Le langage PHP (bases)

Développement web
HENALLUX — IG3

Au programme…

- **Le langage PHP : langage de base**
  - Caractéristiques générales du langage
  - Données, instructions, éléments du langage, tableaux…
- **Le langage PHP : aspects orientés objets**
- **Échanges de données entre le client et le serveur**

PHP : syntaxe et sémantique

- **Données manipulées**
  - Variables, types de données, valeurs et littéraux, opérateurs
- **Instructions**
- **Fonctions**
- **Tableaux**

Ensuite : *l'orienté objet en PHP*

PHP : les fonctions

- **Déclaration de fonctions**
- **Portée des variables**
  - Variables locales et variables globales
- **Variables superglobales**

Ensuite : *les tableaux en PHP*

Définition d'une fonction

- Déclaration d'une fonction

  function *nom* **(** *[$ident [, $ident]*]* **)** **{**

  instructions

  }

- Sortie ou valeur de retour d'une fonction

  return *[expr]*

- Appel de fonction : *nom* **(** *[expr [, expr]*]* **)**

  Pas de vérification de type mais (contrairement à Javascript,) il faut passer *au moins* le nombre d'arguments de la définition !

- Les identificateurs de fonction ne sont pas sensibles à la casse.

  (Mais respectez la tout de même – Clean code !)

Fonctions anonymes

- Fonctions sous la forme d'expressions (fonctions anonymes)

  `function` **`(`** *[$ident [, $ident]*]* **`)`** **`{`** *instructions* **`}`**

  Exemple :

```
$double = function ($x) { return $x * 2; };
echo $double(3);
```

Définition de fonctions : portée

- PHP lit toutes les définitions de fonctions situées au niveau global avant de commencer l'exécution : on peut donc utiliser une fonction définie plus bas.

Exemple :

```
f(3);
function f($x) {
  echo $x*$x;
}
```

(affiche 9)

Définition de fonctions : portée

- Lorsque PHP rencontre une définition de fonction, il définit celle-ci au niveau global.

Ex :
```
function f () {
  function g() { echo 'fonction g'; }
  echo 'fonction f';
}
g();     // produit une erreur : g pas définie
f();
g();     // fonctionne correctement
```

Définition de fonctions : portée

- Lorsque PHP rencontre une définition de fonction, il définit celle-ci au niveau global.

Ex :
```
$x = 1;
f($x);              // produit une erreur : f pas définie

if ($x > 2) {
  function f($x) { echo $x*2; }
}
else {
  function f($x) { echo $x*3; }
}

f($x);              // affiche 3
```

Fonctions : cas particuliers

- En PHP : PAS de redéfinition/surcharge (sinon, erreur fatale !)

- Quelques cas particuliers :
  - Valeurs par défaut
  - Passage par référence/adresse
  - Renvoyer plusieurs valeurs en même temps
  - Nombre d'arguments variable

Fonctions : cas particuliers

- **Valeurs par défaut** pour les arguments (il doit s'agir des derniers arguments dans la liste) :

```
function pourcentage ($cote, $max = 20) {
  return ($cote / $max) * 100;
}
pourcentage(15);
```

Fonctions : cas particuliers

- **Passage par référence** (automatique pour les objets) :

```
function swap (&$x, &$y) {
  $tmp = $x;
  $x = $y;
  $y = $tmp;
}

$nb1 = 3;
$nb2 = 7;
swap($nb1,$nb2);
```

Fonctions : cas particuliers

- Fonctions qui **renvoient plusieurs valeurs** : les assembler dans un tableau (voir plus loin).

```
function plusEtFois ($x, $y) {
  $somme = $x + $y;
  $prod = $x * $y;
  return array($somme, $prod);
}
```

- On peut décomposer le tableau reçu grâce au mot-clef list.

```
list($total,$produit) = plusEtFois(5,7);
```

Fonctions : cas particuliers

- Fonctions à **nombre d'arguments variable**
  - func_num_args() : nombre d'arguments transmis à la fonction
  - func_get_arg(i) : le i^e argument (en comptant à partir de 0)
  - func_get_args : tous les arguments (tableau)

```
function moyenne () {
  $n = func_num_args();
  $total = 0;
  for ($i = 0 ; $i < $n ; $i++) {
    $total += func_get_arg($i);
  }
  return ($total/$n);
}
```

Variables locales et globales

- Portée des variables

```
$intro = 'Le résultat est ';
$outro = '.<br/>';
```

$intro, $outro
portée globale

```
function calculeDouble ($val) {
  $double = $val * 2;
  return $double;
}
```

$val, $double
portée locale

```
function afficheDouble ($v) {
  global $intro, $outro;
  $d = calculeDouble($v);
  echo $intro, $d, $outro;
}
```

$v, $d
(autre) portée locale

Les variables globales ne sont pas automatiquement accessibles en local !

Variables locales et globales

- Portée des variables

```
$prix = 15;

function prixHorsSaison() {
  $prix = 10;
  echo "Le prix passe à $prix.<br/>";
}

echo "Le prix est $prix. <br/>";
prixHorsSaison();
echo "Le prix est $prix. <br/>";
```

$prix
var locale

Affichage

```
Le prix est 15.
Le prix passe à 10.
Le prix est 15.
```

Variables locales et globales

- Portée des variables

```
$prix = 15;

function prixHorsSaison() {
  global $prix;
  $prix = 10;
  echo "Le prix passe à $prix.<br/>";
}

echo "Le prix est $prix. <br/>";
prixHorsSaison();
echo "Le prix est $prix. <br/>";
```

$prix
var globale

Affichage

```
Le prix est 15.
Le prix passe à 10.
Le prix est 10.
```

Variables locales et globales

- Portée des variables

```php
$prix = 15;

function prixHorsSaison(&$prix) {
  $prix = 10;
  echo "Le prix passe à $prix.<br/>";
}
```

$prix
var passée par référence

```php
echo "Le prix est $prix. <br/>";
prixHorsSaison($prix);
echo "Le prix est $prix. <br/>";
```

Affichage

```
Le prix est 15.
Le prix passe à 10.
Le prix est 10.
```

Variables locales et globales

- Portée des variables : variable statique (comme en C)

```php
$prix = 15;
```

$nbModif
var locale statique

```php
function modifPrix($modif) {
  global $prix;
  static $nbModif = 0;
  $prix += $modif;
  $nbModif++;
  echo "Le prix passe à $prix ({$nbModif}e modif).<br/>";
}
```

La variable n'est initialisée à 0 que la première fois.

```php
modifPrix(+5);
modifPrix(-3);
modifPrix(+7);
```

Affichage

```
Le prix passe à 20 (1e modif).
Le prix passe à 17 (2e modif).
Le prix passe à 24 (3e modif).
```

Variables superglobales

- **Variables superglobales** (= prédéfinies et accessibles partout !)
  - `$GLOBALS` : toutes les variables définies dans le contexte global (dont les variables globales déclarées : `$_GLOBALS['mavar']`)
  - `$_SERVER` : informations provenant du serveur web
  - `$_ENV` : informations sur l'environnement et l'OS
  - `$_GET`, `$_POST` : variables reçues dans la requête http
  - `$_FILES` : fichiers attachés à la requête http
  - `$_COOKIE` : variables passées sous la forme de cookies http
  - `$_REQUEST` : l'ensemble des informations attachées à la requête http (redondant avec $_GET, $_POST, $_COOKIE et $_FILES).
  - `$_SESSION` : variables de session

- Ces variables prennent la forme de tableaux associatifs.

- Sécurité contre les injections :

```
$origine = htmlentities($_SERVER["HTTP_REFERRER"]);
```

PHP : les tableaux

- **Tableaux numériques et tableaux associatifs**
- **Tableaux multidimensionnels**
- **Utilisation des tableaux**
  - Résultat d'une fonction, argument d'une fonction, foreach

Tableaux

- En PHP, tous les tableaux sont des **tableaux associatifs** (associations clefs/valeurs).

  Exemple : *père -> Homer*
  mère -> Marge
  fils -> Bart
  fille -> Lisa

- Les tableaux à index numérique sont un cas particulier (les clefs sont 0, 1, 2, 3…) pour lequel il existe certaines facilités syntaxiques.

  Exemple : *0 -> Lisa*
  1 -> Bart
  2 -> Maggie

Tableaux

- En PHP, les **tableaux associatifs** peuvent avoir pour clefs
  - des chaînes de caractères ou
  - des entiers.

  Mais les chaînes correspondant à un entier valide sont automatiquement converties en entier : `$tab[3]` ≡ `$tab["3"]`

- Si on utilise d'autres types de clefs :
  - clefs réelles : converties en entiers
  - clefs booléennes : converties en entiers (0 ou 1)
  - valeur « null » : convertie en une chaîne vide ""
  - tableaux / objets : erreur *Illegal Offset Type* !

Tableaux à index numérique

- Tableaux hétérogènes à index numérique

```
$tab1 = []; // tableau vide
$tab2 = array(); // tableau vide
$tab3 = [2, 7, "hello", TRUE]; // tableau à 4 cases
$tab4 = array(2, 7, "hello", TRUE); // le même tableau
```

- Accès aux éléments : `$tab[index]` ou `$tab{index}`

- Écriture dans la case suivante

```
$tab5 = [2, "oui"];
$tab5[] = "ensuite";
$tab5[] =  47;
```

Aussi : `array_push($tab5, "ensuite", 47);`

Tableaux associatifs

- Tableaux associatifs

```
$simpson['père'] = 'Homer';
$simpson['mère'] = 'Marge';
$fam = array( 'père' => 'Homer',
              'mère' => 'Marge',
              'fils' => 'Bart'    );
$fam2 = [ 'père' => 'Kirk',
          'mère' => 'Luann',
          'fils' => 'Milhouse' ];
```

- Accès aux éléments : `$tab[`*`clef`*`]` ou `$tab{`*`clef`*`}`

- Note : on peut également construire des tableaux mixtes.

```
$tab = array(2, 4, 7, 10);
$tab['info'] = 'Data';
```

Tableaux multidimensionnels

- Simulés par des tableaux dont les éléments sont des tableaux.

- Tableaux numériques multidimensionnels

```
$matrice = array (
    array (1, 0, 0, 0),
    array (0, 1, 0, 0),
    array (0, 0, 1, 0),
    array (0, 0, 0, 1)
);
```

- Accès aux éléments

```
$matrice[2][2]
```

Tableaux multidimensionnels

- Tableaux multidimensionnels

```
$perso = array (
    'Simpson' => array (
        'père' => 'Homer',
        'mère' => 'Marge'),
    'Nahasapeemapetilon' => array (
        'père' => 'Apu',
        'mère' => 'Manjula'),
    'van Houten' => array (
        'père' => 'Kirk',
        'mère' => 'Luann')
);
```

- Accès aux éléments

```
$perso['van Houten']['père']
```

Utilisation des tableaux

- Décomposition (des premières valeurs) d'un tableau

```
$tab = array(4,7,9,10);
list($a,$b) = $tab; // met 4 dans $a et 7 dans $b
```

- Fonction renvoyant plusieurs valeurs

```
function carreCube ($x) {
  $tab[0] = $x * $x;
  $tab[1] = $tab[0] * $x;
  return $tab;
}
$res = carreCube(4);
echo $res[0], ' ', $res[1], '<br/>';
list ($carre, $cube) = carreCube(5);
echo $carre, ' ', $cube, '<br/>';
```

Utilisation des tableaux

- Les tableaux fonctionnent par valeur ! (contrairement à JS)

```
$tab = [1, 2, 3, 4];

$tab2 = $tab; // $tab2 est une copie de $tab
$tab2[0] = 7; // $tab est inchangé

function f($t) { $t[0] = 7; }
f($tab); // $tab est inchangé : f travaille sur une copie
```

- Travailler explicitement par adresse :

```
$tab2 = &$tab; // $tab et $tab2 désignent le même tableau
function f(&$t) { $t[0] = 7; }
```

Utilisation des tableaux

- Parcours d'un tableau à index numérique

```
for ($i = 0 ; $i < 5 ; $i++)
  echo $i, ' -> ', $tab[$i], '<br/>';
```

- Boucle foreach (clef et info)

```
foreach ($tab as $clef => $info)
  echo $clef, ' -> ', $info, '<br/>';
```

- Boucle foreach (info seulement)

```
foreach ($tab as $info)
  echo $info, '<br/>';
```