Cours PHP

Par Gauthier GARNIER
Contact mail et jabber : garnier.gauthier@gmail.com

Bases PHP – Les basiques du langage

- Syntaxe
- Types de données
- Variables
- Constantes
- Opérateurs
- Structure de contrôle
- Erreurs (comment les gérer)

Bases PHP – Syntaxe

Principalement basée sur le C, mais inspiré de Perl et de Java pour les formalismes « objet »

- Tags PHP

Tags Standards	`<?php`
	`... code`
	`?>`

Tags courts	`<?`
	`... code`
	`?>`
	`<?= $variable ?>`

Tags scripts	`<script language="php">`
	`... code`
	`</script>`

Tags ASP	`<%`
	`... code`
	`%>`

Les tags Courts, Scripts et ASP sont considérés comme dépréciés et ne doivent plus être utilisés

Bases PHP – Syntaxe

- Caractères spéciaux

```
\n   A line feed
\r   A carriage return
\t   A horizontal tab
\\   A backslash
\”   A double quote
\nnn A character corresponding to the octal value of nnn
\xnn A character corresponding to the hexadecimal value of nn
```

Bases PHP – Syntaxe

- Caractère '\n' dans les scripts PHP
  - Se rappeler que tous les caractères en dehors des tags PHP sont copiés tels quel par l'interpréteur vers la sortie
  - Non impactant pour HTML, nuit à la lisibilité
  - Peut poser problème si tous les headers n'ont pas été envoyés
  - Bonne pratique pour éviter les lignes vides lors d'un include
    - Ne pas fermer les balises PHP « ?> » en fin de script
- Commentaires

```
// Single line comment
# Single line comment
/* Multi-line
comment
*/
/**
* API Documentation Example
*
* @param string $bar
*/
function foo($bar) { }
```

Bases PHP – Types de données

- Types de données scalaires
  - boolean
  - int
  - Float
  - string
- Types de données composés
  - array
  - Object
- Types de données spéciaux
  - NULL
  - ressource

Bases PHP – Variable

- Nommage des variables

```
$name = 'valid'; // Nom valide
$_name = 'valid'; // Nom valide
$8name = 'invalid'; // Nom invalide (ne doit jamais commencer par un chiffre)
```

- Variable de variable
  - Une *variable de variable* est une variable dont le nom est contenu dans une autre variable.

```
$name = 'foo';
$$name = 'bar';
echo $foo;
// affichera 'bar'
```

- *Déterminer si une variable existe*

```
echo isset ($x);
// Renvoi TRUE si la variable existe ou une valeur différente de NULL sinon
```

Bases PHP – Constantes

- En opposition avec les variable, les constantes ont une valeur fixe et un type de données scalaire (boolean, int, float ou string)
- Par convention leur nom sera toujours en majuscule

- Quelques exemples :

```
define('EMAIL', 'davey@php.net'); // Nom valide
echo EMAIL; // Affiche  'davey@php.net'

define('USE_XML', true);
if (USE_XML) { } // Sera évalué à TRUE

define('6CONSTANT', 'some value');
// Nom Invalide (par de chiffre en début de nom)
```

Bases PHP – Opérateurs

- Opérateurs Arithmétiques
  - Addition
  - Soustraction
  - Multiplication
  - Division
  - Modulo
  - Incrémentation / Décrémentation :

```
$a = 1 + 3.5;
$a = 4 - 2;
$a = 8 * 3;
$a = 15 / 5;
$a = 23 % 7;
++$a;
```

Attention, l'incrémentation sous-entend un casting implicite en cas de besoin :

```
$a = 'Test';
echo ++$a;
// affichera 1
```

Bases PHP – Opérateurs

- Concaténation de strings

```
$string = "foo" . "bar";
// la valeur de $string est maintenant 'foobar'
$string2 = "baz";
// la valeur de $string2 est maintenant 'baz'
$string .= $string2;
// Après concaténation, on obtient : 'foobarbaz'
echo $string;
// Affiche 'foobarbaz'
```

Bases PHP – Opérateurs

- Opérateur sur les bits

Exemple	Nom	Résultat
`$a & $b`	ET (*And*)	Les bits positionnés à 1 dans *$a* ET dans *$b* sont positionnés à 1.
`$a | $b`	OU (*Or*)	Les bits positionnés à 1 dans *$a* OU *$b* sont positionnés à 1.
`$a ^ $b`	*Xor*	Les bits positionnés à 1 dans *$a* OU dans *$b* mais pas dans les deux sont positionnés à 1.
`~ $a`	NON (*Not*)	Les bits qui sont positionnés à 1 dans *$a* sont positionnés à 0, et vice versa.
`$a << $b`	Décalage à gauche	Décale les bits de *$a*, *$b* fois sur la gauche (chaque décalage équivaut à une multiplication par 2).
`$a >> $b`	Décalage à droite	Décalage des bits de *$a*, *$b* fois par la droite (chaque décalage équivaut à une division par 2).

Bases PHP – Opérateurs

- Affectation :

Il est possible d'associer à l'opérateur d'affection tous les opérateurs arithmétiques et opérateurs de bits

```
$variable = 1;
// $variable contient la valeur 1
$variable += 3;
// $variable contient maintenant la valeur 4
```

Bases PHP – Opérateurs

- Affectation par référence :

```
$a = 10;
$b = $a;
$b = 20;
echo $a; // Sortie 10
```

```
$a = 10;
$b = &$a; // par référence
$b = 20;
echo $a; // Sortie 20
```

Bases PHP – Opérateurs

- Affectation par défaut :

  Par défaut, l'opérateur d'affectation fonctionne par valeur pour tous les types de données SAUF pour les objets qui avec ou sans l'opérateur '&' sont affectés par référence.

  En savoir plus :

  http://www.php.net/manual/fr/language.references.php

Bases PHP – Opérateurs

- Opérateurs de comparaison

Opérateur	Description
==	Equivalence : Renvoi *true* si les deux opérandes sont équivalentes, c'est-à-dire qu'elle peuvent être convertie dans un type de données commun dans lequel elles auront la **même valeur**.
===	Identité : Renvoi *true* seulement quand les deux opérandes sont du **même type** de données **ET** on la **même valeur**.
!=	Non-équivalent : Vrai si les deux opérandes ont des valeurs différentes indépendamment de leur type.
!==	Non-identique : Vrai si les deux opérandes n'ont pas la même valeur ou pas le même type.

Bases PHP – Opérateurs

- Opérateurs logiques

Opérateur
&& / and
|| / or
XOR

Bases PHP – Opérateurs

- Autres opérateurs
  - @ : Suppression d'erreurs

```
$x = @mysql_connect();
```

  - Backquote : execution qu'une commade shell

```
$a = `ls -l`;
```

Bases PHP – Structures de contrôle

- Structure conditionnelle IF

```
if (expression1)
{

}
elseif (expression2)
{

}
else
{

}
```

Bases PHP – Structures de contrôle

- Structure conditionnelle CASE

```
$a = 0;
switch ($a) {
case true: // $a est comparé à true
// Evalué à false
break;
case 0: // Comparé to 0
// Evalué à true
break;
default:
// Sera executé si aucun cas ne convient
break;
}
```

Bases PHP – Structures de contrôle

- Structure itérative WHILE et DO WHILE

```
$i = 0;
while ($i < 10) {
echo $i . PHP_EOL; //PHP_EOL = End Of Line
$i++;
}

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

Bases PHP – Structures de contrôle

- Structure itérative FOR

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

Bases PHP – Erreurs

- Configuration de l'error reporting
  - Dans le fichier php.ini
  - Via la fonction : error_reporting()

```
// Rapporte les erreurs d'exécution de script
error_reporting(E_ERROR | E_WARNING | E_PARSE);

// Rapporter les E_NOTICE peut vous aider à améliorer vos scripts
// (variables non initialisées, variables mal orthographiées..)
error_reporting(E_ERROR | E_WARNING | E_PARSE | E_NOTICE);

// Rapporte toutes les erreurs à part les E_NOTICE
// C'est la configuration par défaut de php.ini
error_reporting(E_ALL ^ E_NOTICE);
```

Bases PHP – Erreurs

- Types et niveaux d'erreurs

Constante	Description
E_ERROR	Les erreurs sont aussi affichées par défaut, et l'exécution du script est interrompue. Elles indiquent des erreurs qui ne peuvent pas être ignorées, comme des problèmes d'allocation de mémoire, par exemple.
E_WARNING	Les alertes sont affichées par défaut, mais n'interrompent pas l'exécution du script. Elles indiquent un problème qui doit être intercepté par le script durant l'exécution du script. Par exemple, appeler ereg() avec une expression rationnelle invalide.
E_PARSE	Les erreurs d'analyse ne doivent être générées que par l'analyseur. Elles ne sont citées ici que dans le but d'être exhaustif.
E_NOTICE	Les notices ne sont pas affichées par défaut, et indiquent que le script a rencontré quelque chose qui peut être une erreur, mais peut aussi être un événement normal dans la vie du script. Par exemple, essayer d'accéder à une valeur qui n'a pas été déclarée, ou appeler stat() sur un fichier qui n'existe pas.
E_CORE_ERROR	Elles sont similaires aux erreurs **E_ERROR**, mais elles sont générées par le code de PHP. Les fonctions ne doivent pas générer ce genre d'erreur.
E_CORE_WARNING	Elles sont similaires à **E_WARNING**, mais elles sont générées par le code de PHP. Les fonctions ne doivent pas générer ce genre d'erreur.
E_STRICT	Notices au moment de l'exécution. Permet d'obtenir des suggestions de PHP pour modifier votre code, assurant ainsi une meilleure interopérabilité et compatibilité de celui-ci.
E_RECOVERABLE_ERROR	Erreur fatale qui peut être captée. Ceci indique qu'une erreur probablement dangereuse s'est produite, mais n'a pas laissé le moteur Zend dans un état instable. Si l'erreur n'est pas attrapée par un gestionnaire d'erreur défini par l'utilisateur (voyez aussi set_error_handler(), l'application arrête prématurément comme si cela était une **E_ERROR**.
E_ALL	Toutes les erreurs et alertes supportées sauf le niveau **E_STRICT** dans PHP < 6.

Bases PHP – Erreurs

- Directives de gestion des erreurs
  - *display_errors* (pendant le développement)
  - *log_errors* (en prod)

Bases PHP – Erreurs

- Surcharge de la gestion des erreurs
  - Set_error_handler()
  - http://fr2.php.net/manual/fr/function.set-error-handler.php