

INITIATION PHP

ISTA NTIC MARRAKECH/Mme
oussimour

PHP : C'est QUOI ?

■ Définition

- Un langage de scripts permettant la création d'applications Web
- Indépendant de la plate-forme utilisée puisqu'il est exécuté côté serveur et non côté client.
- La syntaxe du langage provient de celles du langage C, du Perl et de Java.
- Ses principaux atouts sont:
 - La gratuité et la disponibilité du code source (PHP4 est distribué sous licence GNU GPL)
 - La simplicité d'écriture de scripts
 - La possibilité d'inclure le script PHP au sein d'une page HTML
 - La simplicité d'interfaçage avec des bases de données
 - L'intégration au sein de nombreux serveurs web (Apache, Microsoft IIS, ...)

Intégration PHP et HTML (1)

■ Principe

- Les scripts PHP sont généralement intégrés dans le code d'un document HTML
- L'intégration nécessite l'utilisation de balises
 - avec le style xml : `<? ligne de code PHP ?>`
 - Avec le style php: `<?php ligne de code PHP ?>`
 - avec le style JavaScript :
`<script language=«php» ligne de code PHP </script>`
 - avec le style des ASP : `<% ligne de code ASP %>`

Intégration PHP et HTML (2)

■ Forme d'une page PHP

● Intégration directe

```
< ?php
//ligne de code PHP
?>

<html>
<head> <title> Mon script PHP
</title> </head>
<body>
//ligne de code HTML
< ?php
//ligne de code PHP
?>
//ligne de code HTML
...
</body> </html>
```

Intégration PHP et HTML (3)

- Forme d'une page PHP
 - Inclure un fichier PHP dans un fichier HTML : include()

Fichier Principal

```
<html>
<head>
<title> Fichier d'appel </title>
</head>
<body>
<?php
$salut = " BONJOUR" ;
include "information.inc" ;
?>
</body>
</html>
```

Fichier à inclure : information.inc

```
<?php
$chaine=$salut. " , C'est PHP " ;
echo " <table border= \"3\"
<tr> <td width = \" 100%\ " >
    <h2> $chaine</h2>
    </td> </tr></table> ";
?>
```

Intégration PHP et HTML (4)

■ Envoi du code HTML par PHP

- La fonction echo : `echo Expression;`
 - `echo "Chaine de caracteres";`
 - `echo (1+2)*87;`
- La fonction print : `print(expression);`
 - `print("Chaine de caracteres");`
 - `print ((1+2)*87);`
- La fonction printf : `printf (chaîne formatée);`
 - `printf ("Le périmètre du cercle est %d", $Perimetre);`

Syntaxe de base : Introduction

■ Typologie

- Toute instruction se termine par un point-virgule
- Sensible à la casse
 - Sauf par rapport aux fonctions

■ Les commentaires

- `/* Voici un commentaire! */`
- `//` un commentaire sur une ligne

Syntaxe de base : Les constantes

■ Les constantes

- **Define**("nom_constante", valeur_constante)
 - `define ("ma_const", "Vive PHP4") ;`
 - `define ("an", 2002) ;`
- Les constantes prédéfinies
 - `NULL`
 - `_FILE_`
 - `_LINE_`
 - `PHP_VERSION`
 - `PHP_OS`
 - `TRUE` et `FALSE`
 - `E_ERROR`

Syntaxe de base : Les variables (1)

■ Principe

- Commencent par le caractère **\$**
- N'ont pas besoin d'être déclarées

■ 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**

Syntaxe de base : Les variables(2)

■ 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

```
<?
$var = 100;
function test(){
global $var;
return $var;}
$resultat = test();
if ($resultat) echo $resultat; else echo " erreur ";
?>
```

Syntaxe de base : Les variables(3)

■ Les variables dynamiques

- Permettent d'affecter un nom différent à une autre variable

```
$nom_variable = 'nom_var';  
$$nom_variable = valeur; // équivaut à $nom_var =  
valeur;
```

- Les variables tableaux sont également capables de supporter les noms dynamiques

```
$nom_variable = array("val0", "val1", ..., "valN");  
${$nom_variable[0]} = valeur; $val0 = valeur;  
$nom_variable = "nom_var";  
${$nom_variable}[0] = valeur;  
$nom_var[0] = valeur;
```

- Les accolades servent aussi à éviter toute confusion lors du rendu d'une variable dynamique

```
echo "Nom : $nom_variable - Valeur :  
${$nom_variable}";  
// équivaut à echo "Nom : $nom_variable - Valeur :  
$nom_var";
```

Syntaxe de base : Les variables (4)

■ 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_REFERER"]</code>	URL de la page qui a appelé le script PHP
<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["CONTENT_TYPE"]</code>	Type de données contenu présent dans le corps de la requête. Il s'agit du type MIME des données
<code>\$_SERVER["REMOTE_ADDR"]</code>	L'adresse IP du client appelant le script CGI

Syntaxe de base : Les variables (5)

■ Variables prédéfinies

➤ Les variables d'environnement dépendant du

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["DATE_GMT"]</code>	Date actuelle au format GMT
<code>\$_SERVER["DATE_LOCAL"]</code>	Date actuelle au format local
<code>\$_SERVER["\$DOCUMENT_ROOT"]</code>	Racine des documents Web sur le serveur

Syntaxe de base : Les variables (6)

■ Variables prédéfinies

➤ Affichage des variables d'environnement

- la fonction `phpinfo()`

- `<? phpinfo(); ?>`
- `echo phpinfo(constante);`

INFO_CONFIGURATION	affiche les informations de configuration.
INFO_CREDITS du module PHP	affiche les informations sur les auteurs
INFO_ENVIRONMENT	affiche les variables d'environnement.
INFO_GENERAL PHP.	affiche les informations sur la version de
INFO_LICENSE	affiche la licence GNU Public
INFO_MODULES associés à PHP	affiche les informations sur les modules
INFO_VARIABLES	affiche les variables PHP prédéfinies.

- la fonction `getenv()`

- `<? echo getenv("HTTP_USER_AGENT");?>`

Syntaxe de base : Les 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**
- Les flottants : le type **Double**
- Les tableaux : le type **array**
- Les chaînes de caractères : le type **string**
- Les objets

Syntaxe de base : Les types de données (2)

■ Le transtypage

- La fonction **settype()** permet de convertir le type auquel appartient une variable

```
<? $nbre=10;
    Settype($nbre, " double ");
    Echo " la variable $nbre est de type " , gettype($nbre); ?>
```

- Transtypage explicite : le cast

➤ (int), (integer) ; (real), (double), (float); (string); (array); (object)

```
<? $var=" 100 FRF ";
    Echo " pour commencer, le type de la variable est $var, gettype($var);
    $var =(double) $var;
    Echo <br> Après le cast, le type de la variable est $var ", gettype($var);
    Echo "<br> et a la valeur $var ";      ?>
```

■ Détermination du type de données

- Gettype(), Is_long(), Is_double(), Is_string(), Is_array(), Is_object(), Is_bool()

Syntaxe de base : Les chaînes de caractères(1)

■ Principe

- Peuvent être constituées de n'importe quel caractère alphanumérique et de ponctuation, y compris les caractères spéciaux

`\tLa nouvelle monnaie unique, l' €uro, est enfin là...\n\r`

- 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

Car	Code ASCII	Code hex	Description
<code>\car</code>			échappe un caractère spécifique.
<code>" "</code>	32	0x20	un espace simple.
<code>\t</code>	9	0x09	tabulation horizontale
<code>\n</code>	13	0x0D	nouvelle ligne
<code>\r</code>	10	0x0A	retour à chariot
<code>\0</code>	0	0x00	caractère NUL
<code>\v</code>	11	0x0B	tabulation verticale

Syntaxe de base : Les chaînes de caractères(2)

■ Quelques fonctions de manipulation

`chaîne_result = addCSlashes(chaîne, liste_caractères);`

ajoute des slashes dans une chaîne

`chaîne_result = addslashes(chaîne);`

ajoute un slash devant tous les caractères spéciaux.

`chaîne_result = chop(chaîne);`

supprime les espaces blancs en fin de chaîne.

`caractère = chr(nombre);`

retourne un caractère en mode ASCII

`chaîne_result = crypt(chaîne [, chaîne_code])`

code une chaîne avec une base de codage.

`echo expression_chaîne;`

affiche à l'écran une ou plusieurs chaînes de caractères.

`$tableau = explode(délimiteur, chaîne);`

scinde une chaîne en fragments à l'aide d'un délimiteur et retourne un tableau.

Syntaxe de base : les opérateurs (1)

■ Les opérateurs

- les opérateurs de calcul
- les opérateurs d'assignation
- les opérateurs d'incrémentation
- les opérateurs de comparaison
- les opérateurs logiques
- les opérateurs bit-à-bit
- les opérateurs de rotation de bit

Syntaxe de base : Les opérateurs(2)

■ Les 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	$x/3$	2333333
=	opérateur d'affectation	Affecte une valeur à une variable	$x=3$	Mettre la valeur 3 dans la variable x

Syntaxe de base : Les opérateurs(3)

■ Les opérateurs d'assignation

Opérateur	Effet
+=	addition de 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 de 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

Syntaxe de base : Les opérateurs(4)

■ Les opérateurs d'incrémentation

Opérateur	Dénomination	Effet	Syntaxe	Résultat (avec x valant 7)
++	Incrémentation	Augmente d'une unité la variable	\$x++	8
--	Décrémentation	Diminue d'une unité la variable	\$x--	6

Opérateur	Dénomination	Effet	Exemple	Résultat
=	opérateur d'égalité	Compare deux valeurs et vérifie leur égalité	\$x==3	Retourne 1 si \$X est égal à 3, sinon 0
<	opérateur d'infériorité stricte	Vérifie qu'une variable est strictement inférieure à une valeur	\$x<3	Retourne 1 si \$X est inférieur à 3, sinon 0
<=	opérateur d'infériorité	Vérifie qu'une variable est inférieure ou égale à une valeur	\$x<=3	Retourne 1 si \$X est inférieur à 3, sinon 0
>	opérateur de supériorité stricte	Vérifie qu'une variable est strictement supérieure à une valeur	\$x>3	Retourne 1 si \$X est supérieur à 3, sinon 0
>=	opérateur de supériorité	Vérifie qu'une variable est supérieure ou égale à une valeur	\$x>=3	Retourne 1 si \$X est supérieur ou égal à 3, sinon 0
!=	opérateur de différence	Vérifie qu'une variable est différente d'une valeur	\$x!=3	Retourne 1 si \$X est différent de 3, sinon 0

Syntaxe de base : Les opérateurs(5)

■ Les 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)

Opérateur	Dénomination	Effet	Syntaxe	Résultat
&	ET bit-à-bit	Retourne 1 si les deux bits de même poids sont à 1	9 & 12 (1001 & 1100)	8 (1000)
	OU bit-à-bit	Retourne 1 si l'un ou l'autre des deux bits de même poids est à 1 (ou les deux)	9 12 (1001 1100)	13 (1101)
^	OU bit-à-bit	Retourne 1 si l'un des deux bits de même poids est à 1 (mais pas les deux)	9 ^ 12 (1001 ^ 1100)	5 (0101)
~	Complément (NON)	Retourne 1 si le bit est à 0 (et inversement)	~9 (~1001)	6 (0110)

Syntaxe de base : Les opérateurs(6)

■ Les opérateurs de rotation de bit

Opérateur	Dénomination	Effet	Syntaxe	Résultat
<<	Rotation à gauche	Décale les bits vers la gauche (multiplie par 2 à chaque décalage). Les zéros qui sortent à gauche sont perdus, tandis que des zéros sont insérés à droite	6 << 1 (110000) << 1)	12 (1100)
>>	Rotation à droite avec conservation du signe	Décale les bits vers la droite (divise par 2 à chaque décalage). Les zéros qui sortent à droite sont perdus, tandis que le bit non nul de poids plus fort est recopié à gauche	6 >> 1 (011000 >> 1)	3 (0011)

Opérateur	Dénomination	Effet	Syntaxe	Résultat
.	Concaténation	Joint deux chaînes bout à bout	"Bonjour"."Au revoir"	"BonjourAu revoir"
\$	Référencement variable	Permet de définir une variable	\$M=Variable=2;	
->	Propriété d'un objet	Permet d'accéder aux données membres d'une classe	\$MonObjet->Propriete	

Syntaxe de base : Les opérateurs(7)

■ Les priorités

[illegible]

Syntaxe de base : Les instructions conditionnelles(1)

■ L'instruction if

- if (condition réalisée) { liste d'instructions }

■ L'instruction if ... Else

- if (condition réalisée) {liste d'instructions}
else { autre série d'instructions }

■ L'instruction if ... elseif ... Else

- if (condition réalisée) {liste d'instructions}
elseif (autre condition) {autre série d'instructions }
else (dernière condition réalisée) { série d'instructions }

■ Opérateur ternaire

- (condition) ? instruction si vrai : instruction si faux

Syntaxe de base : Les instructions conditionnelles(2)

■ L'instruction switch

```
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;  
}
```

Syntaxe de base : Les instructions conditionnelles(3)

■ La boucle for

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

■ La boucle while

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

■ La boucle do...while

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

■ La boucle foreach (PHP4)

- `Foreach ($tableau as $valeur) {insts utilisant $valeur ;}`