Základy OOP v PHP

Co se v modulu dozvíte?

- Co je to OOP a k čemu slouží objekty?
- Třídy, atributy a metody
- Ukázky objektového programování

Co je to OOP a k čemu slouží objekty?

OOP je zkratkou pro „objektově orientované programování“. Jedná se o přístup, kterým lze vytvářet programy nebo aplikace. V současné době jde o velmi rozšířený způsob návrhu a vývoje SW, neboť umožňuje mimo jiné snadnější rozšíření kódu a větší modularitu. Tím pádem je i samotný vývoj finančně méně náročný.

V PHP lze programovat procedurálně (způsob programování, který jsme používali v příkladech do této doby) nebo objektově. Jak již samotný název napovídá, jedná se o programování, ve kterém používáme tzv. objekty. Objekty jsou běžnou součástí našeho života, mají svoje vlastnosti a chování. Stejným způsobem, jak objekty vnímáme v normálním žití, používáme je i v programování. Program je složen z různých objektů, které mají určité vlastnosti (atributy) a předepsané způsoby chování.

Jako příklad objektu lze uvést například auto. Vlastnostmi může být např. barva, typ motoru, výkon nebo stupeň výbavy. Jde o atributy objektu, které do objektu můžeme vložit nebo je naopak číst. Zjednodušeně řečeno lze říci, že jde o určitý druh „proměnné“, která přísluší objektu. Objekt má také své chování – v našem případě umožňuje jízdu vpřed, vzad, zatáčet nebo např. svítit. Jde o funkce (nebo také metody) objektu, které určují, co daný objekt umí provádět.

Objektově orientované programování je charakteristické tzv. zapouzdřením. Nemusíme znát to, jak daný objekt funguje uvnitř, ale pouze tím, že mu dodáme správná data, ho můžeme používat a snadno tak implementovat do našeho programu. V našem případě – nemusíme znát to, jak uvnitř funguje auto, ale vhodnými vstupy a reakcemi na výstupy objektu můžeme objekt používat a ovlivňovat jeho chování. V praxi tak můžeme využít již předpřipravené knihovny a využívat je v naší aplikaci, aniž bychom museli zkoumat, jak daná knihovna uvnitř pracuje.

Třídy, atributy a metody

Abychom mohli vytvořit nějaký objekt, musíme nejprve vytvořit tzv. třídu. Zjednodušeně se jedná o formu, podle které bude objekt vytvořen. Tím vznikne konkrétní výskyt objektu (tzv. instance). Třídu definujeme v samostatném PHP souboru klíčovým slovem „class“.

Pro vytváření atributů a metod třídy je nutné si ještě říci něco o tzv. řízení přístupu. To souvisí s vlastností objektově orientovaného programování – tzv. zapouzdřením. Zapouzdření umožňuje řídit přístup k atributům a metodám z vnějšího prostředí (z prostředí mimo objekt). Pomocí modifikátorů přístupu lze určit, jaké atributy budou ve třídě veřejně přístupné pro ostatní objekty a jaké budou sloužit pouze danému objektu a ostatní k nim nebudou mít přístup.

Rozlišujeme tři druhy modifikátorů přístupu:

- **public** – atribut nebo metoda s modifikátorem public je přístupná a viditelná odkudkoliv

- **protected** – prvky s modifikátorem protected jsou přístupné pouze ze třídy, ve které je máme definovány a také ze tříd, které jsou od této třídy odvozeny
- **private** – prvky s tímto modifikátorem jsou přístupné pouze uvnitř třídy, ve které jsou definovány

Klíčová slova modifikátorů přístupu jsou uváděna při inicializaci před názvem atributu nebo při definicí metody třídy.

Komunikaci s objektem je v PHP zajištěna pomocí operátoru „->“ (neplést si s operátorem „=>“ pro přiřazení prvku v poli). Tímto operátorem lze z objektu získat hodnotu atributů nebo např. volat metody objektů (spouštět funkce objektu).

Atributy, jak už bylo řečeno, jsou vlastnostmi objektu a reprezentují „proměnné“ do kterých lze ukládat nebo z nich číst data. Že jde o parametr poznáme tak, že není za názvem daného parametru závoka (v tom případě by šlo o metodu). Přečtení atributu objektu může vypadat následovně (viz. příklad dále):

```
1 $sirka_grafu = $graf->width;
```

Naopak metody jsou funkcemi třídy (resp. objektu) a určují chování objektu. Lze jimi např. měnit atributy, které jsou definovány ve třídě a lze je přirovnat ke klasickým funkcím z procedurálního programování. V kódu je poznáme tak, že mají za svým názvem závorky (a případně uvnitř parametry, které mají být metodě předány). Volání metody může vypadat následovně:

```
1 $graf->setSize(400,200);
```

Příklad využití předpřipravené třídy ve vlastním skriptu

Google Chart

Máme definovanou třídu GoogleChart (předpřipravená knihovna pro vytvoření grafu z dodaných dat pomocí služby od Googlu – class.googlechart.php) a budeme ji chtít použít v našem programu. PHP soubor třídy je nejprve nutno do našeho skriptu vložit pomocí příkazu include() a pak v kódu aplikace vytvořit nový objekt (instanci třídy) a vložit potřebná data následujícím kódem:

```
1  <?php
2
3  // vložení třídy do skriptu
4  include "class.googlechart.php";
5
6  // vložení dat grafu do pole
7  $datagrafu = array(
8      array('200' => 'Leden'),
9      array('50' => 'Únor'),
10     array('120' => 'Březen'),
11     array('330' => 'Duben'),
12     array('150' => 'Květen')
13 );
14
15
16 // vytvoření objektu $graf (instance třídy GoogleChart)
17 $graf = new GoogleChart();
18
19 /*
20   zavolání metody třídy GoogleChart pro nastavení typu grafu
21   "lc" - spojnicový, "ls" - spojnicový bez os, "bvs" - svislý sloupcový, "p" - 2D kolácový, "p3" -
22 */
23 $graf->setType('p');
24
25 // zavolání metody pro nastavení atributu width a height ve tříde GoogleChart
26 $graf->setSize(400, 200);
27
28 // vložení dat do objektu
29 $graf->insertData($datagrafu);
```

```php

// zavolání metody pro vykreslení grafu
echo $graf->render();

?>
```

Knihovna pro práci s databází

Dalším příkladem může být jedna z předpřipravených tříd, která ulehčuje práci s databází. Její specifikaci a dokumentaci lze najít na http://code.google.com/p/edb-php-class/

```php
<?php

// vložení třídy do skriptu
include("class.edb.php");

// vytvoření nového objektu $db
$db = new edb('localhost', 'root', '', 'gml');



// ukázka SELECTu jednoho řádku z tabulky a výpis hodnot
$result = $db->line("SELECT * FROM uzivatele WHERE id = '2' LIMIT 1");
echo $result['jmeno'];
echo $result['prijmeni'];

// ----------------------------------------------------------------------



// ukázka SELECTu více řádků tabulky s výpisem
$result = $db->q("SELECT * FROM uzivatele LIMIT 5");
foreach($result as $a){
        echo $a['jmeno'].' '.$a['prijmeni'].' '.$a['login'].' '.$a['mail'].'</br>';
}
// ----------------------------------------------------------------------



// ukázka SELECTu jednoho sloupce a řádku tabulky s výpisem
$name = $db->one("SELECT jmeno FROM uzivatele WHERE id = '3' LIMIT 1");
echo $name;
// ----------------------------------------------------------------------



// ukázka vložení řádku do tabulky
$db->insert('uzivatele', array('mail'=>'neco@mail.cz','jmeno'=>'Honza','prijmeni'=>'Novák'));

// ukázka úpravy řádku tabulky
$db->update('uzivatele', array('heslo'=>'heslo2','jmeno'=>'Eduards'), array('id'=>'3','mail'=>'mail

?>
```

Odeslání e-mailu pomocí PHPmailer

viz také modul Návrhové vzory v PHP

```php
<?php
require 'class.phpmailer.php';

// vytvoření nové instance PHPMaileru
$mail = new PHPMailer();

$mail->IsSMTP(true);      // odeslání přes SMTP protokol
$mail->Mailer   = "smtp";
$mail->Host = "smtp.seznam.cz";  // specifikace SMTP serveru
$mail->SMTPAuth = true;     // vyžadování SMTP autentifikace
$mail->Username = "neco@seznam.cz";  // SMTP uživatelské jméno
$mail->Password = "123"; // SMTP heslo


```

```php
$mail->CharSet = "utf-8";   // kódování e-mailu
$mail->From = $_POST['mail'];    // odchozí e-mailová adresa
$mail->FromName = $_POST['jmeno'];    // jméno odesílatele
$mail->AddAddress("neco@seznam.cz");    // příjemce
$mail->AddReplyTo($_POST['mail']);       // e-mail pro odpověď

$mail->IsHTML(false);    // e-mail bude odeslán jako čistý text

$mail->Subject = "KONTAKTNÍ FORMULÁŘ - ".$_POST['jmeno'];    // předmět
// text e-mailu
$mail->Body    = "E-mail z webového formuláře\n\nKontaktní osoba: ".$_POST['jmeno']."\nE-mail: ".$_

// odeslání e-mailu a kontrola chyby odeslání
if(!$mail->Send()) {
  echo "E-mail se nepodařilo odeslat: " . $mail->ErrorInfo;
} else {
  header("Location: index.php?kontakt=1&zprava=odeslano");
}
?>
```

Kompletní příklady včetně vkládaných souborů ke stažení najdete zde

Tyto materiály vznikly v rámci projektu CZ.2.17/3.1.00/34129
Rozvoj oboru Multimédia v ekonomické praxi pro lepší uplatnění absolventů v praxi

Evropský sociální fond - Praha & EU: Investujeme do vaší budoucnosti