doljnom crtom.
- Naziv funkcije ne može se započeti brojem.
- Slova, brojevi i doljne crte mogu se koristiti nakon prvog slova u funkciji.
- Nazivi funkcija ne razlikuju velika i mala slova (mjau() i Mjau() odnose se na istu funkciju.)

Funkcija u PHP u definira se standardno kao i u ostalim programskim jezicima

```
<?php
function vrijeme() {
$hour = date('G'); // dohvati sat u 24 satnom formatu
  if ($hour < 12) {
    echo 'Prijepodne';
  } else if ($hour < 17) {
    echo 'Popodne';
  } else {
    echo 'Vecer';
  }
}
vrijeme();
```

Kod 20: Definiranje funkcije unutar PHP-a

PHP nudi mogućnost definiranja vrijednost unutar argumenta. Potrebno je definirati osnovni operator dodjele (=) u samoj definiciji funkcije. Ako se argument ne navodi prilikom pozivanja funkcije, koristi se ova zadana vrijednost. Također postoji i deklaracija tipa podatka argumenta koja se koristiti za određivanje vrste podataka za svaki navedeni argument. PHP će baciti grešku ako dođe do pogrešnog tipa podatka. Vrstu podataka potrebno je postaviti prije definiranog argumenta.

```
<?php
function printNumber($number = 10) {
  echo "Broj je: $number <br>";
}
printNumber(2);
printNumber(25);
printNumber(); // Broj je : 10
printNumber(500);
////////////////////////////////////
<?php
function details(string $name, int $age, string $country) {
echo "
  $name <br>
  $age <br>
  $country <br><br>
```

```
";
}
details('Madjaro', 22, 'HUN');
```

Kod 21: Definiranje predefiniranih vrijednosti i tipova argumenta unutar funkcije

Važeće vrste za deklaraciju tipa podatka unutar funkcije su:

- **array**
- **callable** - Argument mora biti poziv (funkcija ili metoda)
- **int**
- **float**
- **bool**
- **string**

Funkcije bez imena nazivaju se anonimnim funkcijama. Anonimne funkcije su korisne u situacijama kada je poželjno napraviti "callback", odnosno funkcija se definira kao argument funkciji koja će se trenutno izvršiti.

```
<?php
function callFunc($callBack) {
  $x = rand();
  $callBack($x);
}
callFunc(function($number) {
// navedeni argument predstavlja anonimnu funkciju
  echo $number;
});
```

Kod 22: Implementiranje anonimne funkcije

11 Operacije sa znakovima

PHP ima puno metoda za rad sa znakovima. Neke od najčešće korištenih metoda su:

- strlen() - koristiti za dobivanje duljine niza
- str_word_count() - vraća broj riječi u nizu.
- strtolower() strtoupper() - transformiraju niz u velika ili mala slova
- trim() - uklanja prazne znakove s početka i kraja niza.
- strrev() - preokreće znakove niza
- strpos() - vraća položaj prvog traženog znaka iz niza.
- str_replace() - zamjenjuje tekst unutar niza.
- str_repeat() - ponavlja vrijednost niza n puta

- sprintf() - koristi se za dobivanje formatiranih nizova. Koristi se %d - za cijele brojeve, %s - za string i %f - za float tip podatka

```php
<?php
$str = 'Lule';
echo strlen($str); // 4
////////////////////////////////////
<?php
$str = 'Koliko ovdi ima rici hehe';
echo str_word_count($str);
////////////////////////////////////
<?php
$str = 'Lule';
echo strtolower($str);
echo '<br>';
echo strtoupper($str);
////////////////////////////////////
<?php
$str = '  Lule    ';
echo strlen($str) . '<br>';
$str = trim($str);
echo strlen($str);
////////////////////////////////////
<?php
$str = 'Kabum';
echo strrev($str);
////////////////////////////////////
<?php
$str = 'Hello World';
echo strpos($str, 'World'); // 6
////////////////////////////////////
<?php
$str = 'Good Morning';
echo str_replace('Morning', 'Evening', $str);
////////////////////////////////////
<?php
echo str_repeat('*', 25);
////////////////////////////////////
$myName = 'Lule';
$age = 25;
echo sprintf("My name is %s. I'm %d years old", $myName, $age);
```

Kod 23: Operacije sa znakovima

12 Polja

U PHP-u postoje tri vrste polja:

- Indeksirana polja: polja s numeričkim indeksima.
- Asocijativna polja: polja s oznakama kao ključevima.
- Višedimenzionalni polja: složena polja

Indeksirana polja definiraju se standardno kao i u ostalim programskim jezicima:

```
<?php
$fruits = ['Apple', 'Banana', 'Orange', 'Mango'];
//////////////////////////////////////
<?php
$fruits[0] = 'Apple';
$fruits[1] = 'Banana';
$fruits[2] = 'Orange';
$fruits[3] = 'Mango';
```

Kod 24: Indeksirana polja

Za razliku od indeksiranih polja, asocijativna polja mogu se deklarirati na dva načina:

```
<?php
$age = array(
'Mirko' => 22,
'Slavko' => 25,
'Maticarka' => 30
);
// ili
$age = [
'Mirko' => 22,
'Slavko' => 25,
'Maticarka' => 30
];
//////////////////////////////
<?php
$age['Mirko'] = 22;
$age['Slavko'] = 25;
$age['Maticarka'] = 30;
```

Kod 25: Asocijativna polja

Višedimenzionalna polja imaju nekoliko stupnjeva dubine odnosno dimenzija:

```
<?php
$people = [
  'Zvonko' => [
    'age' => 22,
    'country' => 'France'
  ],
  'Mirko' => [
    'age' => 25,
```

```
    'country' => 'United Kingdom'
  ],
  'Maticarka' => [
    'age' => 30,
    'country' => 'Croatia'
  ]
];
```

Kod 26: Multidimenzionalna polja

12.1 Iteracija polja

U PHP-u za iteraciju polja koristi se petlja **foreach**. Iteracija indeksiranog polja prikazana je u kodu 27.

```
<?php
$fruits = ['Apple', 'Banana', 'Orange', 'Mango'];
foreach ($fruits as $fruit) {
echo $fruit . '<br>';
}
```

Kod 27: Iteracija indeksiranog polja

Iteracija asocijativnog polja definira se na drugačiji način u odnosu na indeksirano polje (Kod **??**).

```
<?php
$people = [
  'Mirko' => 22,
  'Slavko' => 25,
  'Maticarka' => 30
];
foreach ($people as $name => $age) {
echo "$name, $age" . '<br>';
}
```

Kod 28: Iteracija asocijativnog polja

Pri radu s bazama podataka najčešće se rade iteracije kroz višedimenzionalna polja podataka. Iteracija multidimenzionalnog polja prikazana je u kodu .

```

<?php
$data = [
  'motori' => ['Kawasaki', 'Yamaha', 'Suzuki'],
  'auti' => ['VW', 'Ford', 'Audi']
];
foreach ($data as $vehType => $vehMark) {
  echo "<h2>$vehType</h2>";
  foreach ($vehMark as $mark) {
    echo "<div>$mark</div>";
  }
}
```

Kod 29: Iteracija asocijativnog polja

12.2 Operacije nad poljima

PHP definira mnoge metode za manipulaciju nad poljem. Neke od najčešće korištenih metoda su:

- array_push() - dodajte nove elemente u niz. U ovoj funkciji moguće je istovremeno dodati više elemenata.
- array_unshift() - funkcija se koristi za dodavanje elemenata u niz. Nakon spremanja, indeksi polja bit će u skladu promijenjeni
- count() - metoda koja se koristi se za dobivanje duljine niza u PHP-u.
- array_merge() - koristiti za spajanje dva polja
- is_array() - provjerava je li navedena varijabla sadrži polje
- in_array() - provjerava postoji li u polju određeni element
- array_key_exists() - provjerava postoji li određeni ključ u polju
- array_values() - koristi za dobivanje vrijednosti asocijativnog polja kao indeksirano polje.
- array_keys() - koristi za dobivanje vrijednosti ključa asocijativnog polja kao indeksirana matrica.
- array_map() - metoda koja nakon odrađene definirane radnje, vraća novo generirano polje

```
<?php
$fruits = ['Apple', 'Banana', 'Orange', 'Mango'];
array_push($fruits, 'Pears', 'Watermelon');
var_dump($fruits);
////////////////////////////////////
<?php
$indexedArray = ['Mirko', 'Slavko', 'Maticarka'];
var_dump($indexedArray); // 0 => Mirko
array_unshift($indexedArray, 'Slavko', 'Maticarka');
var_dump($indexedArray); // 0 => Slavko
////////////////////////////////////
<?php
$fruits = ['Apple', 'Banana', 'Orange', 'Mango'];
echo count($fruits); // 4
echo '<br>';
$age = array(
'Mirko' => 22,
'Slavko' => 25,
'Maticarka' => 30
);
echo count($age); // 3
////////////////////////////////////
```

```php
<?php
$array1 = ['red', 'blue'];
$array2 = ['yellow', 'green'];
$arrayMerged = array_merge($array1, $array2);
var_dump($arrayMerged);
////////////////////////////////////
<?php
$array1 = ['name' => 'Zvonko', 'age' => 24];
$array2 = ['country' => 'France'];
$arrayMerged = $array1 + $array2;
var_dump($arrayMerged);
////////////////////////////////////
<?php
$int = 1;
$string = 'String';
$array = ['This', 'is', 'an', 'array'];
var_dump(is_array($int));   // false
var_dump(is_array($string));// false
var_dump(is_array($array)); // true
////////////////////////////////////
<?php
$array = ['Mirko', 'Slavko', 'Maticarka'];
if (in_array('Mirko', $array)) {
echo "Mirko is in the array";
}
////////////////////////////////////
<?php
$array = [
'name' => 'Mirko',
'age' => 25,
'country' => 'France'
];
var_dump( array_key_exists('name', $array) );  // true
var_dump( array_key_exists('birthday', $array) );  // false
////////////////////////////////////
<?php
$array = [
'name' => 'Mirko',
'age' => 25,
'country' => 'France'
];
$values = array_values($array);
var_dump($values);
////////////////////////////////////
<?php
$array = [
'name' => 'Mirko',
```

```
'age' => 25,
'country' => 'France'
];
$keys = array_keys($array);
var_dump($keys);
////////////////////////////////////
<?php
$array1 = [24, 12, 45, 23];
$array2 = array_map(function($val) {
return $val * 2;
}, $array1);
var_dump($array2);
```

Kod 30: Operacije nad poljem

13 Superglobalne varijable

Superglobalne varijable su definirane varijable koje su dostupne u svim okolinama. Za pristup superglobalnim varijablama potrebno je koristiti ključnu riječ **$GLOBALS**. Svi superglobali mogu biti definirani kao indeksirana ili asocijativna polja koja sadrže određeni skup podataka unutar okoline. PHP ima devet definiranih superglobala:

- $GLOBALS - Pohranjuje sve globalne varijable okoline.
- $_SERVER - Pohranjuje podatke o poslužitelju i okruženju izvršenja.
- $_GET - Pohranjuje HTTP GET varijable.
- $_POST - Pohranjuje HTTP POST varijable.
- $_FILES - Pohranjuje varijable za prijenos HTTP datoteka.
- $_ENV - Pohranjuje varijable okoline.
- $_COOKIE - Pohranjuje HTTP kolačiće.
- $_SESSION - Pohranjuje varijable sesije.
- $_REQUEST - pohranjuje varijable zahtjeva ($_GET, $_POST i $_COOKIE varijable).

```
<?php
$x = 'Lule';
$y = 'Dzedaj';
function websiteName() {
  echo $GLOBALS['x'], $GLOBALS['y'];
}
websiteName(); // Lule Dzedaj
```

Kod 31: Primjer definiranja superglobalne varijable

$_SERVER je korisna superglobalna varijabla koja sadrži podatke o trenutnoj izvršenoj skripti, mrežnim adresama, lokacijama itd. Najčešće korišteni elementi unutar navedene super globalne varijable su:

- PHP_SELF - Naziv datoteke trenutno izvršene skripte relativan u odnosu na korijen dokumenta.
- DOCUMENT_ROOT - Korijenski direktorij dokumenta na poslužitelju.
- SERVER_ADDR IP - adresa poslužitelja.
- SERVER_NAME - Naziv poslužitelja
- REQUEST_METHOD - Način zahtjeva sa kojim je stranica zatražena. (Na primjer: POST, GET, HEAD itd.)
- REQUEST_TIME - Vremenska oznaka početka zahtjeva.
- HTTP_USER_AGENT - Zaglavlje Agenta koje šalje preglednik. (Koristi se za otkrivanje OS-a, preglednika itd.)
- REMOTE_ADDR IP - adresa trenutnog korisnika.

```php
<?php
echo $_SERVER['PHP_SELF'] . '<br>';
echo $_SERVER['DOCUMENT_ROOT'] . '<br>';
echo $_SERVER['SERVER_ADDR'] . '<br>';
echo $_SERVER['SERVER_NAME'] . '<br>';
echo $_SERVER['REQUEST_METHOD'] . '<br>';
echo $_SERVER['REQUEST_TIME'] . '<br>';
echo $_SERVER['HTTP_USER_AGENT'] . '<br>';
echo $_SERVER['REMOTE_ADDR'];
```

Kod 32: Primjer korištenja $_SERVER superglobalne varijable

14 Dinamički strukturiranje HTML

Programski jezik PHP je dizajniran za interakciju s HTML-om. Kako bi se PHP kod izvršavao unutar HTML-a potrebno je definirati datoteku **.php** ili **.phtml** ekstenzijom.

U kodu 33 prikazano je dinamičko strukturiranje HTML-a pomoću PHP uvjeta. Navedena sintaksa karakteristična je za korištenje unutar HTML-a, a razlikuje se od standardne sintakse koja se definira u PHP kodu bez HTML-a.

```php
<?php
$user = 'admin';
?>

<?php if ($user === 'admin') : ?>
  <p>Admin Console</p>
<?php elseif ($user === 'developer') : ?>
  <p>Developer console</p>
<?php else : ?>
  <p>User Console</p>
<?php endif; ?>
```

```
<p> Some paragraph content </p>
```

Kod 33: Primjer strukturiranja HTML-a pomoću PHP uvjeta

Osim PHP uvjeta, moguće je definirati **for** i **foreach** petlje za iteriranje HTML elemenata unutar definirane strukture.

```
<html>
<head>
<title></title>
</head>
<body>
<?php for ($i = 1; $i < 6; $i++) : ?>
  <li>List Item <?php echo $i ?></li>
<?php endfor; ?>
</body>
</html>
////////////////////////////////////
<?php
$array = [
['Mirko', 'mirko@gmail.com', 24],
['Slavko', 'slavko@gmail.com', 25],
['Maticarka', 'maticarka@gmail.com', 20]
];
?>

<table>
<tr>
<th>Name</th>
<th>Email</th>
<th>Age</th>
</tr>

<?php foreach ($array as $person) : ?>
  <tr>
    <?php foreach ($person as $detail) : ?>
      <td><?php echo $detail ?></td>
    <?php endforeach; ?>
  </tr>
<?php endforeach; ?>
</table>
```

Kod 34: Primjer strukturiranja HTML-a pomoću PHP iteratora

15 PHP obrasci

Obrasci unutar HTML-u definirani su ulazima kao što su tekstualnih polja, tekstnih područja, potvrdni okviri, radio gumba itd. Dobivene korisnikove podatke potrebno je poslati na poslužitelj i obraditi ih.

U web developmentu, rukovanje HTML obrascem predstavlja najvažniji proces. Dva koraka su potrebna za definiranje obrasca:

- Izrada HTML obrasca
- Izrada odgovarajuće PHP skripte koja će primati i obrađivati podatke obrasca

HTTP je protokol, točnije skup pravila, koji omogućava komunikaciju podataka. Ovaj protokol definira neke metode koje ukazuju na radnju koja se mora izvršiti na poslužitelju.

Postoje dvije glavne HTTP metode koje se koriste za stvaranje PHP obrazaca.

- GET
  - Podaci se šalju u URL-u.
  - Zahtjev se može spremiti u memoriju i u povijest preglednika.
  - Nikada se nije koristio za rad s osjetljivim podacima (npr. E-pošta, lozinke).
  - Koristi se za dobivanje ili dohvaćanje podataka.
  - Ograničene su duljine.
- POST
  - Podaci se šalju u tijelu HTTP zahtjeva.
  - Zahtjev se ne može spremiti u memoriju i spremiti u povijest preglednika.
  - Koristi se za rad s osjetljivim podacima.
  - Koristi se za slanje podataka.
  - Nema ograničenja duljine

Otvaranjem veze unutar preglednika, preglednik se koristi GET metodom za dohvaćanje zadanih postavka. U obje se metode podaci mogu poslati kao parovi ključ i vrijednost.

Kod 35 prikazuje definiranu strukturu obrasca. Kako bi pravilno definirali PHP obrazac potrebno je unutar elementa form definirati sljedeće atribute:

- method="POST" - definira HTTP metodu koji se koristi
- action="action.php" - definira URL za slanje obrasca. Pod pretpostavkom da html datoteka i php skripta se nalazi unutar istog direktorija, koristili smo relativnu putanju do skripte.
- name ="name" - definira naziv navedenog elementa forme koji će se unutar superglobalne varijable koristiti kao ključ (index).

```
<html>
<head>
<title>PHP Forms</title>
</head>
<body>

<form method="GET" action="forms-handler.php">
Name: <input type="text" name="name">
Email: <input type="text" name="email">
<input type="submit" name="submit">
</form>

</body>
</html>
```

Kod 35: Primjer definiranog PHP obrasca sa potrebnim elementima

Unutar iste datoteke moguće je definirati HTML obrazac kao i skriptu koja obrađuje navedeni zahtjev. Postavljanje atributa **action** bez vrijednosti definirat će web pregledniku da pošalje zahtjev navedenog obrasca na istu web lokaciju.

```
<!DOCTYPE html>
<html>
<head>
<title>Form</title>
</head>
<body>

<?php if ($_SERVER['REQUEST_METHOD'] === 'POST') : ?>
  <p>User POST data</p>
  <p><?php echo $_POST['name'] ?>,</p>
  <p><?php echo $_POST['email'] ?></p>
<?php else : ?>

<form method="POST" action="">
  Name: <input type="text" name="name">
  Email: <input type="text" name="email">
  <input type="submit" name="submit">
</form>

<?php endif; ?>

</body>
</html>
```

Kod 36: Primjer obrade zahtjeva na istoj web lokaciji

Do sada u primjerima svaki unos bio je proizvoljan. U stvarnom slučaju potrebno je imati popunjena polja za unos. Primjerice, polje elektroničke pošte ne smije ostati prazno - u suprotnom može doći do pogrešnog upisa u

bazu podataka.

Pomoću funkcije **empty()** moguće je provjeriti korisnički unos na server strani prilikom obrade zahtjeva. Navedena funkcija vraća vrijednost **true** u sljedećim slučajevima:

- "" - prazan znakovni niz
- 0
- 0.0
- "0" - 0 kao znakovni niz
- null
- false
- [] - prazno polje

U primjeru koda 37 prikazana je validacija korisničkih unosa sa server strane. Metode trim i htmlspecialschars služe za uklanjanje praznih polja znakova ("whitespace") i pretvorbu posebnih znakova u HTML entitete radi onemogućavanja potencijalnih napada (XSS ili SQL injection).

```
<?php
if ($_SERVER['REQUEST_METHOD'] === 'POST') {
  $name = $email = '';
  if (empty($_POST['name'])) {
    $nameError = 'Popunite rubriku za ime';
  } else {
    $name = trim(htmlspecialchars($_POST['name']));
  }
  if (empty($_POST['email'])) {
    $emailError = 'Popunite email adresu';
  } else {
    $email = trim(htmlspecialchars($_POST['name']));
  }
}
?>
<html>
<head>
<title>PHP Forms</title>
<style type="text/css">
.error {
color:red;
}
</style>
</head>

<body>
<form method="POST" action="">
```

```
  Name: <input type="text" name="name">
  <span class="error"><?php if (isset($nameError)) echo $nameError ?></span>
  Email: <input type="text" name="email">
  <span class="error"><?php if (isset($emailError)) echo $emailError ?></span>
  <input type="submit" name="submit">
</form>
</body>
</html>
```

Kod 37: Primjer validacije unosa na server strani

16 Cookie i Session

Cookie je datoteka s maksimalnom veličinom od 4 KB koju web poslužitelj pohranjuje na klijentsku stranu. Nakon postavljanja kolačića, svi zahtjevi na stranici koji slijede vraćaju naziv i vrijednost kolačića. Kolačić se može čitati samo iz domene iz koje je generiran. Vrijednost cookie-a može vidjeti samo korisnik kojemu je taj cookie generiran. Drugi korisnici ne mogu vidjeti njegovu vrijednost. Većina web preglednika ima opcije za onemogućavanje cookie-a, cookie-a treće strane ili oboje. Cookie-i se koriste radi prilagođavanje korisničkog iskustva - primjer je stranica koja se personalizira se na temelju zadanih postavki u kolačićima.

U primjeru koda 38 prikazana je sintaksa pomoću koje se definira cookie.

```
<?php
setcookie(cookie_name, cookie_value, [expiry_time], [cookie_path], [domain], [
    secure], [httponly]);
```

Kod 38: Definiranje cookie-a

Parametri cookie_name i cookie_value su obavezni dok su ostali parametri proizvoljni. Parametar [cookie_-path] i [domain] vezani su za direktorije domene servera koje mogu ograničiti pristup kolačiću na lokacijama domene i subdomene.

```
<?php
setcookie("name", "Lukito", time() + 60, '/'); // Istice nakon 60 sekundi, /
    definira da se cookie odnosi na cijeli direktorij
echo 'the cookie has been set for 60 seconds';

// Nakon refresha preglednika
print_r($_COOKIE); // Ispis svih aktivnih cookie-a - Super globalna varijabla
```

Kod 39: Primjer korištenja cookie-a

Za uništavanje cookie-a prije isteka vremena potrebno je postaviti vrijeme isteka na vrijeme koje je već prošlo.

```
<?php
setcookie("name", "Lukito", time() -360, '/');
```

Kod 40: Brisanje cookie-a

Session je globalna varijabla pohranjena na poslužitelju. Svakoj sesiji dodjeljuje se jedinstveni id koji se koristi za dohvaćanje pohranjenih vrijednosti. Prilikom generiranja sesije, kolačić koji sadrži jedinstveni id sesije sprema se na korisnikovo računalo i vraća se sa svakim zahtjevom na poslužitelj. Ako preglednik klijenta ne podržava kolačiće u URL-u se prikazuje jedinstveni PHP sesijski id. Sesije imaju kapacitet za pohranu relativno velikih podataka u odnosu na kolačiće. Vrijednosti sesije automatski se brišu kada se preglednik zatvori. Baš kao i varijabla $_COOKIE polja, varijable sesije su spremljene u polje $_SESSION. Kao i kolačići, sesija se mora započeti prije bilo kojih HTML oznaka.

Sesija se koristi u sljedećim sitacijama:

- ukoliko je potrebno prenijeti vrijednost s jedne stranice na drugu,
- kao alternativa cookie-u na preglednicima koji ne podržavaju kolačiće,
- za pohranu globalne varijable na učinkovit i sigurniji način u usporedbi s unošenjem u URL-a web preglednika.

Kako bi stvorili sesiju, potrebno je pozvati PHP metodu session_start, a zatim pohraniti vrijednosti u varijablu $_SESSION (Kod 41).

```php
<?php
session_start();
if(isset($_SESSION['page_count']))
{
  $_SESSION['page_count'] += 1;
}
else
{
  $_SESSION['page_count'] = 1;
}
echo 'Broj posjeta' . $_SESSION['page_count'];
```

Kod 41: Definiranje i unos vrijednosti u Session

Funkcija **session_destroy()** koristi se za uništavanje svih varijabli sesije. Ukoliko je potrebno obrisati određenu jedinicu sesije, potrebno je koristiti funkciju **unset()**.

```php
<?php
session_destroy();
// ili
unset($_SESSION['page_count']);
```

Kod 42: Brisanje sesije

Osim navedenih primera u laboratorijskoj vježbi, potrebno je proučiti dodatne materijale s poveznica [5, 6, 3, 4, 7] kako bi lakše shvatili zašto se koristi PHP kao programski jezik na server strani (umjesto npr. JavaScripta).

References

[1] *Geek For Geeks - Petlje*. URL: https://www.geeksforgeeks.org/php-loops/.

[2] *Geek For Geeks - Php Operators*. URL: https://www.geeksforgeeks.org/php-operators/.

[3] *Google Developers - Netork Analysis*. URL: https://developers.google.com/web/tools/chrome-devtools/network/reference#record.

[4] *Google Developers - Netork Analysis*. URL: https://www.tutorialspoint.com/http/http_overview.htm.

[5] *MDN - how web works*. URL: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/How_the_Web_works.

[6] *Mkyong - HTTP headers in google chrome*. URL: https://mkyong.com/computer-tips/how-to-view-http-headers-in-google-chrome/.

[7] *Stack Overflow - Cookies and Session*. URL: https://stackoverflow.com/questions/11142882/what-are-cookies-and-sessions-and-how-do-they-relate-to-each-other?#answer-11143263.