Université Sidi Mohamed Ben Abdllah
Faculté des sciences Dhar El Mahraz
Licence professionnelle SIGL

Web dynamique:
Programmation orienté objet en PHP

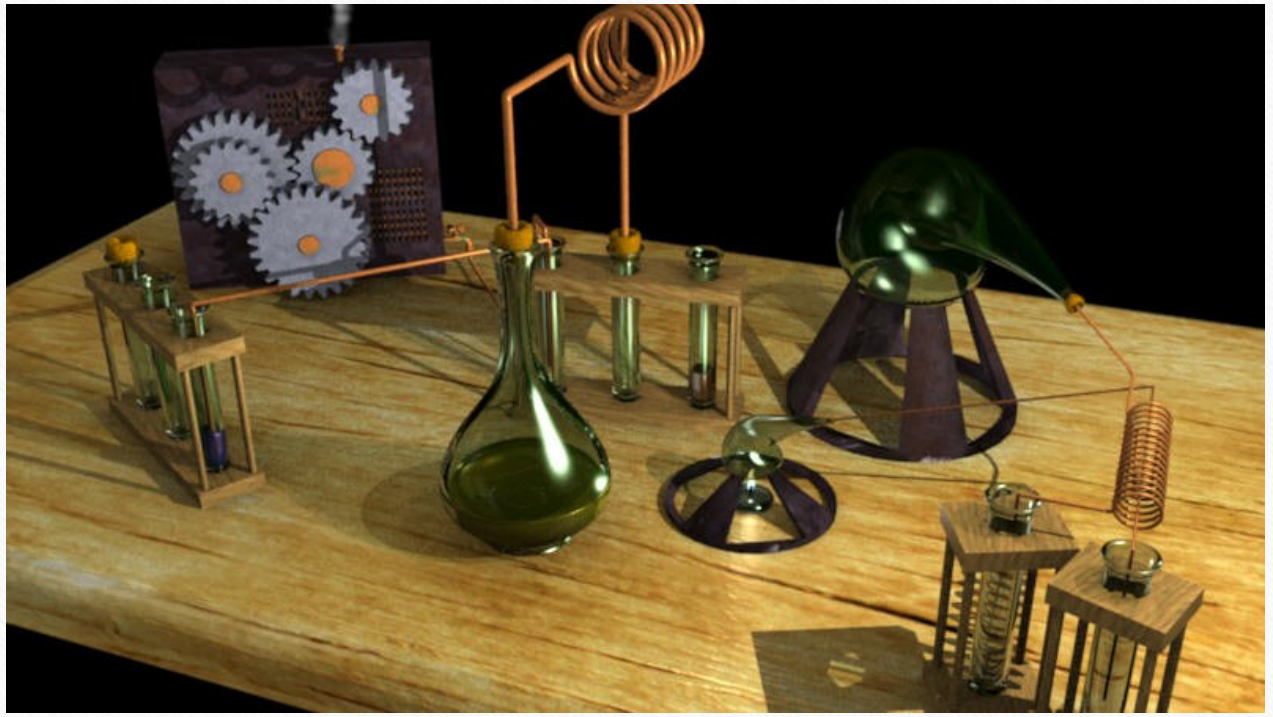
Réalisé par: Pr. Mahraz Med Adnane

Année universitaire:2017/2018

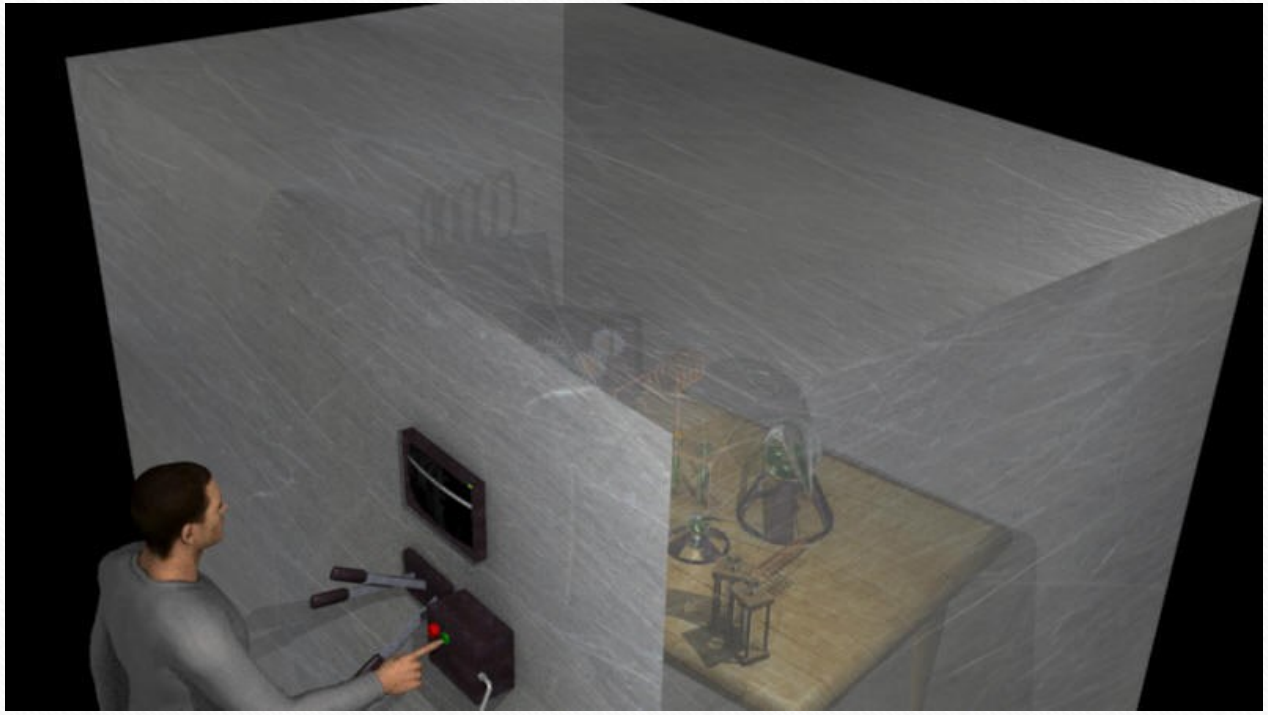
POO en PHP

- La programmation orientée objet est une technique de programmation célèbre qui existe depuis des années maintenant. PHP ne l'a pas inventée : d'autres langages comme le C++, Java et Python l'utilisaient bien avant lui.
- La programmation orientée objet (POO) ne va pas améliorer subitement la qualité de votre site comme par magie. En revanche, elle va vous aider à mieux organiser votre code, à le préparer à de futures évolutions et à rendre certaines portions réutilisables pour gagner en temps et en clarté. C'est pour cela que les développeurs professionnels l'utilisent dans la plupart de leurs projets.

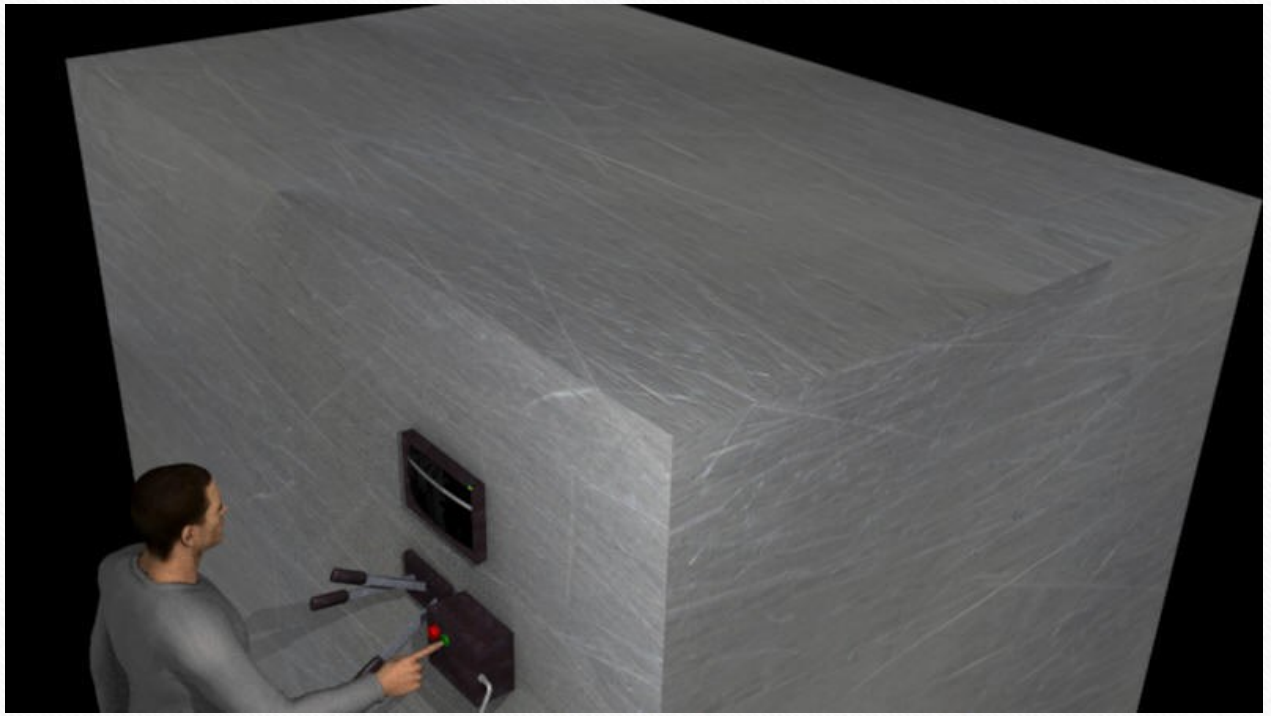
POO en PHP



POO en PHP



POO en PHP



POO en PHP

- En PHP 5, il y a un tout nouveau modele objet. La gestion des objets en PHP a été complètement réécrite, permettant de meilleurs performances ainsi que plus de fonctionnalités.

- Chaque définition de classe commence par le mot-clé **class** , suivi par le nom de la classe, qui peut être quelconque à condition que ce ne soit pas un mot réservé en PHP. Suivent une paire d'accolade contenant la définition des membres et des méthodes. Une pseudo-variable **$this** est disponible lorsqu'une méthode est appelée depuis un contexte objet.

POO en PHP

- **Définition simple d'une classe :**

```php
<?php
Class Ville{
      // déclaration des attributs
     private $name="Fès";
     private $nbHabitant="1.5M";
     // déclaration de la méthode
     function presentToi(){
     return "La ville " . $this->name . " contient " . $this->nbHabitant . "habitants";
     }
7    }
```

POO en PHP

- **Le mot clé new :**
  - Pour créer une instance d'un objet, un nouvel objet doit être créé et assigné à une variable. Pour créer une instance :

```php
<?php
        $ville1=new Ville();
?>
```

POO en PHP

- **Constructeurs** : void __construct (mixed args , ...)
  - Les classes qui possèdent une méthode constructeur appellent cette méthode à chaque création d'une nouvelle instance de l'objet.

```
class Ville{
    private $name;
    private $nbHabitant;

    function __construct($name, $nbHabitant)
    {
        $this->name = $name;
        $this->nbHabitant = $nbHabitant;
    }
}
                    $ville1=new Ville("Fès","1.5");
```

POO en PHP

- **Visibilité**
  - Publique (**public**) : est la visibilité la plus large (accessible directement en dehors de la classe).
  - Privée (**private**): est la plus restrictive (accessible uniquement depuis la classe courante).
  - Protégée (**protected**) : est intermédiaire. L'accès aux éléments protégés est limité aux classes et parents hérités (et à la classe qui a défini l'élément).

POO en PHP

- **Le mot clé extends**
  - Une classe peut hériter des méthodes et des membres d'une autre classe en utilisant le mot clé **extends** dans la déclaration. Il n'est pas possible d'étendre de multiples classes, une classe peut uniquement hériter d'une seule classe de base.
  - Les méthodes et membres hérités peuvent être surchargés, à moins que la classe parent ait défini une méthode comme final. Pour surcharger, il suffit de redéclarer la méthode avec le même nom que celui défini dans la classe parent. Il est possible d'accéder à une méthode ou un membre surchargé avec l'opérateur parent::

POO en PHP

- **Le mot clé extends**

```php
<?php
Class Ville{
private $name;
private $nbHabitant;
function __construct($name, $nbHabitant)
    {
        $this->name = $name;
        $this->nbHabitant = $nbHabitant;
    }
function presentToi(){
    return "La ville " . $this->name .
      " contient " . $this->nbHabitant .
      " habitants";
      }
}
```



```php
// extension de la classe
class Capitale extends Ville
{
    private $pays;
function __construct($name, $nbHabittant,$pays){
    parent::__construct($name, $nbHabittant);
    $this->pays=$pays;
}
    function presentToi(){
        return parent::presentToi() . "est c'est
         la capitale du ". $this->pays;
    }
}
$v=new Ville("Fès","1.5M");
$v->presentToi();
$c=new Capitale("Rabat","0.6M","Maroc");
$c->presentToi();
?>
```

La ville Fès contient 1.5M habitants

La ville Rabat contient 0.6M habitants est c'est la capitale du Maroc

POO en PHP

- **Destructeurs : void __destruct ( )**
  - PHP 5 introduit un concept de destructeur similaire aux autres langages orientés objet, comme le C++ . La méthode destructeur doit être appelée aussitôt que toutes les références à un objet particulier sont effacées ou lorsque l'objet est explicitement détruit.

```php
<?php
    class Ville{
        private $name;
        function __construct($name){
         $this->name = $name;
        }
        function __destruct(){
         $this->name=null;
         echo "je suis dans le destructeur";
                }
    }
       $v=new Ville("Fès");
?>
```

Je suis dans le destructeur

POO en PHP

- **L'opérateur de résolution de portée (::) :**
  - Il fournit un moyen d'accéder aux membres statiques ou constants ainsi qu'aux éléments redéfinis par la classe.
  - Lorsque vous référencez ces éléments en dehors de la définition de la classe, utilisez le nom de la classe.

```
<?php
class MaClass {
    const val= 'Une constante';
}
echo MaClass::val;
?>
```

POO en PHP

Self et parent :

Deux mots-clé spéciaux, self et parent , sont utilisés pour accéder aux membres ou aux méthodes depuis la définition de la classe.

```php
<?php
class Ville{
    private $name;
    public static $nbVille=0;
    function __construct($name){
        $this->name = $name;
        self::$nbVille++;
    }
}
echo Ville::$nbVille;
```

```php
class Capitale extends Ville{
    private $pays;
    private static $nbCapitale=0;
    function __construct($name,$pays){
        parent::__construct($name);
        $this->pays=$pays;
        self::$nbCapitale++;
    }
    function static getNbVille(){
        return parent::$nbVille;}
    function static getNbCapitale(){
        return self::$nbCapitale;}
}
Capitale::getNbCapitale();

echo Capitale::$nbVille;
```

POO en PHP

- **Statique :**
  - Le fait de déclarer des membres ou des méthodes comme statiques vous permet d'y accéder sans avoir besoin d'instancier la classe. Un membre déclaré comme statique ne peut être accédé avec l'objet instancié d'une classe (bien qu'une méthode statique le peut).
  - La déclaration static doit être faite après la déclaration de visibilité. Pour des raisons de compatiblité avec PHP 4, si aucune déclaration de visibilité n'est utilisée, alors le membre ou la méthode sera traité comme s'il avait été déclaré comme public .
  - Comme les méthodes statiques sont appelables sans instance d'objet créée, la pseudo variable $this n'est pas disponible dans la méthode déclarée en tant que statique.
  -

POO en PHP

- **Abstraction de classes**
  - PHP 5 introduit les classes et les méthodes abstraites. Il n'est pas autorisé de créer une instance d'une classe définie comme abstraite. Toutes les classes contenant au moins une méthode abstraite doivent également être abstraites. Pour définir une méthode abstraite, il faut simplement déclarer la signature de la méthode et ne fournir aucune implémentation.
  - Lors de l'héritage depuis une classe abstraite, toutes les méthodes marquées comme abstraites dans la déclaration de la classe parent doivent être définies par l'enfant ; de plus, ces méthodes doivent être définies avec la même (ou plus faible) visibilité .

POO en PHP

- Pour définir une méthode comme étant abstraite, il faut que la classe elle-même soit abstraite.
- Une classe abstraite n'est pas forcément uniquement composée de méthodes abstraites.
- Le fait de déclarer une classe abstraite permet d'empêcher son « instanciation ».
- Le fait de déclarer une méthode abstraite permet de forcer les classes filles à les réécrire.
- Les méthodes abstraites n'ont pas de corps

POO en PHP

```
abstract class Animal
{
 public function manger() {
  return "je mange";
 }
 abstract public function cri();
}
$animal=new Animal();// une classe abstraite n'est pas instanciable.
```

```
class Chien extends Animal
{
 public function cri(){
  return "moi, j'abboie";
 }
}
$chien=new Chien();
$chien->cri();
```

POO en PHP

- **Mot clé ‘final’**
  - PHP 5 introduit le mot-clé " final " qui empêche les classes filles de surcharger une méthode en en préfixant la définition par le mot-clé " final ". Si la classe elle-même est définie comme finale, elle ne pourra pas être étendue.

POO en PHP

- **Interfaces**
  - Les interfaces objet vous permettent de créer du code qui spécifie quelles méthodes et variables une classe peut implémenter, sans avoir à définir comment ces méthodes seront gérées.
  - Les interfaces sont définies en utilisant le mot clé interface , de la même façon qu'une classe standard mais sans aucun contenu de méthode.
  - Toutes les méthodes déclarées dans une interface doivent être publiques.
- **Implements**
  - Pour implémenter une interface, l'opérateur **implements** est utilisé. Toutes les méthodes de l'interface doivent être implémentées dans une classe ; si ce n'est pas le cas, une erreur fatale sera émise. Les classes peuvent implémenter plus d'une interface en séparant chaque interface par une virgule.

POO en PHP

- **Interfaces**

```
<?php
interface ModeDeplacement{
        public function deplacement();
 }
class Bateau implements ModeDeplacement{
 public function deplacement(){
  echo "je navigue";
 } ‘
}
```

```
class Avion implements ModeDeplacement{
 public function deplacement() {
  echo "je vol";
        }
}
```

POO en PHP

- **Parcours d'objets**
  - PHP 5 fournit une façon de définir les objets de manière à ce qu'on puisse parcourir une liste de membres avec une structure foreach . Par défaut, toutes les propriétés visibles seront utilisées pour le parcours.

```
$object = new MyClass();

foreach($object as $key => $value) {
    echo "$key => $value\n";
}
```