Web (Persistance)

Andrea G. B. Tettamanzi
Université de Nice Sophia Antipolis
Département Informatique
andrea.tettamanzi@unice.fr

CM - Séance 5

Introduction à PHP

Plan

- Un peu d'histoire
- Caractéristiques du langage
- Styles de programmation
- Éléments du langage

Avant PHP : CGI

- Au commencement était la Common Gateway Interface (CGI)
- CGI fournit une interface entre un serveur Web et un programme (binaire exécutable) qui génère du contenu Web
- Le serveur Web passe au programme un certain nombre de variables d'environnement, dont REQUEST_METHOD
- Avec la méthode GET,
  - les paramètres de l'URL sont passés au programme dans la variable QUERY_STRING
- Avec la méthode POST, le contenu posté est passé par « stdin »
- Le « stdout » du programme est renvoyé au navigateur comme réponse (page HTML dynamique)

Histoire de PHP

- Créé en 1994 par Rasmus Lerdorf pour sa page Web
- PHP Tools (Personal Home Page Tools)
- Originairement un simple jeu de binaires CGI écrits en langage C
- De plus en plus de fonctionnalités sont ensuite ajoutées
- En juin 1995 Rasmus fourni le code source au grand publique
- ~~PHP~~ → FI (Forms Interpreter), syntaxe à la Perl
- Octobre 1995 : FI → PHP Construction Kit, syntaxe à la C
- Support cookies, BD, fonctions utilisateur
- En 1998, 1 % des serveurs Web avait PHP installé
- PHP 3.0 (PHP Hypertext Processor), avec Gutmans et Suraski
- En janvier 2013, 39 % (2,1 millions) des hôtes utilisait PHP.

Caractéristiques du langage

- PHP est un préprocesseur de pages HTML
- Typage dynamique
- Orienté objet
- Interprété côté serveur
- Nécessite :
  - D'un serveur Web avec support de PHP activé
  - Les fichier avec extension .php sont traités par PHP
  - Les fichiers .php sont comme des fichiers HTML avec en plus des balises « magiques »
  - Pour développer en locale : Apache + PHP + mySQL
- Génération de contenu HTML par « echo » (ou « print »)

Hello World en PHP

```
<html>
 <head>
  <title>Test PHP</title>
 </head>
 <body>
 <?php echo '<p>Bonjour le monde</p>'; ?>
 </body>
</html>
```

Syntaxe

- Entre le C/Java et le Shell Script
  - « ; » comme délimiteur
  - commentaires
    - comme en java/C/C++, entre les signes /* et */
    - comme en java/C/C++, en commençant une ligne par //
    - comme en shell Unix, avec #
- Balises d'ouverture et fermeture PHP
  - <?php …?> (recommandées)
  - <script language="php"> … </script>
  - <? … ?> (« balises courtes » déconseillées)
  - <% … %> (« balises ASP », également déconseillées

Styles de programmation

- On peut distinguer deux styles de programmation en PHP
  - Côté obscur : echo au fur et à mesure
  - Côté lumineux : calcul, mise en forme, puis echo
- Idée de fond : séparation des aspects
  - Code PHP (calcul)
  - Balises HTML (structure du document)

Exemple – style côté obscur

```
<HTML><HEAD>
<TITLE>HTML avec
PHP</TITLE></HEAD>
<BODY>
  <H1>HTML + PHP</H1>
  <p>Nous sommes le
  <?php echo date
        ("j/m/Y"); ?>
</p>
</BODY></HTML>
```

```
<HTML><HEAD>
<TITLE>HTML avec
PHP</TITLE></HEAD>
<BODY>
  <H1>HTML + PHP</H1>
  <p>Nous sommes le
     08/10/2014</p>
</BODY></HTML>
```

Exemple – style côté lumineux

```
<?php //calcul préalable
$date = "<p>Nous sommes
le ".date("j/m/Y").
"</p>"; ?>
<HTML><HEAD>
<TITLE>HTML avec
PHP</TITLE></HEAD>
<BODY>
  <H1>HTML + PHP</H1>
  <?php echo $date; ?>
</BODY></HTML>
```

```
<HTML><HEAD>
<TITLE>HTML avec
PHP</TITLE></HEAD>
<BODY>
  <H1>HTML + PHP</H1>
  <p>Nous sommes le
     08/10/2014</p>
</BODY></HTML>
```

Fonction « date »

- **string date (string format [, int timestamp])**
  - renvoie une date sous forme d'une chaîne, au format donné par la chaîne format. La date est fournie par le paramètre timestamp (un entier), sous la forme d'un timestamp. Par défaut, la date courante est utilisée.

```
<?php // Aujourd'hui, le 12 avril 2006, 10:16:18 am
$aujourdhui = date("F j, Y, g:i a"); // April 12, 2006, 10:16 am
$aujourdhui = date("m.d.y");         // 04.12.06
$aujourdhui = date("j, m, Y");       // 12, 04, 2006
$aujourdhui = date("Ymd");           // 20060412
$aujourdhui = date('\C\'\e\s\t\ \l\e\ jS \j\o\u\r\.');
                                     // C'est le 12th jour.
$aujourdhui = date("D M j G:i:s T Y"); // Wen Apr 12 10:16:18 Paris 2006
$aujourdhui = date("H:i:s");         // 10:16:18
// notation française
$aujourdhui = date("d/m/y");         // 12/04/06
$aujourdhui = date("d/m/Y");         // 12/04/2006
?>
```

Types basiques

- PHP supporte 8 types basiques :
- 4 types scalaires
  - « boolean »
  - « integer »
  - « float » : nombre à virgule flottante (double précision)
  - « string » : chaîne de caractères
- 2 types composés
  - « array »
  - « object »
- 2 types spéciaux
  - « resource », « NULL »

Types

- Le type d'une variable est défini par PHP au moment de l'exécution selon le contexte d'utilisation (typage dynamique)
- « var_dump() » pour vérifier le type d'une expression
- « gettype() » représentation humainement lisible d'un type
- « is_*T*() » : vrai si l'argument est du type *T* :
  - « is_int() », « is_string() », etc.
- Le type d'une variable peut être forcé
  - Avec un transtypage (cast) explicite : (*type*) *variable*
  - Avec la fonction « settype(var, string type) »

Variables

- Représentées par un signe $ suivi du nom de la variable
- Le nom est sensible à la casse
- Les noms de variables suivent les mêmes règles de nommage que les autres entités PHP (outre noms réservés) :
  - Doivent commencer par une lettre ou un souligné _
  - Peuvent contenir des lettres, chiffres ou soulignés
- $this est une variable spéciale qui ne peut pas être assignée
- Le variables sont assignées « par valeur »
- Assignation « par référence » en ajoutant & avant le nom de la variable source
- Variables non assignées ont une valeur par défaut selon leur type

Variables prédéfinies

- PHP fournit un grand nombre de variables pré-définies
- Elles dépendent
  - du serveur sur lequel elles tournent
  - de la version et de la configuration du serveur
  - ou encore d'autres facteurs.
- Exemples :
  - $_SERVER['DOCUMENT_ROOT']
  - $_ENV['HOME']
  - $php_errormsg
  - $argc, $argv

Constantes

- Une constante est un identifiant qui représente une valeur simple
- Cette valeur ne peut jamais être modifiée durant l'exécution
- Par convention, noms de constantes toujours en majuscules
- PHP fournit un grand nombre de « constantes magiques » (qui ne sont pas vraiment des constantes, mais des fonctions...) :
  - __LINE__ : la ligne courante dans le fichier
  - __FILE__ : le chemin complet et le nom du fichier courant
  - __DIR__ : le dossier du fichier courant
  - __FUNCTION__ : le nom de la fonction
  - __CLASS__ : le nom de la classe courante
  - ...

Types scalaires

- Booléens
  - constantes TRUE et FALSE
  - Les constantes 0, 0.0, “”, “0”, NULL, le tableau vide et l'objet vide sont considérés équivalents à FALSE
- Entiers et réels
  - $i = 1; // Entier en notation décimale
  - $f = 3.14116 // Flottant
  - $f = 0.3e-3 // soit 0,0003
  - Conversion :
    - Cast : (int) ou (integer)
    - Fonction intval()

Chaînes de caractères

- Encadrées par des guillemets simples (')
  - Ne contiennent ni variables, ni caractères d'échappement (comme "\n")
  - Acceptent les sauts de lignes

```
'C\'est une chaîne avec guillemets simples
et un saut de ligne.'
```

- Encadrées par des guillemets doubles (")
  - Peuvent contenir des variables remplacées par leur valeur à l'exécution, des caractères d'échappement

Caractères d'échappement

- \n Saut de ligne
- \r Retour chariot
- \t Tabulation
- \\ Le signe ’\’
- \$ Le signe ’$’
- \" Un guillemet double
- \0nn Un caractère en octal
- \xnn Un caractère en hexadécimal

Variables dans les chaînes de caractères

```php
<?php
$boisson = 'vin';
// Correct, car "," n'est pas autorisé dans
// les noms de variables
echo "Du $boisson, du pain et du fromage!";
// Pas correct, car 's' peut faire partie d'un nom
// de variable et PHP recherchera alors $boissons
echo "Il a goûté plusieurs $boissons";
// Correct :
echo "Il a goûté plusieurs {$boisson} s";
?>
```

Chaînes de caractères

- Transtypage explicite : (string)
- Fonction strval( )
- Conversion automatique (cas d'echo)
- Un tableau sera converti en « Array » :
  - utilisation de la fonction var_dump( )
  - print_r( )

- $chaine{$i} : le (i+1) ième caractère

Tableaux

- PHP gère dynamiquement la taille des tableaux
- Les tableaux en PHP peuvent être soit indicés soit associatifs.
- Tableaux indicés : quelques exemples.
  - $tab[0] = " élément 1 "; $tab[1] = "élément 2 "; $tab[2] = 120;
- Une caractéristique importante et très utile : PHP affecte automatiquement un indice à un nouvel élément du tableau.
- Cet indice est le numéro de la première cellule vide. Donc le code ci-dessous est équivalent au précédent.
  - $tab[] = "élément 1 "; // $tab[0] ! ;
  - $tab[] = "élément 2 "; // $tab[1] !
  - $tab[] = 120; // $tab[2] !

Tableaux

- L'instruction array permet d'initialiser facilement un teblea
  - $tab = array ("élément 1 ", "élément 1 ", 120);
- Tableaux associatifs : indexation par clef
- Définition « classique » :
  - $réalisateur["Vertigo"] = "Hitchcock";
  - $réalisateur["Sacrifice"] = "Tarkovski";
  - $réalisateur["Alien"] = "Scott";
- Définition en utilisant « array » :
  - $réalisateur = array ("Vertigo" => "Hitchcock", "Sacrifice" => "Tarkovski", "Alien" => "Scott") ;
- « unset() » efface un élément (une association clef-valeur)

Opérateurs arithmétiques

- $a + $b //Addition de $a et $b
- $a - $b //Soustraction de $b `a $a
- $a * $b //Multiplication de $a et $b
- $a / $b //Division de $a par $b
- $a % $b //$a modulo $b (reste de la division de $a par $b)
- $i++; // incrémenter $i (vaut 5 si son ancienne valeur est 4)
- $j = ++$i; // incrémenter $i puis affecter cette valeur à $j
- $k = $i++; // affecter la valeur de $i à $k puis incrémenter $i
- $k--; // décrémenter $k

Opérateurs sur les chaînes

- $c1 = "Bonjour " ;
- $c2 = " le monde";
- $c = $c1 . " tout " . $c2 ; // → “Bonjour tout le monde” dans $c
- $c .= " ! "; //donne « Bonjour tout le monde !» dans $c

Opérateurs logiques et de comparaison

- $a && $b; $a and $b; // conjonction logique
- $a || $b ; $a or $b ; // disjonction logique
- $a xor $b // OU exclusif
- !$a // négation logique
- $a == $b // Vrai si $a est égal à $b.
- $a != $b // Vrai si $a est différent de $b.
- $a < $b // Vrai si $a est inférieur à $b.
- $a > $b // Vrai si $a est supérieur à $b.
- $a <= $b // Vrai si $a est inférieur ou égal à $b.
- $a >= $b // Vrai si $a est supérieur ou égal à $b

Conditionnels

- if(expression) { … }
- if(expression) { … } else { … }
- if(expression) { … } elseif(expression) { … } … else { … }

- if(expression) : elseif(expression) : else :
- endif;

- switch(expression) { … }
  - case valeur :
  - break;
  - default;

Boucles

- while(expression) { … }
- do { … } while(expression} ;
- for($x = 0; $x <10; $x++) { … }
- foreach(array_expression as $value) { … }
- foreach(array_expression as $key => $value) { … }

- break ;
- continue ;

Inclusion d'autres fichiers

- require( ) et include( ) incluent et exécutent un fichier PHP.
  - La commande require() se remplace elle-même par le contenu du fichier spécifié
  - require( ) et include( ) sont identiques, sauf dans leur façon de gérer les erreurs. include( ) produite une Alerte (warning) tandis que require( ) génère une erreur fatale. Notamment lorsque le fichier manque.
- require_once( ) et include_once( )
  - La principale différence est qu'avec require_once( ), vous êtes assurés que ce code ne sera ajouté qu'une seule fois, évitant de ce fait les redéfinitions de variables ou de fonctions, génératrices d'alertes.

Fonctions

- Les fonctions n'ont pas besoin d'être définies avant d'être utilisées, *sauf* lorsqu'une fonction est définie conditionnellement
- Le nom d'une fonction ne peut pas commencer par '$', et n'est pas sensible à la casse.
- Syntaxe :

```php
<?php
function foo($arg_1, $arg_2, /* ..., */ $arg_n)
{
    echo "Exemple de fonction.\n";
    return $retval;
}
?>
```

- Toutes les fonctions en PHP ont une portée globale
- Les variables à l'intérieur des fonctions ont une portée locale

Passage d'arguments

- Par valeur (comportement par défaut)
- Par référence, en faisant précéder une variable par « & »
- Valeur par défaut
  - En faisant suivre une variable par « = <valeur scalaire> »
  - Utilisation de tableaux possible comme valeurs par défaut
  - La valeur par défaut doit obligatoirement être une constante
  - Les arguments sans valeur par défaut doivent être en 1ers
- Nombre d'arguments variable
  - Mot clé « ... » suivi d'un nom de variable.
  - Les arguments seront passés dans la variable fournie sous forme d'un tableau

Return

- Si appelée depuis une fonction, la commande return() termine immédiatement la fonction, et renvoie l'argument qui lui est passé.
- Si appelée depuis l'environnement global, l'exécution du script est interrompue.
  - Si le script courant était include( ) ou require( ), alors le contrôle est rendu au script appelant, et la valeur renvoyée sera utilisée comme résultat de la fonction include( ).
  - Si return( ) est appelée depuis le script principal, alors l'exécution du script s'arrête.

Merci de votre attention