

Développement d'applications Web

PHP et la programmation Orientée Objet

Php et la programmation Orientée Objet

La notion de classe [1]

Php intègre quelques caractéristiques empruntées aux langages orientés objet, c'est-à-dire la possibilité d'utiliser des objets, entités regroupant des données et des fonctions au sein d'une structure et rendant la programmation plus simple qu'en programmation habituelle (appelée *programmation procédurale* par opposition à la programmation orientée objet).

On appelle classe la structure d'un objet, c'est-à-dire la déclaration de l'ensemble des entités qui composeront un objet. Un objet est donc "issu" d'une classe, c'est le produit qui sort d'un moule. En réalité on dit qu'un objet est une **instanciation** d'une classe, c'est la raison pour laquelle on pourra parler indifféremment d'*objet* ou d'*instance* (éventuellement d'*occurrence*).

Une classe est composée de deux parties :

- **Les attributs** (parfois appelés *données membres*): il s'agit des données représentant l'état de l'objet
- **Les méthodes** (parfois appelées *fonctions membres*): il s'agit des opérations applicables aux objets

Déclaration d'une classe

Pour pouvoir manipuler des objets, il est essentiel de définir des classes, c'est-à-dire définir la structure d'un objet. Avec Php, cette définition se fait de la manière suivante :

```
class Nom_de_la_classe {  
  
    // Déclarations des données membres  
  
    var $Donnee_Membre_1;  
  
  
    var $Donnee_Membre_2;  
  
  
    var $...  
  
    // Déclarations des méthodes
```

```
function Nom_de_la_fonction_membre1(parametres) {  
  
    liste d'instructions;  
  
}  
}
```

Nom_de_la_classe représente bien évidemment le type d'objet désigné par la classe ou du moins le nom que vous leur attribuez.

Contrairement aux langages orientés objet comme le C++, Php n'inclut pas dans sa version 3 de *niveaux de visibilité* des éléments de la classe, il n'y a donc pas de concept d'encapsulation, un des concepts majeurs de la programmation orientée objet.

Remarque :

Contrairement à la déclaration de classes en C++, la déclaration de la classe ne se finit pas par un point-virgule!

Instanciation de la classe

Après avoir déclaré une classe, il faut instancier des objets pour pouvoir l'exploiter. Cette opération se fait à l'aide du mot clé *new* permettant de faire des objets découlant d'une classe. La syntaxe du mot clé new est la suivante :

```
$Nom_de_l_objet = new Nom_de_la_classe;
```

A partir du moment où l'objet est instancié, il possède des propriétés qui lui sont propres, cela signifie que si vous instanciez un nouvel objet, la modification des propriétés de l'un n'influera aucunement sur celles de l'autre.

Il existe une méthode spéciale (portant le même nom que la classe) s'exécutant automatiquement lors de l'instanciation de l'objet. Cette méthode, appelée **constructeur** est très utile pour initialiser les données membres lors de l'instanciation.

Accéder aux propriétés d'un objet

L'accès aux propriétés d'un objet se fait grâce au nom de l'objet, suivi d'une flèche (->) représentée par un moins (-) et un signe supérieur (>), puis du nom de la donnée membre (sans le signe \$). Par exemple :

```
$Nom_de_l_objet->Nom_de_la_donnee_membre = Valeur;
```

Accéder aux méthodes d'un objet

L'accès aux méthodes d'un objet se fait comme pour l'accès aux propriétés, c'est-à-dire par le nom de l'objet, suivi d'une flèche et du nom de la méthode. La méthode est suivie de parenthèses, contenant les paramètres, si il y'en a. L'accès à une méthode se fait donc de la façon suivante :

```
$Nom_de_l_objet->Nom_de_la_fonction_membre(parametre1,parametre2,...);
```

La variable courante *\$this*

Le mot clé *\$this* permet de désigner l'objet dans lequel on se trouve, c'est-à-dire que lorsque l'on désire faire référence dans une fonction membre à l'objet dans lequel elle se trouve, on utilise *this*.

Grâce à cette variable spéciale, il est possible dans une fonction membre de faire référence aux propriétés situées dans le même objet que la fonction membre.

Ainsi, lorsque l'on désire accéder à une propriété d'un objet à partir d'une méthode du même objet, il suffit de faire précéder le nom de la donnée membre par *\$this->*. Par exemple :

```
class Toto{  
  
    var $age;
```

```

var $sexe;

var $adresse;

function DefineTotoAge($Age){

$this->age = $Age;

}

}

$toto_test = new Toto;

$toto_test->DefineTotoAge(10);

echo "L'age de TOTO : " . $toto_test->age . "<br/>";

```

Exemple :

```

<?php
class SimpleClass
{
    // déclaration d'une propriété
    public $var = 'une valeur par défaut';

    // déclaration des méthodes
    public function displayVar() {
        echo $this->var;
    }
}




$a=new SimpleClass; // new SimpleClass()
$a->displayVar();

?>

```

Références Bibliographiques

[1] <https://web.maths.unsw.edu.au/~lafaye/CCM/php/phpclass.htm>

 <https://www.univ-saida.dz/>  e-learning@univ-saida.dz  048931000,1304