



```
$php->simplonBoulogne();
```

Introduction à la POO





Sommaire

- POO
- Création d'une classe et définir un objet
- Notions de propriétés et méthodes
- Héritage
- Challenge



POO

La programmation orienté objet est une tout autre manière de coder, avant nous étions dans la programmation dis procédurale... Nous exécutions des instructions les unes à la suite des autres.

Maintenant on travaille sur un environnement d'objets, on essaye d'imaginer notre site comme étant un ensemble d'objets qui interagisse entre eux.



Classes et objets

Une classe est un « template » définissant ce qu'il y a à l'intérieur d'un objet.

Nous organisons dans notre classe nos constantes, fonctions etc...

Une fois que notre classe est définie nous allons créer notre premier objet !



Vocabulaire

Attention, dans une classe, les fonctions sont appelées « **Méthodes** »

et les variables sont appelées « **Propriétés** »

Classe et objets



Une classe

```
class Product { //Commence par une Majuscule
    public $title; //Propriétés

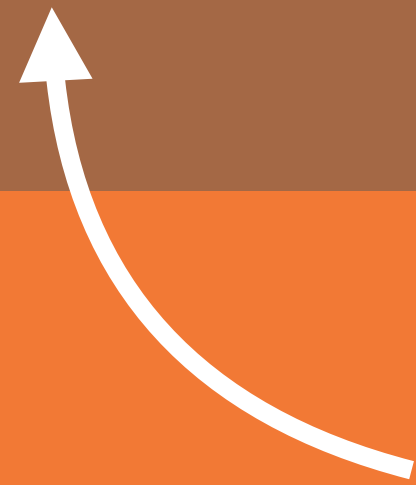
    function getTitle() { //Méthodes
    }
}

$p = new Product(); //Création d'un objet
```



Créer un objet

```
$p = new Product();
```

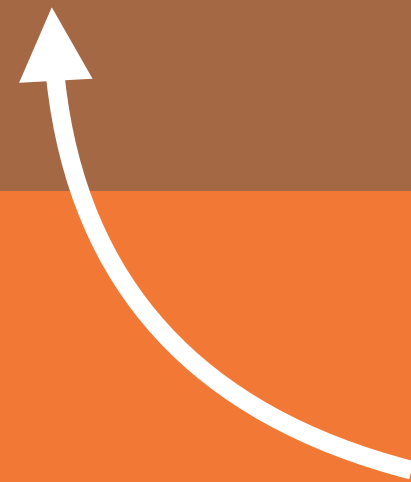


On parle d'une instanciation d'une classe pour créer un objet. (C'est beau...)



Appel d'une propriété

```
$p->titre;
```



On pointe vers la propriété (sans le '\$')

Propriétés



Affectation

```
class Product {  
    public $title; //Propriétés  
  
    function getTitle(){ //Méthodes  
    }  
}  
  
$p = new Product(); //objet de la classe Product  
$p->title('Harry Potter');  
echo $p->title;
```

Non recommandée...



Visibilité des propriétés

- **public** : les propriétés peuvent être accessible depuis n'importe où dans votre code
- **protected** : les propriétés peuvent être accessible seulement à l'intérieur de la classe ET ses classes parentes et hérités
- **private** : les propriétés peuvent être accessible seulement à l'intérieur de la classe.



Visibilité des propriétés

Attention, si vous n'affectez aucune visibilité, par défaut ce sera
public

Pensez à le mettre malgré tout pour une meilleure lisibilité de votre
code notamment pour votre équipe

Propriétés



Exemple

```
class Personne {  
    public $FirstName = "Bill";  
    public $LastName = "Murphy";  
    private $Password = "Poppy";  
    public $Age = 29;  
    public $FavouriteColour = "Purple";  
}  
  
$bill = new Personne();
```



Appel d'une méthode

On n'oublie pas les parenthèses, sauf si c'est pour appeler une propriété !

```
$p->getTitle();
```

On pointe vers une méthode

Méthodes



Création d'une
méthode

```
class Product {  
    public $title = 'Harry Potter';  
    function getTitle(){  
        return "Le nom du produit : " . $this->title;  
    }  
}  
  
$p = new Product();  
$p->getTitle(); //Accès à la méthode
```



"\$this" variable

Lorsque vous êtes à l'intérieur d'une méthode, PHP va automatiquement affecter l'objet à la variable \$this pour pouvoir travailler avec les propriétés de la classe correspondante.

```
$this->title = $title; //Affecte le paramètre  
$title à la propriété $title de la classe
```



Constructeur

Le constructeur est la fonction qui est appelée automatiquement par la classe lorsque vous créez une nouvelle instance d'une classe à l'aide de l'opérateur `new`.

```
$p = new Product("Le seigneur des anneaux");
```


Constructeur



```
class Product {  
    public $title = 'Harry Potter';  
    function __construct($title){  
        $this->title = $title;  
    }  
    function getTitle(){ ... } //retourne le titre  
}  
  
$p = new Product("Le seigneur des anneaux");  
$p->getTitle(); //Retourne quel titre ?
```



Héritage

L'héritage est une manière d'étendre une classe avec une nouvelle et permet d'éviter la saisie de code répétitif.



Voiture



Voiture électrique



Héritage

D.R.Y

Don't Repeat Yourself

Héritage



```
class Voiture {
    public $marque;
    public $puissance;
    function __construct($marque, $puissance){
        $this->marque = $marque;
        $this->puissance = $puissance;
    }
    function getMarque(){ ... } //retourne la marque
}
$v = new Voiture("Renault", "6CV");
$v->getMarque();
```

Héritage



```
class Electrique extends Voiture {
    public $volt;
    function __construct($marque, $puissance, $volt){
        parent::__construct($marque, $puissance);
        $this->volt = $volt;
    }
    function getVolt(){ ... }
}

$ve = new Electrique("Renault", "6CV", "120V");
$ve->getMarque();
```

Challenge



Créer une base de donnée, puis une classe Connexion avec :

- un constructeur pour une connexion PDO à la base
- une méthode `countTable` qui prend une requête SQL en paramètre et retourne le résultat

Ensuite dans un fichier `.php`, créez un nouvel objet en instanciant `Connexion` et appelez la méthode en envoyant une requête sql avec `COUNT ...`



Some ressources

- <http://www.hackingwithphp.com/>
- <https://secure.php.net/manual/fr/index.php>
- <https://openclassrooms.com/courses/programmez-en-orientee-objet-en-php/introduction-a-la-poo>