

Introduction au langage PHP

Qu'est que le PHP ?

- PHP is HyperText Préprocesseur
- PHPPHP est interprété du côté du serveur.
- PHP supporte de nombreux SGBD (MySQL, Oracle, PostgreSQL, etc.)
- PHP est langage de programmation impératif, qui permet de réaliser une séquence d'instruction
- PHP est un langage de programmation procédurale, qui permet de définir des routines et sous-routines (voire orienté objet)
- PHP est un langage libre
- PHP est adapté à la création de pages web dynamique
- est un langage de script

Langages de script-serveur : Définition

- Un langage de script -serveur est :
 - un programme stocké sur un serveur et exécuté par celui-ci,
 - qui passe en revue les lignes d'un fichier source pour en modifier une partie du contenu,
 - avant de renvoyer à l'appelant (un navigateur par exemple) le résultat du traitement.
- La tâche d'interprétation des ordres à exécuter est déléguée à un composant, souvent appelé moteur,
 - installé sur le serveur,
 - qui est doté d'une API et d'un fonctionnement identique quel que soit la plateforme utilisée pour gérer le serveur

Langages de script-client : Définition

- Le script côté client est traité par la machine qui accueille le logiciel de navigation.
 - Ses résultats peuvent varier en fonction de plate-forme utilisée. Un programme en JavaScript pourra fonctionner sous Netscape et poser problème sous Internet explorer.
 - Les résultats peuvent être différents suivant la machine (PC, Mac)
 - Nécessité de tests importants
 - Ne permettent pas de masquer les sources du programme
 - Sont indépendants du serveur et donc de l'hébergement

Historique

- Première version de PHP a été mis au point au début d'automne par Rasmus Lerdorf, un programmeur Groenlandais avec la nationalité canadienne, en 1994
 - Version appelée à l'époque Personal Home Pages
 - Pour conserver la trace des utilisateurs venant consulter son CV sur son site, grâce à l'accès à une base de données par l'intermédiaire de requêtes SQL
- La version 3.0 de PHP fut disponible le 6 juin 1998
- A la fin de l'année 1999, une version bêta de PHP, baptisée PHP4 est apparue
- En 2004 la version 5.0
- Le début du développement de la version 6 de PHP date de 2005 et a été abandonnée en raison de difficultés d'intégration du support Unicode
- La version actuelle de PHP est la 7.2, la version 7 est sortie en Décembre 2015

INTRODUCTION

Comment ça marche ?



Comment ça marche ?

- Le client web demande une page PHP
- Le serveur web identifie ce fichier dans son SGF
- Le fichier est transmis au module d'interprétation PHP du serveur
- Le code HTML est généré par l'interpréteur à partir du code PHP
- Le serveur web répond au client

Comment ça marche ?

3 programmes

- Serveur Apache (<http://www.apache.org>).
- Interpréteur de code PHP (<http://www.php.net>).
- Base de données MySQL (<http://www.mysql.com>).

Il existe plusieurs paquetages tout prêts pour Windows.

- WAMP Server qui a l'avantage d'être régulièrement mis à jour et disponible en français

NB :Télécharger et installer WAMP (avec PHP 5.4 minimum)

Une fois téléchargé, installez-le en laissant toutes les options par défaut. Il devrait s'installer dans un répertoire comme C:\wamp et créer un raccourci dans le menu *Démarrer*.

Premier contact avec PHP

PHP repose sur des modules:

- Le **module de base**, dit standard, permet d'accéder aux instructions élémentaires, aux différents types de données et à un grand nombre de fonctions.
- Des **modules additionnels** spécialisés permettent d'ajouter des fonctionnalités particulières, comme l'accès aux diverses bases de données et leur gestion

Chaque module donne accès à un grand nombre de fonctions spécialisées pour un domaine particulier.

La liste des modules disponibles actuellement est visible dans la documentation générale du langage sur le site officiel de PHP, à l'adresse <http://www.php.net>.

Premier contact avec PHP

- Le script PHP suivant permet de savoir quels modules vous pouvez utiliser sur votre serveur local.

```
<?php  
    phpinfo();  
?>
```

Comment créer un script PHP ?

1. Créer un fichier avec l'extension **.php** ;

- Un fichier PHP (.php) peut contenir
 - du code HTML
 - du code PHP
 - du code JavaScript

2. Y insérer du code HTML et/ou PHP.

Le code PHP doit être délimité par les balises

- `<script langage="php"> ... </script>`
- `<?php ... ?>`

INTRODUCTION

Comment créer un script PHP ?

A screenshot of a code editor with two tabs: 'info.php' and 'helloWord.php'. The 'helloWord.php' tab is active, showing a PHP script. The script is a simple HTML document with a PHP block that echoes 'Hello World'. The code is color-coded: HTML tags are in green, attributes and values are in yellow, and PHP code is in blue. Line numbers 1 through 13 are visible on the left side of the editor.

```
1 <!DOCTYPE html>
2 <html lang="fr">
3   <head>
4     <meta charset="utf-8" />
5     <meta name="description" content="Exemple de cours" />
6     <title>Mon second exemple</title>
7   </head>
8   <body>
9     <?php
10      echo "Hello World"; //Afficher un message
11    ?>
12   </body>
13 </html>
```

INTRODUCTION

Comment créer un script PHP ?



The image shows a code editor with two tabs: 'info.php' and 'helloWord.php'. The 'info.php' tab is active, displaying the following code:

```
1 <!DOCTYPE html>
2 <html lang="fr">
3     <head>
4         <meta charset="utf-8" />
5         <meta name="description" content="Exemple de cours" />
6         <title>Mon premier exemple</title>
7     </head>
8     <body>
9         <?php phpinfo(); ?>
10    </body>
11 </html>
```

Introduction

- Typologie
 - Toute instruction se termine par un point-virgule
 - Sensible à la casse (Sauf par rapport aux fonctions)
- Les commentaires
 - `/*` commentaire sur plusieurs lignes `*/`
 - `//` un commentaire sur une ligne

Les constantes

- `Define("nom_constant", valeur_constant)`
- `Const "nom_constant" = valeur_constant`

Exemple

- `define ("ma_const", "PHP5")`
- `("an", 2012) ;`
- `const an = 2017`

- Les constantes prédéfinies
 - `NULL`
 - `_FILE_`
 - `_LINE_`
 - `PHP_VERSION`
 - `PHP_OS`
 - `TRUE` et `FALSE`
 - `E_ERROR`

Les variables

- Principe
 - Commencent par le caractère \$
 - N'ont pas besoin d'être déclarées : l'affectation détermine le type de variable

Par exemple

```
<?php
```

```
$a = -123;
```

```
$b = 1.234;
```

```
$c = 1.2e3;
```

```
$d = FALSE;
```

```
$e = TRUE;
```

```
$f = 'Hardy dit \'le gros\'';
```

```
$g = "Laurel et $f";
```

```
?>
```


Les variables

- Fonctions de vérifications de variables
 - Doubleval(), empty(), gettype(), intval(),
 - is_array(), is_bool(), is_double(), is_float(), is_int(), is_integer, is_long(), is_object(), is_real(), is_numeric(), is_string()
 - Isset(), settype(), strval(), unset()
- Affectation par valeur et par référence
 - Affectation par valeur : `$b=$a`
 - Affectation par (référence) variable : `$c = &$a`

Les variables

- Visibilité des variables
 - Variable locale
 - Visible uniquement à l'intérieur d'un contexte d'utilisation
- Variable globale
 - Visible dans tout le script
 - Utilisation de l'instruction `global()` dans des contextes locales

Les variables prédéfinies

- Les variables d'environnement dépendant du client

Variable	Description
<code>\$_SERVER["HTTP_HOST"]</code>	Nom d'hôte de la machine du client (associée à l'adresse IP)
<code>\$_SERVER["HTTP_ACCEPT_LANGUAGE"]</code>	Langue utilisée par le serveur (par défaut en-us)
<code>\$_SERVER["HTTP_ACCEPT"]</code>	Types MIME reconnus par le serveur (séparés par des virgules)
<code>\$_SERVER["REMOTE_ADDR"]</code>	L'adresse IP du client appelant le script CGI
<code>\$_SERVER["PHP_SELF"]</code>	Nom du script PHP

Les variables prédéfinies

- Les variables d'environnement dépendant du serveur

Variable	Description
<code>\$_SERVER["SERVER_NAME"]</code>	Le nom du serveur
<code>\$_SERVER["HTTP_HOST"]</code>	Nom de domaine du serveur
<code>\$_SERVER["SERVER_ADDR"]</code>	Adresse IP du serveur
<code>\$_SERVER["SERVER_PROTOCOL"]</code>	Nom et version du protocole utilisé pour envoyer la requête au script PHP
<code>\$_SERVER["\$DOCUMENT_ROOT"]</code>	Racine des documents Web sur le serveur

Types de données

Principe

- Pas besoin d'affecter un type à une variable avant de l'utiliser
 - La même variable peut changer de type en cours de script
 - Les variables issues de l'envoi des données d'un formulaire sont du type string
- Les différents types de données
 - Les entiers : le type **Integer**, **int**
 - Les flottants : le type **Double**, **real**, **float**
 - Les tableaux : le type **array**
 - Les chaînes de caractères : le type **string**
 - Les objets

Types de données

Il est parfois utile de forcer le type d'une variable. On utilise la fonction **settype** ou bien les **opérateurs de casting (int), (string)**

- **settype** renvoie vrai si la conversion a fonctionné, faux sinon

Exemple

```
$a= 3.1415;
```

```
$result= settype( $a, "integer" ); // => $a = 3 , $result = 1
```

Types de données

Les opérateurs de conversion sont :

- (string) conversion en chaîne de caractères
- (int) conversion en entier, synonyme de (integer)
- (real) conversion en double, synonyme de (double) et(float)
- (array) conversion en tableau
- (object) conversion en objet
- (bool) conversion en booléen

Exemple

`$var= 1; // $var est de type "integer" et vaut 1.`

`$chn=(string) $var ; // $var est de type "string" et vaut " 1 "`

Types de données

Règles des conversions implicites :

- Si la chaîne de caractères contient un **point**, un **e** ou un **E** ainsi que des **caractères numériques**, elle est convertie en **décimal**,
- Si la chaîne de caractères ne contient que des **caractères numériques**, elle est convertie en **entier**,
- Si la chaîne de caractères est composée de **chiffres** et de **lettres**, elle est convertie en **entier** et vaut 0,
- Si la chaîne de caractères contient **plusieurs mots**, seul le premier est pris en compte et est converti selon les règles ci-dessus

Les chaînes de caractères

Principe

- Peuvent être constituées de n'importe quel caractère alphanumérique et de ponctuation, y compris les caractères spéciaux
- Une chaîne de caractères doit être toujours entourée par des guillemets simples (') ou doubles (")
 - Ceci est une chaîne de caractères valide."
 - 'Ceci est une chaîne de caractères valide.'
 - "Ceci est une chaîne de caractères invalide.'
- Des caractères spéciaux à insérer directement dans le texte, permettent de créer directement certains effets comme des césures de lignes

Exemple:

- \t : tabulation horizontale
- \n : nouvelle ligne
- \v : tabulation verticale

Les opérateurs

- Opérateurs arithmétiques
+ ou - ou * ou / ou % ou ++ ou --
- Opérateurs d'affectation
= ou += ou -= ou .= ou /= ou % =
- Opérateur de concaténation de chaînes .
- Opérateurs de concaténation de tableaux +
- Opérateurs de comparaison
== ou != ou <> ou > ou < ou >= ou <=
- Opérateurs logiques
&& ou || ou !
- Opérateur conditionnel
variablename=(condition)?value1:value2

Les opérateurs

■ Opérateurs de calcul

Opérateur	Dénomination	Effet	Exemple	Résultat
+	opérateur d'addition	Ajoute deux valeurs	$\$x+3$	10
-	opérateur de soustraction	Soustrait deux valeurs	$\$x-3$	4
*	opérateur de multiplication	Multiplie deux valeurs	$\$x*3$	21
/	plus: opérateur de division	Divise deux valeurs	$\$x/3$	2.3333333
=	opérateur d'affectation	Affecte une valeur à une variable	$\$x=3$	Met la valeur 3 dans la variable \$x

Les opérateurs

■ Opérateurs d'assignation

Opérateur	Effet
+=	addition deux valeurs et stocke le résultat dans la variable (à gauche)
-=	soustrait deux valeurs et stocke le résultat dans la variable
*=	multiplie deux valeurs et stocke le résultat dans la variable
/=	divise deux valeurs et stocke le résultat dans la variable
%=	donne le reste de la division deux valeurs et stocke le résultat dans la variable
 =	Effectue un OU logique entre deux valeurs et stocke le résultat dans la variable
^=	Effectue un OU exclusif entre deux valeurs et stocke le résultat dans la variable
&=	Effectue un Et logique entre deux valeurs et stocke le résultat dans la variable
.=	Concatène deux chaînes et stocke le résultat dans la variable

Les opérateurs

■ Opérateurs logiques

Opérateur	Dénomination	Effet	Syntaxe
ou OR	OU logique	Vérifie qu'une des conditions est réalisée	((condition1) (condition2))
&& ou AND	ET logique	Vérifie que toutes les conditions sont réalisées	((condition1) && (condition2))
XOR	OU exclusif	Opposé du OU logique	((condition1) XOR (condition2))
!	NON logique	Inverse l'état d'une variable booléenne (retourne la valeur 1 si la variable vaut 0, 0 si elle vaut 1)	(!condition)

Les instructions conditionnelles

- L'instruction if
 - **Syntaxe:** if (condition réalisée) { liste d'instructions }
- L'instruction if ... Else
 - **Syntaxe:** if (condition réalisée) {liste d'instructions}
 else { autre série d'instructions }
- L'instruction if ... elseif ... Else
 - **Syntaxe:**if (condition réalisée) {liste d'instructions}
 elseif (autre condition) {autre série d'instructions }
 else (dernière condition réalisée) { série d'instructions }

Les instructions conditionnelles

- L'instruction switch

- **Syntaxe**

```
switch (Variable) {  
    case Valeur1: Liste d'instructions break;  
    case Valeur1: Liste d'instructions break;  
    case Valeurs...: Liste d'instructions break;  
    default: Liste d'instructions break;  
}
```

Les instructions conditionnelles

- La boucle **for**

- **Syntaxe:** `for ($i=1; $i<6; $i++) { echo "$i
"; }`

- La boucle **while**

- **Syntaxe:** `While(condition) {bloc d'instructions ;}`
 `While (condition) :Instruction1 ;Instruction2 ;`
 `.... endwhile ;`

- La boucle **do...while**

- **Syntaxe:** `Do {bloc d'instructions ;}while(condition) ;`

Les fonctions

- Une série d'instructions qui effectue des actions qui sont répétées plusieurs fois dans le code sans avoir à les réécrire à chaque fois
- Lorsqu'on appelle une fonction, il y a trois étapes.
 - L'entrée: on donne des informations à la fonction en lui passant des paramètres avec lesquelles travailler).
 - Le traitement : grâce aux informations qu'elle a reçues en entrée.
 - La sortie : une fois qu'elle a fini son traitement, la fonction renvoie un résultat

Les fonctions

Syntaxe: **function nomFonction (parametres)**
 {
 // Insérez vos instructions ici
 }

- **function** est le mot clef qui déclare la fonction
- **nomFonction**: c'est le nom de la fonction (pas d'accents, pas d'espaces, etc.)
- **parametres** (correspond à l'entrée) : valeurs avec lesquelles la fonction va travailler. Il peut ne pas y avoir de paramètres

Les fonctions

Exemple

```
<?php
```

```
    function add($x,$y){
```

```
        $total=$x+$y;
```

```
        return $total;
```

```
}
```

```
echo "1 + 16 = " . add(1,16);
```

```
?>
```

Les fonctions

<code>int strlen(string \$ch);</code>	<code>// longueur de \$ch</code>
<code>int strcmp(string \$ch1, string \$ch2);</code>	<code>// compare deux chaînes</code>
<code>string trim(string \$ch) ;</code>	<code>// supprime les espaces en début/fin de chaîne</code>
<code>ltrim(string \$ch);</code>	<code>// supprime les espaces en début</code>
<code>string rtrim(string \$ch);</code>	<code>// supprime les espaces en fin</code>
<code>string ucfirst(\$ch);</code>	<code>// la première lettre en majuscule</code>
<code>string ucwords(\$ch);</code>	<code>// la première lettre de chaque mot</code>
<code>string strtolower(string \$ch);</code>	<code>// tout en minuscules</code>
<code>string strtoupper(string \$ch);</code>	<code>// tout en majuscules</code>
<code>string nl2br(string \$string);</code>	<code>// remplace \n par
</code>
<code>int strpos(string \$ch1, mixed \$ch2)</code>	<code>// vérifie si \$ch2 est sous-chaîne de \$ch1 et renvoi la position de la 1ere occurrence de \$ch1 dans \$ch2</code>

Les fonctions

Exemple

```
<?php
```

```
    echo strlen("Hello world!"); // 12
```

```
    echo strpos("Hello world!","world");// 6
```

```
?>
```

Les tableaux

- Un tableau (aussi appelé array) est une variable qui permet d'enregistrer de nombreuses informations. On en distingue deux types de tableaux:
 - Les tableaux à indice ou numérotés
 - Les tableaux associatifs

Les tableaux : tableaux à indice

- Les tableaux à indice ou numérotés associent des clés indexés à une valeur
- Accéder au éléments du tableau a l'intermédiaire de numéros .

- Syntaxe

`$tableau[indice] = valeur;`

Exemple

`$jour[3] = "Mercredi"; $note[0] = 20;`

- Initialisation du tableau avec **array**

- Syntaxe

`$tableau = array(valeur0, valeur1,..., valeurN);`

Exemple

`$jour = array("Dimanche", "Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi", "Samedi");`

Les tableaux : tableaux associatif

- Dans un tableaux associatif ou table de hachage les éléments sont référencés par des chaînes de caractères associatives en guise de nom: la clé d'index.

- Syntaxe

`$tableau[indice] = valeur;`

Exemple

```
$jour["Dimanche"] = 7 ;    $jour["Mercredi"] = "Le jour des enfants";
```

- Initialisation du tableau avec **array**

- Syntaxe

`$tableau = array(ind0 => val0, ind1 => val1,..., indN => valN);`

Exemple

```
$jour = array("Dimanche" => 1, "Lundi" => 2, "Mardi" => 3, "Mercredi" => 4, "Jeudi" => 5,  
"Vendredi" => 6, "Samedi" => 7);
```


Les tableaux : tableaux Multidimensionnel

- Dans un tableaux associatif ou table de hachage les éléments sont référencés par des chaînes de caractères associatives en guise de nom: la clé d'index.

- Syntaxe

`$tableau[indice] = valeur;`

Exemple

```
$jour["Dimanche"] = 7 ;    $jour["Mercredi"] = "Le jour des enfants";
```

- Initialisation du tableau avec **array**

- Syntaxe

`$tableau = array(ind0 => val0, ind1 => val1,..., indN => valN);`

Exemple

```
$jour = array("Dimanche" => 1, "Lundi" => 2, "Mardi" => 3, "Mercredi" => 4, "Jeudi" => 5,  
"Vendredi" => 6, "Samedi" => 7);
```

PHP et formulaires

Principe

- Les formulaires HTML permettent de faire saisir des données par l'utilisateur via son navigateur.
- Ces données sont saisies dans des champs appelés **inputs**, qui sont définis avec la balise `<input>`.
- Les données sont ensuite récupérées dans un script PHP.

Note: Ces données doivent impérativement être testées et filtrées pour des raisons de sécurité.

Principe

- Un formulaire est créé par une balise `<form>` avec les attributs suivants
 - La méthode de transmission des données (GET ou POST)
 - l'action, qui est l'URL du script (ici un script PHP) qui va récupérer les données du formulaire.
 - Le nom du formulaire (optionnel)
- Le contenu peut texte et ou des balises représentant les différents composants de saisies d'un formulaire (champ de texte, bouton radio, liste déroulante, bouton de soumission, etc...)

Principe: envoi de données

- Lorsque l'utilisateur clique sur le bouton d'envoi ou de soumission (de type submit), une requête HTTP est envoyée au serveur à destination du script désigné par l'attribut `action` de la balise `<form>`
- La requête contient les associations
nom du champ \leftrightarrow valeur(s)
- Les associations se trouvent :
 - soit dans l'enveloppe HTTP si la méthode POST est utilisée
 - soit dans l'URL s'il s'agit de la méthode GET

Principe: Réception de données

- La réception des données se fait par la méthode POST (comme indiqué dans la balise <form> avec l'attribut `method`).
- Les données sont récupérées dans un tableau associatif `$_POST`, dont les clefs sont les attributs `name` des `inputs`
- On teste si ces attributs existent bien via la fonction `isset`

Exemple:

Créer un formulaire **formulaire.php** ayant les champ prénom et nom et un fichier **traitement.php** qui affichera le prénom et le nom

```
<!DOCTYPE html>
<html>
<Head>
  <title> Mon formulaire</title>
  <link rel="stylesheet" type="text/css" href="css/bootstrap.css" />
  <link rel="stylesheet" type="text/css" href="css/monSyle.css">
</head>
<body>
<div class="container">
  <form action="traitement.php" method="POST">
    <h3>Formulaire de saisi</h3>
    <div class="row">
      <label class="col-md-1">Pr&eacutenom:</label> <input type="text" name="prenom"
        id="prenom">
    </div>
    <div class="row">
      <label class="col-md-1">Nom:</label><input type="text" name="nom" id="nom">
    </div>
    <input type="submit" name="">
  </form>
</div>
</body>
</html>
```

```
<?php
if (isset($_POST['nom']) && isset($_POST['prenom']))
{
    $nom=$_POST['nom'];
    $prenom=$_POST['prenom'];
    if ($nom!="" && $prenom!="")
    {
        echo "Bonjour " . $prenom . " " . $nom;
    }
    else
    {
        echo "champ(s) vide(s)";
    }
}

?> </html>
```