P H P

Priredio i prilagodio: Dragan Lesić

Uvod

PHP je široko rasprostranjen *Open Source* skript jezik koji se izvršava na strani *web* servera, a namijenjen je za kreiranje dinamičkih *web* stranica, i uopće različitih Internet aplikacija. PHP je nastao 1994. godine od strane Pasmus Ledorfa, a inače je skraćenica od "Personal Home Page Tools". Autor je jezik kreirao radi održavanja vlastitih *web* stranica, a na bazi tada jako zastupljenog *Perl* jezika. Inače, sintaksa samog jezika je vrlo slična sintaksama jezika *C* i *Perl*. Nakon njegovog pojavljivanja, veliki broj programera i dizajnera je uzeo učešća u njegovom daljem razvoju (pošto se radi o *Open Source* projektu), tako da je jezik sada "dogurao" do verzije 4 (ako imate volje i entuzijazma i sami možete raditi na njegovom razvoju).

PHP je podržan od strane velikog broja platformi (gotovo svih), ali se posebno odomaćio na *Unix/Linux* platformi. Direktan konkurent *PHP-Linux-Apache* platformi je *ASP-WindowsNT-IIS* platforma, ali po većini Internet marketing agencija ubjedljivo vodi PHP platforma, uglavnom zato što je besplatna i što je otvorenog koda. Pored svega navedenog, svoju popularnost duguje svojoj sposobnosti da podržava veliki broj sistema za upravljanje bazama podataka (RDBMS), kao što su: na prvom mjestu *MySQL*; pa zatim i ostali: *MS SQL server*, *Oracle*, *Postgre SQL*, *MS Access* i još mnogi drugi.

PHP na strani *web* servera prijedstavlja prijetprocesor kome se prosljeđuju PHP skripte. Ovo u praksi radi na slijedeći način, kreirate HTML stranice i u njih dodate i svoje PHP skripte, date stranice obavezno imaju ekstenziju "*.php". Kada ih postavite na *web* server i korisnik ih zatraži putem svog *browser*-a, *web* server će na osnovu ekstenzije prijepoznati da se radi o PHP stranicama i proslijedit će ih instaliranom prijetprocesoru. Potom će prijetprocesor izvršiti programski kod i rezultat vratiti *web* serveru, koji nakon toga sve šalje *browser*-u. Rezultat procesiranja su najčešće dinamički kreirane HTML stranice, koje se zasnivaju na podacima iz neke od baza podataka.

U slučaju da i sami održavate neki *web site*, sigurno ste se susreli sa problemom održavanja sajta i to kada broj stranica prijevaziđe kritičan broj. U tom slučaju, kada imamo veliki broj stranica, svaki *update* je vrlo mukotrpan. U tom momentu bi trebalo da se okrenete PHP-u ili ASP-u, a sve u zavisnosti od toga na kojoj platformi radi vaš *host* provajder.

Pošto je priča o PHP-u zaista široka, ovdje ćemo stati, a priču ćemo nastaviti u slijedećim prilozima. Prvo što ćemo obraditi je, kako osposobiti vlastiti kompjuter za PHP razvoj.

Instalacija

U prošlom prilogu smo napravili lijepu uvertiru za ono što slijedi, a to je instalacija **PHP** razvojnog okruženja, odnosno svih potrebnih elemenata. Pošto se *PHP* izvršava na serveru, razvoj bi trebao da se odigrava na vašem osobnom kompjuteru koji ima stalnu vezu sa Internetom. Na njemu bi pisali PHP-HTML programski kod i to odmah testirali na serveru. Ali pošto taj "luksuz" većina ne može da priušti rješenje je instalacija vlastitog lokalnog *web* servera sa podrškom za PHP. Ovo će vam pružiti veliku udobnost u radu, a kada završite neki projekt lako ga je poslati na server vašeg host provajdera.

Pošto većina host provajdera radi na platformi "Linux/Apache/PHP/MySQL", i vi bi trebali da na svom osobnom računalu instalirate ovu opciju radi potpune kompatibilnosti (ovo je malo kompliciranije rješenje). Ali, ne mora sve da bude tako, sve ovo može da se odradi i na većinskoj *Windows* platformi. Rješenje se nalazi u instalaciji besplatnih gotovih paketa (namijenjenih *Windows*-u) koji sadrže sve potrebne elemente. Prijeporučićemo vam jedan paket, koji u sebi sadrži sve potrebno:

- Apache web server, verzije 1.3.14
- PHP 4.0.2
- Perl 5
- MySQL 3.23

Paket je proizvod firme "**NuSphere**" koja vlasništvo velike softverske kuće "Progress Software", a instalacioni paket možete "skinuti" sa adrese:

- *www.nusphere.com*

Paket je veličine oko 23 megabajta (zapakiran), a pored gore navedenih komponenti u paketu ćete naći i dosta literature u elektronskom obliku. Kada "skinete" instalaciju i raspakirate biti će dovoljno da startate instalaciju preko fajla "setup.exe". Otvoriće vam se prozor *browser*-a u kome treba samo da kliknete na dugme "install" i krenut će instalacija koja će trajati nekoliko minuta (kao na slici). I to je sve, sada ste sprijemeni za svoje prve korake u *PHP*-u.

Vaša prva skripta

Ako ste pratili prethodna dva priloga i podesili sve onako kako je opisano, onda ste sprijemni da krenete u kodiranje. Pošto je instaliran i *web* server i podrška za *PHP*, prije samog početka potrebno je da pokrenete *Apache* server aktiviranjem fajla:

C:\nusphere\apache\Apache.exe –s

Ovo je *default* putanja do vašeg *web* servera. Nakon ove procedure možemo da krenemo u kreiranje vaših prvih *PHP* stranica.

Za kreiranje *PHP* skriptova potreban vam je bilo kakav tekst editor. To može da bude i onaj koji dolazi uz *Windows* - "*Notepad*", ali vam mi za bilo kakav komotniji rad preporučujemo dva druga editora, a to su:

- **Note Tab Light** - koji možete prijeuzeti sa adrese *www.notetab.com*
- **TextPad** - koji možete prijeuzeti sa adrese *www.textpad.com*

Kodiranje skriptova ćemo započeti od programerskog standarda, programa "Zdravo svijete!" (tj. "Hello World!") iz koga ćemo vidjeti samu sintaksu *PHP*-a. Otvorite neki od editora i ubacite slijedeći kod:

```
<html>
 <head>
 <title>PHP pocetak</title>
 </head>
 <body>
     <?php echo "Zdravo svijete!"; ?>
 </body>
</html>
```

Snimite fajl, obavezno sa ekstenzijom "*.php" ili "*.php3", "*.phtml". Mi vam preporučujemo da to bude ".php", jer je to dio *PHP* standarda. Mada, fajl možete snimiti i sa ekstenzijom *.htm* ili *.html*, s tim što tada morate da podesite server da i takve fajlove, prije slanja u *browser*, propusti kroz *PHP* parser. Fajl, za početak, obavezno snimite u folder "C:\nusphere\apache\Htdocs", koji je *root* folder vašeg servera. Sada ćemo isprobati naš mali *PHP* skript, a ujedno ćemo i istestirati server. Otvorite *browser* i u *Address* liniju ukucajte "http://localhost/proba.php". Dobićete ekran kao na slici, znači samo jednu rečenicu. Izvorni kod ćete vidjeti preko menija "View > Source", a rezultat parsiranja će biti:

```
 <HTML>
 <HEAD>
 <TITLE>PHP pocetak</TITLE>
 </HEAD>
 <BODY>
     Zdravo svijete!
</BODY>
</HTML>
```

Kao što ste i videli u primjeru, svaka *PHP* skripta počinje sa "**<?php**", a završava se sa "**?>**". Skripte se ugnježđavaju u standardne *HTML* stranice, ali mogu biti i u eksternim fajlovima.

Ovdje smo za ispisivanje teksta u *browser*-u iskoristili funkciju "**echo**", a cilj skripta je osnovno upoznavanje sa *PHP* sintaksom. Ovo je bilo dovoljno za početak, nastavak slijedi.

Slanje teksta u browser

Nakon prvog skripta ("Hello World!"), prijeći ćemo na objašnjavanje najjednostavnijih funkcija, to su one koje služe za "slanje" običnog teksta u *HTML* (odnosno *browser* korisnika). Za "slanje" teksta PHP osigurava tri vrste funkcija, a mi smo u prošlom prilogu iskoristili funkciju "***echo***". Za primjer ubacite slijedeći kod u neki ".php" fajl, i pošaljite ga serveru (localhost\fajl.php):

```
<html>
<body>
<?php
print "Ovo je print funkcija.";
print "<p>";
echo "Ovo je echo funkcija.", "  ",
"P.S. Ovdje mozete dodati i drugi string", " ",
"ako stringove razdvojite zarezom.";
print "<p>";
printf ("Ovo je printf funkcija.");
print "<p>";
printf ("Funkcija printf se najcesce
koristi za formatiranje brojki.");
print "<p>";
printf ("Ne zaboravite zagradu sa
funkcijom printf.");
?>
</html>
</body>
```

Rezultat koji će server vratiti će biti:

> Ovo je print funkcija.
>
> Ovo je echo funkcija. P.S. Ovdje mozete dodati i drugi string ako stringove razdvojite zarezom.
>
> Ovo je printf funkcija.
>
> Funkcija printf se najcesce koristi za formatiranje brojki.
>
> Ne zaboravite zagradu sa funkcijom printf.

Kao što vidite upotrijebili smo sve tri funkcije:

- **print** - najjednostavnija funkcija za slanje teksta u prozor *browser*-a. Funkcija može da se navodi i bez, i sa zagradama.
- **echo** - funkcija slična prethodnoj, samo što ovdje možete da kombinirate više stringova, koje obavezno odvajamo zarezima. Sa ovom funkcijom se ne koriste zagrade.
- **printf** - funkcija koja najčešće služi za formatiranje brojki kao integera, decimala itd. Ova funkcija obavezno zahtjeva upotrebu zagrada.

Nakon ove priče probajte da što više eksperimentirate sa ovim funkcijama, pošto su one jako bitne za *PHP/HTML* kombiniranje. U slijedećem prilogu ćemo se baviti varijablama, vrstama varijabli i načinom njihovog definiranja.

Definiranje varijabli

Varijable se u *PHP*-u predstavljaju tako što se ispred naziva varijable postavlja znak dolara (**$**). Pri definiranju varijabli moramo paziti i na upotrebu malih-velikih slova, pošto su nazivi varijabli osetljivi na upotrebu "malih/velikih" karaktera (*case-sensitive*). Pored ovoga moramo paziti i na karakter kojim počinjemo naziv varijable. Iza znaka dolara mogu da idu samo slova alfabeta ili podvučena crta (*underscore*), a iza njih mogu da idu i slova, cifre, ili linije. Evo i primjera pravilnog definiranja *PHP* varijabli:

```
<?php
$varijabla = 'Milan';
$Varijabla = 'Prokic';
// izlaz "Milan, Prokic"
echo "$varijabla, $Varijabla";

$_druga_var = 'pocetak';
// izlaz "pocetak"
echo "<br>$_druga_var" ;
?>
```

Rezultat koji će server vratiti će biti:

Milan,Prokic
pocetak

U verziji jezika *PHP3*, varijablama se uvijek dodjeljuje i vrijednost. Pri ovome, kada jednoj varijabli dodijelimo vrijednost neke druge varijable, njoj se dodjeljuje samo vrijednost prve varijable, bez referenci. Ovo znači, da na primjer, kada jednoj varijabli dodijelimo vrijednost druge varijable, i zatim promenimo vrijednost prvoj varijabli, data promjena neće uticati na vrijednost druge varijable. Sa pojavom verzije *PHP4*, dobija se mogućnost i dodjeljivanja vrijednosti varijablama sa referencom. Ovo znači, da varijabla kojoj smo dodijelili vrijednost druge varijable, uvijek referencira na prvu, tj. uvijek ima njenu vrijednost. Ovo znači, da ako promjenimo vrijednost originalnoj varijabli, promijenit će se i vrijednost druge varijable, ali i obrnuto. Za dodjeljivanje vrijednosti sa referencom, ispred naziva varijable postavlja karakter "ampersand" (**&**). Evo i primjera

```
<?php
$prva = 'Pera';
// Dodela vred. 'Pera' varijabli $prva
$druga = &$prva;
// Referenciranje varijable $prva varijabli $druga
$druga = "Moje ime je $druga";
// Nadgradnja varijable $druga
echo $prva;
// I varijable $prva je promenjena
echo $druga;
?>
```

U ovom primjeru, varijabli "druga" dodjeljujemo vrijednost varijable "prva", i to sa referencom (&$prva). Zatim drugoj varijabli dodjeljujemo novu vrijednost, a samim tim promjena se zbog reference vrši i kod prve varijable. Na kraju će ova mala skripta dva puta "odštampati" jedan isti string (vidi sliku).

Iz ove prethodne priče možete vidjeti da je jako bitno koja verzija *PHP* parsera (interpretera) se koristi na serveru vašeg host provajdera. Jer, ovaj drugi način definiranja varijabli nećete moći da koristite ako vaš provajder radi sa starijim verzijama *PHP*-a. Zato od provajdera obavezno pribavite svu potrebnu dokumentaciju.

Predefinirane varijable

U prethodnom *PHP* prilogu, pisali smo o varijablama i načinu definiranja varijabli. Pored ovih varijabli koje su korisnički definirane, postoje i tzv. "predefinirane" varijable od strane servera ili *PHP* pretprocesora. Sam *PHP* pretprocesor osigurava veliki broj predefiniranih varijabli koje su dostupne svakoj skripti. Pošto ovih varijabli ima stvarno dosta, veliki broj nije dovoljno ni dokumentiran, a i veliki broj zavisi od sistemskog okruženja tj. od: servera, verzije servera, verzije pretprocesora i drugih faktora.

Serverske predefinirane varijable kreira, kao što smo rekli server, pri podizanju sistema. Radi njihovog boljeg iskorišćenja prikazat ćemo vam one najvažnije. Prva među njima je svakako varijabla koja sadrži podatak o serveru i njegovoj verziji - varijabla "**SERVER_SOFTWARE**". A evo i kako možemo da dobijemo podatak o serveru:

- Pokrenite "Apache" *web* server i pokrenite bilo koji tekst editor.
- U novi "*.php" fajl unesite slijedeći kod:

```
<?php
   echo $SERVER_NAME;
?>
```

- Snimite fajl i pozovite ga preko servera u *browser*, radi pretprocesiranja.

Kao rezultat dobićete podataka o serveru i njegovoj verziji. Osim varijable "SERVER_SOFTWARE" još neke bitne varijable su:

- *SERVER_NAME* - koja daje naziv host servera pod kojim se dati skripta izvršava. Ukoliko se skripta izvršava pod virtualnim hostom, ovo će biti vrijednost definirana za virtualni host (*localhost*).
- *SERVER_PROTOCOL* - daje naziv i verziju *web* protokola preko koga se potražuje data stranica. Na primjer "HTTP/1.1".
- *SERVER_PORT* - varijabla kojom dobijamo podatak o portu na serverskoj mašini koji se koristi za komunikaciju. Podrazumjevani port, pri instalaciji je 80.

Pored serverskih varijabli postoje i predefinirane varijable koje kreira sam *PHP* parser. Jedna od ovih varijabli je i *PHP_SELF* - varijabla koja sadrži relativnu putanju do tekućeg dokumenta, u okviru koga se nalazi skript. Primjer, odnosno PHP skript sa predefiniranim varijablama možete vidjeti na slici.

Područje djelovanja varijabli

Do sada smo naučili kako se varijable definiraju i upoznali smo se sa vrstama varijabli. Slijedi vrlo važna priča koja se tiče varijabli, a to je područje djelovanja varijabli i njihov "vijek trajanja". Područje djelovanja varijable predstavlja dio programskog koda unutar koga data varijabla egzistira. Većina PHP varijabli ima **lokalni** karakter, što znači da je data varijabla "vidljiva" samo unutar jedne procedure (ili funkcije), a nije "vidljiva" i za druge procedure u PHP skriptu.

U slijedećem PHP primjeru vidjet ćemo dvije područjei djelovanja varijabli:

```
<?php
$a = 1; /* globalno područje djelovanja */

Function Stampaj () {
   echo $a; /* referenca ka varijabli lokalnog tipa*/
}
Stampaj ();
?>
```

Prethodni skript neće prouzrokovati nikakav izlaz (znači, neće biti štampana jedinica) jer naredba "echo" referencira lokalnu varijablu "$a", a lokalnoj "verziji" ovoj varijabli nije dodjeljena nikakva vrijednost. Zbog ovoga, kada želimo da unutar funkcije upotrebljavamo globalne varijable moramo ih tako i definirati (globalne unutar funkcije):

```
<?php
$a = 1;
$b = 2;

Function Zbrajanje () {
   global $a, $b;

   $b = $a + $b;
}

Zbrajanje ();
echo $b;
?>
```

Rezultat gornjeg skripta bit će "3". Deklariranjem varijabli "$a" i "$b" kao globalne unutar funkcije, mi smo izvršili referenciranje na promjenljive koje se nalaze van funkcije. Pri tome, nema nikakvih ograničenja u broju globalnih varijabli kojima ćete manipulirati unutar jedne funkcije. U slučaju da u gornjem primjeru, u funkciji "Zbrajanje()", niste definirali varijable kao globalne - rezultat bi bio "2".

Prethodni primjer možemo napisati i na drugi način.

```
<?php
```

```
$a = 1;
$b = 2;

Function Zbrajanje () {
   $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}

Zbrajanje ();
echo $b;
?>
```

Ovdje se pristup vanjskim varijablama vrši preko ugrađenog PHP definiranog niza "$GLOBALS". Niz *$GLOBALS* je asocijativni niz naziva globalnih varijabli, gjde indekse predstavljaju nazivi varijabli, te se preko njih dolazi do vrijednosti samih globalnih varijabli.

Još jedna bitna mogućnost kada govorimo o području djelovanja varijabli, je i mogućnost njihovog definiranja kao statičkih varijabli ("**Static**"). Statičke varijable egzistiraju samo u lokalnoj funkciji, s tim što ne gube svoju vrijednost kada programom napustimo datu funkciju. Pogledajte slijedeći primjer:

```
<?php
Function Brojac () {
   $a = 0;
   echo $a;
   $a++;
}
Brojac () ;
echo '<br>' ;
Brojac () ;
?>
```

Ova funkcija je uglavnom beskorisna pošto svaki put kada se pozove (u našem slučaju dva puta), vrijednost varijable "$a" postavlja na nulu i štampa nulu. Instrukcijom "$a++" vršimo povećanje vrijednosti varijable za jedan, ali to ovdje nema nikakvu funkciju, zbog toga što nakon ove instrukcije funkcija završava, a pri njenom ponovnom pozivu vrijednost se vraća na nulu. Da bi funkcija "Brojac()" stvarno bila korisna, varijablu "$a" moramo definirati kao statičku:

```
<?php
Function Brojac () {
   static $a = 0;
   echo $a;
   $a++;
}
Brojac () ;
echo '<br>' ;
Brojac () ;
?>
```

Sada će nakon svakog poziva funkcije, vrijednost varijable "$a" uvećati za jedan (kao na slici). Ovakvim definiranjem vrijednost lokalne varijable čuvamo u memoriji i nakon završetka date funkcije.

Definiranje konstanti

Kao što znate, iz drugih programskih jezika, konstante predstavljaju identifikatore dodjeljenih vrijednosti koji se ne mijenjaju tokom izvršavanja programa. I sam naziv konstanti je logičan, jer "promjenljive" (varijable) programer može mijenjati, dok se vrijednost konstanti ne može mijenjati. PHP u sebi nosi nekoliko predefiniranih konstanti, a naravno osigurava i mehanizam za definiranje novih konstanti tokom *run-time*-a. Inače vrlo su slične varijablama, s tim što se za razliku od njih definiraju uz pomoć funkcije "define()" i kasnije im ne može biti dodjeljena druga vrijednost.

U slijedećim redovima predstavićemo vam neke od predefiniranih konstanti, koje možete koristiti u svakom trenutku:

- *PHP_VERSION* - konstanta koja sadrži vrijednost verzije PHP parsera koja se koristi.
- *PHP_OS* - konstanta koja sadrži string - naziv operativnog sistema na kome se radi PHP parser. Na primjer: Linux
- TRUE - vrijednost true - 1.
- FALSE - vrijednost false - 0.

Da bi isprobali ove konstante, kreiraćemo malu PHP skriptu koja će "odštampati" vrijednost ovih konstanti:

```
<?php
echo PHP_VERSION;
echo TRUE;
echo PHP_OS;
?>
```

Rezultat će zavisiti od vaše konfiguracije.

Kao što smo rekli, pored predefiniranih možemo definirati i naše korisničke konstante. Za definiranje konstanti se koristi ugrađena funkcija "**define()**", čija je sintaksa slijedeća:

```
int define (naziv_konstante, dodjeljena_vrijednost [, int mala_slova])
```

Ovom sintaksom definiramo konstante, koje su vrlo slične varijablama, osim u slijedećem:

- Ispred identifikatora konstanti ne ide znak "$".
- Konstantama možete pristupiti sa bilo kog mjesta u skripti, bez obira na područje prostiranja.
- Kada se jednom definiraju, ne mogu biti redefinirane ili nedefinirane.

Argumenti "naziv_konstante" i "dodjeljena_vrijednost" predstavljaju, sam naziv konstante i njenu vrijednost, a argument "mala_slova" određuje osetljivost na upotrebu malih-velikih slova. Po *default*-u konstanta je osetljiva na upotrebu malih-velikih slova (nije isto "KONSTANTA" i "Konstanta"), a ako argumentu dodjelite vrijednost "1" - onda neće biti. Za primjer definirat ćemo dvije konstante i izvršiti ispis njihovih vrijednosti:

```
<?php
define ("RECENICA", "Recenica koja ce se ispisati.");
echo RECENICA;
?>
```
Skripta će rezultirati ispisom vrijednosti dodjeljenoj konstanti "RECENICA".

Prihvat varijabli iz formulara

Često ćete na stranicama automatiziranih PHP-om, imati formulare. Najčešće će se raditi o anketama, formularima za komentare, glasanje i slično. Na svu sreću, kada se jedan ovakav formular proslijedi u PHP skriptu, sve varijable definirane na formi bit će automatski dostupne PHP skripti. Ovo znači da skripta može da prihvati i varijable koje nisu definirane u samoj PHP skripti. Na primjer, imamo jednostavan formular, koji se metodom "*POST*" šalje na obradu PHP skripti pod nazivom "primjer.php":

```
<FORM ACTION="primjer.php" METHOD="POST">
  IME: <INPUT TYPE="text" NAME="ime"><BR>
  <INPUT TYPE="submit" VALUE="Posalji">
</FORM>
```
Kada ovaj formular popunite i kliknete na dugme "Posalji", metodom "POST" elementi formulara će biti proslijeđeni PHP stranici "primjer.php" na dalju obradu. Stranica "primjer.php" će automatski primiti sadržaj iz proslijeđenog polja "ime" u varijablu "$ime". Evo kako bi mogla da izgleda skripta "primjer.php":

```
<?php
echo "Vase ime je:<BR>";
echo $ime;
?>
```
Znači, kada popunimo polje "IME" na stranici sa formularom, i kliknemo na dugme za slanje, sadržaj ovog polja će primiti varijabla "$ime" u PHP skriptu na stranici "primjer.php", te će se uz pomoć instrukcije "echo" prikazati vrijednost date varijable.

PHP skripte takođe mogu i da prihvataju vanjske nizove varijabli iz formulara. Na primjer, možete imati grupu povezanih varijabli na formularu u niz, ili možete iskoristiti ovu mogućnost za prihvat vrijednosti iz višestrukog *SELECT OPTION* objekta:

```
<FORM ACTION="niz.php" METHOD="post">
  Ime: <INPUT TYPE="text" NAME="osoba[ime]"><BR>
  Email: <INPUT TYPE="text" NAME="osoba[email]"><BR>
  Boja kose: <BR>
  <SELECT MULTIPLE NAME="kosa[]">
    <OPTION VALUE="crna">Crna
```

```
      <OPTION VALUE="smedja">Smedja
      <OPTION VALUE="plava">Plava
      </SELECT>
   <INPUT TYPE="submit" VALUE="Posalji">
</FORM>
```

U slučaju da je uključena PHP mogućnost "*track_vars*", bilo pri samoj konfiguraciji ili pomoću direktive: **<?php_track_vars?>**, sve varijable poslate *POST* i *GET* metodama bit će članovi i globalnih nizova *$HTTP_POST_VARS* i *$HTTP_GET_VARS* (u zavisnosti od upotrebljene metode).

Izrazi (expressions)

Izrazi su jedan od najbitnijih pojmova svakog programskog jezika, pa i samog PHP-a, jer skoro sve što napišete u programskom kodu predstavlja "**izraz**" (engl. *expression*). Najjednostavniju formu izraza predstavljaju varijable i konstante (koje smo upoznali u prethodnim prilozima), odnosno dodjeljivanje vrijednosti istima. Kada napišete "$x=5", vi tada dodjeljujete vrijednost "5" varijabli "$x". Nakon ove dodjele, očekivaćete da "$x" ima vrijednost 5.

Nešto malo kompleksnije izraze predstavljaju **funkcije**. Na primjer, razmotrite sledijeću funkciju:

```
function dodjeli () {

   return 5;
}
```

Ukoliko vam je blizak koncept funkcija, možete pretpostaviti da ova funkcija služi dodjeli vrijednosti (u ovom slučaju vrijednosti 5). Probajte sada da napišete:

```
$x=dodjeli()
```

Ovim izrazom varijabli "$x" dodjeljujemo vrijednost 5, to vam je isto kao da ste napisali "$x=5". Funkcije predstavljaju izraze, koje u sebi sadrže druge izraze, a služe za vraćanje neke vrijednosti. U našem slučaju ova funkcija vraća vrijednost 5, a inače funkcije ne služe samo za jednostavno vraćanje neke statičke vrijednosti, već uglavnom za neka izračunavanja. Ove vrijednosti pri tome ne moraju da budu samo numeričke (*integer*), već mogu da budu bilo koji tip podataka. PHP pri tome podržava, osim spomenutog *integer*-a, i skalarne vrijednosti, stringove i realne brojeve. PHP, takođe, podržava i dva kompozitna tipa (tipovi sastavljeni od drugih tipova): nizove i objekte. Naravno, pri tome, svaki od nabrojanih tipova podataka može biti dodjeljen nekoj varijabli ili može biti vraćen od strane funkcija.

Još jedan dobar primjer izraza predstavljaju operacije uvećanja (*increment*) i smanjenja (*decrement*). Onima koji poznaju sintaksu jezika C sigurno su poznate ove dvije operacije za promjenu vrijednosti, čiji su operatori: **++** i **--**. Ako bi u PHP-u napisali "$x++", ovo ne bi bio izraz pošto prethodno varijabli nismo dodijelili nikakvu vrijednost. Pored ovih operacija, u PHP-u su vrlo uobičajeni i komparacioni izrazi. Rezultat ovih izraza mogu biti samo vrijednosti 0 ili 1, odnosno *FALSE* ili *TRUE*. PHP podržava slijedeće komparacione operatore: **>** (veće), **<** (manje), **==** (jednako), **!=** (nejednako), **>=** (veće ili jednako) i **<=** (manje ili jednako).

Posljednji primjer izraza bi bio kombinacija operator - dodjela. Već smo rekli da vrijednost neke varijable možemo da povećamo za 1, jednostavnim izrazom "$x++". Ali šta ako želite da dodate više od jedan, na primjer četiri? Mogli biste da napišete dodjelu četiri puta, ali to bi bilo neefikasno. Mnogo je komfornije napisati "$x = $x + 4", gde "$x + 4" uvećava vrijednost varijable "$x" za četiri. Međutim, u PHP-u kao i u nekoliko sličnih jezika ovo možete zapisati na kraći, jednostavniji i razumljiviji način: "$x += 4". Kao što vidite ovdje imamo kombinaciju komparacionog operatora i operatora zbrajanja. Treba znati da se na sličan način mogu iskombinirati i ostali operatori, npr. množenje vrijednosti varijable "$x" sa sedam bi bilo "$x *= 7". PHP osigurava potpunu i snažnu implementaciju izraza, a samo njihovo potpuno dokumentiranje prevazilazi ovaj prilog.

Operatori

PHP podržava veći broj operatora, odnosno vrsta operatora. U prethodnom prilogu, kada smo govorili o izrazima spomenuli smo neke operatore poređenja (==) i inkrementalni operator (++). Sada ćemo vam predstaviti najvažnije operatore koji se koriste u kodiranju, a koje ćemo grupirati u nekoliko kategorija:

· **Aritmetički operatori** –

Naravno, nalaze se na prvom mjestu pošto se najčešće upotrebljavaju, a i svih ih poznajemo od ranije. Aritmetičke operatore čini 5 operatora:

- Zbrajanje - ($x + $y) - sumira varijable.
- Oduzimanje - ($x - $y) - predstavlja razliku između varijabli.
- Množenje - ($x * $y) - proizvod dvije varijable.
- Deljenje - ($x / $y) - količnik dijeljenja dvije varijable. Pri dijeljenju dobijamo cjelobrojnu vrijednost ako su operandi cjelobrojne vrijednosti, a ako su decimalne dobijamo decimalnu vrijednost.
- Moduo - ($x % $y) - ostatak pri djeljenju varijabli.

· **Operatori dodjeljivanja** –
Sa njima smo se već sretali. Osnovni operator je "=", koji lijevom operandu dodjeljuje vrijednost desnog operanda. Pri dodjeli vrijednosti sam izraz dodjeljivanja može biti jedan operand. Pogledajte slijedeći primjer:

$x = ($y = 3) + 3
Varijabla "$x" ima vrijednost 6 , što je rezultat zbrajanja vrijednosti 3 i varijable "$y", kojoj je prethodno dodjeljena vrijednost 3. Pored ovog osnovnog operatora imamo i "kombinirane"

operatore, gde kombinujemo operator dodjele i bilo koji aritmetički ili string operator. Na primjer:

```
$x = 3;
$x += 5;
// varijabla $x ima vrijed. 8, jer kao sto smo rekli: $x = $x + 5;

$y = "Cao ";
$y .= "svima!";
// varijabla $y ima vrijed. "Zdravo svima!",
//kombinirali smo operatore "=" i "."
```

· **Komparacioni operatori** –
Ovi operatori, kao što im i ime kaže, služe za komparaciju (upoređivanje) dvije vrijednosti:

- $x == $y - operator "jednako".
- $x === $y - operator "identično". Operator vezan za PHP verzije 4, a označava da su dvije vrijednosti "identične" ako imaju istu vrijednost i ako su istog tipa.
- $x != $y - operator "nejednako".
- $x !== $y - operator "nisu identični", znači suprotno operatoru "identični".
- $x < $y - operator "manje od".
- $x > $y - operator "veće od".
- $x <= $y - operator "manje ili jednako".
- $x >= $y - operator "veće ili jednako".

· **Inkrementalni operatori** –
Kao i kod jezika C, imamo operatore koji služe za povećanje/smanjenje vrijednosti varijabli za jedan:

- ++$x - operator povećava vrijednost varijable $x za jedan, pa tek onda vraća varijablu $x.
- $x++ - vraća se prvo vrijednost varijable, pa se zatim ista uvećava za jedan.
- --$x - smanjuje vrijednost varijable, pa onda vraća njenu vrijednost.
- $x-- - vraća trenutnu vrijednost varijable, pa je zatim smanjuje za jedan.

Evo i jednog primjera:

```
<?php
echo "<h3>Uvecanje nakon dodele</h3>";
$a = 5;
echo "Varijabla a : " . $a++ . "<br>\n";
echo "Varijabla a : " . $a . "<br>\n";
?>
```

U prvom redu će se ispisati pet (5), a u drugom šest (6).

· **String operatori** –
Ovdje postoje samo dva operatora, prvi je operator spajanja stringova - ".", koji kao rezultat vraća spojeni string, lijevog i desnog argumenta. A drugi, je operator dodjele i spajanja - ".=", koji na vrijednost lijevog argumenta dodaje vrijednost desnog argumenta. Na primjer:

```
$a = "Zdravo ";
```

```
$b = $a . "Svijete!";
// sada $b sadrzi "Zdravo Svijete!"

$a = "Zdravo ";
$a .= "Svijete!";
// sada $a sadrzi "Zdravo Svijete!"
```

"If" grananje

Svaka PHP skripta je satkana od serije naredbi, koje mogu biti razna dodjeljivanja, pozivi funkcija, petlje, uvjetna grananja, pa čak i naredbe koje ništa ne rade (tzv. prazne naredbe). Svaka od instrukcija završava, po pravilu, točka-zarezom (;). Pored toga, određeni broj naredbi može graditi "grupu naredbi" koje su same po sebi poseban izraz. Jedan od ovih grupnih izraza je i "IF" grananje. "IF" grananje prijedstavlja jedno od najvažnijih programskih struktura svakog jezika, među njima je i PHP. Ova struktura osigurava uvjetno izvršavanje određenog dela programskog koda. Struktura PHP-ovog "IF" grananja je vrlo slična strukturi C jezika:

```
if (uvjet)
    izrazi
```

Kao što smo rekli u prilogu o izrazima, "uvjet" predstavlja izraz sa određenom istinosnom vrijednošću, koja može biti *TRUE* ili *FALSE*. Ukoliko je vrijednost uvjeta *TRUE*, PHP će izvršiti naredbe koje se nalaze u delu "izrazi", a ako je ta vrijednost *FALSE* - programski kod unutar ove strukture će se ignonirati.

Slijedeći primjer će nam, uz pomoć "IF" grananja, prikazati veću vrijednost od vrijednosti dvije varijable koje se uspoređuju:

```
<?php
  if ($x > $y)
    print "x je vece od y";
?>
```

Često ćete imati situaciju da se u okviru "IF" izraza nalazi više od jedne instrukcije, i tada imamo "grupisane izraze", kao u slijedećem primjeru gde imamo dvije instrukcije:

```
<?php
  if ($x > $y)
    print "x je vece od y";
    $y = $x;
?>
```

Ovdje prvo provjeravamo da li je X veće od Y, i ako jeste izvršava se grupa od dvije instrukcije, gdje prvo ispisujemo poruku, a zatim vrijednost varijable X dodjeljujemo varijabli Y. Ukoliko uvjet nije ispunjen, ove dvije naredbe se neće izvršiti.

Sam "IF" izraz može biti ugrađen u neodređeni broj drugih "IF" izraza, i na taj način vam osigurava veliku fleksibilnost. Primjer za više ugnježdenih "IF" izraza je slijedeći:

```
<?php
  if ($x > $y)
    if ($x > $z)
       print "x je vece i od y, i od z";
?>
```

Često ćete imati potrebu da u slučaju neispunjenja uvjeta izvršite neki drugi kod, a ne samo da ignorirate "IF" izraz. U ovakvim slučajevima koristićemo proširenje "IF" izraza ključnom rječju "ELSE", a koja omogućava izvršenje određenog koda u slučaju da "uvjet" ima vrijednost *FALSE*. Na primjer, u slijedećem primjeru vršimo uspoređivanje dvije varijable i zatim ispisujemo odgovarajuću poruku:

```
<?php
if ($x > $y) {
    print "x je vece od y";
}
else {
    print "x nije vece od y"
}
?>
```

"While" petlja

Nakon "If" strukture prelazimo na obradu najjednostavnijeg tipa petlji u PHP-u, a to je "**While**" petlja. Struktura i sintaksa u PHP-u je vrlo slična sintaksi ove petlje u programskom jeziku C:

```
while (izraz)

    naredbe
```

Semantika same petlje je vrlo jednostavna, ona "kaže" PHP-u da ponavlja izvršavanje ugrađenih naredbi, sve dok je ispunjen uvjet, tj. dok je vrijednost "izraza" *TRUE*. Vrijednost izraza se provjerava samo na svakom početku ponavljanja petlje, tako da ako se ova vrijednost promjeni tokom izvršavanja ugnježdenih instrukcija, izvršenje se neće prekinuti do kraja date iteracije. U slučaju da je vrijednost "izraz"-a *FALSE*, neće se izvršiti ugnježdene naredbe, a izvršenje programa će se prebaciti na slijedeću instrukciju koja dolazi nakon petlje.

Kao i kod "IF" grananja, i ovdje možete grupirati više instrukcija unutar jedne "While" petlje, i to uz pomoć vitičastih zagrada, ili uz korišćenje alternativne sintakse:

```
while (izraz) :
    ...naredbe...
endwhile;
```

U slijedećim redovima možete vidjeti dvije "While" petlje, napisane na dva načina, koje "štampaju" cifre od 1 do 10:

```
<?php
/* 1. primjer */

$a= 1;
while ($a <= 10) {
   print $a++;
}

/* 2. primjer */

$a = 1;
while ($i <= 10):
   print $a;
   $a++;
endwhile;

?>
```

Kao što vidite, ovdje smo upotrijebili inkrementalni operator (++) koji uvećava vrijednost varijable "$a", i to nakon njenog štampanja.

Slična prethodnoj "While" petlji je "**Do...While**" petlja, a razlika je samo u poziciji provjeravanja "izraz"-a. Kod ove petlje provjera istinosne vrijednosti izraza se vrši na kraju petlje svake iteracije. Glavna razlika je u tome što će se kod ove druge petlje (*do...while*) prva

iteracija svakako izvršiti, za razliku od "While" gde se možda neće izvršiti ni jedna iteracija (u slučaju da je vrijednost izraza *FALSE*).

```
<?php
$a = 0;
do {
   print $a;
} while ($a>0);
?>
```

Prethodna petlja će se izvršiti tačno jedanput, jer se nakon prve iteracije provjerava izraz ($a>0), koji će imati vrijednost *FALSE*. Zbog ove vrijednosti izraza petlja će prekinuti dalje izvršavanje.

"For" petlja

Za razliku od "jednostavne" sintakse "While" petlje, "For" petlja ima najsloženiju sintaksu u PHP-u (vrlo sličnu C-u). Struktura i sintaksa je ove petlje je slijedeća:

```
for (izraz1; izraz2; izraz3)

    naredba
```

Prvi izraz (*izraz1*) se provjerava (izvršava) samo jednom (bezuvjetno), i to na početku petlje. Na početku svake iteracije provjerava se drugi izraz (*izraz2*). Ukoliko je istinosna vrijednost ovog izraza *TRUE*, petlja se nastavlja i izvršava se ugnježdena instrukcija(e). Ukoliko je vrijednost izraza *FALSE*, prekinut će se izvršavanje petlje. Također, na kraju svake iteracije petlje provjerava se (izvršava se) treći izraz (*izraz3*).

Svaki od nabrojanih izraza može biti "prazan", tj. ne mora sadržati nikakav izraz. Ako je na primjer, drugi izraz "prazan" petlja će se ponavljati nedefiniran broj puta. Možda vam ova mogućnost na prvi pogled nema smisla, ali sve do momenta kada ovu mogućnost budete upotrijebili sa uvjetnom instrukcijom "break". Razmotrite slijedeći primjer:

```
<?php
/* 1. primjer */

for ($a = 1; $a <= 10; $a++) {
   print $a;
}

/* 2. primjer */

for ($b = 1;;$b++) {
   if ($b > 10) {
      break;
   }
   print $b;
}

/* 3. primjer */

$c = 1;
for (;;) {
   if ($c > 10) {
      break;
   }
   print $c;
   $c++;
}
```

```
/* 4. primjer */

for ($d = 1; $d <= 10; print $d, $d++) ;
?>
```

Od ova četiri primjera, prvi je svakako najjasniji. U drugom primjeru je prikazan "prazan" drugi izraz, pa imamo nedefiniran broj iteracija petlje. Ali, tu smo upotrijebili "If" grananje i instrukciju "break", kojom izlazimo iz petlje (petlja se prekida) kada vrijednost varijable "$b" bude 11 (nakon toga vršimo štampanje vrijednosti varijable). Treći primjer je specifičan jer su sva tri izraza "prazna" (nedefinirana). Tako da će se data petlja izvršavati u zavisnosti od prirode ugnježđenih naredbi. PHP naravno, osigurava i alternativno definiranje "For" petlje (sa dvotočkom):

```
for (izraz1; izraz2; izraz3):
        ...naredba; ...;
endfor;
```

Kontrolna struktura "Switch"

Instrukcija "Switch" je slična seriji "If" naredbi koje se nalaze u istom izrazu. U mnogo situacija, trebat ćete da npr. uspoređujete istu varijablu (ili cio izraz) sa više različitih vrijednosti, i pri tome da u zavisnosti od jednakosti ove dvije vrijednosti izvršite određeni (različit) programski kod. Ova naredba je ekvivalentna naredbi "Select ... Case", *Visual Basic*-a. U naredna dva primjera vidjet ćete dva načina za rješavanje istog problema, prvi način koristi seriju "If" instrukcija, a drugi način naredbu "Switch":

```
 <?php

if ($a == 0) {
   print "a je jednako 0";
}
if ($a == 1) {
   print "a je jednako 1";
}
if ($a == 2) {
   print "a je jednako 2";
}

switch ($a) {
   case 0:
      print "a je jednako 0";
      break;
   case 1:
      print "a je jednako 1";
      break;
   case 2:
      print "a je jednako 2";
      break;
```

```
}
?>
```

Jako je bitno da razumijete način na koji se ova naredba izvršava, jer ćete inače načiniti greške. Naredba "Switch" se izvršava liniju po liniju (zapravo, naredbu po naredbu). U prvom trenutku ne izvršava se ni jedna naredba. Tek kada PHP parser utvrdi koja PHP vrijednost u *Case* izrazu odgovara vrijednosti *Switch* izraza, tada počinje da se izvršava programski kod. PHP izvršava instrukcije do kraja "Switch" bloka, ili do "Break" naredbe. Pazite, ukoliko ne napišete naredbu "Break" na kraju svakog "Case" bloka, PHP će nastavljati izvršavati sve naredne "Case" blokove. Pogledajte slijedeći primjer:

```
<?php
switch ($a) {
   case 0:
   case 1:
   case 2:
      print "a je manje od 3, ali nije negativno";
      break;
   case 3:
      print "a je 3";
}
?>
```

Ukoliko je varijabla *$a* jednaka nuli (0), PHP će izvršiti sve naredbe za ispis (*Print*)! Ako je vrijednost varijable jedan (1), PHP će izvršiti obje, donje naredbe za ispis. Tek ako je vrijednost varijable dva (2), imat ćemo "očekivano" ponašanje ove strukture i ispisat će se samo "*a je manje od 3, ali nije negativno*", jer iza ove naredbe slijedi naredba prekida - "Break". Ovo znači da obavezno morate koristiti naredbu prekida.

Kod naredbe "Switch" uvjet se provjerava samo jedanput, a rezultat se uspoređuje sa svakom "Case" instancom. Naredba "Switch" je bolje rješenje od višestrukih "If ... else if" naredbi, jer se brže izvršava (zato što se uvjet višestruko provjerava). Specijalan slučaj *Case* izraza predstavlja "podrazumjevana" *Case* naredba, čiji se programski kod izvršava ukoliko nije zadovoljena ni jedna prethodna *Case* instanca. Evo i primjera:

```
<?php
switch ($a) {
   case 0:
      print "a je 0";
      break;
   case 1:
      print "a je 1";
      break;
   case 2:
      print "a je 2";
      break;
   default:
      print "a nije ni 0, ni 1, ni 2";
}
?>
```

Funkcije

Ako poznajete bilo koji programski jezik, sigurno ste čuli i za *funkcije*. **Funkcija** prijedstavlja skup iskaza koji je organiziran na poseban način, a *PHP* parser je tretira kao zasebnu programsku cjelinu. Svaka funkcija se sastoji od zaglavlja i tela funkcije, pri tome **zaglavlje** funkcije sadrži identifikator funkcije (naziv same funkcije) i argumente sa njihovim definicijama, a u **tijelu funkcije** sadržane su različite izvršne instrukcije i deklaracije varijabli koje se koriste u funkciji. Naravno, deklaracija promjenljivih se mora napisati prije izvršnih naredbi u kojima se upotrebljavaju date varijable. Evo i primjera jedne vrlo jednostavne funkcije:

```
<?php
function ispisi () {
   echo "Ovo je Vasa prva funkcija!";
}
?>
```

Identifikator "ispisi" u zaglavlju funkcije je sam naziv funkcije, kojim se funkcija identificira i poziva njeno izvršenje. Prazna lista argumenata između zagrada "(" i ")", označava da ova funkcija nema argumenata, tj. iz pozivajućeg dijela *PHP* skripta se ne prenosi ni jedna vrijednost. Između vitičastih zagrada je tijelo funkcije u koje idu deklaracije i instrukcije. U tijelu funkcije "ispisi" imamo samo jednu instrukciju - *echo*, koja će samo ispisati tekst. Kao što vidimo, unutar same funkcije nemamo deklaraciju varijabli.

Ako bi ovakvu funkciju (unutar *HTML* stranice) proslijedili *PHP* parseru, ništa se ne bi desilo, odnosno parser bi vratio praznu *HTML* stranicu jer nigdje u *PHP* skripti nismo pozvali funkciju "ispisi()" na izvršenje. Zato ćemo u prethodnu skriptu dodati i poziv funkcije:

```
<?php
function ispisi () {
   echo "Ovo je Vasa prva funkcija!";
}

ispisi();
?>
```

Kada se dođe do poziva funkcije (*ispisi*), kontrola programa se direktno prenosi u funkciju "ispisi()" i izvršava se tijelo funkcije, odnosno ispisuje se poruka "*Ovo je Vasa prva funkcija!*". Na kraju funkcije, označene zatvorenom vitičastom zagradom, kontrola programa se vraća na prvu instrukciju koja slijedi nakon iskaza kojim se poziva funkcija ***ispisi***.

Svaka funkcija se može pozivati neodređeni broj puta, pa tako ako bi funkciju "ispisi()" pozvali tri puta - tri puta bi se na ekranu ispisala poruka (kao na slici). Što se tiče *PHP*-a, bilo koji valjan kod može da se smjesti u tijelo funkcije, uključujući tu i druge funkcije i klase.

Argumenti funkcija

U prethodnom *PHP* prilogu definirali smo šta su to funkcije i kako se koriste, i šta su argumenti jedne funkcije. Inače, razne informacije se funkciji mogu proslijediti preko liste argumenata, koja prijedstavlja zarezom razdvojenu listu varijabli, i/ili konstanti. Podrazumjeva se da *PHP* omogućava prosleđivanje argumenata funkciji u vidu vrijednosti; argumenti se mogu još proslijediti kao reference (varijable), i kao "podrazumjevane vrijednosti". Takođe, omogućena je i upotreba listi argumenata varijabilne dužine, ali samo u veziji *PHP*-a 4. Kod verzije 3 *PHP*-a, listu argumenata možemo simulirati u vidu niza koji se proslijeđuje funkciji. Na primjer:

```
<?phps
function preuzimanje_niza($input) {
   echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
?>
```

Uglavnom se funkcijama prosljeđuju konkretne vrijednosti, pa tako kada promjenimo vrijednost argumenta unutar funkcije, neće se promjeniti i njegova vrijednost van funkcije. Ukoliko želite da se vrijednosti prosljeđenih argumenata mogu mijenjati unutar funkcije, proslijedite te iste argumente u vidu referenci. A ako želite se argumenti uvijek prosljeđuju kao reference – morat ćete da ispred naziva argumenta dodate znak *ampersand* (&). Evo i primjera prosljeđivanja argumenta u vidu reference:

```
<?php
function dodaj_tekst(&$string) {
   $string .= 'i dodatni dio stringa.';
}
$str = 'Ovo je string, ' ;
dodaj_tekst($str);
echo $str;
?>
```

Rezultat ovog malog skripta će biti ispis - "*Ovo je string, i dodatni dio stringa.*". Slično prethodnoj metodi, varijablu kao referencu možete proslijediti i kada to niste u prvi mah definirali, tada ćete u samom pozivu funkcije dodati znak *ampersand* (&):

```
<?php
function bilo_sta ($rec) {
   $rec .= ' à ovo je dodatni dio.';
}

$str = 'Ovo je string, ';
bilo_sta ($str);
echo $str;
// izlaz ce biti 'Ovo je string, '
bilo_sta (&$str);
echo $str;
```

```
// izlaz 'Ovo je string, a ovo je dodatni dio.'
?>
```

Slično prethodnom definiranju argumenata je i definiranje "podrazumjevanih argumenata" (kao kod jezika *C++*):

```
<?php
function kuhanje_kave ($type = "Grand") {
  return "Najbolja je kava $type.\n";
}

echo kuhanje_kave ();
echo kuhanje_kave ("DonCaffe");
?>
```

Izlaz ove male skripte će biti:

```
Najbolja je kava Grand.
Najbolja je kava Doncaffe.
```

Kao što vidite, ako funkciji ne proslijedimo nikakvu vrijednost argumenta, koristit će se podrazumjevana vrijednost (*Doncaffe*).

Klase i objekti

Klase predstavljaju opću kategoriju u objektno-orijentiranom programiranju, koja opisuje grupu posebnih elemenata koji se nazivaju **objekti** i nalaze se unutar grupe. Klasa je opisni element, i u programiranju služi za definiranje skupa atributa ili skupa akcija (funkcija) koje su dostupne drugim dijelovima programa, a karakteristične su za sve članove jedne klase. Definiranje klasa u objektno-orijentiranom programiranju je vrlo slično definiranju tipova podataka u strukturnom programiranju.

U PHP-u jedna *klasa* je, najjednostavnije rečeno, kolekcija varijabli i funkcija koje manipuliraju ovim varijablama. U slijedećim redovima vidjet ćemo način definiranja jedne klase:

```
<?php
class Racun {
   var $stavke;
   // Stavke koje se kupuju

   // Dodavanje odredj. br $kom artikala na racun br. $ser_br
   function dodaj_stavku ($ser_br, $kom) {
      $this->stavke[$ser_br] += $kom;
   }

   // Skidanje odredj. br $kom artikala sa racuna br. $ser_br
   function ukloni_stavku ($ser_br, $kom) {
      if ($this->stavke[$ser_br] > $kom) {
         $this->stavke[$ser_br] -= $kom;
         return true;
      } else {
         return false;
      }
   }
}
?>
```

Kao što možete vidjeti, definirali smo klasu "Racun" (kupovni račun) koja sadrži određeni broj artikala - stavki (niz *$stavke*) i dvije funkcije:

- **dodaj_stavku()** - funkcija za dodavanje određeneog broja (*kom*) artikala (*ser_br*) na račun
- **ukloni_stavku()**- funkcija za skidanje, također, određenog broja (korisnik ne mora poništiti kupovinu svih) stavki sa računa.

Pošto smo rekli da su klase kao tipovi podataka, da bi ih upotrijebili u programiranju, potrebno je da deklariramo varijable na osnovu klase:

```
$racun = new Racun;
$racun->dodaj_stavku("10", 1);
```

Ovaj kod će kreirati objekat **$racun** koji pripada klasi *Racun*. Zatim smo funkcijom "dodaj_stavku()", na račun dodali 1 komad artikla sa serijskim brojem 10.

Klase inače mogu biti proširenja drugih klasa. Pri tome, proširena klasa ima sve varijable i funkcije osnovne klase (kroz tzv. nasljeđivanje) i može da ima dodatne vlastite varijable i funkcije. Da naglasimo i to da višestruko nasljeđivanje nije podržano. Evo primjera proširivanja prethodno definirane klase *Racun*:

```
class Kupovni_Racun extends Racun {
   var $kupac;

   function unos_kupca ($ime) {
      $this->kupac = $ime;
   }
}
```

Ovim smo definirali novu klasu - *Kupovni_Racun*, koja ima sve varijable i funkcije kao i klasa *Racun*, kao i dodatnu varijablu *$kupac* i funkciju *unos_kupca()*.

PHP - Slanje e-mail poruka sa web-a (I dio)

Nakon osnovnog upoznavanja sa PHP jezikom i njegovom sintaksom, konačno prelazimo na neke ozbiljnije stvari. U ovom i nekoliko narednih priloga obradit ćemo temu "slanje e-mail poruka uz pomoć PHP-a", koja uopće nije "težak zalogaj". Prolaskom kroz ovu temu, prodiskutirat ćemo i još nekoliko pitanja, kao što su: prikaz "rezultirajuće" stranice, provjera ulaznih podataka itd. Radeći na ovoj "e-mail" skripti, javiće vam se više ideja kako da ju upotrijebite i čime možete da ga dopunite. Za početak, evo i same sintakse PHP "mail" instrukcije:

```
mail ([primalac], [tema], [tijelo_poruke], [zaglavlja]);
```

Kao što vidite, nema ničega što bi trebalo da vas uplaši. Ako ste već naprijedniji PHP "skripter", u ovom trenutku ćete već znati šta dalje raditi. Ali, ako ste totalni početnik, onda je je ovo pravo mjesto za vas. Evo i koje točke ćemo obraditi kroz ovu temu:

- Provjera, da li se poziv e-mail forme i izvršenje, vrši sa vašeg sajta, a ne sa neke druge lokacije.
- Dinamičko kreiranje tijela e-mail poruke.
- Osnovna provjera grešaka i provjera popunjenosti svih polja (validacija).
- Kreiranje "rezultujuće" stranice i "rezultujuće" stranice u slučaju greške.
- Kreiranje mail forme (obrasca).

Pa, da krenemo odmah.

Krećemo sa prvom točkom, a to je detekcija "pošiljaoca". Ovo je vrlo važna točka, jer ako ne implementirate ovu mogućnost, neki zlonamjerni PHP "skripteri" mogu lako da iskoriste vašu *web* mail formu za slanje velikog broja *spam* poruka. Nećemo vam prikazati, i kako se to radi, ali verujte - lakše je nego što mislite. Dosta teoretiziranja, imamo "puno posla", pa zato prelazimo na konkretne stvari. Prvo, na prethodnu liniju PHP koda "nakalemićemo" jedno uvjetno "if...then" grananje, tako da možemo zaustaviti slanje e-mail-a u slučaju da nešto nije u redu:

```
if (!$greska) {
mail ([primalac], [tema], [tijelo_poruke], [zaglavlja]);
}
```

Izgleda već kompliciranije, zar ne? Potom, ćemo prijeći na dio koji je vazan za odašiljaoca poruke, gdje prvo kreiramo varijablu - niz:

```
$odasiljaoci = array ('krstarica.com','www.krstarica.com');
```

Gornjom linijom kreiramo niz u kome se nalaze nazivi domena sa kojih će biti dozvoljeno izvršavanje e-mail skripti. Nije bitno kojim redom navedete elemente niza, a umesto naziva domena možete koristiti i IP adrese datih domena. Slijedi PHP kod koji proverava da li se odašiljalac nalazi u našem nizu dozvoljenih odašiljaoca ($odasiljaoci):

```
$privremeni = explode("/",getenv("HTTP_REFERER"));
$brojac = 0;
while($brojac < count($odasiljaoci)){
   if (ereg($odasiljaoci[$brojac], $privremeni[2])) {
      $ispravno = "true";
```

```
    }
    $brojac++;
}
if ($ispravno != "true") { $autorizacija .= "neautoriziran "; }
```

Da vidimo što se tu zbiva:

- Potencijalni pošiljalac se ubacuje u niz "$privremeni[ ]".
- Petljom provjeravamo pošiljaoca, tako što uspoređujemo domen iz varijable "$privremeni[ ]" sa predefiniranim vrijednostima iz niza "$odasiljaoci[]".
- Ukoliko se neki od domena iz niza "$odasiljaoci[]" poklopi sa prosljeđenim domenom, prolazi se kroz ovu "kontrolnu točku" i varijabli "$ispravno" dodjeljuje se vrijednost "true".
- Posljednja instrukcija u "if...then" instrukcija jednostavno popunjava varijablu "$autorizacija" sa stringom "neautoriziran", u slučaju da neko koristi e-mail formu sa drugog servera.

Ovdje ćemo se zaustaviti, te nastavljamo u slijedećem PHP prilogu.

Slanje e-mail poruka sa web-a (II dio)

Funkciju za provjeru odašiljaoca iz prethodnog priloga, vrlo je lako implementirati u vašu PHP skriptu. Potrebno ju je postaviti iznad PHP funkcije "mail()", iz prethodnog priloga. Ako ste dobro pogledali prethodni PHP kod, vjerovatno ste uočili da je nizu "$odasiljaoci[ ]" potrebno malo podešavanja.

Nastavljamo dalje, i u ovom prilogu pozabavit ćemo se kreiranjem tijela e-mail poruke koja će biti odaslana. Prikazat ćemo vam kako možete formirati listu (listing) svih vrijednosti koji se šalju preko formulara, i kako napraviti automatski odgovor za posjetioce vaših stranica. Većina *web* formulara zahtjeva od korisnika unos informacija. Najčešći tip *web* formulara je tzv. "feedback" formular, koji ćemo iskoristiti kao primjer. Kasnije, jednostavno možete prepraviti programski kod i prilagoditi ga drugačijim potrebama.

Slijedeća e-mail poruka (administratorska) najčešće ide *webmaster*-u (odnosno, vama) ili odgovarajućem sektoru vaše firme (ako se radi o firmi). Sadržaj svakog ovakvog mail-a je jednostavna lista sa svim podacima koji su uneseni putem formulara, i odaslati preko skripte. Ove podatke, kao administrator, možete smjestiti u neku bazu podataka, kojoj kasnije možete pristupiti radi analize. Za početak, učitaćemo sve podatke odaslate uz pomoć skripta:

```
function parsiranje_formulara($ulaz) {
  if (count ($ulaz)) {
    while (list ($naziv, $vrijednost) = each($ulaz)) {
        if (is_array ($val)) {
          for ($brojac2=0; $brojac < count ($vrijednost); $brojac2++) {
            $sadrzaj .= "$naziv: $vrijednost [$brojac2]\n";
          }
        } else {
          $sadrzaj .= "$naziv: $vrijednost\n";
```

```
            }
        }
    }
    return $sadrzaj;
}
$sadrzaj = parsiranje_formulara($HTTP_POST_VARS);
```

Ovaj kod nije previše težak. Ono što ovdje treba da dobijemo je varijabla "$sadrzaj" koja je sačinjena od svih ulaznih podatka koji su poslati PHP skripti na obradu . U slučaju da imate više ulaza, dobit ćete listu sličnu ovoj:

```
naziv_ulaza1: vrijednost1
naziv_ulaza2: vrijednost2
naziv_ulaza3: vrijednost3
```

I tako dalje. Pretpostavljate šta bi se desilo da imate samo jedan ulaz. PHP kod koji uobličava sadržaj vaših poruka je slijedeći:

```
$sadrzaj .= "$naziv: $vrijednost [$brojac2]\n";
$sadrzaj .= "$naziv: $vrijednost\n";
```

Varijabla "$naziv" sadrži nazive polja formulara, a varijabli "$vrijednost" se dodjeljuje sadržaj vezan za odgovarajuće polje formulara. Prije nego što krenemo dalje, ispisat ćemo *subject* e-mail poruke koja se upućuje administratoru:

```
$admin_subject = "Odgovor poslat putem PHP-a";
```

Ovdje ćemo se zaustaviti, a nastavit ćemo u slijedećem PHP prilogu. Ako do sada imate utisak da je ovo previše komplicirano, i da je PHP kod koji predstavljamo previše razuđen, ne brinite. Na kraju ove teme predstavićemo vam cio kod, u jednoj celini.

Konekcija na MySQL bazu podataka (I dio)

Nakon više uvodnih članaka o PHP-u i upoznavanja sa sintaksom i osnovnim funkcijama PHP-a, stigli smo i do možda i najkorisnijeg segmenta upotrebe, a to je spajanje sa bazom podataka koja se nalazi na MySQL serveru. Spajanje sa bazama podataka predstavlja i osnovu za izgradnju kompleksnih mrežnih aplikacija. Inače, uz pomoć PHP-a možete se spojiti sa bilo kojim sistemom za upravljanje bazama podataka (*DBMS*), a među PHP programerima su svakako **MySQL** baze podataka najomiljeniji tip. Na vaš budući izbor baza podataka sigurno će uticati i vaš web provajder, tj. tip baza podataka koje on podržava (tj. instaliran server baza podataka).

Kada je riječ o upotrebi MySQL baza podataka, postoji veći broj dobro dokumentiranih PHP funkcija koje će vam pomoći u uspostavljanju veze sa podacima. Međutim, trebat će vam samo nekoliko ovih funkcija u cilju postizanja jednostavne konekcije i selekcije nekih podataka:

- **mysql_connect** - funkcija kojom ostvarujemo vezu sa MySQL serverom; zahtjeva naziv hosta, korisničko ime i lozinku.
- **mysql_select_db** - funkcija za izbor baze podataka od mnogobrojnih koje se nalaze na MySQL serveru.
- **mysql_query** - funkcija za postavljanje SQL instrukcija.
- **mysql_fetch_array** - funkcija za smještanje rezultata SQL upita u niz.
- **mysql_free_result** - funkcija za oslobađanje resursa zauzetih trenutnom konekcijom.
- **mysql_close** - funkcija za prekid veze ka bazi podataka.

Ovo su osnovne funkcije koje vam trebaju za rad sa bazama podataka, a detalje o ostalim funkcijama možete naći u odgovarajućem priručniku koji se nalazi na adresi www.php.net/manual/.

U našem primjeru, pretpostavit ćemo da ste instalirali MySQL server, da ste kreirali bazu podataka i da za datu bazu imate korisničko ime (username) i lozinku (password). U toj bazi kreirajte tabelu "Proizvodi", koja će imati polja: "Naziv", "Tip" i "Kolicina". Samu tabelu popunite sa podacima kao što su ovi dati na slici.

Konekcija na MySQL bazu podataka (II dio)

Nakon upoznavanja sa osnovnim funkcijama za spajanje sa bazom i kreiranja baze i tabele kao pripreme za naš primjer, krećemo i sa samim PHP kodiranjem. Prije nego što počnete, potrebno je da znate naziv servera na kome se nalazi baza podataka (ukoliko radite u lokalu, naziv će biti "local"), validno korisničko ime i lozinka za taj server. Zatim, startamo sa PHP kôdom, kreiranjem konekcione varijable:

```
<php>
$connection = mysql_connect
        ("naziv_servera","kor_ime","lozinka")
          or die("Povezivanje nije moguce.");
```

Funkcija **die()** koristi se za prekidanje skripte i "štampanje" poruke o grešci ukoliko prethodna funkcija "propadne" (u ovom slučaju, ako konekcija nije uspostavljena).

Kada uspostavite vezu, tj. konekciju, slijedeći korak predstavlja izbor baze podataka i kreiranje SQL instrukcije (upita). Pretpostavimo da tabela "Proizvodi" već postoji u MySQL bazi podataka koja se npr. zove "mojaBaza". Potrebno je da eksplicitno izaberete bazu podataka na serveru na koju ćete se spojiti:

```
$db = mysql_select_db("mojaBaza", $connection)
          or die("Baza nije dostupna.");
```

U ovoj točki smo "naredili" PHP parseru da se spoji na MySQL server i selektira bazu podataka. Ako je sve u redu do ove točke, možete odaslati SQL upit i nadati se da će vam se vratiti nekakav rezultat, tj. rezultirajući set slogova. Kreirat ćemo upit koji se zasniva na tabeli "Proizvodi" (definirana u prethodnom prilogu), a koji treba da vrati polja sa nazivima proizvoda, njihovim tipom i količinom proizvoda, s tim da se podaci poredaju po najvećim količinama. Kreirat ćemo varijablu koja će sadržati ovu našu SQL instrukciju:

```
$sql = "SELECT Naziv, Tip, Kolicina
          FROM Proizvodi
          ORDER BY Kolicina DESC";
```

Potom ćemo kreirati varijablu koja će prihvatiti rezultate prethodnog upita, a koji će se dobiti uz pomoć **mysql_query** funkcije. Funkcija mysql_query poseduje dva argumenta; konekciju i SQL varijablu, koje smo prethodno kreirali:

```
$sql_result = mysql_query($sql,$connection)

          or die("Upit nije izvrsen");
```

Ovim bi trebalo da uspostavimo konekciju, selektiramo tabelu, postavimo upit i smjestimo rezultate u varijablu. U slijedećem prilogu ćemo vidjeti kako rezultate upita prikazati korisnicim

PHP - Konekcija na MySQL bazu podataka (III-dio)

Nakon postavljenog i upućenog upita serveru uz pomoć funkcije **mysql_query**, biće potrebno da rezultat sadržan u varijabli **$sql_result** formatiramo, tako da ga podjelimo na odvojene redove, u čemu će nam pomoći funkcija **mysql_fetch_array**:

```
while ($row = mysql_fetch_array($sql_result)) {
//prog. kod
}
```

Kao što vidite, uz pomoć *while* petlje kreirat ćemo niz koji smo nazvali *$row*, i to za svaki slog u rezultirajućem skupu slogova. Da bismo dobili pojedinačne elemente sloga, odnosno polja (*Naziv, Tip, Kolicina*), kreirat ćemo potrebne varijable:

```
$naziv_proiz = $row["Naziv"];
$tip_proiz = $row["Tip"];
$kolicina = $row["Kolicina"];
```

Pošto je osnovna svrha bilo kog upita prikaz i nekakva analiza podataka, smjestićemo dobijene podatke u HTML tabelu, tj. prikazaćemo rezultat u takvom obliku da može da ga iščita bilo koji web browser. Da biste ovo odradili, slijedeći HTML kôd smjestite ispred *while* petlje, kako biste "otvorili" tabelu i kreirali naslovni red tabele:

```
echo "<TABLE BORDER=1>";

echo "<TR>
    <TH>Naziv proizvoda</TH>
    <TH>Tip</TH>
    <TH>Kolicina</TH>
    </TR>
    ";
```

Nakon definiranja varijabli unutar *while* petlje, "odštampat ćemo" ih u tabeli:

```
echo "<TR>

    <TD>$naziv_proiz</TD>
    <TD>$tip_proiz</TD>
    <TD>$kolicina</TD>
    </TR>
    ";
```

Prema tome, konačna *while* petlja izgledat će ovako:

```
while ($row = mysql_fetch_array($sql_result)) {
        $naziv_proiz = $row["Naziv"];
        $tip_proiz = $row["Tip"];
        $kolicina = $row["Kolicina"];
        echo "<TR>
```

```
                <TD>$naziv_proiz</TD>
                <TD>$tip_proiz</TD>
                <TD>$kolicina</TD>
                </TR>
                    ";
}
```

Nakon završetka petlje, slijedi i zatvaranje taga tabele:

```
echo "</TABLE>";
```

I konačno, na kraju je potrebno da oslobodimo sve resurse zauzete izvršavanjem upita, i zatvorimo konekciju ka bazi podataka. Ako ovo ne uradimo, možemo izazvati neželjeno "curenje" memorije ili slične resurs-probleme.

```
mysql_free_result($sql_result);
mysql_close($connection);

?>
```

PHP - Zašto koristiti MySQL kao SUBP?

U jednom od prethodnih PHP priloga mogli ste vidjeti najjednostavniji način za povezivanje PHP stranica i baza podataka. U datom slučaju spojili smo se na *MySQL* server baza podataka, a mogli smo na bilo koji sistem za upravljanje podacima. Ali, zašto smo izabrali baš *MySQL* i zašto se ovaj sistem i koristi u najvećem broju slučajeva?

MySQL predstavlja *relacioni sistem za upravljanje bazama podataka* (RSUBP) koji omogućava da čuvate podatke, da im pristupate i da ih organizirate na najbolji mogući način (ovo zadire u široku temu zvanu "Organizacija podataka"). Pri tome, ovaj RSUBP može da služi u organizaciji podataka i to od nivoa obične liste podataka, pa sve do velike kolekcije tabela sa milionskim brojem slogova. Inače, relacione baze podataka su vrlo moćna alatka koja osigurava da crpite informacije iz višestrukih izvora i pri tome vršite njihovu komparaciju, kombiniranje i sumiranje radi dobijanja prave informacije u bilo koje doba. Na primjer, možete dobiti jednostavnu listu klijenata, ili iskombinirati klijente sa njihovim narudžbinama i pozicijama koje su naručili. Baze podataka osiguravaju potrebnu strukturu i organizaciju koja je potrebna radi operacija za efikasan pristup podacima, čak i kada ovo podrazumjeva veliku količinu informacija i veliki broj tabela (entiteta).

Ipak, šta je to što *MySQL* čini tako posebnim:

- **MySQL je Open Source RSUBP**. Dostupan je na Internetu i pri tome je besplatan. Ovo je veliki kontrast drugim komercijalnim sistemima za baze podataka (kao što su Oracle, MS SQL, Informix i sl.), kreiranih od strane velikih kompanija, koji su pri tome veoma skupi. Međutim, za one koji žele da naruče *MySQL* sistem na CD-u, i pri tome prime odgovarajuću štampanu dokumentaciju, veliki broj firmi osigurava ovu opciju za malu sumu (stotinjak dolara). Pored toga, neke od ovih firmi nude i tehničku podršku i trening kurseve.

- **Brzina**. Svaki od sistema za baze podataka ima područja u kojima se posebno ističe. Za *MySQL*, jedna od ovih područja je brzina - široko je priznato da su odgovori ovog sistema brži nego kod drugih sistema. Upravo zbog brzine, *MySQL* je sistem izbora za Internet aplikacije, gdje se zbog velikog saobraćaja zahtjeva velika brzina.
- **SQL-orijentacija**. *MySQL* podržava standardni *Structured Query Language* (SQL), najkorišćeniji jezik za definiranje i ekstrakciju podataka.
- **Lakoća upotrebe**. Distribucija *MySQL* je relativno mali paket, koji ne zahtjeva stotine i stotine megabajta kao drugi SUBP. Razvojna filozofija ovog sistema fokusirana je na široku i laku upotrebljivost, gdje se u paket ubacuju samo neophodne funkcije. Ovo *MySQL* čini lakim za razumjevanje, lakim za instaliranje, podešavanje i administraciju.
- **Portabilnost**. *MySQL* može da se pokrene na brojnim paltformama, a najvažnije su UNIX, Linux i Windows.
- **Rapidni razvoj**. Suvremeni dodaci za *MySQL* uključuju podršku za transakcije, replikaciju, tekstualno pretraživanje i RAID fajl-sisteme. Pored toga, može se uključiti i podrška za zaključavanje na nivou sloga.
- **Interoperabilnost**. *MySQL* može se koristiti u kombinaciji sa velikim brojem drugog softvera, koji je također besplatan. Ove alatke vam omogućavaju da na lak način iskoristite sve mogućnosti *MySQL*-a.
- **Programibilnost**. Ukoliko postojeći softver ne odgovara vašim potrebama, možete kreirati vlastiti. Dostupni su interfejsi za veliki broj programskih jezika, kao što su: C, C++, Perl, PHP, Python, Java, Ruby, itd.