Strukture za kontrolisanje toka

Naredbe grananja

Naredbe unutar koda se uglavnom izvršavaju redom, onako kako su napisane. Nekad je, međutim, neophodno promeniti tok izvršavanja programa. Naredbe grananja se koriste u slučajevima kada je na osnovu tačnosti nekog uslova potrebni izvršiti jedan ili drugi deo koda.

if...else naredba

Ova naredba u opštem slučaju ima sledeću sintaksu:

```
if (uslov1) {
        naredbe 1
}
elseif (uslov2) {
        naredbe 2
}
else {
        naredbe 3
}
```

Kada program dođe do naredbe *if*, on proverava prvi uslov. Ukoliko je uslov tačan, izvršava se prva grupa naredbi i program prelazi na izvršavanje koda koji sledi iza *if...else* strukture. Ukoliko prvi uslov nije tačan, proverava se uslov unutar *elseif* naredbe (ako postoji). Ako je ovaj uslov tačan, izvršava se druga grupa naredbi. Ako je i ovaj uslov netačan, izvršava se grupa naredbi unutar *else* bloka. Unutar jedne *if...else* strukture može biti više *elseif* blokova. Ukoliko je nekoliko uslova tačno, izvršiće se samo prvi od njih.

```
<?php
$number = 15;
if ($number < 10) {
        echo "Broj je manji od 10.";
}
elseif ($number == 10) {
        echo "Broj je jednak 10.";
}
else {
        echo "Broj je veći od 10.";
}
?>
```

U primeru koji sledi elementi (dan, mesec, godina) određenog datuma se pomoću funkcije *explode()* prebacuju u niz, a zatim se koristi funkcija *checkdate()* za proveru ispravnosti datuma i ispisuje se poruka.

```
<?php
$orig_date = "09/19/2005";
$date = explode("/", "$orig_date");

$month = $date[0];
$day = $date[1];
$year = $date[2];
```

```
$result = checkdate($month, $day, $year);

if ($result == true)
{
	echo "Datum je validan.";
}
else
{
	echo "Datum nije validan.";
}
?>
```

switch naredba

Ova naredba se koristi u situacijama kada uslov koji se testira može da ima više vrednosti, od kojih zavisi dalje izvršavanje programa. Ova naredba ima sledeću sintaksu:

```
switch (izraz) {
	case "vrednost1":
		naredbe1
		break;
	case "value2":
		naredbe2
		break;
	default:
		naredbe3;
}
```

Kod *switch* strukture, prvo se izračunava izraz, a zatim se dobijena vrednost poredi sa vrednostima navedenim u case oznakama. Ukoliko je vrednost izraza jednaka nekoj od vrednosti iz *case* oznake, izvršava se grupa naredbi koja sledi tu oznaku. Ukoliko je vrednost izraza različita od svih navedenih, izvršava se grupa naredbi nakon *default* oznake.

Petlje

U programiranju često je potrebno ponoviti isti blok naredbi nekoliko puta. Ovo ponavljanje se postiže upotrebom petlji. PHP ima nekoliko tipa petlji.

while petlja

Ova petlja ponavlja neki blok naredbi dok god je neki uslov tačan. Sintaksa *while* petlje je

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

Pogledajte sledeći primer:

```
<?php
$number = 5;
while ($number >= 2) {
```

```
        echo $number . "<br/>";
        $number -= 1;
}
?>
```

Promenljiva $*number* na početku koda dobija vrednost 5. Zatim se proverava uslov *($number >= 2)*. Obzirom da je ovaj uslov tačan, program počinje da izvršava blok naredbi unutar *while* petlje. Unutar ovog bloka ispisuje se tekući broj, a zatim se smanjuje za 1. Nakon prvog prolaska kroz blok naredbi, ponovo se proverava uslov. Ponoviće se izvršavanje naredbi unutar petlje sve dok je uslov tačan. Nakon izlaska iz petlje, rezultat će biti sledeći:

5
4
3
2

do...while petlja

Ova petlja je slična prethodnoj, s tom razlikom što se kod *do..while* petlje uslov proverava nakon izvršavanja bloka naredbi, a ne pre. To znali da će se blok naredbi unutar *do..while* petlje izvršiti barem jednom, bez obzira na tačnost uslova.
Kod koji sledi daje isti rezultat kao i prethodni, ali je unutar njega korišćena *do..while* petlja.

```
$number = 5;
do {
        echo $number . "<br/>";
        $number -= 1;
}
while ($number >= 2);
```

for petlja

For petlja se koristi kada se unapred zna tačan broj koliko puta je potrebno izvršiti neki blok naredbi. Ova petlja ima sledeću sintaksu:

```
for (inicijalizacija; uslov; povećanje) {
        blok naredbi;
}
```

Prvi parametar se koristi da kreira i postavi početnu vrednost brojaču. Drugi parametar sadrži uslov za brojač, a treći određuje na koji način se brojač povećava. Sledeći primer koda ispisuje pozdravnu poruku 4 puta:

```
<?php
for ($brojac=1; $brojac < 5; $brojac++) {
        echo "Dobro došli!" . "<br/>";
}
?>
```

Prvo se vrednost promenljive $*brojac* postavlja na 1. Drugi parametar sadrži uslov. Blok naredbi će se izvršavati dok god je ovaj uslov tačan. Treći parametar povećava promenljivu $*brojac* za jedan.

For petlje se često koriste za prolazak kroz nizove. Na primer:

```
<?php
//kreira se novi niz od pet elemenata
$boje = array('crvena', 'zelena', 'plava', 'zuta','bela');

//koristi se for petlja za prolaz kroz niz i ispisivanje njegovih elemenata
for ($i = 0; $i < sizeof($boje); $i++)
{
        echo "Vrednost elementa $i+1 is $boje[$i].";
}
?>
```

Gore kreirani niz ima pet elemenata, pa je *sizeof($boje) = 5*. Pri tome su indeksi elemenata u rasponu od 0 do 4. Zbog toga se u uslovu koristi stroga nejednakost, da program ne bi pokušao da pristupi elementu sa indeksom 5, koji ne postoji.

foreach petlja

Ova petlja je varijacija *for* petlje, namenjena isključivo radu sa nizovima. Ona ima dva oblika. Prvi oblik je:

```
foreach (niz as $vrednost)
        {
        blok naredbi
        }
```

Kod prvog oblika prolazi se kroz niz i prilikom svake iteracije, vrednost tekućeg elementa se smešta u promenljivu $vrednost. U bloku naredbi je zatim moguće koristiti ovu promenljivu, a da pri tom originalna vrednost sadržana u tekućem elementu niza ne bude promenjena.

Pogledajte sledeći primer:

```
<?php
$moj_niz = array('crvena','zelena','plava');

echo "Boje sadrzane u nizu su: ";

foreach($moj_niz as $value)
        {
        $boje .= $value . " ";
        }
echo $boje;

?>
```

Tokom svakog prolaska kroz petlju, vrednost tekućeg elementa se dodeljuje promenljivoj $*value*. Ova vrednost se zatim dopisuje na promenljivu $*boje*, zajedno sa jednim praznim karakterom. Na kraju, rezultat će biti:

Boje sadrzane u nizu su: crvena zelena plava

Drugi oblik *foreach* petlje je:

```
foreach (niz as $indeks => $vrednost)
```

```
{
blok naredbi
}
```

On ima istu funkcionalnost kao i prvi oblik, s tom razlikom što sem vrednosti samog elementa, koristi i vrednost indeksa i smešta ga u promenljivu $*indeks*.