PHP orienté objet

Jean-Claude Charr

Maître de conférences

IUT NFC

Université de Franche Comté

Introduction

- Quelques notions de POO ont été introduites avec PHP3
- Le noyau de PHP5 a été complètement réécrit et de nouveaux modèles objets ont été introduits
- Avantages de la programmation orienté objet :
  - Réutilisable
  - Extensible
  - Facile à maintenir
  - Efficace
  - Facilite la ré-ingénierie logicielle (refactoring)

Terminologie

- Classe et objet
- Propriété
- Méthode
- Constructeur/destructeur
- Héritage
- Super-classe/sous-classe
- Instance
- Interface

Exemple de classe (1/2)

```
<? //class.emailer.php
class Emailer{
   protected $sender, $recipients, $subject, $body;
   function __construct($sender, $subject, $body) {
      $this->sender = $sender;
      $this->subject = $subject;
      $this->body = $body;
      $this->recipients = array();
   }
   public function addRecipients($recipient) {
      array_push($this->recipients, $recipient);
   }
```

Exemple de classe (2/2)

```
  public function sendEmail() {
      foreach ($this->recipients as $recipient) {
         $result = mail($recipient, $this->subject, $this->body, "From:
         {$this->sender}\r\n");
         if ($result) echo "Mail successfully sent to {$recipient}<br/>";
      }
   }
   function __destruct(){
      echo " Object Destroyed."; }
} //end of class
?>
```

Utilisation d'un objet

```
<?php
    include_once("class.emailer.php");
    $emailerObject = new Emailer("hasin@pageflakes.com","Just a
    Test","Hi Hasin, How are you?");
    $emailerObject->addRecipients("hasin@somewherein.net");
    $emailerObject->sendEmail();
?>
```

Héritage

```
<? //class.extendedemailer.php
include_once("class.emailer.php");
class ExtendedEmailer extends Emailer {
   function __construct(){}
   public function setSender($sender) {
      $this->sender = $sender; }
   public function setBody($body) {
      $this->body = $body; }
   public function setSubject($subject) {
      $this->subject = $subject; }
}
?>
```

Utilisation d'un objet de la sous classe

```
<?
include_once("class.emailer.php");
include_once("class.extendedemailer.php");
$xemailer = new ExtendedEmailer();
$xemailer->setSender("hasin@pageflakes.com");
$xemailer->addRecipients("hasin@somewherein.net");
$xemailer->setSubject("Just a Test");
$xemailer->setBody("Hi Hasin, How are you?");
$xemailer->sendEmail();
?>
```

Redéfinition d'une méthode dans la sous-classe

- Redéfinir dans la sous-classe une méthode qui a le même nom et les mêmes paramètres qu'une méthode de la super-classe (overload)
- Le mot clé final interdit la redéfinition ou l'héritage.
  - public final someMethod(){...}
  - final class SomeClass{...}

Accessibilité des propriétés et des méthodes

- **Public :** la propriété ou la méthode est accessible depuis l'intérieur mais aussi depuis l'extérieur de la classe (par défaut).
- **Private :** la propriété ou la méthode est accessible à l'intérieur de la classe seulement.
- **Protected :** la propriété ou la méthode est accessible depuis l'intérieur de la classe et des sous-classes.

Classe avec des constantes (1/2)

```
<?
class WordCounter{
   const ASC=1; //Do not use $ sign before Constants
   const DESC=2;
   private $words;
   function __construct($filename) {
      $file_content = file_get_contents($filename);
      $this->words = (array_count_values(str_word_count (strtolower
      ($file_content),1)));
}
```

Classe avec des constantes (2/2)

```
    public function count($order) {
        if ($order==self::ASC)
            asort($this->words);
        else arsort($this->words);
        foreach ($this->words as $key=>$val)
            echo $key ." = ". $val."<br/>";
    }
}
$wc = new WordCounter("words.txt");
$wc->count(WordCounter::DESC);
?>
```

Méthodes et propriétés statiques (1/2)

```
class StaticTester{
   private static $id=0;
   function __construct() {
      self::$id +=1;
   }
   public static function checkIdFromStaticMehod() {
      echo "Current Id From Static Method is ".self::$id."\n";
   }
   public function checkIdFromNonStaticMethod() {
      echo "Current Id From Non Static Method is ".self::$id."\n";
   }
}
```

Méthodes et propriétés statiques (2/2)

```
$st1 = new StaticTester();
StaticTester::checkIdFromStaticMehod();
$st2 = new StaticTester();
$st1->checkIdFromNonStaticMethod();
$st1->checkIdFromStaticMehod();
$st2->checkIdFromNonStaticMethod();
$st3 = new StaticTester();
StaticTester::checkIdFromStaticMehod();
```

Interface

```
interface DBDriver{
   public function connect();
   public function execute($sql);
}
class MySQLDriver implements DBDriver {
   public function connect() {
      //connect to database
   }
   public function execute($sql) {
      //execute the query and output result
   }
}
```

Classe Abstraite

```
abstract class ReportGenerator{
   Abstract public function generateReport($resultArray);
}
class MySQLDriver extends ReportGenerator implements DBDriver {
   public function connect() { //connect to database }
   public function execute($sql) {
      //execute the query and output result
   }
   public function generateReport($resultArray){
      //generate a report from the query result
   }
}
```

Trait

```
trait TraitName {
public function myFunction() {
    //does something
  }
}
class MyClass {
  use TraitName;
}
(new MyClass())->myFunction();
```

Exception (1/2)

```
<?
Class ConnectionException extends Exception{
   public function __construct() {
     $message = "Sorry, couldn't connect to server:";
     parent::__construct($message, 0000);
   }
}
?>
```

Exception (2/2)

```
class DAL{
    public $connection;
    public function connect($ConnectionString) {
        $this->connection = pg_connect($ConnectionString);
        if ($this->connection==false)
            throw new ConnectionException($this->connection);
    }
}
$db = new DAL();
try{ $db->connect("dbname=golpo user=postgres2"); }
catch(Exception $connectionexception){
echo $connectionexception->getMessage(); }
```

Manipuler une bdd MySQL en POO

```
<?php
$mysqli = new mysqli("localhost", "user" "pwd", "db");
if ($mysqli->connect_errno) {
    echo("Failed to connect, the error message is : ".
    $mysqli->connect_error);
    exit();
}
$result=$mysqli->query("select * from users") ;
while ($data = $result->fetch_object())
    echo $data->name." : '".$data->pass."' \n";
?>
```

Préparer une requête

- Une requête préparée est pré-compilée par le serveur MySQL et peut être plus performante qu'une requête normale

```
$stmt = $mysqli->prepare("select name, pass from users");
$stmt->execute();
$stmt->bind_result($name, $pass);
while ($stmt->fetch())
    echo $name."<br/>";
```

PHP Data Objects (PDO)

```
$dsn = 'mysql:dbname=test;host=localhost;';
try {
    $pdo = new PDO($dsn, 'user', 'password');
} catch (PDOException $e) {
    echo 'Connection failed: ' . $e->getMessage();
}
$result = $pdo->query("select * from users");
foreach ($result as $row)
    echo $row['name'];
```