SIN6U5 : Développement web 2
(Prog. web coté serveur)
Le langage PHP

Luigi Santocanale
LIF, Aix-Marseille Université

Transparents basés sur le cours de Bertrand Estellon

24 janvier 2018

Plan

Plan

Le langage PHP

- PHP pour `Personal Home Page (tools)/(Personal) Hypertext Preprocessor`
- Langage de programmation libre ;
- utilisé pour produire des pages dynamiques ;
- compilé et executé à la volée ;
- typage dynamique (faible) ;
- créé en 1994 par Rasmus Lerdorf.

```
<html>
  <body>
    Deux plus deux = <?php echo 2+2; ?>
  </body>
</html>
```

Interprétation d'une page

Contenu du fichier `test.php` présent dans le répertoire courant :

```
<html>
  <body>
    Deux plus deux = <?php echo 2+2; ?>
  </body>
</html>
```

La commande `php test.php` produit la sortie suivante :

```
<html>
  <body>
    Deux plus deux = 4
  </body>
</html>
```

*Pourquoi utiliser **PHP** ?*

1

R. Le Breton :

- PHP se démarque de ses concurrents par une importante communauté qui peut vous aider (blogs, packages, . . .)
- C'est un langage facile à utiliser, idéal pour les débutants . . .
- . . . comme pour les professionnels (Wikipédia, Yahoo et Facebook).

1. Source : http://blog.stoneriverelearning.com/top-5-programming-languages-used-in-web-development/

Historique

1994, PHP Tools : créé par Rasmus Lerdorf, jeu de binaires CGI écrits en langage C, pour son CV en ligne. Suite de scripts "Personal Home Page Tools", Outils pour page personnelle, en francais.

1997, PHP 3 : renommé PHP : Hypertext Preprocessor ; collab. Andy Gutmans et Zeev Suraski, avec Lerdorf ; capacités d'extension par modules ;

1998, PHP 4 : nouveau moteur, le 'Zend Engine' ;

2004, PHP 5.0 : objets ; typage explicite des paramètres des fonctions ;

2009, PHP 5.3 : fonctions anonymes (prog. fonctionnelle) ; espaces de noms ;

2012, PHP 5.4 : traits ;

2017, PHP 7.0 : performance améliorée (deux fois plus vite que PHP 5.6) ; typage explicite des valeurs de retour des fonctions.

Remarque

Le but du cours

- n'est pas maîtriser **PHP**,
- mais plutôt d'arriver à s'en servir de façon intelligente.

Plan

Développement en local avec PHP

Lancement d'un serveur local interprétant du PHP :

```
php -S localhost 8080
```

Contenu du fichier `test.php` présent dans le répertoire courant :

```
<html>
  <body>
    Deux plus deux = <?php echo 2+2; ?>
  </body>
</html>
```

URL pour accéder à la page test.php :

```
http://localhost:8080/test.php
```

Les variables

- En C ou en Java, une variable est définie par :
  - Un nom (ou identifiant)
  - Un type
  - Une zone mémoire (désignée par une adresse)

```
int myInteger;
myInteger = 2;
```

- En PHP, une variable est définie par :
  - Un nom (ou identifiant) commençant par un $
  - Un conteneur de valeur

```
<?php
$my_integer = 2;
?>
```

Les types des valeurs

- Les variables ne sont pas typées mais les valeurs ont un type :
  - integer : 7, 14, 255, 0xFF
  - boolean : TRUE, FALSE
  - double : 1.95, 1.12e4
  - string : "bonjour", 'bonjour'
  - array : array(1,2,3)
  - object : new maclasse
  - ressource : mysql_connect('localhost', 'moi', '')
  - null : null, NULL
- L'opérateur = permet d'affecter une variable

```php
<?php
$integer_variable = 2;
var_dump($integer_variable); // affiche int(2)
?>
```

Instructions, opérations et fonctions

Construire des expressions :

```php
<?php
$radius = 10;
$perimeter = 2*3.14*$radius;
?>
```

Définir et d'utiliser des fonctions :

```php
<?php
function computePerimeter($radius) {
    return 2*3.14*$radius;
}
$radius = 10;
$perimeter = computePerimeter($radius);
?>
```

État d'une variable

- La fonction `isset($var)` retourne :
  - FALSE si la variable n'est pas initialisée ou a la valeur NULL ;
  - TRUE sinon.
- La fonction `empty($var)` retourne :
  - TRUE si une des conditions suivantes est vérifiée :
    - la variable n'est pas initialisée
    - la variable a la valeur `""` (chaîne vide)
    - la variable a la valeur 0 (entier)
    - la variable a la valeur 0.0 (ottant)
    - la variable a la valeur "0"
    - la variable a la valeur NULL
    - la variable a la valeur FALSE
    - la variable a la valeur `array()` (tableau vide)
  - FALSE sinon.
- La fonction `unset($var)` détruit une variable.

Opérateurs numériques

Négation	-$a	Opposé de $a
Addition	$a + $b	Somme de $a et $b
Soustraction	$a - $b	Différence de $a et $b
Multiplication	$a * $b	Produit de $a et $b
Division	$a / $b	Quotient de $a et $b
Modulo	$a % $b	Reste de $a divisé par $b

Pre-incrémente	++$a	Incrémente $a de 1, puis retourne $a
Post-incrémente	$a++	Retourne $a, puis incrémente $a de 1
Pre-décrémente	–$a	Décrémente $a de 1, puis retourne $a
Post-décrémente	$a–	Retourne $a, puis décrémente $a de 1

Opérateurs logiques

et	$a and $b	TRUE si $a et $b valent TRUE
ou	$a or $b	TRUE si $a ou $b valent TRUE
ou exclusif	$a xor $b	TRUE si $a ou $b est égal TRUE mais pas les deux en même temps
non	!$a	TRUE si $a n'est pas égal à TRUE
et	$a && $b	TRUE si $a et $b sont égaux TRUE
ou	$a || $b	TRUE si $a ou $b est égal à TRUE

Attention à la précédence des opérateurs :

```php
<?php
$b = false || true; // (b = (false || true))
$b = false or true; // ((b = false) or true)
$b = false && true; // (b = (false && true))
$b = false and true; // ((b = false) and true)
?>
```

Opérateurs de comparaison

égal	$a == $b	TRUE si $a est égal à $b
identique	$a === $b	TRUE si $a et $b sont égaux et ont le même type
différent	$a != $b	TRUE si $a est différent de $b
différent	$a <> $b	TRUE si $a est différent de $b
non identique	$a !== $b	TRUE si $a et $b sont différents ou n'ont pas le même type
plus petit	$a < $b	TRUE si $a est strictement plus petit que $b
plus grand	$a > $b	TRUE si $a est strictement plus grand que $b
inférieur ou égal	$a <= $b	TRUE si $a est plus petit ou égal à $b
supérieur ou égal	$a >= $b	TRUE si $a est plus grand ou égal à $b

```php
<?php
var_dump(0 == 'a'); // bool(true)
var_dump(0 === 'a'); // bool(false)
?>
```

Opérateurs d'affectation combinée

addition	$a += $b	additionne $a et $b puis affecte le résultat à $a
soustraction	$a -= $b	soustrait $a et $b puis affecte le résultat à $a
multiplication	$a *= $b	multiplie $a et $b puis affecte le résultat à $a
division	$a /= $b	divise $a et $b puis affecte le résultat à $a
modulo	$a %= $b	divise $a et $b puis affecte le reste à $a

```php
<?php
$my_integer = 13;
$my_integer += 12;
echo $my_integer; // affiche 25
?>
```

Plan

Les structures de contrôle

```php
<?php
$my_integer = 10;
while($my_integer<=200)
    $my_integer*=2;
?>
```

```php
<?php
for ($i = 0; $i < 100; $i++) {
    if (is_bad_integer($i)) continue;
    if (is_empty()) break;
    $element = get_element_at_position($i);
    echo $element;
}
?>
```

Syntaxe alternative

```php
for($j=$i=0;$i<10;$i++){
    if($i % 2 == 0){
        $j+=$i;
    }else{
        $j++;
    }
}
echo "$j\n";
```

```php
for($j=$i=0;$i<10;$i++):
    if($i % 2 == 0):
        $j+=$i;
    else:
        $j++;
    endif;
endfor;
echo "$j\n";
```

Les blocs d'instructions

```
<table>
    <?php for($i = 1; $i < 10; $i++){ ?>
        <tr>
            <?php for($j = 1; $j < 10; $j++){ ?>
                <td>
                    <?= $i*$j ?>
                </td>
            <?php  } ?>
        </tr>
    <?php } ?>
</table>
```

```
<table>
    <?php for($i = 1; $i < 10; $i++): ?>
        <tr>
            <?php for($j = 1; $j < 10; $j++): ?>
                <td>
                    <?= $i*$j ?>
                </td>
            <?php endfor; ?>
        </tr>
    <?php endfor; ?>
</table>
```

Plan

Les tableaux

- Les tableaux sont associatifs : ils associent des valeurs à des clés
- (Les clés sont des entiers ou des chaînes de caractères)
- Les valeurs sont de n'importe quel type
- Les clés ont un ordre

Création et affectation d'un tableau à une variable :

```
<?php $my_array = array('cal"=>"M2_CCI',
↪  'getdate"=>"20150908'); ?>
```

```
<?php $my_array = array(1=>true, 3=>false, 121=>true); ?>
```

Depuis la version 5.4 de PHP :

```
<?php $my_array = ['cal'=>'M2_CCI', 'getdate'=>'20150908']; ?>
```

Les tableaux

Consultation de la valeur d'un tableau :

```php
<?php
$key = 'cal';
$cal = $my_array[$key];
$getdate = $my_array['getdate'];
?>
```

Modification de la valeur associée à une clé :

```php
<?php $my_array['getdate'] = '20151009'; ?>
```

Si la clé n'existe pas, une nouvelle association est crée

Les tableaux

Nombre d'associations :

```php
<?php $number_of_pairs = count($my_array); ?>
```

Tester l'existence d'une clé :

```php
<?php
$isCalExist = array_key_exists('cal', $my_array); ?>
```

Itération sur les couples d'un tableau :

```php
<?php
foreach ($my_array as $key=>$value) echo "$key=>$value";
foreach ($my_array as $value) echo $value;
?>
```

Les tableaux

Le tableau

$$[0=>10,1=>20,2=>30]$$

peut être créé comme ceci :

```php
<?php $my_array = array(10,20,30); ?>
```

Utilisation d'un tableau comme une pile ou une file :

```php
<?php
array_push($my_array, 123); // ajoute à la fin
$value = array_pop($my_array); // retire à la fin
array_unshift($my_array, 123); // ajoute au début
$value = array_shift($my_array); // retire au début
?>
```

Les tableaux

Existence d'une valeur dans un tableau :

```php
<?php
$needle = 10;
$strict = true; /* facultatif */
$is_in_array = in_array($needle, $my_array, $strict);
?>
```

- Il existe également des fonctions pour trier un tableau, faire faire la somme de ses éléments, chercher une valeur...
- Documentation de PHP sur les tableaux :

  `http://php.net/manual/fr/ref.array.php`

Plan

La portée des variables

Par défaut, une variable est locale à l'exécution d'une fonction :

```php
<?php
function sum_recursively($array) {
    $sum = 0;
    foreach ($array as $value) {
        if (is_array($value))
            $sum += sum_recursively($value);
        else
            $sum += $value; }
    return $sum;
}
?>
```

La fonction a le comportement souhaité car la variable `$sum` est locale à chacune des ses exécutions.

Les variables globale et super-globales

Dans une fonction, il est possible de préciser qu'une variable est globale :

```php
<?php
function my_action() {
    global $database;
    $results =
        $database->query("SELECT * FROM USERS");
...
}
?>
```

Les variables super-globales sont globales par défaut :

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