

Apprentissage et Pratique du Langage PHP

Programmation orientée Objets

*Par
Elhassan Abdelwahed
Département d'Informatique
Faculté des Sciences Semlalia Marrakech*

Evolution PHP OO (5)


La version PHP 5 offre de véritables concepts OO :

- Constructeur, destructeur,
- Objets en tant que références
- Droits d'accès aux membres d'une classe (public/protected/private)
- Interfaces, Héritage
- Classe abstraite,
- Méthodes magiques,
- ... etc.

Classe & Objets

```
<?php
class Produit {
    private $designation;
    public $prixHT;
    function SetDesignation( $Desg ){ $this->designation = $Desg; }
    function GetDesignation ( ) {
        echo "La Classe courant est : ".__CLASS__."<br>";
        echo "La Méthode courante est : ".__METHOD__."<br>";
        return ( $this->designation ) ; }
}

$prd = new produit( );
$prd->SetDesignation( "Ordinateur" );
$prd->prixHT = 4500;
echo "Prix de ".$prd->GetDesignation( )." est ".$prd->prixHT;
?>
```



La Classe courant est : Produit
La Méthode courante est : Produit::GetDesignation
Prix de Ordinateur est 4500

Classe & Objets

Constantes d'une classe

Accès à une constante depuis l'extérieur de la classe : **<Nom Classe>::<Nom Constante>**

Accès à une constante depuis une méthode de la classe : **self::<Nom Constante>**

```
<?php
class Livre {
    const INFORMATIQUE = 1;
    const PHYSIQUE = 2;
    const MATHEMATIQUE = 3;
    const BIOLOGIE = 4;
    public $Thematique;
    function SetThematique( ) {$this->Thematique = self::BIOLOGIE;}
}

$livbio = new Livre( );
$livbio->SetThematique( );
$livinfo = new Livre( );
$livinfo->Thematique = Livre::INFORMATIQUE;
echo "Thématique 1: ".$livbio->Thematique." Thématique 2: ".$livinfo->Thematique;
?>
```

Classe & Objets

Exemple de méthode magique: Affichage d'un objet

```
<?php
class Etudiant {

    private $nom;
    private $moyenne;

    function Affectation( $n, $m ) {    $this->nom = $n;
                                        $this->moyenne = $m;
    }

    function __toString( ) {
        echo "Etudiant ".$this->nom."<br>";
        echo " a une moyenne de ".$this->moyenne;
    }

}

$setd = new Etudiant( );
$setd->Affectation( "Hamid", 12);
echo $setd;
?>
```

Clonage d'objets

Clonage: copie explicite d'objet

```
<?php
class Livre {
    private $titre = "Langage PHP5";
    private $prix = 150;
    function AfficheLivre( ) {
        echo "Titre: ".$this->titre." Prix: ".$this->prix."<br>";
    }
    function __clone( ){ $this->prix = 100; }
}

$original = new Livre ( );
$copie = clone ( $original );
$original->AfficheLivre( );
$copie->AfficheLivre( );
?>
```

Clone & Référence


Egalité & Identité

Test de l'égalité se fait par ==

Test de l'identité (les deux variables référencent le même objet) se fait par ===

```
<?php
class Livre {
    private $titre = "Langage PHP5";
    private $prix = 150;
    function AfficheLivre( ) {
        echo "Titre: ".$this->titre." Prix: ".$this->prix."<br>";
    }
    function ChangePrix( $nouveauprix){ $this->prix = $nouveauprix; }
}

$original = new Livre ( );
$unecopie = clone ( $original ); $unereference = $original;
$original->AfficheLivre(); $unecopie->AfficheLivre(); $unereference->AfficheLivre();
if ( $original == $unecopie ) echo "Objets égaux"."<br>";
if ( $original === $unereference ) echo "Objets identiques"."<br>";
$unecopie->ChangePrix( 100 ); $unereference->ChangePrix( 120 );
if ( $original == $unecopie ) echo "Objets égaux"."<br>";
if ( $original === $unereference ) echo "Objets identiques"."<br>";
$original->AfficheLivre(); $unecopie->AfficheLivre(); $unereference->AfficheLivre();
?>
```



Titre: Langage PHP5 Prix: 150
Titre: Langage PHP5 Prix: 150
Titre: Langage PHP5 Prix: 150
Objets égaux
Objets identiques
Objets identiques
Titre: Langage PHP5 Prix: 120
Titre: Langage PHP5 Prix: 100
Titre: Langage PHP5 Prix: 120

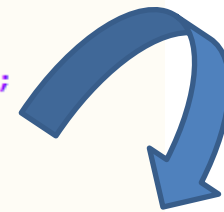
Constructeur & Destructeur

Exemple

```
<?php
class Livre {
    private $titre;
    private $prix;
    function AfficheLivre( ) {
        echo "Titre: ".$this->titre." Prix: ".$this->prix."<br>";
    }
    function __construct ( $t, $p = 150) {
        $this->titre = $t; $this->prix = $p;
        echo "Construction livre: " . $this->titre."<br>";
    }
    function __destruct ( ) {
        echo "Destruction livre: " . $this->titre."<br>";
    }
}

echo "Début constructeurs L1 et L2 <br>";
$L1 = new Livre ( "Langage PHP", 250);
$L2 = new Livre ("Pratique MySQL");
echo "Création L3 (référence L1) et L4 (copie de L2) <br>";
$L3 = $L1; $L4 = clone ( $L2 );
echo "Destruction de L1 et de L2 <br>";
unset( $L1); unset( $L2);
echo "Destruction de L3 et de L4 <br>";
unset( $L3); unset( $L4);
echo "Fin de la fonction <br>";
```

?>



Début constructeurs L1 et L2
Construction livre: Langage PHP
Construction livre: Pratique MySQL
Création L3 (référence L1) et L4 (copie de L2)
Destruction de L1 et de L2
Destruction livre: Pratique MySQL
Destruction de L3 et de L4
Destruction livre: Langage PHP
Destruction livre: Pratique MySQL
Fin de la fonction

Héritage – Exemple (1/2)

Surcharge des méthodes, Constructeur & Destructeur

```
class Produit {  
    protected $designation;  
    protected $prix;  
    function PrixTTC( ) {  
        $TVA = 0.10; // TVA = 10%  
        $PrTTC = $this->prix + $this->prix * $TVA;  
        return( $PrTTC );  
    }  
    function __construct ( $d, $p ) {  
        $this->designation = $d; $this->prix = $p;  
        echo "Fin Constructeur de la classe de base <br>";  
    }  
    function __destruct( ) { echo " Fin Destruction de la classe de base <br>"; }  
    function Affiche () { echo $this->designation." ".$this->prix."<br>"; }  
}
```

Classe Produit

Désignation

Prix

+PrixTTC // TVA = 10%

+Affiche

Classe Produit de Luxe

catégorie

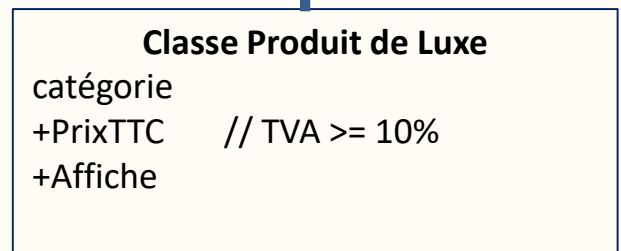
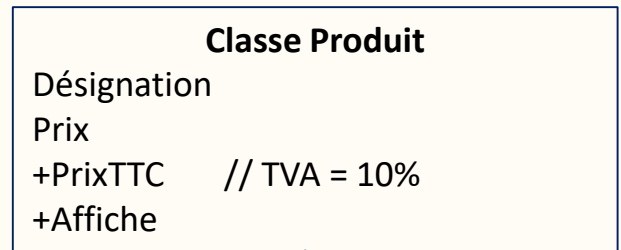
+PrixTTC // TVA >= 10%

+Affiche

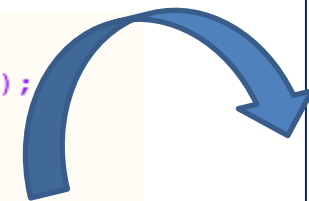


Héritage – Exemple (2/2)

```
class ProduitLuxe extends Produit {  
    private $categorie;  
    function PrixTTC( $TVA = 0.10 ) {  
        $PrTTC = $this->prix + $this->prix * $TVA;  
        return( $PrTTC );  
    }  
    function __construct ( $d, $p, $c ) {  
        parent::__construct( $d,$p);  
        $this->categorie = $c;  
        echo "Fin Constructeur de la classe dérivée <br>";  
    }  
    function __destruct ( ) {  
        parent::__destruct( );  
        echo " Fin Destruction de la classe dérivée<br>";  
    }  
    function Affiche () { echo $this->designation." ".$this->prix." ".$this->categorie."<br>"; }  
}
```



```
$P = new Produit ( "Ordianteur", 5000);  
$PL = new ProduitLuxe( "Mac", 15000, "Informatique");  
$P->Affiche( );  
$PL->Affiche( );  
echo "Prix TTC Produit: ".$P->PrixTTC( )."<br>";  
echo "Prix TTC Produit Luxe: ".$PL->PrixTTC(0.20)."<br>";
```



Fin Constructeur de la classe de base
Fin Constructeur de la classe de base
Fin Constructeur de la classe dérivée
Ordianteur 5000
Mac 15000 Informatique
Prix TTC Produit: 5500
Prix TTC Produit Luxe: 18000
Fin Destruction de la classe de base
Fin Destruction de la classe dérivée
Fin Destruction de la classe de base

Résolution de portée

Surcharge des méthodes, Opérateurs de résolution de portée

Les mots clefs **parent** et **self** combinés à l'opérateur **::** permettent aux classes dérivées d'accéder (et de distinguer) leurs propres méthodes (surchargées) des méthodes (initiales) de la classe de base.

```
<?php
class Logement {
    // Autres membres de la classe
    protected function QuiSuisJe( ) { return( "Je suis un logement" ); }
}

class Appartement extends Logement {
    // Autres membres de la classe
    protected function QuiSuisJe( ) { return( "Je suis un appartement" ); }
    public function IdentifierParent ( ) { return( parent::QuiSuisJe( ) ); }
    public function IdentifierSelf ( ) { return( self::QuiSuisJe( ) ); }
}

$appart = new Appartement ( );
echo $appart->IdentifierParent( )."<br>";
echo $appart->IdentifierSelf()."<br>";
?>
```

Classe abstraite - Exemple

```
1 <?php
2 abstract class Produit {
3     protected $Designation;
4     protected $Famille;
5     protected $PrixHT;
6     abstract function PrixTTC ( ) ;
7     // Autres membres de la classe mais pas de constructeur
8 }
9
10 class Ordinateur extends Produit {
11     private $OS;
12     public function __construct ($d,$f,$p) {
13         $this->Designation=$d; $this->Famille=$f;$this->PrixHT=$p;}
14     public function SetOS ( $SysExploitation ) {
15         $this->OS = $SysExploitation; }
16     public function GetOS ( ) { return( $this->OS ); }
17     public function PrixTTC ( ){
18         $ttc = $this->PrixHT+ $this->PrixHT * 0.10;
19         return ( $ttc ) ;} // TVA = 10%
20     public function Specifications( ) { return ( $this->Designation.", ".$this->Famille ); }
21 }
22
23 $pc = new Ordinateur ( "Acer", "Informatique", 4500);
24 $pc->SetOS ( "Linux" );
25 echo $pc->Specifications( )." OS installé: ".$pc->GetOS( )." Prix TTC: ".$pc->PrixTTC( );
26 ?>
```

Interface - Exemple

```
1 <?php
2     interface TextEditor
3     {
4         public function addParagraph($text);
5         public function addTitle($text, $level);
6     }
7     class HTMLText implements TextEditor
8     {
9         private $html;
10        public function __construct($title) {
11            $this->html = "<html> <head> <title>'".$title."</title>    </head>";
12            $this->html .= "<body> <h1>".$title."</h1>";
13        }
14        public function addParagraph($text) { $this->html .= "<p>".$text."</p>"; }
15
16        public function addTitle($t, $l) { $this->html .= "<h".$l.">".$t."</h".$l.">"; }
17
18        public function AfficheHTML() {
19            $this->html .= "</body> </html>";
20            echo $this->html;
21        }
22    }
23
24    $doc = new HTMLText("HTML Test");
25    $doc->addTitle("Ceci est un titre de paragraphe", 2);
26    $doc->addParagraph ("Ceci est le contenu du paragraphe");
27    $doc->AfficheHTML( );
28 ?>
```



HTML Test

Ceci est un titre de paragraphe

Ceci est le contenu du paragraphe

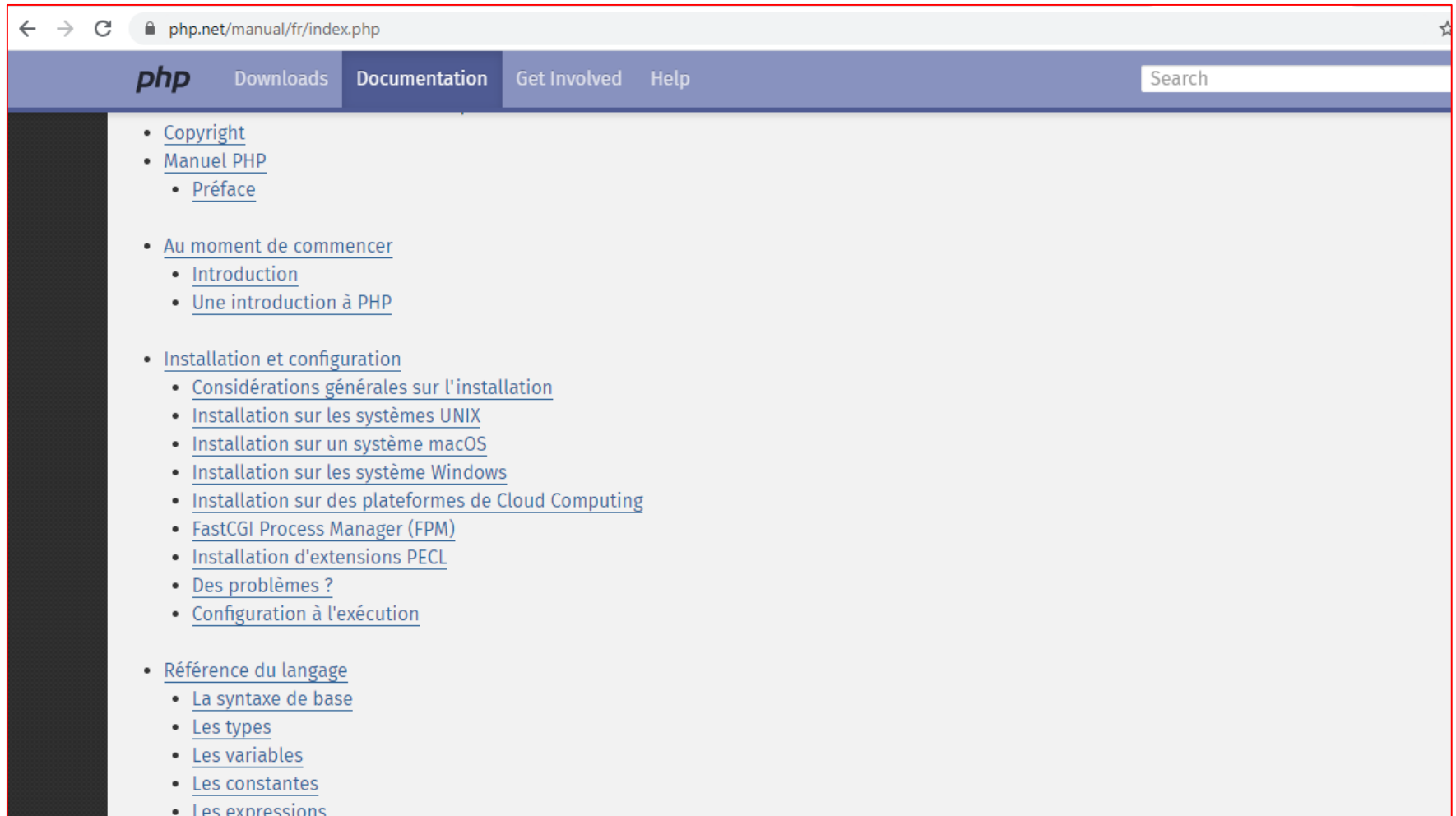
Accès & Membres statiques – Exemple

```
1  <?php
2
3  class Produit
4  {
5      public static $NbObjets = 0;
6      private $Designation;
7      public function __construct($d) {
8          $this->Designation = $d;
9          self::$NbObjets = self::$NbObjets + 1;
10     }
11     public static function CombienProduits ( ) { return( self::$NbObjets ); }
12 }
13
14 $TV = new Produit("TV");
15 $Ordinateur = new Produit("Ordinateur");
16 echo Produit::CombienProduits ( );
17
18 ?>
```

Manuel officiel de PHP

Pour plus d'informations (fonctions, ...etc.) Consultez le manuel officiel de PHP :

<https://www.php.net/manual/fr/index.php>



AOO: Classe, Instances, ...etc.

Objectifs de l'atelier 5

- Mise en œuvre des différentes notions acquises (POO en PHP) dans le cadre du projet de gestion des notes.
- Reprendre les travaux réalisés dans le cadre de l'atelier 4 pour la gestion des notes et l'améliorer en utilisant les classes et un seul tableau contenant les instances pour gérer les données au niveau de la RAM.
- Les données seront sauvegardées au niveau du disque dur sous forme de fichier CSV.

Remarque

Le programme de gestion des notes va évoluer en y intégrant à chaque nouveau chapitre les différentes notions étudiées.