



Architecture N-tiers et Développement Web

Plan du cours

- Chapitre 01: Introduction aux technologies de Web et Architecture N-tiers
 - Chapitre 02: Le langage HTML
 - Chapitre 03: Le langage CSS
 - Chapitre 04: Le Langage Javascript
 - **Chapitre 05: Le langage PHP et connexion MySQL**
-
- The diagram illustrates the course structure. It features a vertical red bracket on the right side, spanning from the end of Chapter 4 up to the start of Chapter 5. This bracket groups the first four chapters (Chapitre 01 to Chapitre 04) under the label "Développement frontend". Below this, another vertical bracket groups Chapter 5 under the label "Développement backend". The background behind the brackets is a light orange for the frontend section and a light blue for the backend section.
- Développement frontend
- Développement backend

Introduction

- **PHP : Hypertext Preprocessor** (proposé en 1995)
- Proche du C,Java
- Un langage de scripts: gestion des sites web dynamiques
- Le but du langage est d'écrire rapidement des pages HTML dynamiques.
- Créer des pages interactives : saisir des données, vérifier, etc.
- Transmission des données vers le serveur
- Créer des sites de diffusion, de collecte d'information, e-commerce, blogs, etc.
- Très bonnes performances (améliorations de l'ordre de 50% de la vitesse d'exécution)
- Grand succès et utilisation par de très grands sites
- beaucoup de code libre disponible.
- des dizaines de millions de sites Web l'utilisent à travers le monde...

Introduction

- PHP : indépendant de la plateforme utilisée (exécuté côté serveur)
 - *il est interprété par le serveur Web*
- PHP : Open source et gratuit
- PHP : accéder aux base de données
- PHP : exécution sur le serveur et renvoie du code HTML au client
- Nécessité d'avoir un serveur distant ou local
- Plusieurs éléments :serveur apache, interpréteur code PHP, Base de données
- MySQL, Utilitaire phpMyAdmin, etc.
- Exemple de paquetage complet : XAMPP, WampServer

Introduction

- PHP a besoin d'un serveur Web pour fonctionner
- Les pages demandées par un client sont construites par le serveur Web en fonction des paramètres transmis avant d'être retournés au client
- **Site dynamique** : Pour des traitements plus lourds et plus complexe nécessitant l'accès à une base de données, ...
- PHP et MySQL...



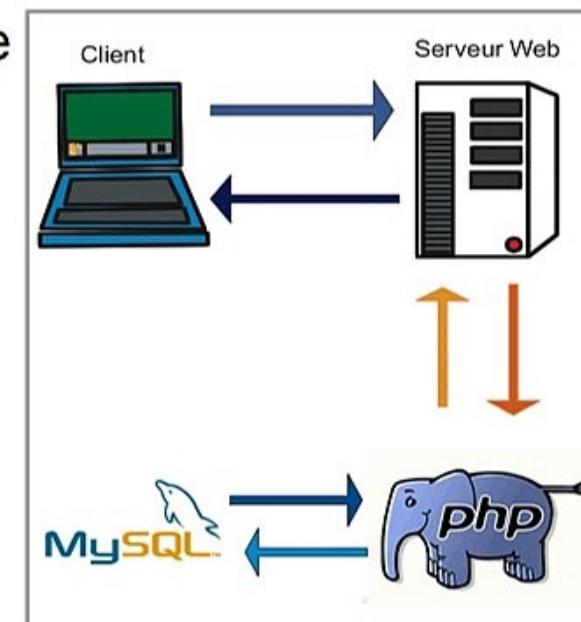
Introduction

- Voici, en simplifiant, ce qui se passe lorsque vous consultez une page html:
 - Le navigateur envoie l'adresse URL tapée
 - Le serveur web est un "ordinateur" présent sur l'Internet et qui héberge la page demandée
 - Sur ce serveur, on trouve **Apache**, logiciel apte à traiter les requêtes HTTP
 - Apache cherche le fichier demandé et renvoie à votre navigateur la page HTML
 - Votre navigateur interprète les différents langages se trouvant dans ce fichier (HTML, JavaScript, etc.) et affiche la page

Introduction

■ Maintenant, voyons ce qui se passe lorsque votre page HTML contient du code PHP :

- Le serveur regarde si le fichier envoyé contient une extension .php
- Si oui, il transmet le fichier à PHP qui l'analyse et l'exécute
- Si le code contient des requêtes vers MySQL, PHP envoie la requête SQL à MySQL
- La base de données renvoie les informations voulues au script qui peut les exploiter
- PHP continue d'analyser la page, puis retourne le fichier dépourvu du code PHP au serveur web
- Le serveur web renvoie donc un fichier ne contenant plus de PHP, donc seulement du HTML au navigateur qui l'interprète et l'affiche



Environnement de travail

■ Utiliser PHP sur son ordinateur

- Pourquoi installer PHP sur son ordinateur ?
 - Pour tester vos script PHP, vous allez être amenés à les envoyer sur votre hébergeur, sur Internet
 - Cependant il devient vite très lourd de sans cesse renvoyer ces fichiers par FTP
 - C'est pourquoi installer un serveur web sur son ordinateur est utile, et permet de tester ses scripts plus souplement
- Concrètement, votre ordinateur sera à la fois client et serveur
 - Ainsi vous pourrez programmer en PHP sans avoir besoin d'être connecté à Internet
- Pour cela, il existe plusieurs utilitaires très pratiques qui installeront Apache

Environnement de travail

- **WAMP/LAMP?:**

Windows/Linux/ — Apache — MySQL — PHP

- Apache : c'est un serveur web. Il s'agit du plus important de tous les programmes, car c'est lui qui est chargé de délivrer les pages web aux visiteurs. Cependant, Apache ne gère que les sites web statiques (il ne peut traiter que des pages HTML). Il faut donc le compléter avec d'autres programmes.
- PHP : c'est un plug-in pour Apache qui le rend capable de traiter des pages web dynamiques en PHP.
- En clair, en combinant Apache et PHP, notre ordinateur sera capable de lire des pages web en PHP.
- **MySQL** : c'est le logiciel de gestion de bases de données.

Environnement de travail

- Il existe plusieurs paquetages tout prêts pour **Windows**. Tel que:
 - **XAMPP**
 - **WAMP Server**
 - **EasyPHP**
 - ...

Les bases du PHP: Intégration PHP et HTML (1)

- Les scripts PHP sont généralement intégrés dans le code d'un document HTML
- Le fichier porte le suffixe .php
- L'intégration nécessite l'utilisation de balises
 - Avec le style php: <?php ligne de code PHP ?>

Intégration PHP et HTML (2)

- Exemple:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Premier exemple PHP</title>
</head>
<body>
<?php
    echo "Hello World ! ";
?>
</body>
</html>
```



```
<html>
<head>
    <title>Titre</title>
</head>
<body>
    Hello World !
</body>
</html>
```

Intégration PHP et HTML (3)

- Forme d'une page PHP

- Inclure un fichier PHP dans un fichier HTML : include() ou require()

Fichier Principal

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

Fichier à inclure : information.php

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

Intégration PHP et HTML (3)

▪ Exécution :

- Démarrer le service Apache
- `http://localhost/Votre_dossier/exemple0.php`

Fichier Principal

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

Fichier à inclure : information.php

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

BONJOUR,C'est PHP

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 caractères");
➤ print ((1+2)*87);

- La fonction **printf** : printf (chaîne formatée);

➤ printf ("Le périmètre du cercle est %d",\$Perimetre);

```
<?php  
for ($i=1; $i<=6;$i++)  
{  
echo "<br>";  
echo "<font size= $i >";  
echo "voici une commande  
<b>echo</b> avec des  
<i>balises</i>html";  
}  
?>
```

voici une commande echo avec des baliseshtml
voici une commande echo avec des baliseshtml

Syntaxe de base : Introduction

- Toute instruction se termine par un point-virgule ;
- Les commentaires:
 - */* Voici un commentaire! */*
 - *// un commentaire sur une ligne*

Syntaxe de base : Les variables (1)

- Une variable commence par un dollar « **\$** » suivi d'un nom de variable.
Le nom est **sensible à la casse**
=> **var** et **Var** ne sont pas les mêmes variables
- Les variables ne sont pas typées au moment de leur création.

Voici les règles à respecter :

- Une variable peut commencer par une lettre
- Une variable peut commencer par un underscore « **_** »
- Une variable **ne doit pas** commencer par un chiffre.

Syntaxe de base : Les variables (2)

- // Déclaration et règles

Exemple:

```
$var=1;
```

```
$Var=2;
```

```
$_toto='Salut';
```

```
$3number=5; // Invalide : commence par un chiffre
```

Syntaxe de base : Les variables (3)

- Le type des variable dépend de sa valeur et de son contexte d'utilisation.
- Mais on peut forcer (**cast**) ponctuellement une variable à un type de données, ce qui s'appelle le **transtypage**.
- PHP effectue **automatiquement** un **transtypage**

Syntaxe de base : Les variables (4)

■ Déclaration et transtypage

```
$var='2'; // Une chaîne 2
```

```
$var+=1; // $var est maintenant un entier 3
```

```
$var=$var+0.3; // $var est maintenant un réel de type double 3.3
```

```
$var=5 + "3 petits enfants"; // $var est un entier qui vaut 8
```

// il cast 3 petits... en entier => 3

Syntaxe de base : Les variables (5)

■ Fonctions de vérifications de variables

- **empty()**: permet de vérifier si la variable passée en paramètre est vide ou non.
- **isset()**: permet de vérifier si la variable passée en paramètre existe ou non.
- **unset()**: permet de supprimer la variable passée en paramètre.
- **gettype()**: permet de retourner le type de la variable passée en paramètre
- **Doubleval()**: renvoie la valeur flottante d'une variable,
- **intval()** : Retourne la valeur numérique entière équivalente d'une variable,
- **settype()**: Affecte un type à une variable
- **strval()** Récupère la valeur d'une variable, au format chaîne
- **is_numeric()**: vérifie si la variable passée en paramètre ne contient qu'une suite de caractères numériques. **-valeur-**
- **is_int()**: vérifie si la variable passée en paramètre est de type entier ou non. **-type-**
- **is_array(), is_bool(), is_double(), is_float(), is_integer, is_long(), is_object(), is_string()**

Syntaxe de base : Les variables (5)

- **Affectation par valeur et par référence**

- Affectation par valeur : **\$b=\$a**
- Affectation par (référence) variable : **\$c = &\$a**

- **Exemple:**

```
$a = 0;  
$b= 2;  
$b= &$a;  
$b=4;
```

```
//$a= 4;  
//$b = 4;
```

■ Portée des variables

- Par défaut, toutes les variables sont locales
- Leur portée se réduit à la fonction ou au bloc de leur déclaration
- Une variable déclarée dans un script et hors d'une fonction est **globale** mais par défaut sa portée est limitée au script courant, ainsi qu'au code éventuellement inclus (include, require)
- Pour accéder à une variable globale dans une fonction, il faut utiliser le mot-clé **global**.

Syntaxe de base : Les variables (7)

- Exemple:

```
<?php
$a=1; // globale par défaut
$b=2; // idem
function test() {
global $a,$b;
                    // on récupère les variables globales
$b=$a+$b;
}
test(); // appel de la fonction
echo $b;
// affiche 3
?>
```

Syntaxe de base : Les variables (7)

- **Variables statiques**

- PHP accepte les variables **statiques**. Comme en C une variable statique ne perd pas sa valeur quand on sort d'une fonction.

```
function test_static() {  
    static $a=0;  
    echo $a; // +1 à chaque passage dans la fonction  
    $a++;}  
  
test_static(); //affiche 0  
test_static(); //affiche 1  
test_static(); //affiche 2
```

Syntaxe de base : Les variables (7)

- Les variables dynamiques : Permettent d'affecter un nom différent à une autre variable

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

```
$nom_variable = 'nom_var';
$$nom_variable=10;
echo $nom_variable . "<br>";
echo $nom_var . "<br>";
```

```
nom_var
10
```

- Variables dynamiques: créer des variables par indiçage

Exemple :

```
<?php

$v1 = "15 €";
$v2 = "30 €";
$v3 = "dvd";
for($i=1;$i<=3;$i++)
echo ${"v".$i}. "<br/>";
?>
```

```
15 €
30 €
dvd
```

Syntaxe de base : Les variables (7)

▪ Constantes :

- On les définit à l'aide de la fonction **define()**

• Exemple:

```
define("LST", "GI");
echo LST;      //affiche GI
```

▪ Interprétation des variables

- À l'intérieur d'une chaîne entre guillemets, les variables sont automatiquement remplacées par leur valeur

• Exemple

```
$lst = "GI";
$chaine ="Dép. Math & Info, Filière $lst , FST SETTAT ";
echo $chaine;
//affiche Dép. Math & Info, Filière GI , FST SETTAT
```

Syntaxe de base : Les variables (8)

■ Variables Prédéfinies

- PHP dispose d'un grand nombre de variables prédéfinies. Ces variables sont généralement des tableaux.
- Elles sont souvent de type **superglobales**, c'est à dire accessible depuis n'importe où sans notion de portée.

Syntaxe de base : Les variables (8)

■ Variables Prédéfinies

Voici quelques tableaux prédéfinis :

- **`$_GLOBALS`** : tableau des variables globales. La clé est le nom de la variable.
- **Exemple**

```
<?php
function test() {
    $var = "variable locale";

    echo '$var dans le contexte global : ' . $_GLOBALS["var"] . "<br>";
    echo '$var dans le contexte courant : ' . $var . "<br>";
}

$var = "Exemple de contenu";
test();
?>
```

\$var dans le contexte global : Exemple de contenu
\$var dans le contexte courant : variable locale

Syntaxe de base : Les variables (8)

▪ Variables Prédéfinies

Voici quelques tableaux prédéfinis :

- **`$_GLOBALS`** : tableau des variables globales. La clé est le nom de la variable.
- **`$_SERVER`** : variables fournies par le serveur web, par exemple 'SERVER_NAME'
`echo $_SERVER['SERVER_NAME'] // Affiche localhost`
- **`$_GET`** : variables fournies par HTTP par la méthode GET (formulaires)
- **`$_POST`** : idem mais pour la méthode POST
- **`$_COOKIE`** : les variables fournies par un cookie
- **`$_FILES`** : variables sur le téléchargement d'un fichier (upload)
- **`$_ENV`** : accès aux variables d'environnement du serveur
- **`$_SESSION`** : les variables de session

Syntaxe de base : Les types de données (1)

- **Principe**

- 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 **Float**
- Les tableaux : le type **Array**
- Les chaînes de caractères : le type **String**
- Les objets: Le type **Object**
- Les booléens : le type **Boolean**
- « Nul » : **NULL**

Syntaxe de base : Les types de données (2)

■ Le transtypage (Conversion de type)

Le transtypage permet de convertir le type d'une variable dans un autre type explicite.

```
$nbre=10;  
settype($nbre, "double");  
echo "la variable $nbre est de type".gettype($nbre);  
//la variable 10 est de type double  
-----  
  
$nb = 10; // $nb est un entier  
$nbr= (double) $nb; // $nbr est un double
```

Syntaxe de base : Les types de données (4)

■ Le transtypage

- Transtypage explicite : le cast
 - (int), (integer) ; (double), (float); (string); (array); (object)

```
<?php  
$v="100 FRF";  
echo "pour commencer, le type de la variable $v est ".gettype($v);  
$v =(double)$v;  
echo "<br> Après le cast, le type de la variable est ".gettype($v);  
echo "<br>la valeur est ".\$v;  
?>
```

pour commencer, le type de la variable 100 FRF est string
Après le cast, le type de la variable est double
la valeur est 100

Syntaxe de base : Les chaînes de caractères(1)

■ Exemple :

```
echo 'On aura l\'examen de l\'algorithmique la  
semaine prochaine. ';  
  
$var=2345;  
  
echo "la valeur de \$var est $var ";
```

On aura l'examen de l'algorithmique la semaine prochaine. la valeur de \$var est 2345

Syntaxe de base : Les chaînes de caractères(2)

- On peut facilement concaténer deux chaînes avec l'opérateur point « . ».
- On peut ajouter du texte à une chaîne avec l'opérateur point égal « .= ».

Exemple1 :

```
<?php  
$str="Salut ! ";  
$str.="Comment ça va ?"; // "Salut ! Comment ça va ?  
$str2=$str." .....!"; // "Salut ! Comment ça va ? .....!"  
echo $str2;  
?>
```

Salut ! Comment ça va ?!

Syntaxe de base : Les chaînes de caractères(3)

- **Exemple 2** : écrire le nom de la variable au milieu du texte et il sera remplacé par sa valeur en cas de double guillemets.

```
<?php  
$age_du_visiteur = 17;  
echo "Le visiteur a $age_du_visiteur ans";  
?>
```

Le visiteur a 17 ans

- **Exemple 3** : écrire la variable en dehors des guillemets et séparer les éléments les uns des autres à l'aide d'un point.

```
<?php  
$age_du_visiteur = 17;  
echo 'Le visiteur a ' . $age_du_visiteur . ' ans';  
?>
```

Le visiteur a 17 ans

Syntaxe de base : Les chaînes de caractères(4)

- **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.*//un alias de la fonction rtrim()*

- `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'incrémantation
- les opérateurs de comparaison
- les opérateurs logiques

Syntaxe de base : les opérateurs (2)

■ Les opérateurs de calcul

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

Syntaxe de base : les opérateurs (3)

■ Les opérateurs d'assignation

Opérateur	Effet
<code>+=</code>	addition deux valeurs et stocke le résultat dans la variable (à gauche)
<code>-=</code>	soustrait deux valeurs et stocke le résultat dans la variable
<code>*=</code>	multiplie deux valeurs et stocke le résultat dans la variable
<code>/=</code>	divise deux valeurs et stocke le résultat dans la variable
<code>%=</code>	donne le reste de la division deux valeurs et stocke le résultat dans la variable
<code> =</code>	Effectue un OU logique entre deux valeurs et stocke le résultat dans la variable
<code>^=</code>	Effectue un OU exclusif entre deux valeurs et stocke le résultat dans la variable
<code>&=</code>	Effectue un Et logique entre deux valeurs et stocke le résultat dans la variable
<code>.=</code>	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)
++	Incrémantation	Augmente d'une unité la variable	\$x++	8
--	Décrémentation	Diminue d'une unité la variable	\$x--	6

■ Les opérateurs de comparaison

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)

Syntaxe de base : Les instructions conditionnelles

- 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
 - **Exemple:**
`$note= 14;
$valide= ($note >= 10) ? true : false;`

Syntaxe de base : Les instructions conditionnelles

- 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 boucles

- **La boucle for**

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

- **La boucle while**

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

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

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

- **La boucle foreach**

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

Syntaxe de base : Les fonctions

■ Déclaration et appel d'une fonction

Function *nom_fonction*(\$arg1, \$arg2, ...\$argn)

{

 déclaration des variables ; bloc d'instructions ;

 //fin du corps de la fonction

return \$resultat ;

}

Appel : *nom_fonction(arg1, arg2, ...)*

■ Fonction avec nombre d'arguments inconnu

- **func_num_args()** : fournit le nombre d'arguments qui ont été passés lors de l'appel de la fonction
- **func_get_arg(\$i)** : retourne la valeur de la variable située à la position \$i dans la liste des arguments passés en paramètres.
 - Ces arguments sont numérotés à partir de 0

Syntaxe de base : Les fonctions

■ Fonction avec nombre d'arguments inconnu

```
<?php
function produit()
{
$nbarg = func_num_args();
$prod=1;
echo " Le nombre d'arguments:$nbarg <br>";
// la fonction produit a ici $nbarg arguments
for ($i=0; $i<$nbarg; $i++)
{
$prod *= func_get_arg($i);
}
return $prod;
}
echo "le produit est : ".produit(3,77,10,5,81,9)." ";
// affiche le produit est 8 419 950
?>
```

Le nombre d'arguments:6
le produit est : 8419950

Syntaxe de base : Les fonctions

■ Variables locales et variables globales

- variables en PHP : global, static, local
- toute variable déclarée en dehors d'une fonction est globale
- utiliser une variable globale dans une fonction, l'instruction **global** suivie du nom de la variable
- Pour conserver la valeur acquise par une variable entre deux appels de la même fonction : l'instruction **static**.

➤ Les variables statiques restent locales à la fonction et ne sont pas réutilisables à l'extérieur.

```
<?php
Function cumul($prix){
static $cumul = 0;
static $i=1;
echo "Total des achats $i = ";
$cumul+=$prix;
$i++;
return $cumul;
}
echo cumul(175)."<br/>";
echo cumul(65), "<br/>";
echo cumul(69), "<br/>";
?>
```

Total des achats 1 = 175
Total des achats 2 = 240
Total des achats 3 = 309

Syntaxe de base : Les tableaux

■ Principe

- Création à l'aide de la fonction **array()**
- Les éléments d'un tableau peuvent appartenir à des types distincts
- L'index d'un tableau en PHP commence de 0
- Pas de limites supérieures pour les tableaux
- La fonction **count()** pour avoir le nombre d'éléments d'un tableau
- Déclaration : **plusieurs manières** :

```
<?php
// Déclaration d'un tableau vide
$filieres= array();

// Déclaration d'un tableau indexé numériquement
$langages= array('Java','PHP','C','C++');

// Déclaration d'un tableau mélangeant les types entier et chaîne
$variable = "test";
$tab = array($variable, "texte", 153, 56);
?>
```

Syntaxe de base : Les tableaux

■ Déclaration

- En PHP, la déclaration est implicite, nul besoin de préciser à l'avance le nombre d'éléments du tableau

➤ Par affectation

```
$t[0] = "bonjour";  
$t[1] = "bonsoir";  
$t[2] = "bla bla bla";
```

➤ Utilisation

```
echo "case numéro 2 : ".$t[2]."<BR>\n";  
for ($i=0 ; $i<count($t) ; $i++) {  
    echo "case numéro $i : ".$t[$i]."<BR>\n";  
}
```

Les tableaux: Tableau associatif

■ Crédit

Pour créer un tableau associatif, il faut donner pour chaque élément, le couple : (clé => valeur)

```
<?php  
$t = array(  
    "prénom" => "Ahmed",  
    "ville" => "Casa",  
    "travail" => "informatique"  
);  
  
?>
```

Les tableaux: Tableau associatif

- **Exemples de tableaux simples :**

- clé => valeur

```
$fruits = array ("a"=>"orange", "b"=>"banane", "c"=>"pomme");  
$num = array (1=>"premier", 2 => "second", 3 => "troisième");
```

- **Exemples de tableau de tableaux:**

- clé =N° de département => sous-tableau

- Chaque sous-tableau est composé de 4 éléments (Réf, nom_dep,nbr_prof,nbr_etudiants)

```
var $departement = array (  
    "01" => array ( "A","Math","10","222" ),  
    "02" => array ( "B","Info","12","180" ),  
    "03" => array ( "C","Indus","8","200"),  
    etc.
```

- Exemple: // Article avec différentes caractéristiques...

Les tableaux: Tableau associatif

- **Fonctions relatives : isset**

Pour tester l'existence d'un élément, on utilise la fonction **isset()**

- **Exemple:**

```
<?php
$calories["pommes"] = 300;
$calories["bananes"] = 130;
$calories["oranges"] = 30;
if( isset( $calories["pommes"] ) ) {
echo "une pomme contient ", $calories["pommes"] ,
" calories";
}
else{
echo "pas de calories définies pour la pomme";
}

?>
```

Les tableaux: Tableau associatif

- **Parcours :**

La méthode classique ne fonctionne pas. Il faut utiliser les fonctions : **foreach, list**

- **Exemple:**

```
<?php
$t = array(
    "prénom" => "Ahmed",
    "ville" => "Casa",
    "travail" => "informatique"
);

foreach ($t as $cle => $val) {
    echo " $cle : $val <BR>";
}
?>
```

prénom : Ahmed
ville : Casa
travail : informatique

```
<?php
$my_array = array("JAVA", "PHP", "C");

list($a, $b, $c) = $my_array;
echo "Compétences:, $a, $b et $c.";
```

Compétences:, JAVA, PHP et C.

Utilisation des tableaux

- **Fonctions de tri**

- **Tri sur les clés**

- La fonction **ksort()** trie les clés du tableau → ordre croissant
 - La fonction **krsort()** effectue la même action mais en ordre inverse → ordre décroissant

- **Exemple:**

```
<?php
$tab2= array("2006"=>"Ahmed","2004"=>"Sofia","2000"=>"Amine");
ksort ($tab2);
echo "<h3>Tri sur les clés de tab2</h3>";
foreach ($tab2 as $cle=>$valeur) {
echo " l'élément a pour clé :".$cle." et pour valeur :".$valeur." <br/>";
}
?>
```

Tri sur les clés de tab2

l'élément a pour clé :2000 et pour valeur :Amine
l'élément a pour clé :2004 et pour valeur :Sofia
l'élément a pour clé :2006 et pour valeur :Ahmed

Utilisation des tableaux

➤ Rechercher un élément: Présence d'un élément :

- `in_array(expression, tableau)`

➤ Calculer la clé :

- `Array_search(expression, tableau)`

➤ Vérifier l'existence d'une clé

- `array_key_exists()`

▪ Exemple:

```
<?php
$colors = array('rouge', 'vert',
'bleu');
if (in_array('vert', $colors)){
echo '<br>Trouvé, vert';
}
?>
```

```
<?php
$colors = array('rouge', 'vert', 'bleu');
$cle = array_search('vert', $colors);
echo "La valeur 'vert' est à la clé $cle";
//Affiche : la valeur 'vert' est à la clé 1
?>
```

```
<?php
$colors = array('ff0000' =>
'rouge', '00ff00' => 'vert',
'0000ff' => 'bleu');
if(array_key_exists('00ff00',
$colors)){
echo 'La clé "00ff00" existe';
}
?>
```

Utilisation des tableaux

➤ Calculer le nombre d'occurrences d'un élément:

- `Array_count_values()`

➤ Extraire et remplacer un élément

- Par utilisation de : `list()`: Assigne des variables comme si elles étaient un tableau

■ Exemple:

```
<?php
$tab = array('ahmed', 'Said', 'ahmed' , 'Amine', 'ahmed');
//tableau contenant les éléments
$cpt = array_count_values($tab);
echo "L'élément 'ahmed' apparaît ", $cpt['ahmed'],
"fois.<br>";
//Affiche L'élément 'ahmed' apparaît 3fois.
?>
```

L'élément 'ahmed' apparaît 3fois.

1-2-3-4

```
<?php
$tab = array(1, 2, 3, 4);
list($a, $b, $c, $d) = $tab;
echo "$a-$b-$c-$d";
?>
```

Utilisation des tableaux

➤ Lister les clés utilisées

- `array_keys()`

➤ Fusionner de plusieurs tableaux

- `array_merge()`

➤ Extraire des indices

- `extract()` permet de faire des clés, des variables, et de leur donner la valeur de leur indice

▪ Exemple:

```
<?php
$tab = array('a'=>1, 'c'=> 5);
$cles = array_keys($tab);
echo implode('-', $cles); //implode(séparateur, tableau)
//Affiche a-c
?>
```

```
$a = "Original";
$my_array = array("a" => "JAVA", "b" => "PHP", "c" =>
"C++");
extract($my_array);
echo "\$a = $a; \$b = $b; \$c = $c";
```

```
$result_2002 = array(12250, 12000);
$result_2003 = array(1520, 25000);
$result_2002_2003 =
array_merge($result_2002, $result_2003);
print_r($result_2002_2003);
```

```
Array ( [0] => 12250 [1] => 12000 [2] => 1520 [3] => 25000 )
```

```
$a = JAVA; $b = PHP; $c = C++
```

Utilisation des tableaux

➤ Séparer

- **array_chunk(\$tab,n)** : sépare \$tab en tableaux de n éléments chacun

➤ Calculer des différences

- Différence : **array_diff()**

➤ intersection : des indices

- **array_intersect()**

▪ Exemple:

```
<?php
$tab1 = array(1, 2, 3, 4, 5, 6, 7);
$tab2 = array(1, 3, 5, 7);
$tab3 = array(1, 2, 3);
$diff = array_diff($tab1,$tab2,$tab3);
echo implode('-', $diff) // affiche 4-6
?>
```

```
<?php
$tab1 = array(1, 2, 3, 4, 5, 6, 7);
$tab2 = array(1, 3, 5, 7);
$tab3 = array(1, 2, 3);
$inter = array_intersect($tab1,$tab2,$tab3);
echo implode('-', $inter); //Affiche 1-3
?>
```

Utilisation des tableaux

➤ Gérer des piles et des files

- Fonctions : array_push et array_pop



▪ Exemple:

```
<?php
$tab = array();
array_push($tab,1, 3, 5);
print_r( $tab);
echo "<br>";
/*équivalent à */
$tab = array();
$tab[0]=1;
$tab[1]=3;
$tab[2]=5;
print_r( $tab);
?>
```

Array ([0] => 1 [1] => 3 [2] => 5)
Array ([0] => 1 [1] => 3 [2] => 5)

```
<?php
$tab = array();
array_push($tab,1, 3, 5);
echo array_pop($tab); //Affiche 5;
echo "<br>";
echo array_pop($tab); //Affiche 3;
echo "<br>";
print_r($tab); //affiche Array ( [0] => 1 )
?>
```

5
3
Array ([0] => 1)

L'interactivité en PHP

Les formulaires

■ Intérêt

- Dans un contexte Web, les données échangées avec le serveur se font à travers des formulaires
- Les formulaires HTML sont la méthode la plus simple pour avoir des interactions avancées avec l'utilisateur
- Ils permettent, par exemple, de :
 - Créer un espace sécurisé
 - Donner aux clients la possibilité de modifier eux mêmes leurs sites
 - Interagir avec le visiteur en lui demandant des informations complémentaires...

Les formulaires

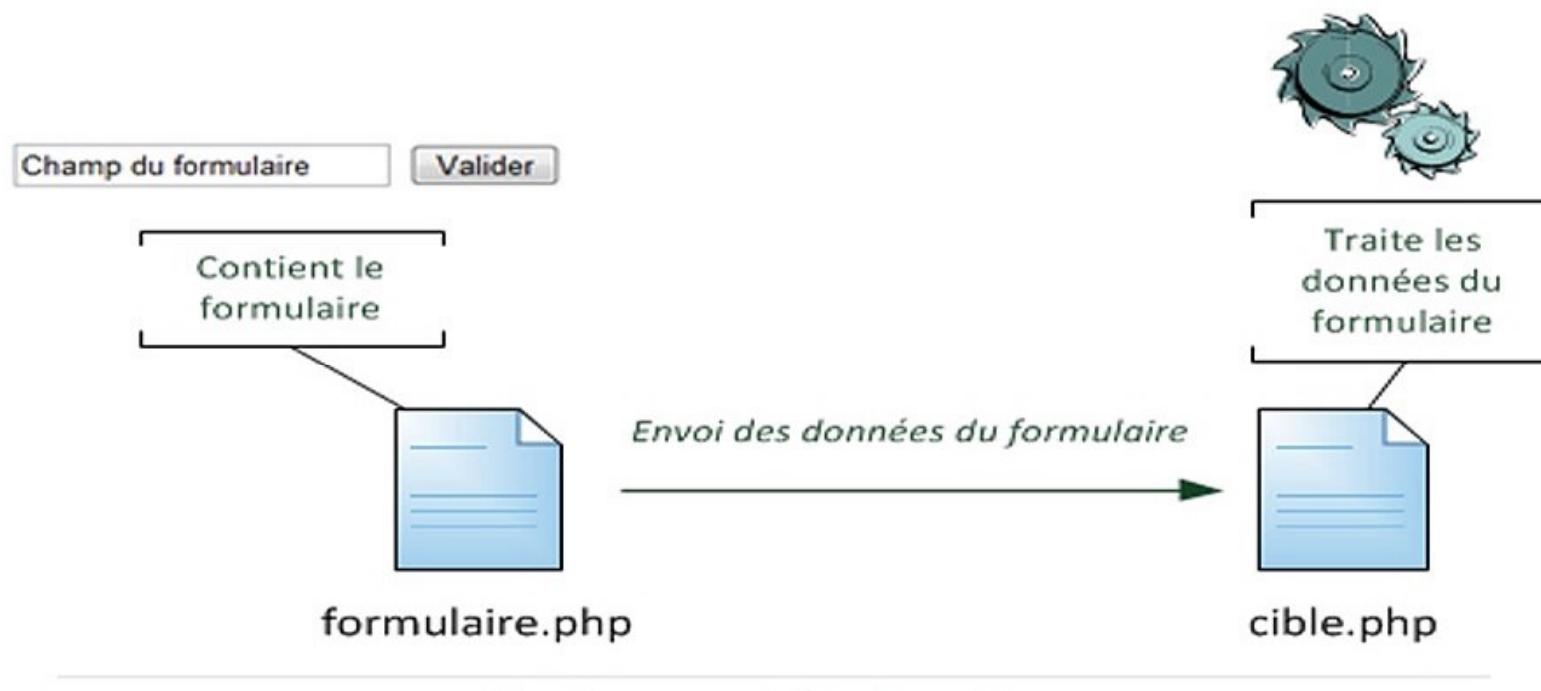
- **Création : balise <form>**

```
<form action="reception_formulaire.php" method='GET' ou 'POST'>
<!-- différents champs -->
</form>
```

- **action** : désigne la page vers laquelle seront envoyées les informations rentrées par l'utilisateur une fois le bouton d'envoi actionné
- **method** : définit le mode d'envoi des informations au serveur
 - Deux méthodes existent:
 - **GET** et **POST**

Les formulaires

GET et POST : Le but est de récupérer le contenu des champs d'un formulaire HTML dans notre code PHP pour pouvoir le traiter. Lorsqu'un formulaire est envoyé à un script PHP, toutes les variables seront disponibles automatiquement dans le script. Les formulaires peuvent être de type GET ou POST.



Les formulaires

■ Méthodes d'envoi GET et POST

- La méthode GET place les informations d'un formulaire directement à la suite de l'adresse URL de la page appelée.
 - <http://www.site.com/cible.php?champ=valeur&champ2=valeur>
- Le point d'interrogation "?" dit au navigateur que les objets suivants sont des variables
- **inconvénients :**
 - Les données sont visibles dans la barre d'adresse du navigateur.
 - De plus, la longueur totale est limitée à 255 caractères, ce qui rend impossible la transmission d'un volume de données important
 - La méthode **POST** assure une confidentialité efficace des données

Les formulaires

■ Récupération par tableaux

- Chaque champ du formulaire en PHP est défini par un **nom** et une **valeur**.
- Dans un script, PHP va récupérer ces noms et ces valeurs dans des **tableaux associatifs** dit superglobaux (accessibles depuis partout),
- Pour la méthode GET, le tableau est **\$_GET**,
 - **\$_GET ne contiendra que les variables de type GET.**
- pour la méthode POST le tableau est **\$_POST**.
 - **\$_POST ne contiendra que les variables de type POST.**
- Si vous ne souhaitez pas vous soucier de la méthode, vous pouvez utiliser le tableau **\$_REQUEST**. En index on aura le nom du champ de formulaire (ou de la variable passée en URL) et en valeur la valeur du champ.
 - **\$_REQUEST contient les variables de type POST et GET mais aussi les variables de cookies.**

Les formulaires

■ Récupération par tableaux: méthode POST

- Exemple:

Index.php

```
<form action="cible.php" method="post">  
Name: <input type="text" name="username"><br>  
Email: <input type="text" name="email"><br>  
<input type="submit" name="submit"  
value="Submit me!">  
</form>
```

cible.php

```
<?php  
echo "<h4> Ces infos sont envoyés par le  
formulaire </h4>";  
  
echo $_POST['username'];  
echo "<br>";  
echo $_REQUEST['email'];  
?>
```

Name: Amine

Email: Amine_gi@gmail.com

Submit me!



Ces infos sont envoyés par le formulaire

Amine
Amine_gi@gmail.com

Les formulaires

■ Récupération par tableaux: méthode GET (transmission par URL)

- Exemple:

Imaginons l'appel d'une page test.php par une URL comme ceci :

http://localhost/test_get.php?id=1

Ici on transmet une variable via une URL et donc la méthode implicite GET.

Pour récupérer « id » dans un code PHP on peut donc faire :

```
<?php  
echo $_GET['id'];  
echo "<br>";  
echo $_REQUEST['id'];  
echo "<br>";  
echo $_GET['idb'];  
?>
```



Les formulaires

- La fonction : **isset()**

La fonction : **isset()** teste si une variable existe. Nous allons nous en servir pour afficher un message spécifique si une variable est définie ou non.

- Exemple 1:

[index.php](#)

```
<form method="GET" action="cible.php">
    <input name="nom" type="text"/>
    <input name="prenom" type="text"/>
    <input name="valider" type="submit" value="OK"/>
</form>
```

[cible.php](#)

```
<?php
if (isset($_GET['prenom']) AND isset($_GET['nom'])) // si on a tous les param
{
    echo 'Bonjour ' . $_GET['prenom'] . ' ' . $_GET['nom'] . ' !';
}
else // Il manque des paramètres, on avertit le visiteur
{
    echo 'Il faut renseigner un nom et un prénom !';
}
?>
```

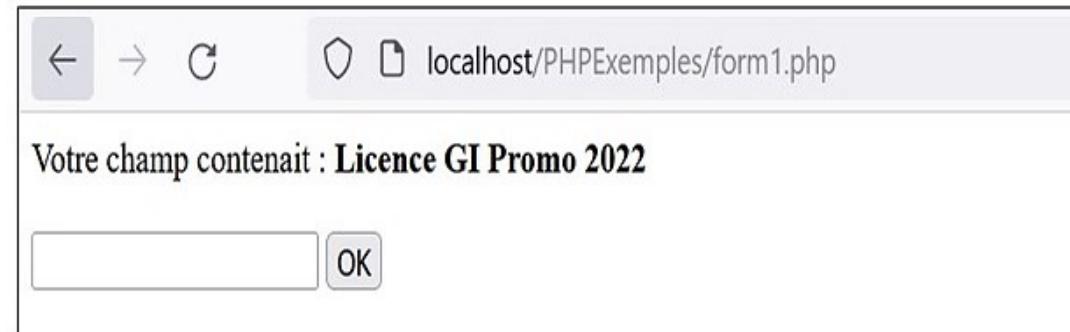
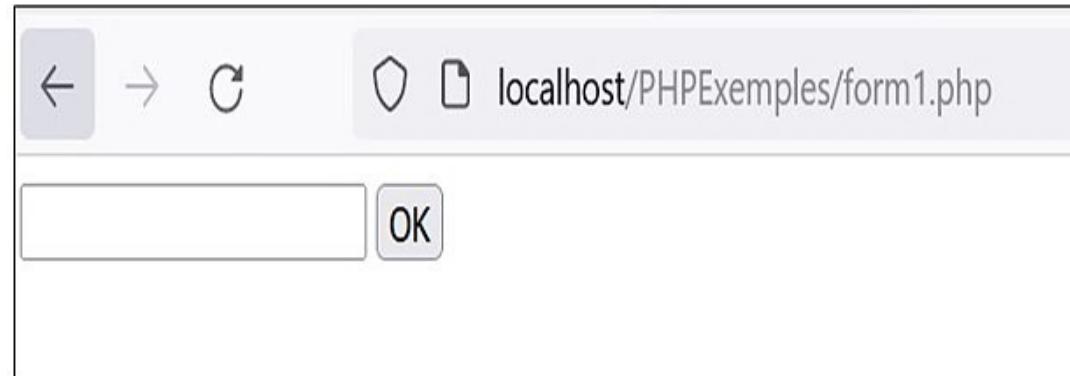
Les formulaires

- La fonction : **isset()**

La fonction : **isset()** teste si une variable existe. Nous allons nous en servir pour afficher un message spécifique si une variable est définie ou non.

- Exemple 2: (**Sur la même page**)

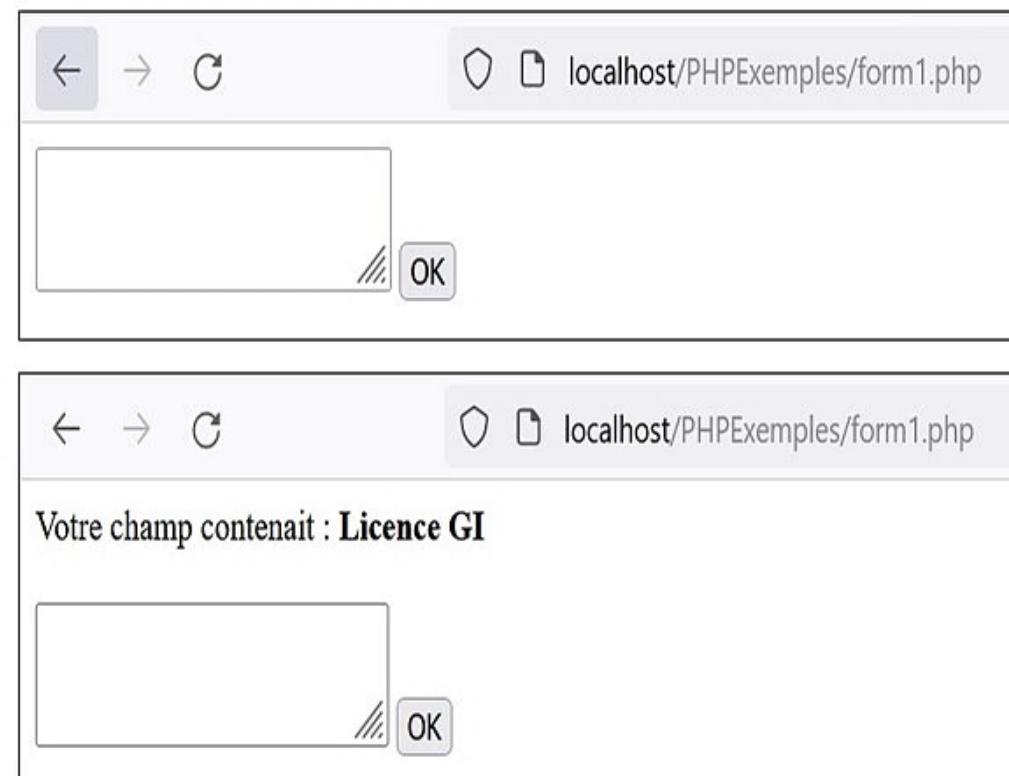
```
<?php  
if (isset($_POST['mon_champ'])) {  
?  
    Votre champ contenait :  
    <b> <?php echo $_POST['mon_champ']; ?></b>  
    <br/><br/>  
    <?php  
}  
?  
<form method="POST" action="">  
    <input name="mon_champ" type="text"/>  
    <input name="valider" type="submit"  
        value="OK"/>  
</form>
```



Les formulaires

- Les éléments du formulaire
- Exemple 2: Afficher le contenu de la zone de texte si celle-ci n'est pas vide

```
<?php  
$mon_champ = isset($_POST['mon_champ']) ?  
$_POST['mon_champ'] : '';  
if ($mon_champ) {  
?  
    Votre champ contenait :  
    <b><?php echo $mon_champ; ?></b>  
    <br/><br/>  
?>  
<?php  
}  
<?>  
<form method="POST">  
    <textarea name="mon_champ"></textarea>  
    <input type="submit" value="OK"/>  
</form>
```



Les formulaires

- Les éléments du formulaire
- Exemple 2: Un peu plus compliqué maintenant, car nous allons réafficher dans le même contrôle, son contenu

```
<?php  
$mon_champ = isset($_POST['mon_champ']) ?  
$_POST['mon_champ'] : '';  
if ($mon_champ) {  
?>  
Votre champ contenait :  
<b><?php echo $mon_champ; ?></b>  
<br/><br/>  
<?php  
}  
?>  
<form method="POST">  
    <textarea name="mon_champ">  
        <?php echo $mon_champ; ?>  
    </textarea>  
    <input type="submit" value="OK"/>  
</form>
```



Les formulaires

- **Les éléments du formulaire**
 - Les boutons d'options (radio)
 - Les cases à cocher
 - Les listes déroulantes à sélection simple
 - Les listes déroulantes à choix multiples
 - Les champs cachés
 - C'est un code dans votre formulaire qui n'apparaîtra pas aux yeux du visiteur, mais qui va quand même créer une variable avec une valeur.

```
<input type="hidden" name="pseudo" value="Amine_LST" />
```

Exercice TP (1)

← → C 127.0.0.1/PHPExemples/Tp2_formulaire_interactif/formulaire.php

Sexe : Homme Femme

Nom : Ahmed

Prénom : Amine

Âge : 23

Pseudo : Amine_GI

Mot de passe : *****

Mot de passe (confirmation) : *****

Domaine d'activité: Informatique Gestion Pédagogie

Les langages préférés Java C C++ PHP (Pour faire plusieurs choix maintenir la touche CTRL enfoncée)

Pays : Maroc

Je désire recevoir la newsletter chaque mois.

← → C 127.0.0.1/PHPExemples/Tp2_formulaire_interactif/traitement.php

Bonjour !

Votre nom et prénom sont: Ahmed Amine !

Vous êtes: Homme !

Si tu veux changer le nom, [clique ici](#) pour revenir à la page formulaire.php.

Votre Age est : 23

Votre password crypté est : olEkgq6nj7RYlGJ0mjOzuw==

Votre password décrypté est : lstgi2022

Votre domaine d'activité est : Informatique - Gestion -

Les compétences : Java - C - PHP -

Votre pays est : Maroc

Votre réponse pour les newsletter : OUI

Votre pseudo caché : Amine_LST

Exercice TP (2)

Exercice:

Protéger le contenu d'une page par mot de passe.

On veut mettre en ligne une page web pour donner des informations confidentielles à certaines personnes. Cependant, pour limiter l'accès à cette page, il faudra connaître un mot de passe.