**Promenljive – varijable**

**00:19:13**

Jedinica: 3 od 19

**Rezime**

* Promenljive u PHP-u moraju ispred imena sadržati znak $ ($mojaPromenljiva);
* Promenljive u PHP-u kao prvi znak imena ne smeju imati broj ili specijalni karakter (*$1mojaPromenljiva* – nepravilno, *$%mojaPromenljiva* – nepravilno);
* Konstante u PHP-u ne moraju imati znak $ ispred imena. *define("PI", 3.14);*
* Konstante u PHP-u mogu biti bilo kog tipa (koji je podržan od strane jezika).

U ovoj lekciji akcenat će biti na samoj definiciji promenljive i njenoj implementaciji u PHP jeziku.

**Promenjive (varijable) u PHP-u**

Varijable su privremeni kontejneri u koji se skladište neke vrednosti. Nakon što je vrednost uskladištena u memoriju, ona biva reprezentovana kroz varijablu (promenljivu).

Tipovi podataka koje jedna varijabla može reprezentovati su, naravno, svi tipovi obrađeni u prethodnoj lekciji.

U prethodnoj lekciji smo takođe pomenuli da je PHP u stanju da automatski – implicitno konvertuje jedan tip podatka u drugi, u zavisnosti od operacije koja je nad njim urađena. Ova osobina jezika naziva se **Loosely typed**(postoji i **Strong typed**).

To znači da nećemo morati mnogo da brinemo o tome kako smo deklarisali određeni tip, sve dok se tom tipu obraćamo na adekvatan način.

Zapravo, kada govorimo o promenljivama u PHP-u, postoji tek nekoliko pravila kojih se moramo držati:

* Promenljive moraju imati **$** znak pre konkretnog naziva;
  + Na primer: $x ili $first\_name

* Naziv promenljive mora započeti slovom ili donjom crtom (\_);
  + Na primer: $a ili $\_a

* Naziv promenljive može posedovati slova (a–z,A–Z), cifre (0–9) i znak (\_);
* PHP je osetljiv na velika i mala slova kada su nazivi promenljivih u pitanju;
  + Na primer: $c\_number i $C\_number su dve zasebne promenljive

* Nazivi za promenljive ne smeju sadržati prazan prostor (razmak / white space), već se umesto toga može koristiti donja crta;
  + Primer nepravilnog naziva: $first name
  + Primer pravilnog naziva: $first\_name

Iako pri imenovanju promenljivih ne moramo poštovati nikakva jezička pravila, poželjno je da se koristi jedna od nekoliko konvencija za pisanje. Jedna od najpopularnijih je Camel Case notacija, koja podrazumeva veliko slovo na početku svake reči u promenljivoj. Npr. *$MojaPromenljiva* ili *$ovoJeMojaPromenljiva*.

Recimo da hoćemo da kažemo da u programu postoji neka promenljiva čiji je naziv $x i koja ima vrednost 10. Napisaćemo:

|  |  |
| --- | --- |
| 1 | $x = 10; |

U nastavku programa biće nam na raspolaganju promenljiva $x sa vrednošću 10. I ako napišemo:

|  |  |
| --- | --- |
| 1 | echo $x; |

Na strani će biti ispisan broj 10.

Ako napišemo:

|  |  |
| --- | --- |
| 1  2 | $myVar = "Hello!";  echo $myVar; |

Na strani će biti napisana poruka Hello!

U PHP-u je moguće dodeliti nazivu varijable vrednost neke druge varijable. Na primer:

|  |  |
| --- | --- |
| 1  2 | $mp = "myVar";  $$mp = "content of myVar"; |

U ovom slučaju, dodelili smo nazivu druge varijable vrednost prve varijable (mp), tako da je sada moguće doći do vrednosti druge varijable kroz varijablu:

|  |  |
| --- | --- |
| 1 | $myVar; |

Ovu vrednost takođe je moguće dobiti i kroz varijablu:

|  |  |
| --- | --- |
| 1 | $$mp; |

Ali i prilikom ovakvog načina definisanja promenljivih mora se pridržavati pravila o nazivima varijabli. Na primer, ako bismo definisali varijablu na sledeći način:

|  |  |
| --- | --- |
| 1  2 | $mp = "1myVar";  $$mp = "content of myVar"; |

Prilikom poziva te varijable ($1myVar), došlo bi do greške, zato što brojevi u početku imena varijable nisu dozvoljeni. Ipak, i pored toga, ova varijabla i vrednost koja joj je dodeljena bile bi uskladištene u memoriji i dostupne kroz sledeću sintaksu:

|  |  |
| --- | --- |
| 1 | ${'1myVar'} |

Treba biti naročito oprezan prilikom ovakvog pristupa, jer ukoliko je neadekvatan, može lako doći do bezbednosnih propusta u aplikaciji.

U PHP-u često ne postoji način da se potpuno osigura postojanje neke promenljive u određenoj tački programa, što može dovesti do funkcionalnih i bezbednosnih problema. Zato se, kako bi se ublažio problem uzrokovan ovim nedostatkom, koristi funkcija **isset()**, koja ispituje postojanje neke promenljive.

Ova funkcija prihvata naziv promenljive kao parametar i vraća vrednost tipa boolean kao rezultat.

isset($myVar)

Ako je $myVar prethodno definisana u kodu, funkcija isset će vratiti kao vrednost **true**. U suprotnom, ova funkcija će vratiti vrednost **false** (o funkcijama ćemo više govoriti u narednim lekcijama).

Ukoliko ipak budete rukovali nepostojećim promenljivama, PHP će biti u stanju da obradi ovakve promenljive na adekvatan način. Na primer:

$a + 2

U slučaju da promenljiva $a ne postoji nigde u prethodnom kodu, PHP će je tretirati kao da joj je vrednost nula, pa će rezultat ovog izraza biti 2. Slična stvar će se dogoditi i ukoliko je izraz:

$a + $b

Tada će rezultat biti nula, jer su obe promenljive tretirane kao nule, iako nisu prethodno definisane.

Ovaj rezultat će zavisiti od podešavanja u PHP.ini fajlu. Nepostojanje promenljive može izazvati grešku.

**Domet promenljive**

Deklarisanje promenljivih nije ograničeno na deklarisanje po vrednosti ili referenci i možete ih deklarisati bilo gde, naravno, u okviru php skriptova. Međutim, kada je u pitanju lokacija same promenljive, može se javiti problem dostupnosti. Problem se javlja ukoliko promenljivu deklarišete u jednom delu aplikacije, a pozivate je u drugom zatvorenom delu. Na primer, ako ste promenljivu deklarisali u okviru neke funkcije ili klase i želite da joj pristupite van tih okvira, program će se pobuniti i reći da promenljiva ne postoji. Tako dolazimo do domenskog pristupa promenljivih.

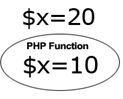
Php promenljive mogu pripadati jednom od četiri dometa:

* + Lokalne promenljive;
  + Parametri funkcija;
  + Globalne promenljive;
  + Statičke promenljive.

**Lokalne promenljive**–Promenljive koje su deklarisane samo u okviru funkcije nazivaju se lokalnim jer mogu da se koriste samo u okviru te funkcije. Kada bismo pokušali da promenljivoj dodelimo neku vrednost van tela funkcije, PHP bi pomenutu promenljivu tretirao kao potpuno drugačiju, novu promenljivu, koja nema veze sa deklarisanom u telu funkcije.

Lokalne promenljive se koriste da bi eliminisale eventualne bočne efekte koji predstavljaju, između ostalog, i slučajne ili namerne izmene globalno dostupnih promenljivih.

Na sledećoj ilustraciji vidimo promenljivu definisanu unutar php funkcije i jednu definisanu izvan nje.



*3.1. Istoimene promenljive definisane izvan i unutar funkcije*

Ove promenljive neće uticati jedna na drugu.

Ako želite to i da dokažete na primeru, možete napisati sledeći kod (neki elementi primera još nisu razjašnjeni u ovom delu kursa, ali su neophodni da bi primer bio kompletan):

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | // Defines the variable with name x and the value 20  $x = 20;  //defines function f  function f(){      //Within a function, defines the variable x with value 10      $x = 10;      //Printing variables on the screen      echo $x;  }  //Calling function f  f();  //Printing variable x  echo $x; |

**Parametri funkcija –**Što se tiče funkcija, objašnjava ćemo ih u nastavku kursa detaljnije. Sada ćemo se zadržati samo na parametrima funkcije, odnosno, argumentima funkcija. Kao što je već poznato, svaka funkcija koja prihvata neke parametre (na primer, sa ulaza) u svom zaglavlju mora imati određene argumente (zapravo, ti argumenti su parametri). Bitno je napomenuti da ovi argumenti dobijaju vrednosti izvan tela funkcije kojoj pripadaju i njima nije moguće pristupati nakon završetka izvršavanja funkcije. Ovakvi tipovi promenljivih, parametri funkcije, deklarišu se nakon navođenja naziva funkcije, unutar zagrada.

Primer:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | //Function f is defined  function f($x)  {  //Variable x increases by 5  $x = $x + 5;  } |

Da bi ova funkcija bila startovana, morali bismo da napišemo:

|  |  |
| --- | --- |
| 1 | f(10); |

Odnosno, mogli bismo napisati:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | //Variable x is defined with the value 10  $x = 10;  //Calling function f i with argument x  f($x);  //Value of variable x shows on display  echo $x; |

Ako bismo startovali prethodni kod (u kombinaciji sa prethodno definisanom funkcijom), na izlazu bi bio prikazan broj 10. Upravo zbog toga što promenljiva koja je ušla u funkciju kao parametar, i tretirana je unutar nje, nema uticaja na spoljašnju funkciju.

**Globalne promenljive –**Možemo reći da su globalne promenljive suprotan pojam od lokalnih promenljivih, koje smo objasnili. Globalnim promenljivama se može pristupati iz bilo kog dela programa. Da bismo imali mogućnost da modifikujemo promenljivu, moramo je eksplicitno deklarisati kao globalnu u okviru funkcije u kojoj želimo da vršimo izmenu. Proglašavanje promenljive globalnom potrebno je da se navede pomoću ključne reči **global** ispred željene promenljive.

Primer:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | $my\_variable = 10;    //Function without arguments  function f()  {      //Variable my\_variable is global      global $my\_variable;      //Variable my\_variable increases by 1      $my\_variable++;  }  f();  //Result shows on display  echo "Result is " . $my\_variable; |

Rezultat nakon izvršenja funkcije je 11, međutim, kada bismo izostavili proglašavanje promenljive globalnom (global $my\_variable) promenljiva $my\_variable bi imala nedefinisanu vrednost nakon izvršavanja funkcije jer se u ovom slučaju tretira kao lokalna promenljiva.

Drugi način korišćenja globalnih promenljivih je deklarisanje promenljive kao $GLOBALS polje. Sada ćemo modifikovati prethodni primer tako da promenljiva $my\_variables bude globalna uz pomoć niza $GLOBALS:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | $my\_variable=10;  function f() {      $GLOBALS["my\_variable"]++;  }  f();  echo "Result is ". $my\_variable; |

Nakon izvršenja ovog koda, na strani dobijamo sledeći prikaz:

Result is 11

jer unutar funkcije nismo kreirali novu promenljivu već smo pristupili globalnoj kroz niz $GLOBALS.

**Statičke promenljive –**Nasuprot promenljivama koje smo navodili u okviru domena parametara funkcija, statičke promenljive se ne uništavaju nakon izvršavanja izračunavanja koje se obavlja u samoj funkciji. Statička promenljiva ne gubi svoju vrednost nakon izvršavanja funkcije, vrednost promenljive ostaje i prilikom narednog pozivanja iste funkcije.

Veliku primenu korišćenja statičkih promenljivih pronalazimo u okviru kreiranja rekurzivnih funkcija. Rekurzivne funkcije su najprostije rečeno funkcije koje pozivaju same sebe dok ne bude ispunjen odgovarajući uslov koji se unapred zadaje.

**Korišćenje statičkih promenljivih**

Kada želimo da lokalna promenljiva zadrži svoju vrednost između poziva funkcija, potrebno je da je proglasimo statičkom, odnosno deklarišemo je pomoću rezervisane reči **static,**na primer:

static $a = 1;

Primer bez korišćenja statičke promenljive:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <?php  function incrementFunc(){      $x = 10;      echo ++$x . "<br>";  }  incrementFunc();  incrementFunc();  incrementFunc();  ?> |

Kako se pri svakom pozivu funkcije kreirala nova lokalna promenljiva $x koja nije statička, tako je i njena početna vrednost uvek bila 10. Zato je rezultat na ekranu sledeći:

11  
11  
11

Primer sa korišćenjem static promenljive:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <?php  function incrementFunc(){      static $x = 10;      echo ++$x . "<br>";  }  incrementFunc();  incrementFunc();  incrementFunc();  ?> |

Rezultat na ekranu je sledeći:

11  
12  
13

Statičke promenljive su blisko vezane za upotrebu klasa (objektno orijentisani koncept).

**Superglobalne promenljive**

Korišćenjem PHP-a otvara se mogućnost korišćenja velikog broja predefinisanih promenljivih kojima se pristupa unutar skripta koji se izvršava, a koje vam mogu omogućiti da imate pregled dodatnih informacija vezanih za okruženje. Informacije takve vrste mogu biti na primer informacije o klijentu, serverskom softveru itd... Pokušajte da izvršite kod iz sledećeg primera i analizirajte dobijeni zapis kako biste uočili povratne informacije od servera.

Primer:

|  |  |
| --- | --- |
| 1  2 | foreach ($\_SERVER as $var => $value)      echo $var . ":" . $value . '<br />'; |

Izvršavanjem koda dobićete veliki broj informacija koje verovatno neće imati veliki značaj za vas, međutim, ukoliko vam je potrebna specifična informacija, na primer, o IP adresi vašeg računara, možete je preuzeti upravo odavde:

|  |  |
| --- | --- |
| 1 | printf('Your IP address is: %s ', $\_SERVER['REMOTE\_ADDR'] ); |

**Konstante**

Kada su u pitanju korisnički definisane vrednosti, osim promenljivih, postoji još jedan tip kome je vrednost, kada ih jednom inicijalizujemo, fiksna. Takvi tipovi se nazivaju **konstante**. Konstante upotrebljavamo kada znamo da se inicijalizovana vrednost sigurno neće menjati tokom izvršavanja programa. Za konstante je, takođe, karakteristično (osim standardnih pravila za promenljive) i to da ne moraju sadržati $ ispred imena promenljive. Najbolji primer za vrednost konstante je 3.14 (odnosno PI).

Konstante se kreiraju pomoću naredbe define("ime konstante", vrednost). Na primer:

|  |  |
| --- | --- |
| 1  2  3 | define("CURRENCY", 80.2);  define("UNIVERSITY", "ITS");  define("PI", 3.14); |

Po konvenciji, konstantama se dodeljuju imena pisana velikim slovima, ali to nije pravilo. PHP prihvata i mala slova u imenu konstante. Pri imenovanju konstanti treba voditi računa da se ne koriste ključne reči PHP-a.

PHP ima puno ugrađenih konstanti koje možete da koristite u svom kodu. Neke od njih su:

*\_\_LINE\_\_*– broj linije na kome je konstanta pozvana

*\_\_FILE\_\_*– ime fajla u kome je konstanta pozvana

Prilikom inicijalizacije treba obratiti pažnju da konstanta može prihvatiti samo skalarni tip podatka.

**Deklaracija promenljivih**

U nekim drugim programskim jezicima neophodno je da se promenljiva deklariše pre nego što joj se dodeli vrednost. U PHP-u je dovoljno samo dodeliti vrednost promenljivoj. Pre samog smeštanja vrednosti u memoriju, PHP sam određuje tip vrednosti i snima je u odgovarajućem formatu. Ukoliko je potrebno, PHP automatski i konvertuje jedan tip podatka u drugi. Na primer:

|  |  |
| --- | --- |
| 1  2  3 | $first\_number = 1; //  like integer  $second\_number = 1.1; // like float  $sum = $first\_number + $second\_number; |

U trećoj naredbi sabiraju se dve vrednosti različitog tipa. Pre samog sabiranja, PHP konvertuje celobrojnu vrednost u realnu, kako bi sabiranje bilo moguće.

Da bismo nekoj promenljivi dodelili ili promenili vrednost, koristimo operatore.

**Greške prilikom korišćenja operatora = i ==**

U slučaju da imate potrebu da poredite neku vrednost sa promenljivom, potrebno je obratiti pažnju da tu vrednost ne dodelite promenljivoj sa kojom vršite poređenje. Iako zvuči zbunjujuće, evo objašnjenja koje će vas sprečiti da napravite najčešću početničku grešku.

Potrebno je da vaš kod izgleda ovako:

if($my\_variable==10)  
{ ...  }

a nikako ne ovako:

if($my\_variable=10)  
      { ... }

Naredbom if ćemo se više baviti u narednim lekcijama, tako da će za potrebe objašnjenja ovog primera biti dovoljno da kažemo da se naredbom if proverava da li je neki uslov ispunjen (u našem slučaju da li promenljiva $my\_variable ima vrednost 10). Ukoliko je uslov ispunjen, blok koda naveden u vitičastim zagradama će biti izvršen. Sa druge strane, ukoliko uslov nije ispunjen, blok koda će biti preskočen. Upravo ovakvu proveru vršimo operatorom ==. Da smo želeli da promenljivoj $my\_variable dodelimo vrednost, iskoristili bismo operator dodele =. Svrha ovog dela lekcije je samo istaći razliku između ova dva operatora.

**Dodeljivanje podrazumevanih vrednosti promenljivama**

Dodeljivanje podrazumevanih vrednosti promenljivama možemo izvršiti na više načina. U prvom, koristićemo funkciju isset() na sledeći način:

|  |  |
| --- | --- |
| 1  2  3  4  5 | // Checking existence of variable  if (!isset ($cars)){  // If the variable does not exist, it will be created  $cars = $default\_cars;  } |

Kada želimo da dodamo podrazumevanu vrednost promenljivoj kojoj vrednost prethodno nije bila zadata, takođe možemo da iskoristimo ternarni operator **(a ? b : c)** na sledeći način (o ternarnom operatoru će biti još reči u narednim lekcijama):

|  |  |
| --- | --- |
| 1 | $cars = isset($\_REQUEST['cars'])? $\_REQUEST['cars']:$default\_cars; |

Ukoliko $\_REQUEST['cars'] ne postoji, promenljivoj će biti dodeljena vrednost $default\_cars, u suprotnom, vrednost promenljive $cars biće dodeljena iz superglobalne promenljive $\_REQUEST['cars'].

**Šta ispituje funkcija isset()?**

 Ispituje postojanje neke promenljive

 Ispituje da li je neka promenljiva celobrojna vrednost

 Ispituje da li je neka promenljiva string

 Ispituje da li je korisnik ulogovan

**Vežba 1**

***Problem:***

U postojećoj aplikaciji, postoji set promenljivih vezanih za bazu podataka kreiran na sledeći način:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <?php    $s = "localhost";    $k = "root";    $s = "";    $b = "myDB";  ?> |

Potrebno je modifikovati promenljive u cilju bolje intuitivnosti.

***Rešenje:***

Ove promenljive nemaju intuitivna imena i vrlo je moguće da će se ukrstiti sa drugim promenljivama u okviru aplikacije. Bolja imena bila bi, na primer:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <?php    $dbServer = "localhost";    $dbKorisnik = "root";    $dbSifra = "";    $dbBaza = "myDB";  ?> |

**Vežba 2**

***Problem:***

Jednostavna aplikacija koja sabira dva broja i emituje ih na izlazu ne funkcioniše. Potrebno je ispraviti grešku:

|  |  |
| --- | --- |
| 1  2  3  4  5 | <?php    $x = 10;    $y = 20;    echo $x+$Y;  ?> |

***Rešenje:***

PHP je osetljiv na velika i mala slova, a u poslednjoj liniji aplikacije se umesto $y nalazi $Y, što dovodi do greške, jer promenljiva $Y ne postoji.

|  |  |
| --- | --- |
| 1  2  3  4  5 | <?php    $x = 10;    $y = 20;    echo $x+$y;  ?> |

**Vežba 3**

***Problem:***

Potrebno je napraviti set konstanti koje će biti korišćene za rukovanje bazom podataka. U ovom trenutku, postoje promenljive:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <?php    $dbServer = "localhost";    $dbKorisnik = "root";    $dbSifra = "myPass";    $dbBaza = "myDB";  ?> |

***Rešenje:***

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <?php    define("DB\_SERVER","localhost");    define("DB\_KORISNIK","root");    define("DB\_SIFRA","myPass");    define("DB\_BAZA","myDB");  ?> |

Kreiranje konstanti vrši se upotrebom funkcije define(). Ova funkcija podrazumevano kreira case-sensitive konstante. Prvi parametar funkcije je naziv konstante, dok je drugi njena vrednost. Moguće je postaviti i treći, opcioni parametar, kojim bi se moglo odrediti da li je naziv konstante neosetljiv na velika i mala slova (case-insensitive).