PHP

Les bases de PHP

Nicolas BLAUDEZ

2018

PHP - SOMMAIRE

- Histoire
- Installation et environnement
- Les variables
- Les conditions
- Les boucles
- Les fonctions
- La programmation objet

PHP – Histoire

- PHP: Hypertext Preprocessor
- Crée en 1994 par Rasmus Lerdorf
- Version 3 en 1997 par Andi Gutmans et Zeev Suraski
- Création du moteur Zend Engine qui amene à la version 4 de php.
- Version 5 : Programmation objet
- Version 7 : Amélioration des performances

PHP – Installation / Environnement

- PHP CLI (ligne de commande / terminal)

  apt-get install php7.1-cli

  cd /var/www

  php -S localhost:4000 .

  http://localhost:4000/

- PHP Apache

  apt install php7.1 libapache2-mod-php7.1 php7.1-mysql php-common php7.1-cli php7.1-common php7.1-json php7.1-opcache php7.1-readline apache2

  apachectl start

  http://localhost/

PHP - Syntaxe

- Le code php se trouve entre ces balises

  <?php et ?>

- Il est possible de mettre des commentaires dans son code. Ils ne sont pas interprétés par PHP.
- Les fonctions « echo() » et « var_dump() » permettent d'afficher du texte ou le contenu d'une variable.

```
<?php
// commentaire sur une ligne (pas interprété)
# commentaire sur une ligne (pas interprété)
/*
* Commentaire sur plusieurs lignes (pas interprété)
*/
$variable= 'Bonjour' ;
echo 'ceci est un texte' ; // affiche a l'ecran : 'ce texte est un texte'
var_dump($variable) ; // affiche a l'ecran : string(7) "bonjour"
```

PHP – Les variables

- Une variable permet de stoker une information.
- Nom des variables précédés par $
- Le nom de la variable peut être composé de chiffre et de lettre minuscule ou majuscule ainsi que du caractère underscore « _ »
- Types de variables :

Array/Tableau	Contient plusieurs elements de n'importe quel type	$array = ['nicolas' , 3,14 , new Human() , 8] ;
Int / Entier	nombres entiers	$int = 8 ;
String / Chaine	Chaine de caractere	$string = 'Il y a 3 pommes dans l'arbre' ;
Object / Objet	Instanciation d'une classe	$obj = new Human() ;
Float / Flottant	nombres à virgule flottante	$float = 3.14 ; // il faut utiliser un . pas une ,
Resourece/Ressource	Liens vers un ressources système (fichier, connexion reseau)	$conn = mysql_connect('localhost','root','pwd) ;

PHP – Les variables – Tableau

- Un tableau est composé de plusieurs éléments contenant chacun une information.
- Il peut etre indéxé ou associatif.

Tableau indexé :	Tableau associatif :

$array = [‘nico’,’greg’,’louis’] ;
 	
echo $array[0] ; // nico
echo $array[1] ; // greg
	$array= [‘prenom’=> ‘nicolas’,’nom’=>’blaudez’] ;
	
	eho $array[‘prenom’] ; // nicolas
	echo $array[‘nom’] ; // blaudez

Infos : http://php.net/manual/fr/language.types.array.php

PHP – Les variables – Tableau multidimensionnel

Un tableau multidimensionel est un tableau composé d'autres tableaux

```
$eleves = [
                ['prenom'=>'prenom1','nom'=>'nom1'],
                ['prenom'=>'prenom2','nom'=>'nom2']
            ] ;

echo $eleves[0]['prenom'] ; // prenom1
echo $eleves[1]['nom'] ; // nom2

$pays = [ 'france'=> ['capitale'=>'Paris','population'=>'60M'],
            'allemagne' =>  ['capitale'=>'berlin','population'=>'80M']
            ] ;

echo $pays['france']['capitale'] ; // Paris
echo $pays['allemagne']['population'] ; // 80M
```

PHP – Les variables – Chaine de caractère

- Une chaîne de caractère (string en anglais) et une variable comprenant des caractères alphanumériques
- Il est possible de les « concaténer » c'est à dire les « liés » avec l'opérateur '.'

```
<?php

$string = 'Bonjour' ;
$prenom= 'Nicolas' ;

// On concatène  les deux variables avec un espace au milieu
$phrase = $string.' '.$prenom ;

// On affiche le resultat
echo $phrase ; // Affiche 'Bonjour Nicolas' ;
```

Infos : http://php.net/manual/fr/language.types.string.php

PHP – VARIABLES - SUPERGLOBALES

Les superglobales sont des variables définies par PHP. Elles contiennent chacune des informations spécifiques.

- Accéder aux variables de la requête :

  $_GET
- Accéder aux variables du formulaire :

  $_POST
- Accéder aux variables serveur :

  $_SERVEUR
- Accéder aux variables des cookies :

  $_COOKIE

ces superglobales sont accessibles partout même dans les fonctions et objets. Elles sont « globales »

Infos : http://php.net/manual/fr/language.variables.superglobals.php

PHP – Les conditions

Il est possible d'exécuter des portions de code suivant certaines conditions. Pour cela les mot clefs ci-dessous sont disponibles :

- IF
- ELSEIF
- ELSE
- SWITCH

- Opérateurs logiques : http://php.net/manual/fr/language.operators.logical.php
- Opérateurs de comparaison : http://php.net/manual/fr/language.operators.comparison.php

PHP – Les conditions - IF

- Le mot clef **IF** permet d'exécuter du code si l'expression entre parenthèse est vraie (true).
- Le mot clef **ELSEIF** permet d'ajouter une autre condition si la première n'est pas vérifiée
- Le code présent après le mot clef **ELSE** est executé si aucune des conditions précédentes sont vraies.

```
$a = 10 ;
$b = 15 ;

if( ($a  ≤ 20) && ($b!= $a) ) {
      echo '$a est inférieur à 20 et différent de $b' ;
} elseif( ($a > 20) && ($b!= $a) )  {
      echo '$a est supérieur à 20 et différent de $b' ;
} else {
      echo '$a est égal a 20' ;
}
```

Infos : http://php.net/manual/fr/control-structures.if.php

PHP – Les conditions - SWITCH

Le mot clef switch permet d'exécuter du code en fonction du résultat de l'expression passée en argument.

```
$a = 'rouge' ;
switch($a) {
   case 'rouge' : echo '$a est rouge';break ;
   case 'bleu' : echo '$a est bleu';break ;
   default :  // Executé si aucun des cas précédent est vérifié
      echo 'on ne connaît pas la couleur de $a';
      break ;
}
```

Infos : http://php.net/manual/fr/control-structures.switch.php

PHP – Les boucles

Une boucle permet d’effectuer un traitement plusieurs fois de suite.

Mot clefs utilisés pour faire des boucles :

- FOR
- FOREACH
- WHILE
- DO

PHP – Les boucles - FOR

Une boucle **FOR** permet d'itérer c'est à dire jouer plusieurs fois une portion de code.
D'abord on initialise une variable qui permettra de stopper la boucle une fois la condition vérifiée
Ensuite on incrémente la variable (ou on la décrémente) .

```
for(initialisation ; condition ; modification de la variable) {
        code a exécuter
}
```

```
// affiche les nombres de 0 a 10
for($i=0; $i<=10; $i++) {
   echo $i. '\n' ;
}
```

```
// affiche les nombres de 10 a 0
for($i=10; $i>=10; $i--) {
   echo $i. '\n' ;
}
```

```
// Affiche les nombres de 0 a 20
// 5 par 5
for($i=0; $i<=20; $i+=5) {
   echo $i. '\n' ;
}
```

```
// Affiche les nombres de 20 a 0
// 5 par 5
for($i=20; $i>=0; $i-=5) {
   echo $i. '\n' ;
}
```

Infos : http://php.net/manual/fr/control-structures.for.php

PHP – Les boucles - FOREACH

Permet de parcourir un tableau.

Dans l'exemple ci-dessous la clef est passé dans la variable $key et le contenu de l'élément dans $value.

```
// Avec un tableau indéxé
$array = ['paris','london','madrid'] ;
foreach($array as $key=>$value) {
    echo $key. '='. $value. '\n' ;
}
// Affiche :
// 0 = paris
// 1 = london
// 2 = madrid
```

```
// Avec un tableau associatif
$array =  [
        'pays'=>'france',
        'capitale'=>'paris
] ;
foreach($array as $key=>$value) {
    echo $key. '='. $value. '\n' ;
}
// Affiche :
// 'pays' = 'france'
// 'capitale' = 'paris'
```

Infos : http://php.net/manual/fr/control-structures.foreach.php

PHP – Les boucles - WHILE

Une boucle **WHILE** permet d'exécuter du code tant que la condition entre parenthèse renvoie vraie (true)

```
// Affiche les nombres de 0 à 100

$i=0 ; // Initialisation de la variable $i
while($i<100) {   // ($i < 100) = condition d'exécution
   echo $i.'\n' ;  // on affiche $i
   $i++ ; // on incremente $i sinon boucle infinie
}
```

Infos : http://php.net/manual/fr/control-structures.while.php

PHP – Les boucles - WHILE

- Exemple de boucle while

```
tant que  $i inferieur à $n
    si $i est pair
        afficher $i
    fin si
fin tant que
```

```
$i = 1 ;
while($i<=10) {
    if($i %2 == 0) {
        echo "$i";
    }
}
```

PHP – Les boucles - DO

Une boucle DO permet d'exécuter le code tant que la condition renvoie vraie, true.

A l'inverse de **WHILE** la condition est testée après la première exécution du code.

```
$i = 0 ;
do {
    echo $i;
    $i++ ;
} while ($i < 100);
```

Infos : http://php.net/manual/fr/control-structures.do.while.php

PHP – Les fonctions

Une fonction permet d'effectuer la même tache sur des jeux de données différents.

Une fonction est définie par le mot clef **function** puis son nom et une liste de **paramètres** (variables) entre parenthèse.

Les paramètres sont locales a la fonction, c'est à dire que ces variables ne peuvent pas être utilisées en dehors de la fonction.

Une fonction peut retourner une valeur a l'aide du mot clef **RETURN**

```
function add($a,$b) {
   return $a + $b ;
}
echo add(5,4) ; // affiche 9
echo add(3,4) ; // affiche 7
```

Infos : http://php.net/manual/fr/language.functions.php

PHP – Fonctions - Paramètres

- Une fonction prend des paramétrés (ici $a et $b). Les paramétres sont des variables que vous pouvez utiliser dans la fonction.
- Il est possible de donner une valeur par défaut au paramétrés ($b a une valeur de 5 par defaut). Cette valeur est utilisé si le paramètres n'est pas renseigné dans l'appel de la fonction.

```
function test($a,$b =5) {
   return $a+$b ;
}
echo test(4) ; // affiche 9 car $b = 5 par defaut
echo test(4, 2 ) ; // afffiche 6 car ici $b = 2
```

PHP – Fonctions – porté des variables

- A l'intérieur d'une fonction les variables sont dites « locales » c'est à dire qu'elles ne sont pas accessible à l'extérieur de la fonction.
- Les variables du programme principal sont « globales » c'est a dire accessible dans tout le programme SAUF dans les fonctions et méthodes de classe.

```
<?php
$a = 'bonjour nicolas'

function maFonction($parametre) {
        $a = 'Aurevoir Nicolas' ;
        $phrase = $parametre . ' ' . $a ;
        return $phrase ;
}

echo $a ; // Affiche 'Bonjour nicolas'
echo maFonction('test ') ; //Affiche 'Aurevoir Nicolas test'
```

PHP – Fonctions - Référence

- La référence (&) permet de changer la valeur d'un paramètre dans l'ensemble du script. Elle se place devant la variable dans la liste des paramètres.

```
function test(&$a, $b =5) {
   $a = $a + 10 ;
   return $a+$b ;

}
echo test(4) ; // affiche 19
echo $a ; // affiche 14 car $a est passé par référence et à été
                // modifié dans la fonction
```

PHP - POO

Une classe est un ensemble de méthodes (fonctions) et d'attributs (variables) servant de modèle à des futurs objets.

Bien que les objets sont issus de la même classes ils sont compléments indépendants. (leurs propriétés sont différentes)

```php
<?php

Class Personne() {

        public $nom ;
        private $adresse;
        protected $age ;

       public function setNom($nom) {
             $this→nom = $nom ;
       }

      public function getNom() {
            return $this→nom ;
      }
}
```

```php
// Création d'un objet Personne
$obj1 = new Personne()
$obj1→setNom('Blaudez') ;
echo $obj1→getNom() ; // affiche « Blaudez »

// Création d'un autre objet Personne
$obj2 = new Personne()
$obj2→setNom('Tarantino') ;
echo $obj2→getNom() ; // affiche « Tarantino »
```

Infos : http://php.net/manual/fr/language.oop5.php

PHP – POO - Modifiers

Une méthode ou un propriété d'objet peut être soit :

- public (accessible dans et en dehors de la classe)
- **private** (accessible uniquement dans la classe)
- **protected** (accessible dans la classe et ses classes héritières)

ces mots clefs sont appelés des **modifiers**.

Par défaut une méthodes ou une variables est public

PHP – POO - Héritage

- Une classe peut étendre, c'est à dire hériter des méthodes (fonctions) et propriétés (variables) d'une autre classe. Cela se fait grace au mot clef **EXTENDS**
- Lorsqu'une classe hérite d'une autre classe elle inclue ses propriétés et méthodes. Une classe ne peut hérité que d'UNE autre classe.

```php
<?php

Class Human() {

        public $nom = 'Blaudez';
        private $adresse;


       public function setNom($nom) {
            $this→nom = $nom ;
       }

      public function getNom() {
           return $this→nom ;
      }
}
```

```php
<?php
Class Male extends Human {

        protected $age = 35;

        public function getAge() {
             Return $this→age ;
        }

}

$obj= new Male() ;
echo $obj→getAge() ; // affiche '35'
// getNom() est un  méthode héritée de la classe // Human
echo $obj→getNom() ; // affiche 'Blaudez'
echo $obj→adresse ; // ERROR - (private)
```

Infos : http://php.net/manual/fr/language.oop5.inheritance.php

PHP – POO - TRAITS

- Un trait est un ensemble de propriétés (variables) et méthodes (fonctions) que vous pouvez inclure dans vos classes.
- Une classe peut introduire plusieurs traits.
- Pour introduire un **trait** il faut utiliser le mot clef **USE**

```
Traits Human {
    protected $age;
    public function getAge() {
        return $this→age;
    }
}

Traits Terrian {
    protected $planet = 'heart';
    public function getPlanet() {
        return $this→planet;
    }
}
```

```
Class Male {
    Use Human;
    Use Terrian ;

    Protected  $sexe = 'Homme' ;
    public function getSexe() {
          return $this→sexe ;
    }

}

$obj = new Male() ;
echo $male→getPlanet() ; // affiche 'heart'
echo $male→getSexe() ; // affiche 'homme'
echo $male→getAge() ; // affiche '35'
```

Infos : http://php.net/manual/fr/language.oop5.traits.php

PHP – POO - Interface

- Une interface impose des méthodes aux classes qui l'implémentent
- Si ces méthodes ne sont pas définies dans la classe alors PHP émet une erreur et le script s'arrête,

```
Interface Human {
     public function getAge() ;
}
// Cette classe n'est pas valide car elle ne definie pas la methode getAge()
// alors qu'elle implemente l'interface Human
Class Male implements Human {
   public function getName() {
   }
}

// Cette classe est valide car elle definie la méthode getAge() ;
Class Female implements Human {
   protected $age='35' ;
   public function getAge() {
       Return $this→age ;
   }
}
```

Infos : http://php.net/manual/fr/language.oop5.interfaces.php

PHP – POO - Constructeur

- Le constructeur est la méthode qui est appelé lorsque l'objet est crée.

```
Class Human {
    protected $sexe;
    protected $age ;

    public function __construct($age,$sexe) {
        $this→age = $age ;
        $this→sexe= $sexe ;
    }
    public function __toString() { // Méthode magique appelée lorsqu'on affiche l'objet.
        Echo 'Je suis un '.$this→sexe.' de '.$this→age.' ans.' ;
    }
}

$obj = new Human('35','homme') ;
echo $obj ; // Je suis un homme de 35 ans.
```

PHP – POO – Espace de nom

- Les espaces de nom (namespace) permettent d'isoler des classes, interfaces et traits.
- Cela permet d'avoir deux classes comportant le même nom mais dans des espaces de nom différents au sein d'une même application.

```php
<?php
namespace Simplon\Web;

Class Prof {

}
```

```php
<?php
namespace Simplon\Literature;

Class Prof {

}
```

```php
<?php

use Simplon\Web\Prof;
use Simplon\Literature\Prof as Lprof;

$obj1 = new Prof();
var_dump(get_class($obj1); // Simplon\Web\Prof

$obj2 = new Lprof() ;
var_dump(get_class($obj2); // Simplon\Literature\Prof
```