Objektno orijentisano programiranje u PHP-u

Definisanje klase

Klasa u PHP-u se definiše upotrebom naredbe *class*.

```
class imeKlase
{
Naredbe koje definišu osobine
Metodi klase
}
```

Unutar tela klase, sva podešavanja osobina i definicije metoda su uokvirene velikim zagradama.
PHP podržava nasleđivanje klasa, pa je moguće novu klasu kreirati kao naslednika neke već postojeće pomoću ključne reči *extends*:

```
class novaKlasa extends staraKlasa
{
Naredbe koje definišu osobine
Metodi klase
}
```

Objekti klase *novaKlasa* će imati pristup svim osobinama i metodima klase *staraKlasa*, dok obrnuto ne važi.

Deklaracija osobina

U telu klase prvo se navode sve osobine. Na primer:

```
class Auto
{
var $boja;
var $gume;
var $gorivo;
Metodi klase
}
```

PHP ne zahteva da deklarišete promenljive. U prethodnim primerima koda nismo deklarisali promenljive, već smo ih odmah koristili. To je moguće uraditi i sa promenljivima unutar klase. Međutim, mnogo je bolje deklarisati promenljive koje se odnose na osobine klase. Time je kod mnogo razumljiviji i deklarisanje osobina se smatra dobrom programerskom praksom.

Osobinama je moguće postaviti inicijalne, podrazumevane vrednosti. Međutim, tu postoje određena ograničenja. Dozvoljeno je dodeljivanje prostih vrednosti osobinama, ali ne i izraza.
Na primer, sledeće deklaracije su dopustive:

```
var $boja = "crna";
var $gorivo = 10;
var $gume = 4;
```

Sledeće deklaracije nisu dopustive:

```
var $boja = "plava"." crna";
var $gorivo = 10-3;
var $gume = 2*2;
```

Nizove je dozvoljeno koristiti u deklaracijama osobinama, sve dok su vrednosti elemenata proste. Na primer:

```
var $vrata = array("prednja", "zadnja");
```

Postavljanje i promena vrednosti osobina nekog objekta klase se vrši ili pomoću konstruktora, ili pomoću metoda specijalno namenjenih u te svrhe, o čemu će biti reči kasnije.

Unutar klase postoji promenljiva $*this* koja se odnosi na samu klasu. Ova promenljiva ne može biti korišćena izvan klase. Dizajnirana je da se koristi u naredbama unutar klase kako bi omogućila pristup promenljivima same klase. Obično se koristi u formatu *$this->varname.* Na primer:

```
$this->gorivo
$this->gorivo = 20;
if($this-> gorivo > 10)
$product[$this->gorivo] = $cena
```

Obratite pažnju da se znak $ koristi ispred promenljvie *this*, ali ne i ispred promenljive *gorivo*.

Dodavanje metoda

Metodi određuju ponašanje nekog objekta. Definišu se unutar klase kao funkcije. Za klasu *Auto* mogli bismo da definišemo metod za sipanje goriva. Prethodno definisana osobina $*gorivo* predstavlja trenutni sadržaj goriva u automobilu. Metod za sipanje goriva bi trebalo da poveća ovaj sadržaj za određenu vrednost.

```
class Auto
{
        var $gorivo = 0;
        function sipajGorivo($kolicina)
        {
                $this->groivo = $this->gorivo + $kolicina;
                echo "$kolicina litara je dosuto u rezervoar.";
        }
}
```

Sem metoda kojim definišete ponašanje neke klase, PHP obezbežuje i neke specijalne metode za rad sa klasama. Njihova imena počinju sa __ (dve donje crte). Jedan od njih je tzv. konstruktor. On se poziva prilikom kreiranja objekta neke klase. On nije neophodan i ukoliko ne želite da podešavate osobine objekta prilikom kreiranja, ne morate da ga koristite. Dozvoljeno je postojanje samo jednog konstruktora unutar klase. Njegovo ime mora da bude *__construct()*, kako bi PHP znao koji metod da izvršava prilikom kreiranja objekta. Na primer:

```
function __construct()
{
        $this->gorivo = 10; # pun rezervoar
```

```
        $this->otvoriVrata();
    }
```

Ovaj konstruktor definiše novi auto, sa punim rezervoarom goriva i otvorenim vratima.
U prethodnim verzijama PHP-a, konstruktor je imao isto ime kao i klasa. PHP 5 prvo traži konstruktor nazvan *__construct(),* a ako ga ne nađe traži metod koji ima isto ime kao klasa, kako bi i stare klase mogle da se izvršavaju.

Sledeći primer predstavlja klasu koja kreira jednostavnu HTML formu sa tri metoda.

```
<?php
class Form
{
    var $fields=array(); # sadrži imena i oznake polja
    var $processor; # ime programa koji procesira formu
    var $submit = "Submit Form"; # vrednost dugmeta za potvrdu
    var $Nfields = 0; # broj polja dodatih na formu
    /* Konstruktor: Korisnik prosleđuje ime skripta kome
    * se šalju podaci iz forme ($processor) i tekst koji će
    * biti prikazan na dugmetu za potvrdu.
    */
    function __construct($processor,$submit)
    {
        $this->processor = $processor;
        $this->submit = $submit;
    }
    /* Funkcija za prikazivanje forme.
    */
    function displayForm()
    {
        echo "<form action='{$this->processor}' method='post'>";
        echo "<table width='100%'>";
        for($j=1;$j<=sizeof($this->fields);$j++)
        {
            echo "<tr><td align=\"right\">
            {$this->fields[$j-1]['label']}: </td>\n";
            echo "<td>
            <input type='text'
            name='{$this->fields[$j-1][`name']}'>
            </td></tr>\n";
        }
        echo "<tr><td colspan=2 align='center'>
            <input type='submit'
            value='{$this->submit}'></td></tr>\n";
        echo "</table>";
    }
    /* Funkcija koja dodaje polje na formu. Korisnik treba da
    * prosledi ime polja i oznaku koja će se prikazati.
    */
    function addField($name,$label)
    {
        $this->fields[$this->Nfields][`name'] = $name;
        $this->fields[$this->Nfields][`label'] = $label;
        $this->Nfields = $this->Nfields + 1;
    }
```

```
}
?>
```

Osobine i metode klase mogu biti javne ili privatne. Javne osobine i metodi su dostupne izvan same klase. Ukoliko su osobine i metodi definisani kao u gore navedenim primerima, oni će biti javni. Međutim, nije dobra praksa dozvoljavati promenu osobina objekta direktno iz skripta. Osobine nekog objekta, tj. klase bi trebalo da budu sakrivene od ostatka koda, a pristup da im bude omogućen preko odgovarajućih metoda klase. Zbog toga bi osobine trebalo deklarisati kao privatne, pomoću ključne reči *private*. Na primer:

```
class Auto
{
        private $gorivo = 0;
        private function sipajGorivo($kolicina)
        {
                $this->gorivo = $this->gorivo + $kolicina;
                echo "$kolicina litara je dosuto u rezervoar.";
        }
        function kupiGorivo($kolicina)
        {
                $this->sipajGorivo($kolicina);
        }
}
```

Ukoliko je klasa *Auto* definisana kao u navedenom primeru, jedini način da se promeni količina goriva nekog objekta klase jeste preko metoda *kupiGorivo*(). To je jedini javni metod. On ima pristup metodu *sipajGorivo*() koji povećava vrednost promenljive $*gorivo*, jer se ova dva metoda nalaze unutar iste klase. Metodu *sipajGorivo*() nije moguće pristupiti izvan klase.

Korišćenje klase

Kod koji definiše klasu mora da bude dostupan kodu koji tu klasu koristi. Često se klase definišu u posebnim fajlovima koji se zatim uključe u kod koji ih koristi.
Kada postoji definicija klase, na osnovu nje je moguće kreirati objekat te klase naredbom koja ima sledeću sintaksu:

```
$imeObjekta = new imeKlase(vrednost1,vrednost2,...);
```

Na primer:

```
$Nikola = new Osoba("musko");
$NikolinAuto = new Auto("crvena");
$PerinAuto = new Auto("zelena");
$klijent1 = new Klijent("Petrovic","Petar",$IDklijenta);
```

Prilikom kreiranja objekat se smešta u navedenu promenljivu i poziva se konstruktor klase. Nakon toga, kada je objekat kreiran, moguće je pozivati različite metode klase ili mu menjati osobine:

```
$Nikola ->idiNaPosao();
$NikolinAuto ->parkiraj("zabranjeno");
$PerinAuto ->prefarbajAuto("plava");
$ime = $klijent1->uzmiIme();
```

Sledeći primer koda demonstrira korišćenje prethodno definisane klase *Form*.

```
<?php
require_once("form.inc");
echo "<html><head><title>Telefonski imenik</title></head><body>";
$phone_form = new Form("process.php","Zapamti broj");
$phone_form->addField("first_name","Ime");
$phone_form->addField("last_name","Prezime");
$phone_form->addField("phone","Telefon");
echo "<h3>Popunite sledeću formu:</h3>";
$phone_form->displayForm();
echo "</body></html>";
?>
```

U prvoj naredbi u kod se uključuje fajl u kome se nalazi definicija klase *Form*. Zatim se nakon ispisivanja sekcije *<head>* na HTML stranu, kreira novi objekat klase *Form* nazvan $*phone_form*. Dodaju se tri polja na formu, a zatim se forma prikazuje.

Kopiranje i uništavanje objekata

PHP obezbeđuje metod koji možete da koristite za kopiranje objekata. Ovaj metod se naziva *__clone().* Unutar klase možete da napišete svoj metod *__clone()* ukoliko želite da precizirate šta se dešava prilikom kopiranja. Ako ga izostavite, PHP će koristiti podrazumevani metod *__clone()* koji kopira osobine kakve jesu.

Jednom kreiran objekat možete da uništite pomoću naredbe unset:

```
unset($imeObjekta);
```

Nakon izvršenja ove naredbe, objekat više ne postoji. Prilikom uništavanja objekta, PHP izvršava metod *__destruct()*. Ovaj metod možete da redefinišete unutar klase, kako biste izvršili još neke potrebne naredbe. Na primer, možete da prikažete poruku da je objekat uništen, da zapamtite broj uništenih objekata ili da upišete neki podataka u bazu.