M1105 - Cours n°6

Introduction au langage **PHP**

I - PHP (Hypertext Preprocessor)

Langage interprété Open Source :

- Tire son origine de PHP/FI (1995, Rasmus Lerdorf)
- PHP 3.0 (1998, Zeev Suraski et Andi Gutmans)
- PHP 4.0 (2000, début des aspects objets)
- PHP 5.0 (2004, modèle objet complet)
- PHP 7.0 (2015, performance, qq modifications)

- spécialement conçu pour le développement d'applications web

  ⇒utilisé pour produire des *pages dynamiques* via un serveur web

  ⇒mais peut aussi fonctionner comme n'importe quel langage interprété de façon locale

PHP (Hypertext Preprocessor)

Langage interprété :

- interpréteur PHP = le « Zend Engine »
- interprété côté serveur (≠ Javascript qui s'exécute côté client)
- extrêmement simple pour les néophytes,
- mais offre des fonctionnalités avancées pour les experts.

Pour connaître la version installée: *phpversion()*

⇒ Sites documentation PHP:

http://www.phpdebutant.org/ http://www.phpfrance.com/

http://www.php.net/manual/fr/ http://www.manuelphp.com/

…

3

PHP (Hypertext Preprocessor)

- Langages de programmation côté serveur les plus utilisés par les sites web (World Wide Web Technology Surveys - https://w3techs.com/)

© W3Techs.com	usage	change since 1 December 2017
1. PHP	83.1%	+0.1%
2. ASP.NET	14.1%	-0.1%
3. Java	2.5%	
4. static files	1.4%	
5. ColdFusion	0.6%	

percentages of sites

4

PHP et les Bases de Données

Une des grandes forces de PHP :

– le support de nombreuses bases de données:

- **PostgreSQL**
- Ingres
- Oracle (OCI7 et OCI8)
- Sybase
- IBM DB2
- MySQL
- Informix
- ...

5

II - Syntaxe de base PHP

- délimité par des balises d'échappement (début et fin)
  <?php ... ?>
- instructions terminées par un point-virgule ;
- bloc d'instructions délimité par des accolades { }
- des commentaires :

// ou #	commentaire sur une ligne
/*… */	commentaire sur plusieurs lignes

- 4 types simples en PHP : booléen, entier, réel (flottant), chaîne de caractères
- **typage faible et dynamique** : le type d'une variable est déterminé par la valeur qu'on lui donne et peut changer au cours du programme
- Pour connaître le type:

```
is_double() is_float() is_string() is_int() is_integer
  is_boolean() is_array() is_object() is_resource()
```

6

Les variables et constantes PHP

Les variables :

- représentées par un signe dollar "$" suivi du nom de la variable : `$NomDeVariable`
- attention : **$x != $X**
- existence d'une variable : `isset()`
- valeur NULL : `is_null()`
- Variables dynamiques (variables variables):

```
$pomme = 'golden';
$fruit = 'pomme';
print ${$fruit}; //équivalent à print $pomme;affiche golden !
echo ${$fruit}; //équivalent à print $pomme;affiche golden !
```

Les constantes : `define`

pas de $, par convention => en majuscule

```
define("TAUX",6.55957);
```

7

Les opérateurs

- Opérateurs arithmétiques

  `+ - * / %`

- Opérateurs de comparaison

  `== != < <= > >= === !==`

- Opérateurs booléens

  `! AND && OR || XOR`

- Opérateurs d'incrémentation, de décrémentation

  `++ --`

- Opérateurs d'affectation, concaténation

  `= . .=`

8

Exemple: PHP « en-ligne »

Fichier *hello.php*

```
<?php
print ('Hello world ! \n ');
 ?>
```

1. Vérification syntaxique:

```
php -l hello.php
```

2. Exécution

```
php hello.php
```

9

Exemple: PHP + HTML

Fichier *hello2.php*

Dans un fichier PHP, tout ce qui est en dehors des balises <?php ... ?> est produit en sortie *sans modification*.

Le serveur web Apache intègre un module "interpréteur PHP":

Exécution du programme :

http://www-etu-info… /hello2.php

```
<!DOCTYPE html>
<html lang="fr">
<head>
      <title> Un exemple </title>
      <meta charset … />
      <link … />
</head>
<body>
<h1> Voici le résultat : </h1>
<p>
            <?php
               print ("Hello world !\n");
            ?>
</p>
</body>
</html>
```

10

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <title> Un exemple </title>
    <meta charset … />
    <link … />
</head>
<body>
<h1> Voici le résultat : </h1>
<p> Hello world !</p>
</body>
</html>
```

11

Structures de contrôle

- if/ if .. else / if .. elseif …
- switch, …
- while / for / **foreach**…

```
$i = 0;
while ($i <= 9){
   print $i;
   $i++;
}
```

```
switch ($val) {
 case  "pomme":
       $i=0;
       break;
 case  "poire":
       $i=1;
       break;
 default:
       $i=2;
       break;
}
```

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

```
if ($i == 0) {print "i vaut 0";}
  elseif ($i == 1) {print "i vaut 1";}
    elseif ($i == 2) { print "i vaut 2";}
else { print "i différent";}
```

12

III - Les chaînes de caractères

Attention! Interprétation des variables… ou non! :

- Interprétation des variables

```
$fruit = "la pomme";    (ou $fruit = 'la pomme';
$chaine1 = "mon fruit préféré est $fruit";
print $chaine1;   ???
```

- Pas d'interprétation des variables

```
$chaine2 = 'mon caractère préféré est le $';
$chaine3 = 'mon fruit préféré est $fruit';
echo $chaine2;   ???
echo $chaine3;   ???
```

13

Les chaînes de caractères

- Caractère d'échappement \

```
$chaine4 = "mon fruit préféré est \$fruit";
echo $chaine4;   ???
```

- Concaténation de chaînes

```
$chaine1 = 'ceci est ';
$chaine2 = 'une concaténation';

$chaîne5 = $chaîne1.$chaîne2;
$chaîne1 .= $chaîne2

echo $chaine5;   ???
echo $chaine1;   ???
```

14

Traitement des chaînes

- Affichage simple : `echo() ou print()`

```
echo('Bonjour les étudiants');
print('Bonjour les étudiants');
echo 'Bonsoir les étudiants';
print 'Bonsoir les étudiants';
$val='chouette'; echo 'PHP',5,' est super ',$val;
```

- Affichage avec masque `printf() sprintf()`

```
$masque = 'la dernière version de %s est %s';
printf ($masque,'PHP', '5.4.17');
$chaine = sprintf ($masque,'PHP', '5.4.17');
```

- Accès à un caractère d'une chaîne

```
$chaine = 'vive les pommes !';
echo $chaine[1]  //affiche i
```

15

Traitement des chaînes

- Taille d'une chaîne `strlen()`
- Position d'un motif dans une chaîne `strpos()`
- Remplacer un motif dans une chaîne `str_replace()`
- Comparaison d'une chaîne à un motif `ereg()`

```
ereg("^[A-Za-z]",$chaîne); ereg obsolète
  preg_match
```

- Changement de casse `strtoupper() strtolower()`
- Echappement et inverse `addslashes() stripslashes()`

```
$prenom = 'Eddie';
$nom = addslashes ("O'Sullivan");
$req = "INSERT INTO selectionneur (prenom,
  nom)VALUES ('$prenom','$nom')";
```

- … de l'ordre de 100 fonctions!

16

IV - Les tableaux (array)

2 catégories principales de tableau:

1. Tableau indexé numériquement

```
$tab = array('pomme','poire','pêche');
$tab[0]= 'pomme'; $tab[1]= 'poire'; $tab[2]= 'pêche'
$tab[]= 'pomme'; $tab[]= 'poire'; $tab[]= 'pêche';
echo $tab[1] ???
```

2. Tableau associatif : associe une chaîne de caractère a un élément

```
$tab = array (
    'prenom'      => 'toto',
    'age'         => 10,
    'note'        => 0
);
echo $tab['prenom']." a eu la note ".$tab['note'];  ???
```

17

Traitements des tableaux

- Afficher un tableau:

```
print_r($nomTableau); // ou var_dump
```

- Taille d'un tableau:

```
$n=count($nomTableau); // ou sizeof
```

- Convertir une chaîne en tableau (et inversement)

```
$chaine = "Paul,Dupont,3 rue des prés,38999,Chaille";
$nomTableau = explode(",",$chaine);
(implode)
```

- Parcourir un tableau (foreach, while, for):

```
foreach ($nomTableau as $clef => $valeur)
    {
    echo $clef.' : '.$valeur.'<br />';
    }
```

Et beaucoup d'autres choses!

⇒ notion de tableau très riche en PHP

18

Exemple de parcours d'un tableau associatif

```
$tab = array (
   'prenom' => 'toto',
   'age'  => 10,
   'note' => 0);

   foreach ($tab as $clef => $valeur)
      {
      echo $clef.' : '.$valeur.'<br />';
      }
```

???

19

V - Variables prédéfinies PHP

- Il existe des variables prédéfinies, toujours accessibles (en anglais : ***superglobals***)
- Ce sont des tableaux qui contiennent des informations sur le contexte de communication entre le client et le serveur

$_GET	
$_POST	$_SERVER
$_COOKIE	$_ENV
$_REQUEST	$GLOBALS
$_FILES	
$_SESSION	

20

Variables prédéfinies PHP

$_GET : Contient la liste des variables transmises via la *méthode GET dans un formulaire*

$_POST : Contient la liste des variables transmises via la *méthode POST dans un formulaire*

$_COOKIE : Contient la liste des variables transmises par le navigateur via les cookies

$_REQUEST : Ce tableau agrège $_POST $_GET $_COOKIE en un seul tableau.

21

Variables prédéfinies PHP

$_FILES : Dans un formulaire, il est possible de télécharger des fichiers. Ce tableau décrits ces fichiers.

$_SESSION : Ce tableau permet de manipuler directement les données de session utilisateur

$_SERVER : Permet de connaître le détail de la requête en cours (nom, chemin de la page,…) et les éléments spécifiques au serveur web (nom, version) et à la connexion (IP paramètres du navigateur,…)

$_ENV : Contient les variables de l'environnement d'exécution de l'interpréteur PHP

$GLOBALS : Contient l'ensemble des variables globales définies.

22

VI - Traitement d'un formulaire

```
<form method= "get" action="traite.php">
    <fieldset>
        <legend>Vos coordonnées</legend>

        <label for="nom">Votre nom: </label>
        <input type="text" name="nm" id="nom" />
        <br />

        <label for="prenom">Votre prénom: </label>
        <input type="text" name="pn" id="prenom" />
        <br />

    </fieldset>
    <p>
        <input type="submit" value=  "Envoyez" />
        <input type="reset" value="Raz" />
    </p>
</form>
```

traite.php : principe

```
<!DOCTYPE html>
<html lang="fr">
<head>
        <meta charset="utf-8">
        <title> essai PHP </title>
        <link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
…/...
<?php
        $nom = $_GET['nm'];
        $prenom= $_GET['pn'];
        …/..
?>
…/...
</body>
</html>
```

24

Extrait de traite.php

```php
<?php
        $nom = $_GET['nm'] ;
        $prenom = $_GET['pn'] ;
        if ( !empty($nom) && !empty($prenom) )  {
                echo "Bonjour $prenom $nom";
        } else {
                echo 'Bonjour toi !';
        }
?>
```

25

Sécurité XSS

```php
<body>
<p>
<?php
  $nom = htmlentities($_GET['nm']);
  $prenom=htmlentities($_GET['pn']);
   if (  !empty($nom) && !empty($prenom) )  {
      echo "Bonjour $prenom $nom";
   } else {
     echo 'Bonjour toi !';
}
?>
</p>
</body>
</html>
```

Faille de sécurité PHP
XSS (cross-site scripting)

La fonction ***htmlentities()*** remplace tous les caractères possibles en leur équivalent HTML:

- & (ampersand) devient &
- " (double quote) devient "
- ' (single quote) devient '
- < (less than) devient <
- > (greater than) devient >
- ...

26

Programme générique de traitement

1. On sait que la méthode est "GET" :

On souhaite afficher toutes les variables/valeurs; on parcourt le tableau associatif $_GET

```
<?php
echo "<ul>";
        foreach($_GET as $key => $value) {
        echo "<li> $key: ".htmlentities($value)."</li>";
  }
}
echo "</ul>";
?>
```

27

Programme générique de traitement

2. On sait que la méthode est "POST" :

On souhaite afficher toutes les variables/valeurs; on parcourt le tableau associatif $_POST

```
<?php
echo "<ul>";
        foreach($_POST as $key => $value) {
        echo "<li> $key: ".htmlentities($value)."</li>";
  }
}
echo "</ul>";
?>
```

28

Programme générique de traitement

3. Un programme générique:

```
<?php
echo "<ul>";
If ( ! empty($_POST) ) {
        foreach($_POST as $key => $value) {
        echo "<li> $key: ".htmlentities($value)."</li>";}
}
else {
        foreach($_GET as $key => $value) {
        echo "<li>$key: ".htmlentities($value)."</li>";}
}
echo "</ul>";
?>
```

29

Travail personnel

Cherchez comment envoyer un mail en PHP

30