PHP - Développement Web

CM2: PHP

Mickaël Martin Nevot

V1.14.0



Cette œuvre de <u>Mickaël Martin Nevot</u> est mise à disposition selon les termes de la <u>licence Creative Commons Attribution - Pas d'Utilisation Commerciale - Partage à l'Identique</u>
3.0 non transposé.

PHP – Développement Web

- Présentation du cours
- Rappels: Web, HTML, CSS et JavaScript
- III. PHP
- IV. PHP « avancé »

PHP/PHP 5

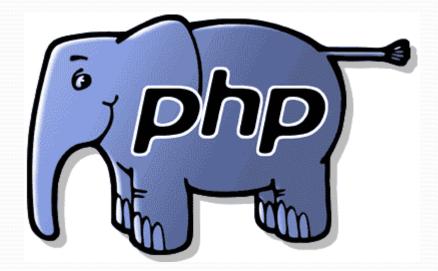
- Historique :
 - 1994-1995 : PHP par Rasmus Lerdorf
 - 2004 : PHP version 5
 - 2020 : PHP version 8
- Principales caractéristiques :
 - Langage de programmation de script interprété
 - Conçu pour le développement d'applications Web
 - Exécuté **côté serveur** (code non accessible côté client)
 - Génération de pages Web dynamiques
 - Possibilité d'être couplé à une base de données
 - Langage objet faiblement typé
 - Open source



Utilisation

- Fichier texte avec l'extension : .php
- Script PHP: langage PHP et/ou langage HTML
- Balise <?php ?>:

```
<?php
    instruction_1
    instruction 2
?>
```



Commentaires

Non interprétés

• Commentaires de type C/C++ et *shell* Unix

• Exemples :

```
// Commentaire PHP (une seule ligne complète).
# Autre commentaire PHP (une seule ligne complète, déprécié).
/* Autre commentaire PHP (une ou plusieurs lignes). */
   Commentaire PHP
    (sur plusieurs lignes).
```

Définition de type

- Pas de déclaration explicite du type d'une variable
- Le type d'une variable est déterminé par le contexte d'utilisation
- Conversion automatique



Variables

- Représentées par un signe dollar \$ suivi du nom de la variable: \$x
- Sensibles à la casse : \$x est différent de \$X
- Affectation :
 - Simple :

```
$a = 10;
$b = $a;
```

• Par référence :

```
$a = 10;
$b = &$a; // Ici, $a et $b pointent sur le même contenu.
```

Destruction : unset (\$var)

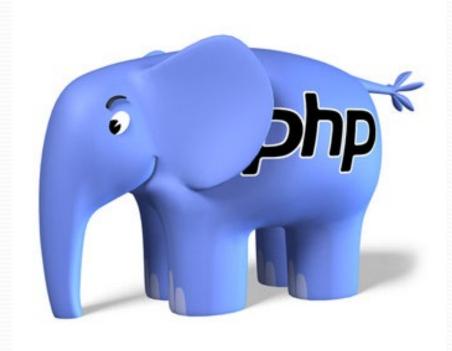
La valeur NULL

- La valeur spéciale NULL représente l'absence de valeur
- Une variable considérée comme NULL n'a pas de valeur
- Une variable est considérée comme NULL si :
 - On lui a affecté la constante NULL
 - Aucune valeur ne lui a été attribuée



Les types

- Types scalaires (simples):
 - Booléen
 - Entier
 - Nombre décimal
 - Chaîne de caractères
- Types composés :
 - Tableau
 - Objet



Booléen

- Peut prendre la valeur true ou false
- Valeurs considérées comme fausses :
 - Le booléen false lui-même
 - L'entier 0
 - Le nombre décimal 0.0
 - La chaîne de caractères vide et la chaîne de caractères "0"
 - Le tableau vide

A cause de la définition de type

- La valeur spéciale NULL
- Toutes les autres valeurs sont considérées comme true
- Opérateurs de conversion : (bool), (boolean)

Entier

- Un type entier est un entier naturel
- Les entiers peuvent être spécifiés en base 10, 8 ou 16
- Les entiers peuvent être optionnellement précédés par le signe plus ou moins (+ ou -):

```
$a = 1234; // Nombre entier en base 10.
a = -123; // Nombre entier négatif.
$a = 0123; // Nombre entier en base 8, octale (83 en base 10).
\$a = 0x12; // Nombre entier en base 16, hexadécimale (18 en base 10).
```

- Conversion :
 - Opérateurs de conversion : (int), (integer), intval()
 - Lors de la conversion entre un nombre décimal et un entier, le nombre sera arrondi:
 - À la valeur inférieure s'il est positif, à la supérieure s'il est négatif

Nombres décimaux

- Aussi appelés :
 - Double
 - Float
 - Nombres réels :

```
$a = 1.234;
$b = 1.2e3;
$c = 7E-10;
```

- Conversion en nombre décimal :
 - Opérateurs de conversion : (real), (double) et (float)
 - Fonctions floatval() et doubleval()

Chaîne de caractères

- Séquence de caractères
- Une chaîne peut être spécifiée de trois manières :
 - Guillemets doubles "
 - Guillemets simples '
 - Syntaxe *heredoc* (plus de détails : documentation PHP) :

```
$a = "hello world";
$b = '$a : ';
SC = <<<EOT
    Hello world
EOT;
```



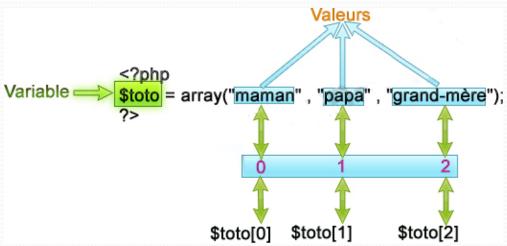
Chaîne de caractères

- Guillemets doubles / syntaxe heredoc:
 - Variables présentes dans la chaîne évaluées
 - Caractères spéciaux pris en compte :
 - Nouvelle ligne : \n
 - Retour à la ligne : \r
 - Tabulation horizontale : \t
 - *Antislash* : \\
 - Caractère \$:\\$
 - Guillemets doubles : \"
- Guillemets simples :
 - Variables présentes dans la chaîne non évaluées
 - Caractère spécial pris en compte : \ ' guillemet simple

Tableau

• Syntaxe :

```
// array(clef => valeur , ...)
// clef ne peut être que :
// - entier (tableau indexé) ;
// - chaîne de caractères (tableau associatif).
// Valeur peut être de n'importe quel type.
$arr = array('foo' => 'bar', 12 => true);
echo $arr['foo']; // bar
echo $arr[12];
```



Tableau

• Syntaxe (création/modification avec des crochets) :

```
// $arr[clef] = valeur;
// $arr[] = valeur;
// clef peut être un entier ou une chaîne de caractères.
// valeur peut être n'importe quel type.
\$arr = array(5 \Rightarrow 1, 12 \Rightarrow 2);
$arr[] = 56; // Identique à $arr[13] = 56; à cet endroit du script.
$arr['x'] = 42; // Ceci ajoute un nouvel élément au
                 // tableau avec la clef 'x'.
```

• Fonction sizeof():

```
sizeof($arr); // Retourne le nombre d'éléments du tableau.
```

• Fonction unset():

```
unset($arr[5]); // Ceci efface un élément du tableau.
unset($arr); // Ceci efface tout le tableau.
```

Objet

• Instancié avec new:

```
class Foo
    function do_foo($a)
        return $a / 2;
\text{sbar} = \text{new Foo()};
b = \frac{1}{2}
```

- Opérateurs de conversion : (object)
 - Conversion en un tableau : les propriétés sont les clefs
 - Conversion dans un autre type : membre scalar

```
$obj = (object) 'hello';
echo $obj->scalar; // Affiche : hello.
```

Constante

- Ni modifiable, ni effaçable
- Ne commence pas par un \$
- Par convention : écrite en majuscule
- Accessible globalement
- Valeur scalaire uniquement
- Définie par define():

```
define('FOO', 'something');  // Valide.
define('2FOO', 'something'); // Invalide.
define('__FOO__', 'something'); // A éviter (à cause de __..._).
```

Variables prédéfinies

- Variables prédéfinies « superglobales » :
 - \$GLOBALS: toutes les variables du contexte global
 - \$_SERVER : variables serveur et d'exécution
 - \$_GET: HTTP GET
 - \$_POST: HTTP POST
 - \$_SESSION : variables de session
 - \$_COOKIE : cookies HTTP
 - \$_FILE : fichier chargé
- Variables globales et configuration : phpinfo()

Opérateurs 1/2

- Unaires
 - Signes : +, -
 - Négation : !
 - Incrémentation/décrémentation (pré et post) : ++, --
 - Conversion: (int), (bool), etc.
 - Clonage: clone
 - Contrôle d'erreur : @

```
$a = -0.5;
--$a;
$b = (bool) $a;
$c = !$b;
```

Opérateurs 2/2

- Binaires
 - Arithmétiques : +, -, *, /, %
 - Concaténation de chaîne : .
 - Affectations : =, +=, .=, *=, etc.
 - Comparaisons : ==, >, >=, <, !=, <>, ===, !==, etc.
 - Logiques: &&, ||, xor, etc.
 - Binaires : &, |, ~, ^, <<, >>

```
$a = 5;
$b = 10;
$c = '$a : ' . $a; // Pas de caractère '+' !
d = (a = b) & (0 != b % a) | (b < 20);
```

Structures conditionnelles

Structure conditionnelle:

```
if (0 = \$a)
    ++$a;
else if (1 == \$a)
    --$a;
else
    $a = 0;
```

Branchement conditionnel:

```
switch($i)
  case 'Jambon':
     echo 'Salé';
     break;
  case 'Tarte':
  case 'Bonbon':
     echo 'Sucré';
     break;
  default:
     echo 'Autre';
```

Boucles

• Boucle for:

```
for (\$i = 0 ; \$i < 10 ; ++\$i)
    echo $i;
```

• Boucle foreach :

```
\$arr = array(1, 2, 3, 4, 5);
foreach ($arr as $key => &$val)
   $val *= 2;
// $arr = (2, 4, 6, 8, 10).
```

• Boucle while:

```
while ($i < 5)
    ++$i;
```

De 0 à n fois

• Boucle do ... while:

```
do
    --$i;
} while (\$i > 0);
```

De 1 à n fois

Break/continue

Break

 Sortir d'une boucle ou d'un branchement conditionnel:

```
while(1)
  if (10 = $i)
     break;
  ++$i;
// Exemple de break dans un
// switch sur la diapositive
// de structures conditionnelles.
```

Continue

 « Sauter » à l'itération suivante d'une boucle:

```
$a = 0;
for (\$i = 0 ; \$i < 10 ; ++\$i)
{
    if (0 = \$i \% 2)
        continue;
    $a += $i;
// ?
```

Fonctions

```
function foo(\$a, \$b, \$c = 2)
    // Rappel : $a, $b et $c sont des paramètres.
    // = 2 : valeur par défaut (uniquement de type scalaire).
    // Un paramètre par défaut est optionnel.
    ++$a;
    --$b;
    // Termine la fonction en retournant la valeur calculée.
    return ($a * $b) / $c;
    // Le code éventuellement placé ici n'est pas exécuté.
}
$a = 5;
b = foo(3, a); // Rappel : 3 et a sont des arguments.
// $a = 5.
// \$b = 8.
```

Portée des variables

- Globale : définie dans un script PHP
- Locale : définie dans une fonction

```
$a = 1; // Portée globale.
function test()
    echo $a; // Portée locale : rien n'est affiché à l'écran.
test();
```

Statique:

```
function compteur()
    static $a = 0; // Prend une valeur scalaire, pas une expression.
   // Qu'affiche cet echo au cinquième appel de la fonction compteur ?
   echo $a;
    ++$a;
```

Instruction

- Se termine par un ;
- Types d'instruction :
 - Déclaration : \$a;
 - Affectation : \$a = 10;
 - Appel de fonction : myFunction();
 - Instruction conditionnelle: if (\$a == \$b) ...;
 - Instruction vide : ;
 - Bloc (d'instructions) :

```
instruction 1
instruction 2
```

Inclusion de fichiers

- Inclut et exécute un fichier PHP en remplacement de l'instruction : include/require
- include_once/require_once : comme leur version sans le suffixe « _once » mais suppriment l'instruction sans remplacement si le fichier a déjà été inclus
- Seule différence entre include et require : la manière dont les erreurs sont produites :
 - include: produit une simple alerte
 - require : produit une erreur fatale

Exemple

Solution 1 (« totalement PHP ») :

```
<?php
\$arr = array(1, 2, 3, 4, 5);
echo '';
for (\$i = 0; \$i < 5; ++\$i)
   // PHP_EOL : retour à la ligne dans le code source (pas dans la page Web)
   echo PHP_EOL . ' ' . $arr[$i] . '';
echo PHP EOL . '';
?>
```

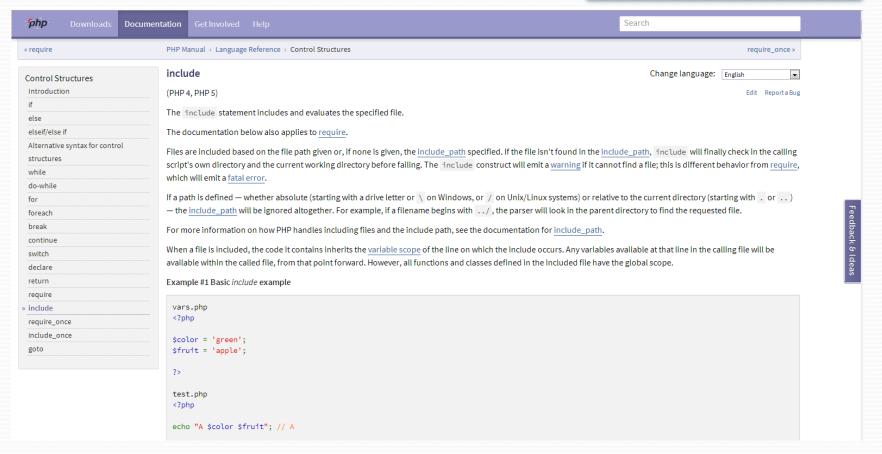
Solution 2 (mixte):

```
<?php
   \$arr = array(1, 2, 3, 4, 5);
   for (\$i = 0 ; \$i < 5 ; ++\$i)
   ?>
   <?php echo $arr[$i];?>
<?php } ?>
```

Les solutions sont équivalentes

Manuel PHP

De préférence, en anglais



http://www.php.net

Convention de codage

- Convention de codage :
 - http://pear.php.net/manual/fr/standards.php
- Style d'écriture de code :
 - http://www.php-fig.org/psr/
 - https://tommcfarlin.com/what-is-psr/



Liens

- Documents électroniques :
 - http://php.net/manual
 - http://www.manuelphp.com
 - http://php.developpez.com/cours
 - http://www.vulgarisationinformatique.com/variables-predefinies.php
 - http://code18.blogspot.fr/2009/05/3-facons-de- lire-du-xml-en-php.html
- Document classique :
 - Ilia Alshanetsky. *Introducing PHP 5.4*.

Crédits



