

Cours de PHP



Jean-Luc Collinet

Haute École Léonard de Vinci : Paul Lambin
Bloc 1 du Bac en Informatique

Vu en semaine 1

- Formulaires HTML
- Variables en PHP
- Constantes en PHP
- Utilisations de fonctions en PHP
- Architecture MVC Orientée Objet
- Introduction à Git

Au menu de la semaine 2

- L'orienté objet en PHP
- Créer et utiliser des fonctions
- Traiter les données des formulaires HTML
- Gérer des tableaux
 - Scalaires
 - Associatifs
- Anticiper les erreurs courantes de programmation en PHP

L'instruction if

```
<?php
    $a = 2019;
    $b = 2019;
    if ($a == $b) {
        $msg1 = '$a est égal à $b';
        $msg2 = "$a est égal à $b";
    }
?>
```

L'instruction else

```
if ($a == $b) {  
    $msg = '$a est égal à $b';  
}  
else {  
    $msg = '$a est différent de $b';  
}
```

L'instruction elseif

```
if ($privilege == "admin") {  
    $msg = "Interface d'administration";  
}  
elseif ($privilege == "user") {  
    $msg = "Interface utilisateur";  
}  
else {  
    $msg = "Privilège non défini";  
}
```

L'instruction de boucle for

```
$somme = 0;
```

```
for ($i = 1; $i <= 10; $i++) {  
    $somme += $i;  
}
```

L'instruction de boucle while

```
$somme = 0;  
$i = 1;  
$stop = false;  
while ($i != 10 and !$stop) {  
    $somme += $i;  
    if ( # test à écrire # ) $stop = true;  
    $i++;  
}
```


Définition d'attributs d'une classe

```
<?php
class Personnage {
    private $_force;           // La force du personnage
    private $_experience;      // Son expérience
    private $_vie;             // Sa vie

    // Suivi de la définition de méthodes etc.
    // cf. diapositives suivantes
}
?>
```

Accesseur (getteur)

```
public function experience() {  
    return $this->_experience;  
}
```

- `$this` est l'objet courant
- Le symbole `->` est l'équivalent du point en Java
- On n'écrit pas le `$` de la variable privée après le symbole `->`

Mutateur (setteur)

```
public function setExperience($experience) {  
    if (!is_int($experience)) {  
        trigger_error('L\'expérience d\'un  
        personnage doit être un nombre entier',  
        E_USER_WARNING);  
        return;  
    }  
    $this->_experience = $experience;  
}
```

Définition d'une méthode

```
public function gagnerExperience() {  
    $this->_experience++;  
}
```

- Rappel de la variable privée déclarée précédemment :
private \$_experience;

Définition d'un constructeur

```
public function __construct() {  
    $this->_force=80;  
    $this->_vie=100;  
    $this->_experience=1;  
}
```

- __ est un double souligné (deux *underscores* qui se suivent)
- C'est une méthode dite en PHP magique

Création d'objets

```
<?php // dans un contrôleur  
require('models/Personnage.class.php');  
  
$perso1 = new Personnage();  
$perso2 = new Personnage();
```

```
?>
```



Require des classes automatisé

```
<?php // dans index.php
function chargerClasse($classe) {
    require_once('models/' . $classe . '.class.php');
}
```

```
spl_autoload_register('chargerClasse');
/* La fonction 'chargerClasse' sera appelée dès qu'on
instanciera un objet d'une classe */
```

```
$perso = new Personnage();
```

```
?>
```

Les constantes de classe

// Déclarations de constantes

```
const FORCE_PETITE = 20;
```

```
const FORCE_MOYENNE = 50;
```

```
const FORCE_Grande = 80;
```

- En majuscules

Utilisation d'une constante de classe

```
<?php
```

```
// On envoie une « FORCE_MOYENNE »  
// en guise de force initiale.
```

```
$perso = new
```

```
Personnage(Personnage::FORCE_MOYENNE);
```

```
?>
```

Mutateur avec constantes de classe

```
public function setForce($force) {
```

```
/* On vérifie qu'on nous donne bien en paramètre soit une « FORCE_PETITE », soit une « FORCE_MOYENNE », soit une « FORCE_Grande » QUI SONT DES CONSTANTES DE CETTE CLASSE */
```

```
    if (in_array($force,  
        array(self::FORCE_PETITE,  
              self::FORCE_MOYENNE, self::FORCE_Grande))) {
```

```
        $this->_force = $force;
```

```
    }
```

```
}
```

Use \$this to refer to the current object.
Use self to refer to the current class.
In other words, use \$this->member for non-static members,
use self::\$member for static members.

Les attributs et méthodes statiques

```
class Compteur {  
    // Déclaration l'attribut statique $compteur  
    private static $_compteur = 0;  
  
    public function __construct() {  
        // Modification de la variable statique $compteur  
        self::$_compteur++;  
    }  
  
    // Méthode statique  
    public static function getCompteur() {  
        return self::$_compteur;  
    }  
}
```

Les attributs et méthodes statiques

```
$test1 = new Compteur();
```

```
$test2 = new Compteur;
```

```
$test3 = new Compteur;
```

```
echo Compteur::getCompteur();
```

Fonctions

- PHP offre plein plein plein de **fonctions prédéfinies**
 - Manipulation des **chaînes de caractères**
 - Gestion des **dates** et des heures
 - Gestion des **tableaux**
 - Gestion des fichiers et des dossiers
 - Accès aux **bases de données**
 - Fonctionnalités pour des réseaux et **Internet**

Définition de fonction

```
<?php
    function mafonction($arg_1, $arg_2, /* ..., */
                        $arg_n)
    {
        // Traitement(s) à faire
        // ...
        return $valeuraretourner;
    }
?>
```

Valeur de retour d'une fonction

```
<?php
function additionner($a, $b) {
    $c = $a + $b;
    return $c;
}
```

```
# Utilisation de la valeur retour de la fonction
$result = additionner(8, 6);
?>
```

Paramètres GET passés via URL

`http://localhost/monsiteweb/
index.php?mavARIABLE=777`

- Le tableau `$_GET` n'est pas vide
- Une variable `$_GET['mavARIABLE']` vaut 777

`http://localhost/monsiteweb/
index.php?action=livres&no=33`

- Une variable `$_GET['action']` vaut 'livres'
- Une variable `$_GET['no']` vaut 33

Formulaires HTML

<!-- Passage des paramètres selon la méthode GET -->

```
<form action="index.php" method="GET">
```

Prénom: <input type="text" name="prenom">

```
<input type="submit" value="Envoyer">
```

```
</form>
```

<!-- Passage des paramètres selon la méthode POST -->

```
<form action="index.php" method="POST">
```

Prénom: <input type="text" name="prenom">

```
<input type="submit" value="Envoyer">
```

```
</form>
```

Paramètres POST d'un formulaire

```
<?php
```

```
// contrôleur PHP qui traite un formulaire  
// cf. dia 24 utilisant la méthode POST
```

```
if (!empty($_POST)) {
```

```
    // Traitements des paramètres
```

```
    // Ici, une variable $_POST['prenom'] vaut
```

```
    // ce que l'utilisateur a entré dans la zone de
```


```
    // texte dont l'attribut name est prenom
```

```
}
```

Utiliser GET ou POST ?

- GET
 - Valeurs des paramètres visibles dans l'URL
 - Pour demander des informations
- POST
 - Invisibilité des paramètres (Sécurité 😊)
 - Pour poster (envoyer) des informations
 - Donne généralement lieu à des changements dans la base de données

Tableaux

- **Scalaire**
 - Indicés par des entiers positifs
 - A partir de 0
 - `$pays[3]`
- **Associatifs** 
 - Indicés par des valeurs quelconques
 - `$users['admin']`

Tableaux à une dimension

```
<?php
$pays = array('Belgique', 'France');
// ajout d'un élément en dernière position
$pays[] = 'Suisse';

$membres[0] = 'INSTANT Justin';
$membres[1] = 'SAHALOR Aubin';
// ajout d'un élément en dernière position
$membres[] = 'MANVUSSA Gérard';
?>
```

Parcours classique mais Attention

```
<?php
$nb_membres = count($membres);
for ($i = 0 ; $i < $nb_membres ; $i++) {
    // $i est l'indice
    // $membres[$i] est l'élément d'indice $i
    // ...
}
```

Attention ce parcours ne fonctionne plus s'il y a des « trous » dans le tableau suite à la suppression de valeur(s) !

Parcours du tableau \$membres

```
foreach ($membres as $cle => $valeur) {  
    // $cle est l'indice  
    // $valeur est l'élément d'indice $cle  
}
```

Exemple de code PHP dans une vue

```
<p>
<?php foreach ($membres as
                    $i => $membre) {
    echo $i . ' : ' . $membre . '<br />';
}
?></p>
```


Anticiper les erreurs courantes

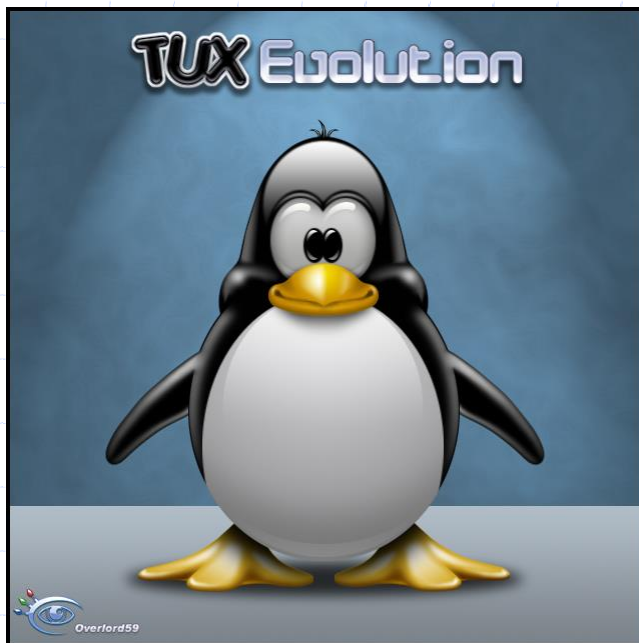
- Erreurs **syntaxiques**
 - Oubli du séparateur d'instructions ;
 - Délimitation incorrecte des chaînes de caractères entre " " ou entre ' '
- Erreurs **fonctionnelles**
 - Analyse erronée
 - Oubli de l'**echo** d'une variable dans une vue
- Erreurs **à l'exécution**
 - Référence à une variable qui n'existe pas
 - Chemin erroné vers un fichier sur le disque

« Copy and Paste »

- **Comprendre** ce que vous recopiez d'ailleurs !
- **Ne pas copier-coller d'un PowerPoint :** cela provoque parfois des erreurs indétectables !

Bonnes évolutions à tous

- Lorsqu'une chose évolue, tout ce qui est autour d'elle évolue...



Paulo Coelho