

# TP 1 – Introduction à la POO en PHP

## Classes, objets, instance, héritage et encapsulation

Imaginons qu'on possède un site sur lequel les visiteurs peuvent s'enregistrer pour avoir accès à un espace personnel par exemple. Quand un visiteur s'enregistre pour la première fois, il devient un utilisateur du site.

Ici, on va essayer de comprendre comment faire pour créer le code qui permet cela. Pour information, ce genre de fonctionnalité plus simple à réaliser en programmation orienté objet.

Qu'essaie-t-on de réaliser ici ? On veut « créer » un nouvel utilisateur à chaque fois qu'un visiteur s'enregistre à partir des informations qu'il nous a fournies. Un utilisateur va être défini par des attributs comme son nom d'utilisateur ou son mot de passe. En programmation, ces attributs vont être traduits par des variables.

Ensuite, un utilisateur va pouvoir réaliser certaines actions spécifiques comme se connecter, se déconnecter, modifier son profil, etc. En programmation, ces actions sont représentées par des fonctions.

A chaque fois qu'un visiteur s'inscrit et devient utilisateur, on va donc devoir créer des variables « nom d'utilisateur », « mot de passe », etc. et définir leurs valeurs et donner les permissions à l'utilisateur d'utiliser les fonctions connexion, déconnexion, etc.

Pour cela, on va créer un formulaire d'inscription sur notre site qui va demander un nom d'utilisateur et un mot de passe, etc. On va également définir les actions (fonctions) propres à nos utilisateurs : connexion, déconnexion, possibilité de commenter, etc.

Sur notre site, on s'attend à avoir régulièrement de nouveaux visiteurs qui s'inscrivent et donc de nouveaux utilisateurs. Il est donc hors de question de définir toutes ces choses manuellement à chaque fois.

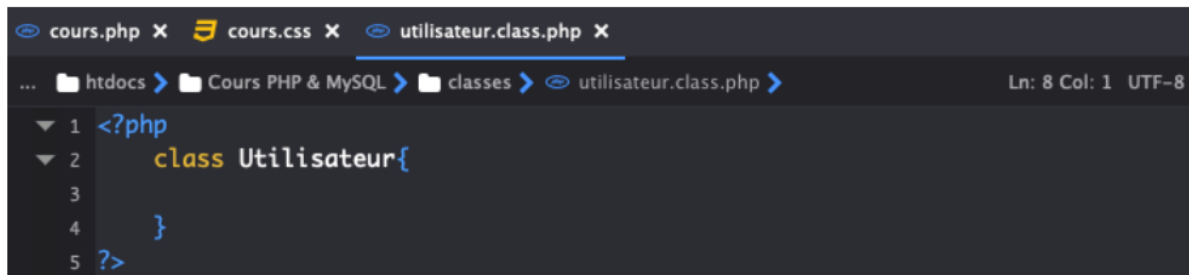
A la place, on va plutôt créer un bloc de code qui va initialiser nos variables nom d'utilisateur et mot de passe par exemple et qui va définir les différentes actions que va pouvoir faire un utilisateur.

Ce bloc de code est le plan de base qui va nous servir à créer un nouvel utilisateur. On va également dire que c'est une classe. Dès qu'un visiteur s'inscrit, on va pouvoir créer un nouvel objet « **utilisateur** » à partir de cette classe et qui va disposer des variables et fonctions définies dans la classe.

On n'aura ensuite qu'à inclure les fichiers de classes nécessaires à l'exécution de notre script principal dans celui-ci grâce à une instruction **require** par exemple.

On va donc créer un nouveau fichier en plus de notre fichier principal **cours.php** qu'on va appeler **utilisateur.class.php**. Notez qu'on appellera généralement nos fichiers de classe «maClasse.class.php» afin de bien les différencier des autres et par convention une nouvelle fois.

Dans ce fichier de classe, nous allons donc pour le moment tout simplement créer une classe **Utilisateur** avec le mot clef **class**.



```
1 <?php
2 class Utilisateur{
3
4 }
5 ?>
```

Nous allons également directement en profiter pour inclure notre classe dans notre fichier principal cours.php avec une instruction require. Ici, mon fichier de classe est dans un sous-dossier « classes » par rapport à mon fichier principal.

Rajouter des propriétés à l'utilisateur (nom d'utilisateur, mot de passe, etc).

Rajouter les accesseurs (getters et setters) et le constructeur.

Créer une nouvelle classe Admin qui va étendre notre classe Utilisateur.

Définir de nouvelles propriétés et méthodes spécifiques à notre classe Admin. On pourrait par exemple permettre aux objets de la classe Admin de bannir un utilisateur ou d'obtenir la liste des utilisateurs bannis.

*Attention cependant : afin d'être utilisées, les classes doivent déjà être connues et la classe mère doit être définie avant l'écriture d'un héritage. Il faudra donc bien penser à inclure les classes mère et fille dans le fichier de script principal en commençant par la mère.*

*Si on souhaite que des classes étendues puissent manipuler les propriétés d'une classe mère, alors il faudra définir le niveau de visibilité de ces propriétés comme **protected** dans la classe mère.*

```
<?php
class Utilisateur{
    protected $user_name;
    protected $user_pass;

    public function __construct($n, $p){
        $this->user_name = $n;
        $this->user_pass = $p;
    }

    public function __destruct(){
        //Du code à exécuter
    }

    public function getNom(){
        return $this->user_name;
    }
}
?>
```

```
<?php
class Admin extends Utilisateur{
    protected $ban;

    public function setBan($b){
        $this->ban[] .= $b;
    }
    public function getBan(){
        echo 'Utilisateurs bannis par ' . $this->user_name . ' : ';
        foreach($this->ban as $valeur){
            echo $valeur . ', ';
        }
    }
}
?>
```

```
<!DOCTYPE html>
<html>
  <head>
    <title>Cours PHP & MySQL</title>
    <meta charset="utf-8">
    <meta name="viewport"
      content="width=device-width, initial-scale=1, user-scalable=no">
    <link rel="stylesheet" href="cours.css">
  </head>

  <body>
    <h1>Titre principal</h1>
    <?php
      require 'classes/utilisateur.class.php';
      require 'classes/admin.class.php';

      $pierre = new Admin('Pierre', 'abcdef');
      $mathilde = new Utilisateur('Math', 123456);

      echo $pierre->getNom(). '<br>';
      echo $mathilde->getNom(). '<br>';

      $pierre->setBan('Paul');
      $pierre->setBan('Jean');
      echo $pierre->getBan();
    ?>
    <p>Un paragraphe</p>
  </body>
</html>
```