Programmation Web avancée
Introduction à PHP

Thierry Hamon

Bureau H202 - Institut Galilée
Tél. : 33 1.48.38.35.53
Bureau 150 – LIM&BIO – EA 3969
Université Paris 13 - UFR Léonard de Vinci
74, rue Marcel Cachin, F-93017 Bobigny cedex
Tél. : 33 1.48.38.73.07, Fax. : 33 1.48.38.73.55
thierry.hamon@univ-paris13.fr
http://http://www-limbio.smbh.univ-paris13.fr/membres/hamon/PWA-20112012

Introduction

PHP :

- Définition de pages Web dynamiques
- Langage interprété
- Scripts PHP inclus dans les pages Web
- Scripts interprétés coté serveur
- Envoi du résultat (HTML, CSS, Javascript, etc.) de l'interprétation par le serveur au client

Introduction

Disponible sur de nombreuses plateformes (Windows, Linux, BSD, Mac OS X, etc)
Intégré comme module dans Apache et IIS
Propose des supports pour

- les bases de données (Oracle, MySql, Postgresql, etc.)
- les fonctions Web (cookies, authentification, session, etc.)
- des librairies variées (manipulation du XML, LDAP, PDF, etc.)

Exemple

Page Web contenant du script PHP

coté serveur :

```
<html>
  <body>
  <?php
     echo "Hello World";
   ?>
  </body>
</html>
```

résultat coté client

```
<html>
  <body>
Hello World
  </body>
</html>
```

Inclusion de fichiers externes

- `include(fichier)` : inclusion d'un fichier (HTML ou PHP) au moment où l'instruction est évaluée
- `require(fichier)` : inclusion d'un fichier (HTML ou PHP) au moment où le fichier PHP est chargé
- `include_once()` et `require_once()` : idem mais le fichier n'est pas inclus s'il a déjà été inclus précédemment

Exemple

```
<html>
   <body>
   <?php include("fichierHW.php");
   ?>
   </body>
</html>

<?php
      echo "Hello World";
?>
```

Commentaires

Commentaires : syntaxe du C, C++ et shell

- `// commentaires`
- ```
  /*
     commentaires
   */
  ```
- `# commentaires`
  ```

Casse et instructions

- Pas de prise en compte de la casse dans le nom de fonction

```
echo "bonjour";

ECHO "bonjour";
```

- ATTENTION : Prise en compte de la casse dans le nom des variables

```
// deux variables différentes

echo $nom

echo $NOM
```

Instructions : terminées par un point-virgule

Variables

- Nommage : `$nom`
  Le nom de la variable est précédé de $
- Langage faiblement typé
  l'affectation définit le type de la variable
- Initialisation/utilisation standard :

Commentaires : `// commentaires`

Variables

- Nommage : `$nom`
- Langage faiblement typé
- Initialisation/utilisation standard

```
$i = $i * 4;

$i++;

echo $i;
```

Types supportés

Booléens, entiers et flottants

- Booléens : `true` / `false` (valeurs 0, chaîne de caractère, tableau, objet vides, constante NULL)
- Entiers positifs ou négatifs

```
$j = 1;
$k = -5;
$l = 010;
$m = 0xA3;
```

Remarque : division entière après transtypage (cast)

```
$i = (int) (11/4)
```

- Flottants

```
$f = -2.6; $f = 2.1e-3;
```

Opérateurs

Affectation : =
Opérateurs arithmétiques : standard (+, -, *, /, ++, ...)
Opérateurs de comparaison : standard (<, >, <=, ...)
Opérateurs de concaténation : .

Types supportés

Chaînes de caractères

- Définition à l'aide simples (pas de substitution de variables) ou double quote (substitution de variables) :

```
$str1 = hello';

$str2 = "$str1 world\n";
```

- Protection de nom de variables :

```
$str1 = hellp';

echo  "$str1world\n";

echo "${str1}world\n";
```

- `echo "\n";` : un saut de ligne dans le code HTML généré
  MAIS
  Pas de saut de ligne HTML (nécessite `<br/>`)

Chaînes de caractères

conversion en nombre

Deux règles :

- Conversion en flottant si elle contient `.`, `e` ou `E`
  sinon conversion en entier

```
$i = 1 + "0.1"; // contient 1.1
$i = 1 + "1E2"; // contient 101
```

- Définition de la valeur par la première partie de la chaîne (0 si c'est du texte)

```
$i = 1 + "toto"; // contient 1
$i = 1 + "2 + toto"; // contient 3
$i = 1 + "toto + 2"; // contient 1
```

Chaînes de caractères

Fonctions de manipulation

- Affichage de chaînes :
  - `echo`, `print`, `printf`

```
echo "variable $i\n";
printf ("variable %d\n", $i);
```

  - Affichage de données complexes (tableaux) `print_r` (contenu) et `var_dump` (structure et contenu)

```
print_r ($a);

var_dump($a);
```

- Accès aux caractères :

```
$str = "Bonjour\n";
echo $str{4};
```

Chaînes de caractères

Fonctions de manipulation

- Comparaison de chaînes :`strcmp($str1, $str2)`
- Longueur d'une chaîne : `strlen($str)`
- `addslashes($str)` et `stripslashes($str)` : déspécialisation des caractères spéciaux dans la chaîne courte, et opération inverse
- Substitution :
  `str_replace($substr, $replace_str, $str)` :
  remplacement dans `$str`, toutes les occurrences de la chaîne `substr`.

```
$str = "Bonjour\n";
echo str_replace("jour", "soir", $str);
```

Chaînes de caractères

Fonctions de manipulation

- Décomposition d'une chaîne
  - `explode($sep, $str)` (voir également `split`) : décomposition d'une chaîne (`$str`) en fonction d'un séparateur `$sep`
  - `implode($sep, $tab)` ou `join()` : (fonction inverse) concaténation des éléments du tableau `tab`, séparés par la chaîne `$sep`.
  - `chop($str)`, `rtrim($str)`, `ttrim($str)`, `ltrim($str)` : suppression des caractères d'espacement dans `$str`
  - `parse_str($str)` : Analyse de la chaîne type `QUERY_STRING` et retourne des couples nom/valeur
- ...

Expressions régulières (1)

Définition d'un patron plus ou moins déterminé
Caractères spéciaux :

.	n'importe quel caractère (joker)
$	fin de ligne
^	début de ligne
\	dé-spécialisation
[ ]	groupement exclusif
[^]	négation du regroupement
(|)	disjonction sur plusieurs caractères

Expressions régulières (2)

?	0 ou 1 fois
+	1à n fois
*	0 à n fois
{n}	nombre précis d'occurrences
-	plage de caractères

Utilisation des expressions régulières

- `ereg($regexp, $str, $tab)` : recherche des chaînes correspondant à `$regexp` dans `$str` (le tableau `$tab` contient le résultat)

  `eregi($regexp, $str, $tab)` : idem mais insensible à la casse
- `ereg_replace($regexp, $replace, $str)` : recherche de la chaîne correspondant à `$regexp` dans `$str` et replacement par `$replace`.

  Retour : la chaîne modifiée

  `eregi_replace($regexp, $replace, $str)` : idem mais insensible à la casse

Utilisation des expressions régulières

- `split($regexp, $str)` : décomposition d'une chaîne en éléments (stockés dans un tableau) en fonction d'une expression régulière `$regexp`

  `spliti($regexp, $str)` : idem mais insensible à la casse

Tableaux

- Tableaux associatifs
  ensemble de couples clés/valeurs
- la clé peut être un entier ou une chaîne de caractères
- Création :
  - par affectation de valeurs dans le tableau
  - en utilisant la fonction `array()`

```
$tab[0] = "aa";
$tab[1] = 2;

$tab2 = array("a" => 97, "b" => 98, c => "99");
```

Notion de pointeur de tableau

Un pointeur interne est associé à chaque tableau
$\rightarrow$ Référence de l'élément courant
Fonctions associées :

- `current($tab)` : : l'élément courant
- `next($tab)` : positionnement du pointeur sur l'élément suivant
- `prev($tab)` : positionnement du pointeur sur l'élément précédent
- `end($tab)` : positionnement du pointeur sur le dernier élément
- `reset($tab)` : positionnement le pointeur sur le premier élément

Notion de pointeur de tableau

exemple

```
$tab2 = array("a" => 97, "b" => 98, c => "99");

echo current($tab2);
echo next($tab2);
echo end($tab2);
echo reset($tab2);
```

Manipulation des tableaux

- `sizeof($tab)` ou `count($tab)` : Nombre d'éléments d'un tableau
- `key($tab)` : extraction de la clé de l'élément pointé par le pointeur interne du tableau

```
$tab = array("a" => 97, "b" => 98, c => "99");
next($tab);
echo key($tab);
```

- `extract($tab)` : extraction des valeurs d´un tableau
chaque valeur est recopiée dans une variable ayant pour nom la valeur de la clé

```
$tab = array("a" => 97, "b" => 98, c => "99");
extract($tab);
echo "$a $b $c\n";
```

Manipulation des tableaux

- `list($tab)` : extraction des valeurs d'un tableau (association de variables à des éléments d'un tableau)

```
$tab = array("a" => 97, "b" => 98, c => "99");
list($v1, $v2) = $tab;
```

- `each($tab)` : extraction du couple clé/valeur courant (pointé par le pointeur interne du tableau)
  NB : le couple est retourné dans un tableau de 4 éléments : `0=>clé, 1=>valeur, key=>clé et value=>valeur`

```
$tab = array("a" => 97, "b" => 98, c => "99");
$tab2 = each($tab);
echo "$tab2[0] $tab2[1] $tab2[key] $tab2[value]";
```

Manipulation des tableaux

- `foreach()` : parcours de l'ensemble des éléments du tableau (en fait un corpie du tableau)

```
$tab = array("a" => 97, "b" => 98, c => "99");
foreach($tab as $v) {
   echo "$v<p>\n";
}
```

- `unset($tab)` : Suppression d'un élément

```
$tab = array("a" => 97, "b" => 98, c => "99");
unset($tab["b"]);
```

Manipulation des tableaux

- Tris de tableaux (sur les clés ou les valeurs)
  - `sort($tab)` : tri du tableau par ordre croissant de valeurs et réassignation des clés
    NB : perte de l'assocation clé/valeur
  - `asort($tab)` / `arsort($tab)` : tri par ordre croissant / décroissant de valeurs
  - `ksort($tab)` / `krsort($tab)` : tri par ordre croissant / décroissant de clés
  - `uasort($tab)` / `uksort($tab)` / `usort($tab)` : idem mais il faut fournir la fonction de comparaison des éléments

Structures de contrôle

- choix : `if () { } else { }`

```
if ($i == 9) {
    echo $i;
}
```

- boucle Pour : `for(;;) { }`

```
for($i=0; $i < 10; $i++) {
    echo "$i"
}
```

- boucle tant que : `while(test) {}`

```
$i = 0;
while($i < 10) {
   $i++;
    echo $i;
}
```

Fonctions

- Définition à n'importe quel endroit du fichier PHP
- Introduite par le mot clé `function`
- Valeur de retour : n'importe quel type de variable (tableau, chaîne de caractères, objet, etc.)
- Arguments :
  - passage par valeur : mention de la variable
  - passage par référence : mention de la variable préfixé par & (dans la définition)

Fonctions

exemples

```
function printInfoStr($str) {
  print "chaine: $str<p>\n";
  print "taille chaine: " . strlen($str) . "<P>\n";

  return(strlen($str));
}

$ch = "Hello World";
$taillestr = printInfoStr($ch);

function printInfoStrRef(&$str) {
  print "chaine: $str<p>\n";
  print "taille chaine: " . strlen($str) . "<P>\n";
  $str = "taille($str) = " . strlen($str);

  return(strlen($str));
}
```

Fonctions

(suite)

- Valeur par défaut d'un argument :
  NB : les argments avec valeur par défaut doivent être placés après les arguments sans valeur par défaut

```
function printInfoStr($e, $str = "bonjour") {
```

- Définition d'un nombre variable d'arguments
  Fonctions associées pour la gestion de la liste des arguments :
  - `func_num_args()` : nombre d'arguments passé à la fonction
  - `func_get_arg($n)` : retoune l'argument à la $n^{ème}$ position la liste des arguments
  - `func_get_args()` : tableau contenant la liste des arguments

Fonctions

(suite)

- Récupération de la référence de la variable retournée par une fonction

```
function printInfoStr($str) {
   print "chaine: $str<p>\n";
   print "taille chaine: " . strlen($str) . "<P>\n

   $t = strlen($str);

   return($t);
}

$ch = "Hello World";
$taillestr =& printInfoStr($ch);
```

Fonctions

(suite)

- Pointeurs de fonction (?)

```
function printInfoStr($str = "bonjour") {
   print "chaine: $str<p>\n";
}

$fonction = 'printInfoStr';
$fonction();
$fonction("hello");
```

Programmation objet

Remarques sur les objets :

- Les attributs et méthodes sont publics
- Pas de destructeurs
- Libération automatique des objets s'ils ne sont pas utilisés

Programmation objet

- Définition d'une classe (introduite par le mot clé `class`)

```
class etudiant {
  // attributs
  var $nom;
  var $prenom;
  var $initiales;

  // contructeur
  function etudiant() {
    $this->prenom = $_GET['prenom'];
    $this->nom = $_GET['nom'];

    $this->calcul_initiales();
  }

  function calcul_initiales() {
    $this->initiales = $this->nom{0} . $this->prenom{0};
  }

}
```

Programmation objet

- Création d'un objet :

```
$etudiant1 = new Etudiant;
```

- Accès aux méthodes et attributs

```
echo "initiales: " . $etudiant1->initiales . "<p>";
```

Programmation objet

- Héritage de classe

```
class INFO3 extends Etudiant {
  var $ecole = "Sup'Galilée";
  var $stage;

  function set_stage () {
    $this->stage = $_GET['stage'];
  }
}
$info31 = new INFO3();

$info31->set_stage();
echo "Stage: " . $info31->stage . "<p>";
```

XML, XSLT

- Chargement d'un fichier XML : `DOMDocument()`, `load()`

```
<?php
    // Creation du document XML (vide)
    $xmlDoc = new DOMDocument();
    // Chargement d'un document XML
    $xmlDoc->load("note.xml");
?>
```

- Transformation XSL : `XSLTProcessor()`, `importStyleSheet()`, `transformToXML()`

```
<?php
// Creation du processeur XSLT
$proc = new XSLTProcessor;

// Chargement des règles de transformation XSL
$proc->importStyleSheet($xsltDom);
```

Pour aller plus loin

- Plusieurs bibliothèques/classes : gestion de la connexion MySQL, XML, etc.
- Documentation en ligne : `http://www.manuelphp.com/`, `http://www.php.net/manual/en/index.php`