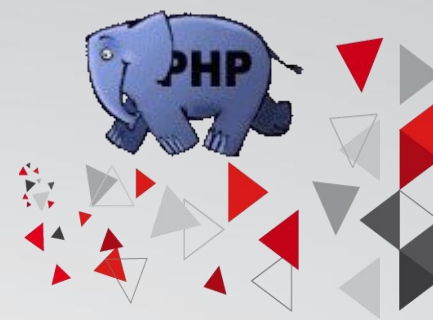




# P00 avec PHP5

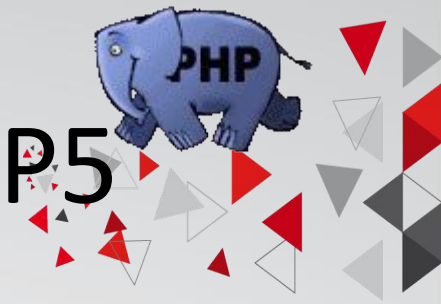
Année universitaire  
2017-2018

# Plan



- ❖ Migration de PHP4 vers PHP5
- ❖ Les nouveautés dans PHP5
- ❖ Notion de classe
- ❖ Constructeur
- ❖ Instanciation d'un objet
- ❖ Les droits d'accès
- ❖ L'encapsulation
- ❖ Redéfinition
- ❖ L'héritage

# Migration de PHP4 vers PHP5



Pourquoi?

- Nouvelles fonctionnalités
- Meilleures performances
- Meilleure sécurité
- Meilleure stabilité
- PHP 5 est fortement supporté

# Les nouveautés dans PHP5 (1/4)

- Prise en charge complète de la POO
- Prise en charge de XML
- Intégration de la BD embarquée SQLite
- Intégration de nouvelles extensions (JSON, Filter, ZIP, ...)
- Gestion des appels aux bases de données : PHP Data Object (PDO)
- Le nouveau modèle objet (passage par référence)
- La gestion des exceptions a fait son apparition en PHP5
- Apparition de la SPL (Standard PHP Library), un rassemblement de classes internes utiles

# Les nouveautés dans PHP5 (2/4)

- Prise en charge de la POO
  - Nouvelles méthodes
    - visibilité
      - Public
      - Private
      - Protected
    - Encapsulation
    - Interfaces
    - Héritage
    - classes abstraites...

# Les nouveautés dans PHP5 (3/4)

## Le nouveau modèle Objet



- Le mot clef « var » utilisé en PHP 4 ne fonctionne plus en PHP 5, il est traduit en « public nom de la variable ».
- Le constructeur de classe:
  - En PHP 4 : fonction ayant le même nom que la classe
  - En PHP5: `__construct()`, `__destruct()`

# Les nouveautés dans PHP5 (4/4)

## XML



- La version 4 de PHP impliquait une utilisation relativement lourde pour qui souhaitait manipuler des flux XML
- Avec la version 5, il existe deux nouveautés :
  - L'intégration d'un nouveau gestionnaire XML : la bibliothèque libxml2, qui amène une implémentation DOM standard complète (ce qui n'était pas le cas en PHP 4)
  - L'extension SimpleXML



# PPO: Programmation Orienté Objet









# PОО: Notion de classe (1/2)

- Une **classe** est une représentation abstraite d'un **objet**.
- Une classe peut généralement être rendue concrète au moyen d'une **instance de classe**, que l'on appelle **objet**.
- Une classe s'écrit au moyen du mot "**class**" suivi du nom de la classe et d'accolades.

```
Class Personne
{
    public $nom;
    public $prenom;
}
```

# P00: Notion du classe (2/2)

Point	
 x	
 y	
 getX()	
 getY()	
 setX()	
 setY()	

```
class Point
{
    //attributs
    private $x;
    private $y;

    //méthodes
    public function getX()
    {
        return $this->x;
    }

    public function setX($x)
    {
        $this->x = $x;
    }

    public function getY()
    {
        return $this->y;
    }

    public function setY($Y)
    {
        $this->y = $Y;
    }
}
```

# POO: Le constructeur

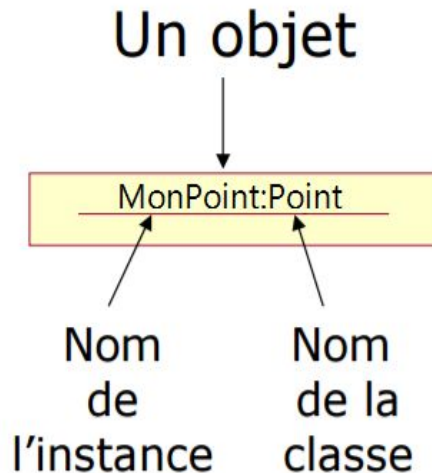


- Le constructeur est la méthode qui va être appelée à l'instanciation de l'objet.
- Il doit être implémenté dans la classe elle-même de la manière suivante :

```
public function __construct()  
{  
  
}
```

# POO: instantiation d'un objet

- Exemple:



//Déclaration d'une instance  
`$MonPoint = new Point();`

//Utilisation  
`$MonPoint->setX("2");`

//Vérification  
`if($MonPoint instanceof Point)  
echo "L'objet \ $MonPoint est du type Point" ;`

# POO: Les propriétés d'un objet

- Une propriété est une variable associée à un objet.
- On peut demander « le nom de cette personne » et non le nom en général

```
<?php  
  
class Personne { }  
  
$michel = new Personne();  
  
echo $michel->nom;
```

Pas de caractère \$ devant le nom de la propriété

La propriété nom est liée intrinsèquement à l'objet ; cette liaison est notée par la flèche

# POO: Remarque



- Constructeur
    - ✓ **PHP 3 et 4** : une fonction portant le même nom que la classe
    - ✓ **PHP5** : une fonction membre spécifique. PHP5 ne permet pas **la surcharge(overloading)** de fonction (ou de méthodes) et donc on ne peut attribuer le même nom à plusieurs fonctions. Par contre la redéfinition est possible.
  - Destructeur
    - ✓ **PHP3 et 4** : il n'y a pas de destructeur
    - ✓ **PHP5**: introduit la notion de destructeur
- Destruction : on utilisera la fonction **unset(\$MonPoint)**.

# POO: Les droits d'accès



- Les méthodes et les variables "**public**" sont visibles et manipulables par tous les objets, même s'ils sont relatifs à d'autres classes.
- Les méthodes et les variables "**protected**" concernent les objets de la même classe ainsi que ses dérivés, mais pas ceux des classes étrangères.
- Les classes ont des variables et des méthodes internes et qui ne concernent pas l'extérieur. Ces propriétés sont déclarées en tant que "**private**".

# POO: L'encapsulation



- L'encapsulation est la pratique consistant à regrouper des attributs au sein d'une même classe.
  - Pour améliorer la lisibilité des programmes, les attributs encapsulés sont souvent **privés** (inaccessibles aux autres classes)
  - Les données et méthodes accessibles sont dites **publiques**



# POO: La redéfinition



- PHP permet la redéfinition des méthodes, c'est-à-dire la possibilité de redéclarer les mêmes attributs et opérations d'une super classe au sein d'une sous classe. Nous pouvons aussi modifier la signature de la fonction, et son traitement

```
class DeuxRoues
{
    //Attribut
    protected $_couleur ;

    //Méthode
    public function getCouleur()
    {
        return $this->_couleur;
    }
}
```

```
class Bicyclette extends DeuxRoues
{
    //redéfinition de la méthode
    public function getCouleur()
    {
        echo "Nouvelle méthode !";
        return $this->_couleur;
    }
}
```

```
//Déclaration d'une instance :
$MonVelo = new Bicyclette;

$MonVelo->getCouleur();
```

Appel de cette méthode



# POO: L'héritage



- L'**héritage** consiste à définir différents niveaux d'abstraction permettant ainsi de **factoriser** certains attributs et/ou méthodes communs à plusieurs classes.
- Une classe générale définit alors un ensemble d'attributs et/ou méthodes qui sont partagés par d'autres classes, dont on dira qu'elles héritent de cette classe générale.