



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. A red bracket groups the first four chapters (Chapitre 01 to Chapitre 04) under the heading "Développement frontend". A blue bracket groups the fifth chapter (Chapitre 05) under the heading "Développement backend".
- 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...



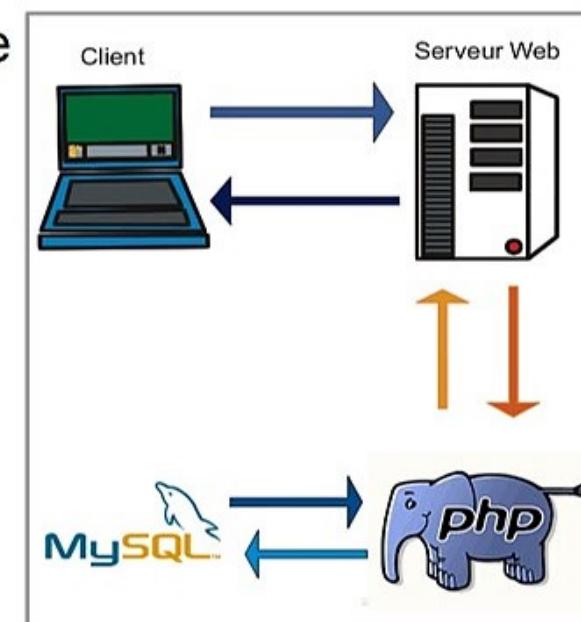
Introduction

- 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] = "test";
```

➤ Utilisation

```
echo nl2br("case numéro 2 : ".$t[2]."\n");  
for ($i=0 ; $i<count($t) ; $i++) {  
    echo nl2br("case numéro $i : ".$t[$i]."\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)

```
$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 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
?>

<?php
$tab1 = array(1, 2, 3, 4, 5, 6, 7);
$diff = array_chunk($tab1,3);
foreach($diff as $a){
    foreach($a as $b){
        echo "$b"; }
    echo nl2br("\n");
} ?>
```

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

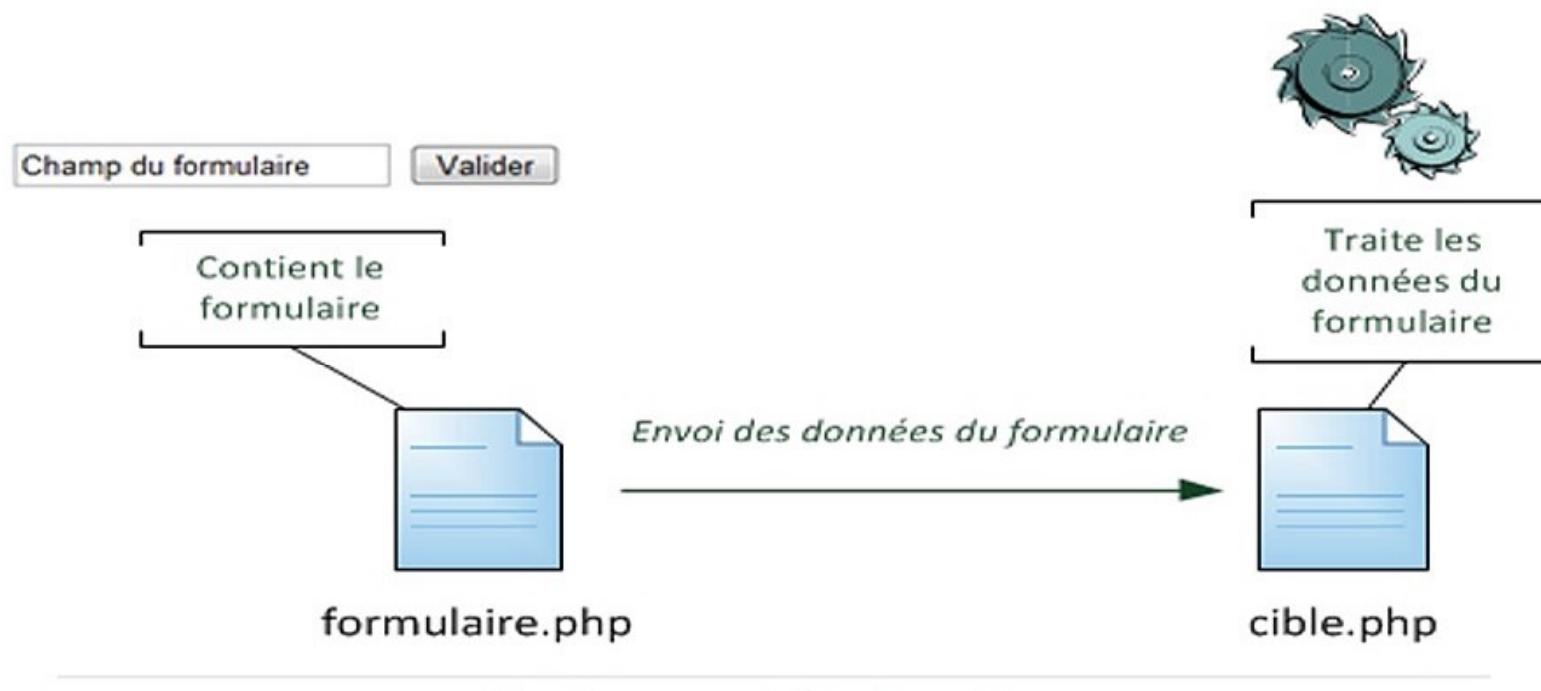
- **Création : balise <form>**

```
<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 (par url).**
- 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 seulement.**
 - **RQ: la nouvelle version du PHP ne contient pas le stockage de cookies sur \$_REQUEST**

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=4

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 !';
}
?>
```

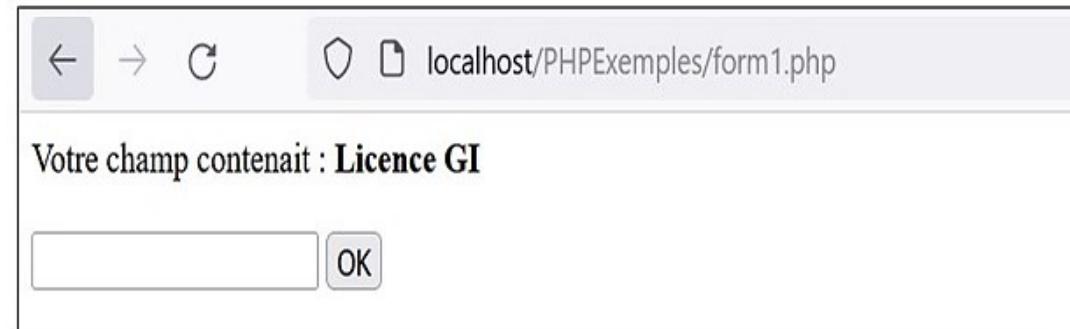
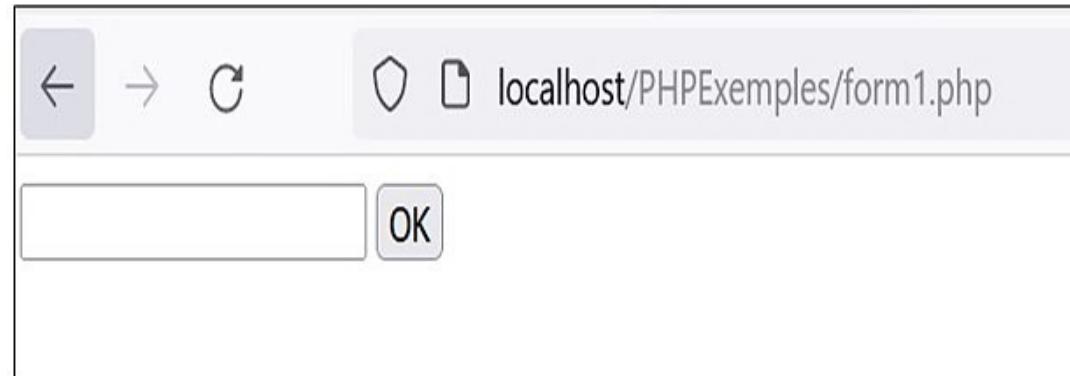
Les formulaires

- 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**)

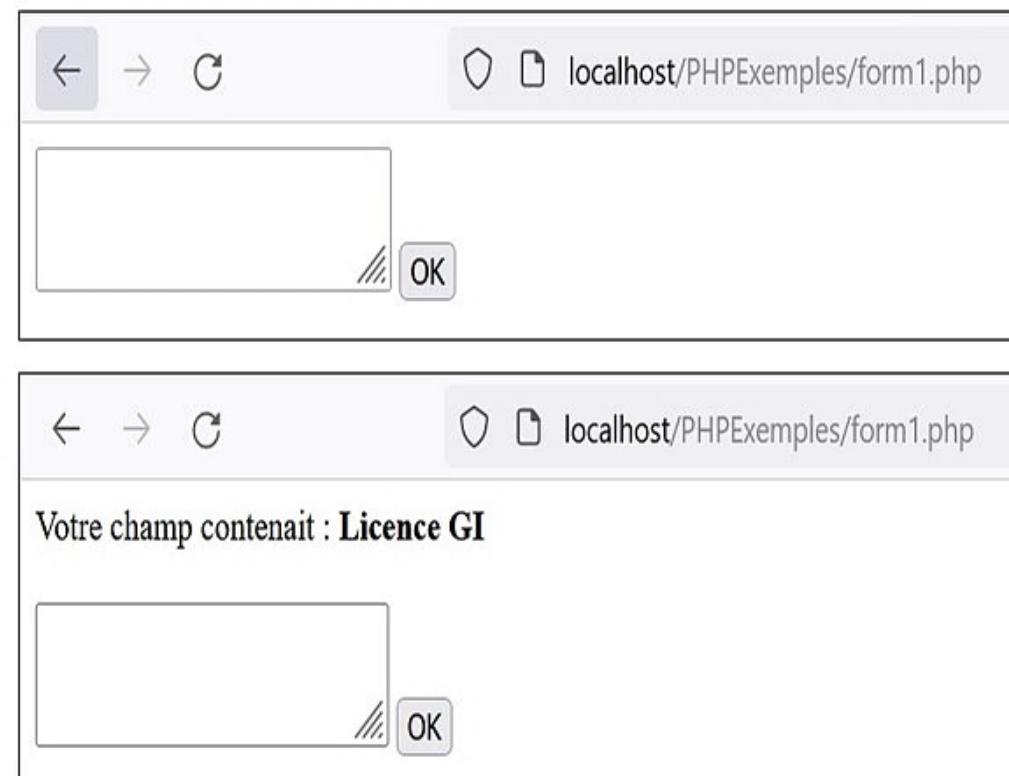
```
<?php  
if (isset($_POST['mon_champ'])) {  
?  
    Votre champ contenait :  
    <b> <?php echo $_POST['mon_champ']; ?></b>  
    <br/><br/>  
    <?php  
}  
?  
<form method="POST" action="">  
    <input name="mon_champ" type="text"/>  
    <input name="valider" type="submit"  
        value="OK"/>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



Les formulaires

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

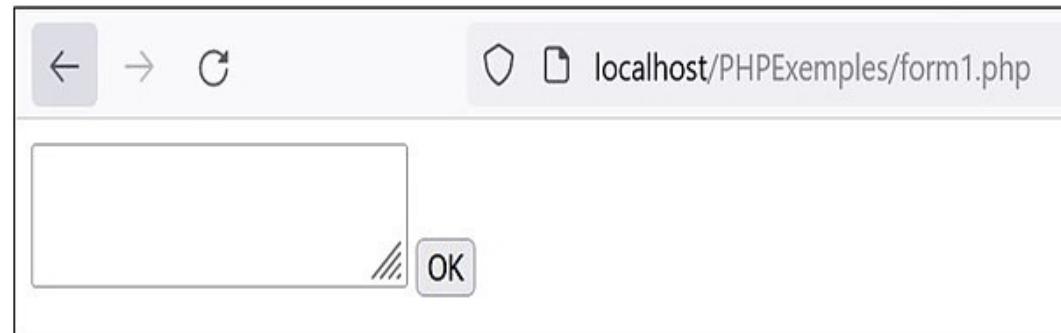
```
<?php  
$mon_champ = isset($_POST['mon_champ']) ?  
$_POST['mon_champ'] : '';  
if ($mon_champ) {  
?  
    Votre champ contenait :  
    <b><?php echo $mon_champ; ?></b>  
    <br/><br/>  
?>  
<?php  
}  
?  
<form method="POST">  
    <textarea name="mon_champ"></textarea>  
    <input type="submit"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

Les cookies

■ Principe

- Un cookie, est un **petit fichier** que l'on enregistre sur l'ordinateur du visiteur.
- Ce fichier contient du texte et permet de « **retenir** » des informations sur le visiteur.
- Par exemple, vous inscrivez dans un cookie le **pseudo du visiteur**, comme ça la prochaine fois qu'il viendra sur votre site, vous pourrez lire son pseudo en allant regarder ce que son cookie contient.
- Chaque cookie stocke généralement une information à la fois.
- Si vous voulez stocker le pseudonyme du visiteur et sa date de naissance, il est donc recommandé de créer deux cookies

Les cookies

■ Principe

- La date d'expiration des cookies est définie de manière explicite par le serveur web chargé de les mettre en place.
- Les cookies disponibles sont importés par PHP sous forme de variables identifiées sous les noms utilisés par ces cookies
- La variable globale du serveur **`$_COOKIES`** enregistre tous les cookies qui ont été définis

Les cookies

■ Où sont placés les cookies pour Firefox et Chrome

The image shows two browser windows side-by-side, illustrating the privacy settings for Firefox and Chrome.

Firefox Privacy Settings (Left):

- Header: Firefox | about:preferences#privacy
- Message: Votre navigateur est géré par votre organisation.
- Search bar: Rechercher dans les paramètres
- General tab (selected):
 - Cookies and site data section:
 - Storage usage: Utilise actuellement 181 Mo d'espace disque.
 - Buttons: Effacer les données..., Gérer les données...
 - Identifiers and passwords section:
 - Checkboxes: Proposer d'enregistrer les identifiants et les mots de passe pour les sites web, Renseigner automatiquement les identifiants et les mots de passe, Suggérer et créer des mots de passe robustes, Afficher des alertes pour les mots de passe de sites concernés par des fuites de données.
 - Buttons: Exceptions..., Identifiants enregistrés...
- Other tabs: Accueil, Recherche, Vie privée et sécurité, Synchronisation, Autres produits de Mozilla, Extensions et thèmes.

Chrome Privacy Settings (Right):

- Header: Chrome | chrome://settings/privacy
- Search bar: Rechercher
- Paramètres tab (selected):
 - Google et vous
 - Saisie automatique
 - Confidentialité et sécurité** (selected):
 - Information: Chrome peut vous aider à vous protéger contre les violations de données et les extensions malveillantes. Button: Vérifier maintenant
 - Sub-sections: Effacer les données de navigation (Effacer l'historique, supprimer les cookies, vider le cache, etc.), Cookies et autres données des sites (Les cookies tiers sont bloqués lorsque vous utilisez le mode navigation privée), Sécurité (Navigation sécurisée (protection contre les sites dangereux) et autres paramètres de sécurité).
 - Apparence
 - Moteur de recherche
 - Navigateur par défaut
 - Au démarrage
- Paramètres avancés

Les cookies

■ Écriture de cookies

Un cookie s'envoie avec la fonction **setcookie()**. Les arguments sont les suivants :

- **Nom_var** : nom de la variable qui va stocker l'information sur le poste client et qui sera utilisée pour récupérer cette information dans la page qui lira le cookie.
 - C'est la seule indication obligatoire pour un cookie
- **Valeur_var** : valeur stockée dans la variable. Par exemple une chaîne de caractères ou une variable chaîne, en provenance d'un formulaire
- **Date_expiration** : la date à laquelle le cookie ne sera plus lisible et sera effacé du poste client
 - (ex. : $365*24*3600$).(on définit une durée de vie de notre cookie (en **secondes**), donc un an dans notre cas)
 - Si l'attribut n'est pas spécifié, le cookie expire à l'issue de la session

Les cookies

■ Écriture de cookies

- **Chemin (facultatif):** définit la destination (incluant les sous-répertoire) à laquelle le navigateur doit envoyer le cookie.
Par défaut, le cookie est disponible pour tout le site.
- **Domain (facultatif):** le nom du domaine à partir duquel peuvent être lus les cookies.
 - On peut aussi utiliser la variable d'environnement **`$_SERVER['SERVER_NAME']`** à la place.
 - Par défaut, le cookie est disponible pour le domaine actuel.
- **Secure (facultatif):** Spécifie si le cookie doit être envoyé uniquement via HTTPS. un nombre qui vaut 0/false si la connexion n'est pas sécurisée, sinon, il vaut 1/true pour une connexion sécurisée

■ Syntaxe:

```
Setcookie("nom_var", "valeur_var", date_expiration, "chemin", "domain", "secure")
```

■ Exemple:

```
Setcookie("cookie_test", "12345", time() + (7 * 24 * 60 * 60), "/", "", true);
```

Les cookies

■ Écrire des cookies : limitations

- Un cookie n'est pas disponible dans la page qui l'a créé. Il faut soit **recharger** la page, soit **pointer** sur une autre.
- Un cookie ne peut être supprimé qu'avec les mêmes paramètres qui ont servi à sa création. C'est le navigateur qui supprime le cookie.

■ En général:

- **Taille limitée** : les cookies ont une taille maximale de 4 Ko, ce qui limite la quantité de données que vous pouvez stocker dans un cookie.
- **Sécurité** : Les cookies sont stockés sur le navigateur de l'utilisateur, ce qui les rend vulnérables aux attaques de piratage.
- **Dépendance du navigateur** : Les cookies dépendent du navigateur de l'utilisateur, ce qui signifie que si le navigateur n'accepte pas les cookies ou s'ils sont désactivés, le site ne peut pas utiliser de cookies.
- **Confidentialité** : Les cookies stockent des informations sur l'utilisateur et son comportement en ligne, ce qui peut soulever des préoccupations en matière de confidentialité. Les sites doivent respecter les lois et les réglementations en matière de confidentialité des données et informer les utilisateurs des données collectées.
- **Expérience utilisateur** : Les cookies peuvent avoir un impact sur l'expérience utilisateur, car ils peuvent ralentir la vitesse de chargement des pages et peuvent causer des problèmes de compatibilité avec certains navigateurs

Les cookies

■ Écrire des cookies

- Exemple:

```
<?php
/* on définit une durée de vie de notre cookie (en
secondes), donc un an dans notre cas*/
$temps = 365*24*3600;
/* on envoie un cookie de nom pseudo portant la valeur Amine */
setcookie ("pseudo", "Amine", time() + $temps);
?>
```

- On envoie par ce code chez le client (donc le visiteur du site) un cookie de nom pseudo portant la valeur **Amine**
- De plus, avec `time()`, on impose que le cookie ait une durée de vie de un an (soit en fait l'instant présent plus un an, donc un an)
- Maintenant, si le visiteur ne supprime pas ce cookie, et bien, dans toutes les pages WEB de notre site, on pourra accéder à la variable `$pseudo` qui contiendra la chaîne de caractères **Amine** ou celle que vous avez rentrée dans votre formulaire

Les cookies

■ Accès à un cookie

On accède au cookie grâce à la variable globale **`$_COOKIE`** qui est un tableau.
L'index du tableau est le nom du cookie.

```
$valeur=$_COOKIE['pseudo'];
```

```
echo $valeur; // par rapport à l'exemple précédent : Amine
```

Note : on peut placer les tableaux avec les cookies. Il suffit de nommer son cookie avec une **notation par crochets**.

```
<?php  
setcookie("testcookie[1]", 'Nom', time() + 3600);  
setcookie("testcookie[2]", 'Prenom', time() + 3600);  
setcookie("testcookie[3]", 'Email', time() + 3600);  
foreach($_COOKIE as $tab) {  
    if(is_array($tab))  
        foreach($tab as $key => $value) echo "$key => $value". "<br/>";  
}  
?>
```

```
1 => Nom  
2 => Prenom  
3 => Email
```

Les cookies

■ Supprimer un cookie

- Il suffit de donner une date antérieure à la date actuelle à celui-ci.

```
<?php
if(!isset($_COOKIE['pseudo'])){ setcookie("pseudo",'toto',time()+3600);
}
else{
setcookie("pseudo",'',time()-1);
}
if(isset($_COOKIE['pseudo'])){
echo "OK";
}
else {
echo "Pas de cookie";
}

?>
```

- Dans cet exemple, les appels au script vont provoquer successivement l'affichage de « pas de cookie » et « OK ».

Les cookies

■ Exemple

- Supposons que sur une page de notre site WEB , nous souhaitons faire en sorte que si le visiteur vient pour la première fois (ou qu'il a supprimé ses cookies), il aurait alors, la possibilité de **saisir son nom dans un formulaire**,
- ou bien s'il ne s'agit pas de sa première visite, d'afficher tout simplement **Bonjour** puis son **nom**
- On aurait alors le code suivant pour notre page (par exemple index.php)

Les cookies

■ Exemple

index.php

```
<html>
<head>
<title>Index du site</title>
<body>
<?php
if (isset($_COOKIE['mypseudo'])) {
    // si le cookie existe
echo 'Bonjour '.$_COOKIE['mypseudo'].' !';
}
else {
echo 'Notre cookie n\'est pas déclaré.';
    // si le cookie n'existe pas, on affiche un formulaire permettant
    // au visiteur de saisir son nom
echo '<form action=".cookies.php" method="post">';
echo 'Votre nom : <input type = "texte" name = "nom"><br />';
echo '<input type = "submit" value = "Envoyer">';
}
?>
</body>
</html>
```

Notre cookie n'est pas déclaré.

Votre nom :

Les cookies

■ Exemple

cookies.php

```
<?php
If (isset($_POST['nom'])) {
$temps = 365*24*3600;
// on envoie un cookie de nom mypseudo portant la valeur de la
variable $nom,
//c'est-à-dire la valeur qu'a saisie la personne qui a rempli le
formulaire
setcookie ("mypseudo", $_POST['nom'], time() + $temps);

// on effectue une redirection vers la page d'accueil
redirection ('index.php');
}
else {
echo 'La variable du formulaire n\'est pas déclarée.';
// au cas où on accède directement à la page cookies.php
}
```

Notre cookie n'est pas déclaré.

Votre nom :

Bonjour Amine_LST !

The screenshot shows the browser's developer tools open. On the left, there's a context menu with options like 'Enregistrer sous...', 'Code source de la page', and 'Inspecter'. A blue arrow points from the 'Inspecter' button to the top navigation bar. The 'Stockage' tab is highlighted in blue. Below it, a table lists a single cookie: 'mypseudo' with value 'Amine_LST', domain '127.0.0.1', path '/PHPExemples/...', expiration date 'Fri, 31 Mar 2023 15:07:55 ...', size '17', 'HttpOnly' set to 'false', 'Secure' set to 'false', and 'SameSite' set to 'None'.

| Nom | Valeur | Domain | Path | Expiration / Durée maximum | Taille | HttpOnly | Secure | SameSite |
|----------|-----------|-----------|------------------|-------------------------------|--------|----------|--------|----------|
| mypseudo | Amine_LST | 127.0.0.1 | /PHPExemples/... | Fri, 31 Mar 2023 15:07:55 ... | 17 | false | false | None |

Exercice TP

- Exemple: **Créer un panier à l'aide du cookies**
- Créer un premier fichier **cookie_init.php** qui servira à mettre à 0 les cookies
- On met à 0 tous les éléments du panier
- On fait ensuite une redirection vers **cookie_ajout.php**.
- Créer un fichier **cookie_ajout.php** : on met à jour le bon cookie (grâce à la méthode GET)
- Créer un troisième script (**cookie_calcul.php**) pour calculer et afficher le prix total du panier

Exercice TP

■ Exemple: Créer un panier à l'aide du cookies

The image shows two side-by-side browser windows. Both have the address bar set to 127.0.0.1/PHPExemples/cookies/cookie_ajout.php.

Left Window: This window displays a form for adding items to a cart. It has five rows, each containing a product name with its price in parentheses and a quantity input field. The first row shows "1Kg pommes (1 E)" with "0Kg pomme(s)". The second row shows "1Kg poire (1,5 E)" with "0Kg poire(s)". The third row shows "1Kg pêche (2 E)" with "0Kg pêche(s)". The fourth row shows "1Kg banane (1 E)" with "0Kg banane(s)". Below the form are two links: "vider mon panier" and "calculer le total".

Right Window: This window shows the result of calculating the total. It displays the message "Le total de votre panier: 2 Euro." and two links: "Modifier mon panier" and "Vider mon panier".

Les sessions

- **Idée: Aller plus loin que les cookies**

- Permettre à plusieurs pages de partager les mêmes données
- Conserver plusieurs données lors d'une session
- Ranger les données de la session sur le serveur
- Sécuriser le passage des données vers le serveur en
 - donnant un identifiant à la session
 - cryptant l'identifiant
- Disposer de plusieurs sessions et donner un nom à chacune

Les sessions

- **Qu'est-ce qu'une session ?**
- Les sessions permettent de préserver des données lors de la visite d'un site.
- Chaque personne se voit attribué un identifiant unique appelé **identifiant** de session, ou **SID**.
- On peut définir un nombre infini de variables qui seront accessibles durant toute la durée de la session.
- Notez que si vous fermez et relancez votre navigateur, vous changez d'identifiant, et donc la précédente session est perdue,
- Si on souhaite transmettre des variables sur toutes les pages de notre site pendant la durée de la présence d'un visiteur. Ce ne serait pas facile avec **GET** et **POST** car ils sont plutôt faits pour transmettre les informations une seule fois, d'une page à une autre. C'est pour cela qu'on doit utiliser les sessions.

Les sessions

- **Qu'est-ce qu'une session ?**
- Une session est plus ou moins similaire aux cookies HTTP
- Les données d'une session sont cependant stockées sur le serveur et non chez le client, ce qui l'empêche de pouvoir les modifier manuellement, comme il peut le faire pour un cookie

Les sessions

- **A quoi peut bien servir une session ?**
- **Exemple**
 - Permettre à un visiteur privilégié de pouvoir accéder à des pages particulières de votre site :
 - dans ce cas, l'utilisateur, se connectant avec son login et mot de passe, est reconnu au travers de ces variables sauvegardées dans un cookie, reçoit cette permission
 - Cela permet de réserver des liens à des gens qui ont une inscription à mon site

Les sessions

■ Fonctionnement des sessions

Comment sont gérées les sessions en PHP ? Voici les trois étapes à connaître.

- Un visiteur arrive sur votre site. Une session est créée pour lui. PHP génère alors un numéro unique. Ce numéro est souvent très gros et écrit en hexadécimal, par exemple : **a02bbffc6198e6e0cc2715047bc3766f** (SID). *PHP transmet automatiquement cet ID de page en page en utilisant généralement un cookie.)*
- Une fois la session générée, on peut créer une infinité de variables de session pour nos besoins. Par exemple, on peut créer une variable **\$SESSION['nom']** qui contient le nom du visiteur, etc. Le serveur conserve ces variables même lorsque la page PHP a fini d'être générée. **Cela veut dire que, quelle que soit la page de votre site, vous pourrez récupérer par exemple le nom et le prénom du visiteur via la superglobale **\$SESSION** !**
- Lorsque le visiteur se déconnecte de votre site, la session est fermée et PHP « oublie » alors toutes les variables de session que vous avez créées.

Les sessions

■ Fonctionnement des sessions

On doit connaître deux fonctions :

- **session_start()** : démarre le système de sessions. Si le visiteur vient d'arriver sur le site, alors un numéro de session est généré pour lui. **Vous devez appeler cette fonction au tout début de chacune des pages où vous avez besoin des variables de session.**
- Cette fonction doit être en mesure d'envoyer des entêtes **HTTP**
- **session_destroy()** : ferme la session du visiteur. Cette fonction est automatiquement appelée lorsque le visiteur ne charge plus de page de votre site pendant plusieurs minutes (c'est le timeout), mais vous pouvez aussi créer une page « Déconnexion » si le visiteur souhaite se déconnecter manuellement

Les sessions

- **Initialiser une session**
- Il faut appeler **session_start()** sur chacune de vos pages AVANT d'écrire le moindre code HTML.
- Si vous oubliez de lancer `session_start()` , vous ne pourrez pas accéder aux variables **superglobales** `$_SESSION` .
- **Code:**

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

Les sessions

- **Une session portant un nom personnalisé**
 - Une session porte par défaut le nom "**PHPSESSID**"
 - Ce nom sera utilisé comme nom de cookie ou comme paramètre GET dans les liens
 - Pour le changer, utiliser la fonction `session_name()`

```
<?php
$v = session_name();
echo "Nom du session : ".$v;
session_name('session_LST_GI');
$var = session_name();
echo "<br> Nom du session après: ".$var;
?>
```

```
Nom du session : PHPSESSID
Nom du session après: session_LST_GI
```

Les sessions

- **Lire et écrire des données dans la session**
 - Les données de la session sont très facilement accessibles au travers d'un simple tableau PHP
 - On utilise le tableau super-global **`$_SESSION`**
 - Tout ce qui est stocké à l'intérieur est sauvegardé et accessible depuis toutes les pages PHP utilisant les sessions

Les sessions

- **Supprimer une variable d'une session**
- Utiliser **unset()** qui prend en paramètre la variable à détruire
- **Exemple:**

```
<?php
    unset($_SESSION['prenom']); // La variable de
    //session "prenom" a été supprimée, on ne peut plus
    //y avoir accès !
?>
```

Les sessions

- Détruire une session
 - Utiliser **session_destroy()** qui ne prend aucun paramètre et qui renvoie vrai en cas de succès et faux en cas d'erreur
 - Fonctionnement

```
<?php

if (session_destroy()) {
echo 'Session détruite !';
} else {
echo 'Erreur : impossible de détruire la session !';
}

?>
```

Les sessions

- **Détruire toutes les variables d'une session**
- Il est aussi possible de détruire toutes les variables de session, ce qui permet de conserver votre session :
 - il suffit tout simplement de réinitialiser le tableau `$_SESSION`

```
<?php
$_SESSION = array(); // $_SESSION est désormais un tableau vide, toutes les
variables de session ont été supprimées

unset($_SESSION['my_variable']); // supprimer juste la variable «my_variable»
?>
```

- Il ne faut jamais utiliser `unset()` directement sur `$_SESSION`, cela rendrait impossible l'accès aux variables de la session courante jusqu'à sa fermeture

Syntaxe: `session_unset();`

■ Regénérer les identifiants de session

- Il est possible de demander à PHP un nouvel identifiant de session grâce à la fonction **session_regenerate_id()**
 - Les données ne sont alors pas effacées de la session en cours
 - L'identifiant de session est simplement modifié
 - L'ancien devenant invalide

```
<?php
session_start();
// Regénérer session ID
session_regenerate_id();

$new_session_id = session_id();

echo "New session ID: " . $new_session_id;
?>
```

- Cela est utile lorsque, par exemple, l'utilisateur change de niveau d'accès, s'il a accès à de nouvelles possibilités ou au contraire n'a plus accès à d'anciennes pages
- De cette manière, si quelqu'un avait réussi à trouver l'ancien identifiant de session, il n'aura pas accès aux nouveaux priviléges de l'utilisateur
- À l'inverse, si les identifiants de sessions sont régénérés trop souvent, cela peut poser des problèmes au niveau du référencement des pages web, surtout dans les cas où ils sont passés via l'URL

Les sessions

- Utiliser plusieurs sessions dans la même page
 - Il est impossible d'ouvrir simultanément plusieurs sessions
 - Cependant, on peut tout à fait ouvrir plusieurs sessions l'une après l'autre
 - Dans ce cas, il faut :
 - fermer la première session sans la détruire, grâce à **session_write_close()**
 - puis assigner les nouveaux **session_name** et **session_id**
 - et enfin ouvrir la nouvelle session avec **session_start()**

Les sessions

■ Utiliser plusieurs sessions dans la même page

```
<?php
session_name('utilisateur');
session_start(); // Création de la première session
[...]
// Utilisation de la première session
session_write_close(); // Fermeture de la première session, ses données sont
sauvegardées.
session_name('admin'); // Indication du nom de la seconde session
session_start(); // Ouverture de la seconde session
[...] // Utilisation de la seconde session.
?>
```

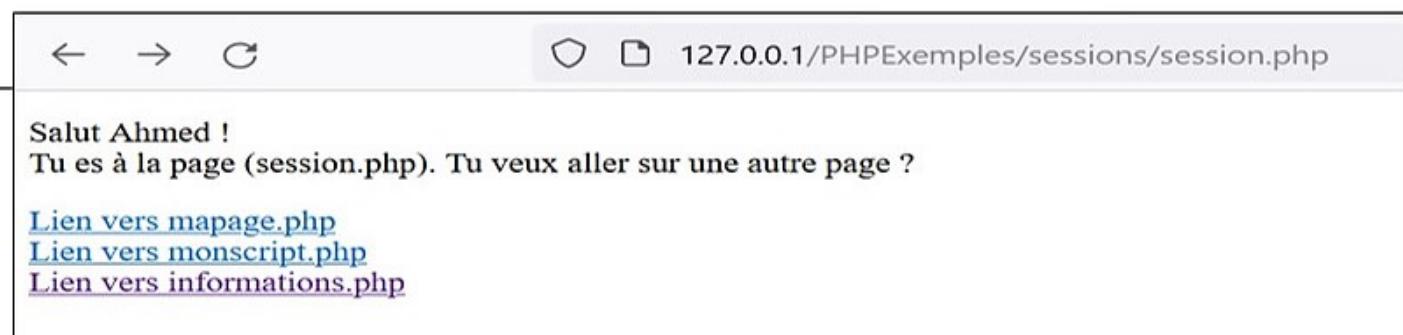
- Une fois la session fermée, il est toujours possible d'accéder en lecture (les modifications ne seront pas prises en compte) aux variables de l'ancienne session
- **`$_SESSION`** ne sera vidé et rempli qu'au prochain appel à **`session_start()`**

Les sessions

■ Fonctionnement des sessions

session.php

```
<?php
session_start();
$_SESSION['prenom'] = 'Ahmed';
$_SESSION['nom'] = 'Amine';
$_SESSION['age'] = 24; ?>
<!DOCTYPE html><html><head><meta charset="utf-8" /><title>Titre de ma
page</title></head><body>
    <p> Salut <?php echo $_SESSION['prenom']; ?> !<br />Tu es à la page (session.php).
    Tu veux aller sur une autre page ?</p><p>
        <a href="mapage.php">Lien vers mapage.php</a><br />
        <a href="monscript.php">Lien vers monscript.php</a><br />
        <a href="informations.php">Lien vers informations.php</a>
    </p></body></html>
```



Les sessions

■ Fonctionnement des sessions

- Voici par exemple le code source de la page informations.php

```
<?php
session_start(); // On démarre la session
?>
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8" /><title>Titre de ma page</title></head>
<body>
    <p>Re-bonjour !</p>
    <p> Tu t'appelles <?php echo $_SESSION['prenom'] . ' ' . $_SESSION['nom']; ?> <br/>
        Et ton âge ... Tu as <?php echo $_SESSION['age']; ?> ans, c'est ça ?
    </p>
</body>
</html>
```

Re-bonjour !

Tu t'appelles Ahmed Amine
Et ton âge ... Tu as 24 ans, c'est ça ?

La POO en PHP

Introduction

■ Evolutions et grands principes :

- Les objets existent en PHP à partir de la version 4
- Changements importants en PHP 5 : Convergence vers le modèle objet de Java
- Introduction comme en Java d'interfaces et de classes abstraites
- emploi des modificateurs private et public comme en java
- On retrouve aussi **`__toString()`**, **`__clone()`** et un mécanisme de traitement des exceptions semblable à celui de Java.
- Les constructeurs s'appellent désormais : **`__construct()`** et les destructeurs **`__destruct()`**
- les méthodes portent le mot clef **function** mais ne signalent pas leur type de retour
- les commentaires de documentation se font à la manière de Java

La POO en PHP

- **Intérêt**
- **Permet de:**
 - Rassembler autour d'un même objet (concept), une définition, des attributs et des méthodes d'action
 - de réutiliser des objets existants en les complétant par ce qui est nouveau
- **D'où des avantages liés à :**
 - La clarté du code
 - La modularité
 - La réutilisabilité
 - L'interopérabilité

La POO en PHP

■ Qu'est ce qu'un objet ?

- Un objet est toute donnée manipulée
 - Exemples
 - ❖ la voiture de mon voisin
 - ❖ mon compte bancaire

■ Qu'est ce que les attributs et les méthodes ?

- Chaque objet a des attributs qui lui sont propres
 - Mon compte bancaire a 3 attributs
 - ❖ Le numéro du compte
 - ❖ Le solde actuel
 - ❖ Liste des différentes opérations
- Les objets peuvent avoir des méthodes
 - Ce sont des actions que l'on peut appliquer à un objet
 - ❖ Solder
 - ❖ Créditer...

Classes et instances

■ Qu'est ce qu'une classe ?

- C'est un modèle de donnée
- On peut la voir comme une famille d'objets
- Tous les objets de la classe sont similaires
 - Partagent les mêmes attributs et les mêmes méthodes

■ Exemple

- Classe rassemblant toutes les voitures
- **Attributs**
 - Toutes les voitures (les objets) ont des plaques d'immatriculation, un moteur, un nombre de portes...
- **Méthodes**
 - Toutes les voitures (les objets) ont des méthodes pour démarrer, freiner, accélérer...

Classes et instances

- **Qu'est ce qu'une instance ?**
 - Une instance est une représentation particulière d'une classe
- **Exemple**
 - La voiture **Mégane** est une classe
 - La voiture que vous venez d'acheter est une instance
 - Elle est **bleue**, sans options
 - Une autre instance est
 - Une voiture **rouge** toutes options

Classes et instances

- **Utilisation simple des objets**

- Déclarer une classe

```
class voiture {  
    //contenu de la classe  
}
```

- Attributs

- ❖ Le contenu de la classe est structuré en deux parties
 - ❖ La première partie permet de déclarer les attributs
 - ❖ Les attributs sont déclarés en utilisant la syntaxe des variables et un des mots-clés suivant : **public, private ou protected**

- Exemple

```
class voiture {  
    public $marque;  
}
```

Classes et instances

- Il est possible d'utiliser une valeur par défaut

```
<?php
    class voiture {
        public $marque='Ferrari';
    }
?>
```

- Méthodes

- La deuxième partie du contenu d'une classe permet la déclaration des méthodes
- Ces méthodes se déclarent exactement comme des fonctions

- Exemple

```
<?php
    class voiture {
        public $marque='Ferrari';
        function freiner($force_de_freinage){
            //instructions pour faire freiner
        }
    }
?>
```

Exemple Récapitulatif

- **Un Objet Simple Etudiant**
 - Ecrivons un objet représentant un étudiant avec
- ses données :
 - **identifiant**
 - **nom**
 - **date de naissance**
- et des méthodes pour opérer sur ces données :
 - **constructeur**
 - **getters et setters**
 - **toString() pour affichage**

Démo

La gestion des fichiers avec PHP

La gestion des fichiers

■ Généralités

- De nombreuses applications travaillent avec des fichiers
- Les fichiers permettent de sauvegarder des données sur disque et les rendent en quelque sorte pérennes
- On peut les utiliser dans des sessions ultérieures
- Il existe de nombreuses fonctions pour les lire, les remplir, les supprimer et même changer leurs attributs
- La communication entre le script PHP et le fichier **est repérée par une variable**, indiquant l'état du fichier et qui est passée en paramètre aux fonctions spécialisées pour le manipuler

La gestion des fichiers avec PHP

■ Principe

- PHP prend en charge l'accès au système de fichiers du système d'exploitation du serveur
- Les opérations sur les fichiers concernent la création, l'ouverture, la suppression, la copie, la lecture et l'écriture de fichiers

La gestion des fichiers avec PHP

■ Fichier texte

- PHP ne lit et n'écrit dans les fichiers que sous forme de texte
- Un fichier texte est un fichier caractère, construit sous un éditeur de texte
- Il est organisé sous la forme d'une suite de lignes de caractères
- Le fichier est suffixé par .txt

La gestion des fichiers avec PHP

■ Ouverture de fichiers

- Pour travailler sur un fichier texte, il faut l'ouvrir
- On utilise pour cela fonction fopen
- La fonction **fopen()** permet d'ouvrir un fichier, que ce soit pour le lire, le créer ou y écrire : **fopen(chaine nom du fichier, chaine mode);**
 - **mode** : indique le type d'opération qu'il sera possible d'effectuer sur le fichier après ouverture.
 - **r** (read): indique une ouverture en lecture seulement
 - **w** (write): indique une ouverture en écriture seulement (la fonction crée le fichier s'il n'existe pas)
 - **a** (append): indique une ouverture en écriture seulement avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas)
 - lorsque le mode est suivie du caractère **+** celui-ci peut être lu et écrit

La gestion des fichiers avec PHP

■ Ouverture de fichiers

| Mode | Description |
|------|--|
| r | ouverture en lecture seulement |
| w | ouverture en écriture seulement (la fonction crée le fichier s'il n'existe pas) |
| a | ouverture en écriture seulement avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas) |
| r+ | ouverture en lecture et écriture |
| w+ | ouverture en lecture et écriture (la fonction crée le fichier s'il n'existe pas) |
| a+ | ouverture en lecture et écriture avec ajout du contenu à la fin du fichier (la fonction crée le fichier s'il n'existe pas) |

La gestion des fichiers avec PHP

■ Ouverture de fichiers

- Exemple

```
$fp=fopen ("fichier.txt", "r") ;
```

```
//lecture
```

```
$fp=fopen ("fichier.txt", "w") ;
```

```
//écriture depuis début du fichier
```

La gestion des fichiers avec PHP

■ Ouverture de fichiers

- Il est généralement utile de tester si l'ouverture de fichier s'est bien déroulée sinon stopper le script PHP si cela n'est pas le cas
- Un fichier ouvert avec la fonction **fopen()** doit être fermé, à la fin de son utilisation, par la fonction **fclose()** en lui passant en paramètre l'entier retourné par la fonction **fopen()**
- **Exemple:**

```
<?php
if (!$fp=fopen("fichier.txt","r")){
    echo "Echec de l'ouverture du fichier !!!";
    exit("Unable to open file!");
}
else {
    echo "l'ouverture du fichier s'est bien déroulée !".$fp;
}

fclose($fp);

?>
```

l'ouverture du fichier s'est bien déroulée !Resource id #3

La gestion des fichiers avec PHP

■ Écrire dans un fichier

- L'écriture dans un fichier texte se fait avec la fonction fwrite(). Pour faire des retours à la ligne vous devez utiliser : "\n"

```
<?php
$f = 'exemple1.txt';
$text = "Licence GI 2022";
$var_entier=34;
$var_decimal=2500.567;
$file = fopen($f,"w");    //w: Tentative de création si celui ci n'existe pas
fwrite($file, $text);
fwrite($file, "\n" );
fwrite($file, $var_entier);
fwrite($file, "\n" );
fwrite($file, $var_decimal);
fclose($file);
echo "Done, Check your file !";
?>
```

- Les entiers et les décimaux sont écrits sous forme de chaînes de caractères et non sous forme binaire

La gestion des fichiers avec PHP

■ Écrire dans un fichier, de manière contrôlée

```
<?php
$f = 'exemple.txt';
$text = "ma chaine de caractères";
$file = fopen($f,"w");
// Regarde si le fichier est accessible en écriture
if (is_writable($f)) { // Écrit $text et vérifie
if (fwrite($file, $text) === FALSE) {
echo 'Impossible d\'écrire dans le fichier '.$f.'';
exit;
}
echo 'Ecriture terminée';
fclose($file);
}
else {
echo 'Impossible d\'écrire dans le fichier '.$f.'';
}
?>
```

La gestion des fichiers avec PHP

■ Trouver la fin d'un fichier

- La fonction feof() est utilisée pour tester la fin du fichier

```
<?php  
if (feof($f))  
echo 'Fin du fichier';  
?>
```

La gestion des fichiers avec PHP

■ Comment lire caractère par caractère ?

- Par **fgetc()**
- Exemple:

```
<?php
if (!($f=fopen("exemple.txt","r")))
exit("Impossible d'ouvrir le fichier.");
while (!feof($f))
{
$texte=fgetc($f);
echo $texte;
}
fclose($f);
?>
```

La gestion des fichiers avec PHP

■ Lecture et écriture de fichiers (ligne par ligne)

- Il est possible de lire le contenu d'un fichier et d'y écrire des informations grâce aux fonctions:
 - **fputs()** permet d'écrire une chaîne de caractères dans le fichier. Elle renvoie 0 en cas d'échec, 1 dans le cas contraire
 - **fputs(entier Etat_du_fichier, chaine Sortie);**
 - **fgets()** permet de récupérer une ligne du fichier. Elle renvoie 0 en cas d'échec, 1 dans le cas contraire
 - **fgets(entier Etat_du_fichier, entier Longueur);**
 - Le paramètre Longueur désigne le nombre de caractères maximum que la fonction est sensée récupérer sur la ligne
 - Pour récupérer l'intégralité du contenu d'un fichier, il faut insérer la fonction **fgets()** dans une **boucle while**. On utilise la fonction **feof()**, fonction testant la fin du fichier.

La gestion des fichiers avec PHP

■ Lecture et écriture de fichiers

```
<?php
$fp=fopen("fichier.txt","r");
if(!$fp)
    {echo "Echec de l'ouverture du fichier";}
else {
    $Fich="";
    while(!feof($fp)){//tant que l'on est pas à la fin du fichier

        // On récupère une ligne - 25 la taille max de la ligne
        $Ligne = fgets($fp,25);
        echo $Ligne;
        // On stocke l'ensemble des lignes dans une variable
        $Fich.= $Ligne."<br>";
    }
    // On ferme le fichier
    fclose($fp);
    echo $Fich; }
?>
```

La gestion des fichiers avec PHP

- **Lecture et écriture de fichiers**
- Pour stocker des informations dans le fichier, il faut dans un premier temps ouvrir le fichier en écriture en le créant s'il n'existe pas
 - Deux choix : le mode 'w' et le mode 'a'.

```
<?php
$nom="Amine"; $email="email@gmail.fr";
$fp=fopen("fichier.txt","a");
// ouverture du fichier en écriture
fputs($fp, "\n"); // on va a la ligne
fputs($fp, $nom."|".$email);
// on écrit le nom et email dans le fichier
fclose($fp);
?>
```

La gestion des fichiers avec PHP

■ Les tests de fichiers

- **is_dir()** permet de savoir si le fichier dont le nom est passé en paramètre correspond à un répertoire.
 - La fonction **is_dir()** renvoie 1 s'il s'agit d'un répertoire, 0 dans le cas contraire

booléen **is_dir(chaine Nom_du_fichier);**

```
$file = "images";
if(is_dir($file)) {
    echo ("$file is a directory");
} else {
    echo ("$file is not a directory");
}
```

La gestion des fichiers avec PHP

■ Les tests de fichiers

- **is_executable()** permet de savoir si le fichier dont le nom est passé en paramètre est exécutable.
 - La fonction `is_executable()` renvoie 1 si le fichier est exécutable, 0 dans le cas contraire

booléen `is_executable(chaine Nom_du_fichier);`

```
<?php  
$myfile = "gfg.exe";  
  
if (is_executable($myfile))  
    echo ("$myfile: executable!");  
else  
    echo ("$myfile: non executable!");  
  
?>
```

La gestion des fichiers avec PHP

■ Les tests de fichiers

- **is_file()** permet de savoir si le fichier dont le nom est passé en paramètre ne correspond ni à un répertoire, ni à un lien symbolique.
 - La fonction **is_file()** renvoie 1 s'il s'agit d'un fichier, 0 dans le cas contraire

booléen **is_file(chaine Nom_du_fichier);**

```
<?php
myfile = "fichier.txt";

// Vérifier si le paramètre est un fichier
if (is_file($myfile)) {
    echo ("$myfile: c'est un fichier !");
} else {
    echo ("$myfile: c'est n'est pas un fichier !!");
}
?>
```

La gestion des fichiers avec PHP

■ Lire un fichier sous forme de tableau

- La fonction **file()** permet de retourner dans un tableau l'intégralité d'un fichier en mettant chacune de ces lignes dans un élément du tableau
- Ceci est très pratique quand vous avez besoin d'une ligne en particulier dans votre fichier (dont vous connaissez le numéro) ou encore pour faire des opérations de tri alphabétique, numérique...

➤ **Tableau = file(chaine nomdufichier);**

- **Exemple** : parcourir l'ensemble du tableau afin d'afficher le fichier

```
<?php
$Fichier = "fichier.txt";
if (is_file($Fichier)) {
    if ($TabFich = file($Fichier)) {
        for($i = 0; $i < count($TabFich); $i++)
            echo $TabFich[$i];
    } else {echo "Le fichier ne peut pas être lu...<br>"}
} else {echo " Erreur ! le fichier n'est pas valide<br>"}
?>
```

La gestion des fichiers avec PHP

■ Lire un fichier sous forme de tableau

• Tableau file(chaine nomdufichier);

- Exemple complet : parcourir l'ensemble du tableau afin d'afficher le fichier

```
<?php

```

La gestion des fichiers avec PHP

■ Le téléchargement de fichier

- Le langage PHP dispose de plusieurs outils facilitant le téléchargement vers le serveur et la gestion des fichiers provenant d'un client
- Un simple formulaire comportant un champ de type file suffit au téléchargement d'un fichier qui devra être traité par la suite par un script PHP adapté

```
<form method="POST" action="traitement.php">
<input type="hidden" name="MAX_FILE_SIZE" value="Taille_Octets">
<input type="file" name="fichier" size="30"><br>
<input type="submit" name="telechargement" value="telecharger">
</form>
```

La gestion des fichiers avec PHP

■ Le téléchargement de fichier en PHP

- Un champ caché doit être présent dans le formulaire afin de spécifier une taille maximum (`MAX_FILE_SIZE`) pour le fichier à télécharger. Cette taille est par défaut égale à deux mégaoctets **2Mo**.
- En PHP, le tableau associatif global `$FILES` contient plusieurs informations sur le fichier téléchargé.
 - `$_FILES['fichier']['name']` : fournit le nom d'origine du fichier.
 - `$_FILES['fichier']['type']` : fournit le type du fichier.
 - `$_FILES['fichier']['size']` : fournit la taille en octets du fichier.
 - `$_FILES['fichier']['tmp_name']` : fournit le nom temporaire du fichier.
 - `$_FILES['fichier']['error']`: fournit le code de l'erreur associé au téléchargement du fichier.

La gestion des fichiers avec PHP

■ Le téléchargement de fichier

- Par défaut, le fichier envoyé par le client est stocké directement dans le répertoire indiqué par l'option de configuration `upload_tmp_dir` dans le fichier `php.ini`.

```
$uploadTempDir = ini_get('upload_tmp_dir');
```

- Plusieurs fonctions spécialisées permettent la validation d'un fichier téléchargé pour son utilisation ultérieure.

➤ La fonction `is_uploaded_file` indique si le fichier a bien été téléchargé par la méthode **HTTP POST**.

```
$boolean=is_uploaded_file($_FILES['fichier']['tmp_name']);
```

- La fonction `move_uploaded_file` vérifie si le fichier a été téléchargé par la méthode HTTP POST, puis si c'est le cas le déplace vers l'emplacement spécifié.

La gestion des fichiers avec PHP

```
<!-- Fichier : formulaire.html -->
<html><body>
    <form method="POST" action="traitement.php" enctype="multipart/form-data">
        <input type="hidden" name="MAX_FILE_SIZE" value="1000000">
        <input type="file" name="fichier" size="30"><br>
        <input type="submit" name="telechargement" value="telecharger">
    </form> </body></html>

<?php /* Fichier : traitement.php*/
if (is_uploaded_file($_FILES['fichier']['tmp_name'])) {
    $fichier_temp = $_FILES ['fichier']['tmp_name'];
    $uploadTempDir = ini_get('upload_tmp_dir');

    echo "<h3>Le fichier a été téléchargé avec succès " . "à l'emplacement suivant : <br>" .
$uploadTempDir. "'</h3>";
    $nom_fichier = $_FILES ['fichier']['name'];
    echo "<h3>Le nom d'origine du fichier est '" . $nom_fichier . "'.</h3>";
    echo "<h3>Le type du fichier est '" . $_FILES['fichier']['type'] . "'.</h3>";
    echo "<h3>La taille du fichier est de '" . $_FILES['fichier']['size']."'octets </h3>";
    echo "<h3>ERROR '" . $_FILES['fichier']['error']."' </h3>"; // 0 means success
}

?>
```

La gestion des fichiers avec PHP

```
<!-- Fichier : formulaire.html -->
<html><body>
<form method="POST" action="traitement.php" enctype="multipart/form-data">
<input type="hidden" name="MAX_FILE_SIZE" value="1000000">
<input type="file" name="fichier" size="30"><br>
<input type="submit" name="telechargement" value="telecharger">
</form> </body></html>

<?php /* Fichier : traitement.php*/
if (is_uploaded_file($_FILES['fichier'][tmp_name])) {
    $fichier_temp = $_FILES ['fichier'][tmp_name];
    $uploadTempDir = ini_get('upload_tmp_dir');

    echo "<h3>Le fichier a été téléchargé avec succès " . "à l'emplacement suivant : <br>" .
$uploadTempDir. "</h3>";
    $nom_fichier = $_FILES ['fichier'][name'];
    echo "<h3>Le nom d'origine du fichier est '" . $nom_fichier . "'.</h3>";
    echo "<h3>Le type du fichier est '" . $_FILES['fichier'][type] . "'.</h3>";
    echo "<h3>La taille du fichier est de '" . $_FILES['fichier'][size].".octets </h3>";
    echo "<h3>ERROR '" .$_FILES['fichier'][error]." </h3>"; // 0 means success
}

?>
```

Le fichier a été téléchargé avec succès à l'emplacement suivant :
'C:\xampp\tmp'

Le nom d'origine du fichier est 'Capture ____ .PNG'.

Le type du fichier est 'image/png'.

La taille du fichier est de '13870octets

ERROR : 0

La gestion des fichiers avec PHP

■ Les fonctions de système de fichiers

`true | false = copy(fichier, nouveau_fichier);` copie un fichier vers une nouvelle destination.

`delete(fichier);` efface le fichier.

`$chaine = dirname(chemin);` retourne le nom du dossier parent.

`$nombre = disk_total_space(dossier);` retourne la taille totale d'un dossier.

`true | false = fclose(ID_fichier);`

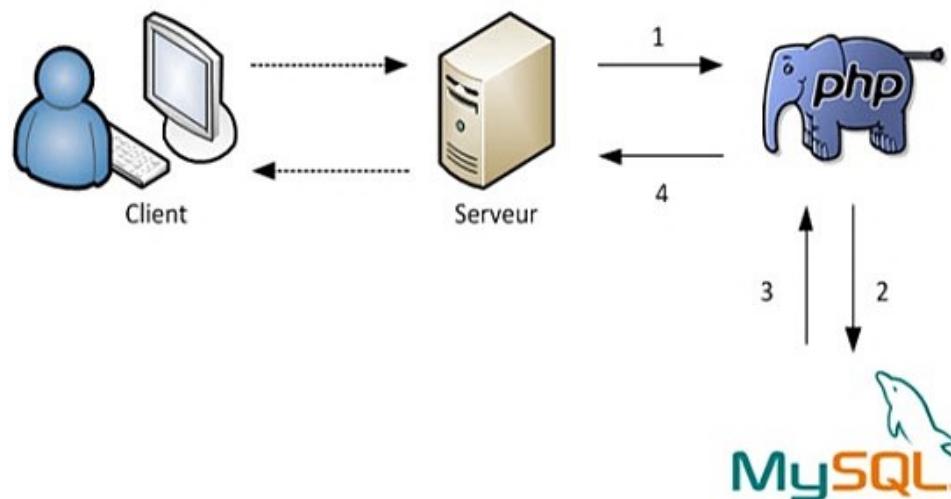
ferme un fichier indiqué par un identificateur retourné par fopen.

`true | false = feof(ID_fichier);` teste la fin du fichier.

PHP et connexion MySQL

PHP et MySQL

- **Présentation de MySQL**
- MySQL est un SGBDR
- MySQL est un produit Open Source
- On doit communiquer avec le SGBD pour lui donner l'ordre de récupérer ou d'enregistrer des données. Pour cela on utilise le langage SQL.
- Et c'est PHP qui va faire l'intermédiaire entre l'utilisateur et MySQL.



PHP et MySQL

■ Présentation de MySQL

- Par exemple le serveur a reçu une demande d'un client qui veut poster un message sur un forum:
 1. le serveur utilise toujours PHP, il lui fait donc passer le message ;
 2. PHP effectue les actions demandées et se rend compte qu'il a besoin de MySQL. En effet, le code PHP contient à un endroit « Va demander à MySQL d'enregistrer ce message ». Il fait donc passer le travail à MySQL ;
 3. MySQL fait le travail que PHP lui avait soumis
 4. PHP renvoie au serveur que MySQL a bien fait ce qui lui était demandé.

■ Se connecter à la base de données en PHP

- PHP propose plusieurs moyens de se connecter à une base de données MySQL:
- **L'extension mysql_** : ce sont des fonctions qui permettent d'accéder à une base de données MySQL et donc de communiquer avec MySQL. Toutefois, ces fonctions sont vieilles et on recommande de ne plus les utiliser aujourd'hui.
- **L'extension mysqli_** : ce sont des fonctions améliorées d'accès à MySQL. Elles proposent plus de fonctionnalités et sont plus à jour.
- **L'extension / Classe PDO (PHP Data Object)** : c'est un outil complet qui permet d'accéder à n'importe quel type de base de données. On peut donc l'utiliser pour se connecter aussi bien à MySQL que PostgreSQL ou Oracle,...
 - L'approche PDO permet d'étendre les fonctions d'accès à la base facilement et de manière transparente

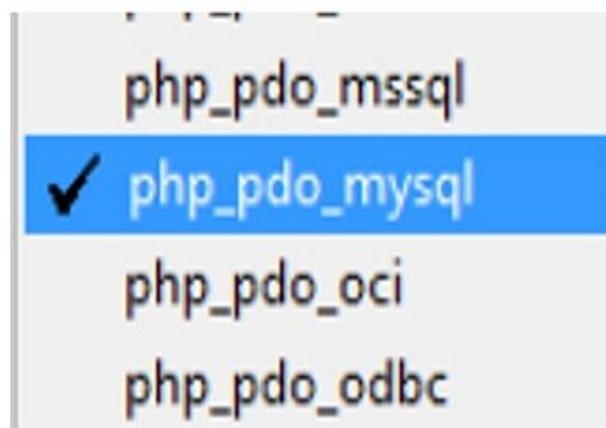
■ Utiliser votre base de données

- L'utilisation de la BD avec PHP se fait en 5 étapes
 - Connexion
 - Sélection de la BD
 - Requête
 - Exploitation des résultats
 - Fermeture de la connexion
- Structure des classes de PDO
 - PDO propose 3 classes
 - ❖ PDO : lien à la BD
 - ❖ PDOStatement : requêtes et leurs résultats
 - ❖ PDOException : pour la gestion des erreurs

- **Se connecter à la base de données en PHP**

Activer PDO (Si désactivé):

Normalement, PDO est activé par défaut. Pour le vérifier (voir la figure suivante), faites un clic gauche sur l'icône de WAMP dans la barre des tâches, puis allez dans le menu PHP / Extensions PHP et vérifiez que `php_pdo_mysql` est bien coché.



PHP et MySQL

■ Connexion au serveur de données

- La première étape consiste à déclarer les variables qui vont permettre la connexion à la base de données (ce sont les paramètres des fonctions de connexion à la base)
- Ces variables sont :
 - \$user : le nom d'utilisateur
 - \$passwd : le mot de passe
 - \$host : l'hôte (ordinateur sur lequel le SGBD est installé)
 - \$bdd : le nom de la base de données
- La deuxième étape est de construire la DSN (Data Source Name)
 - `$dsn = 'mysql:host=localhost;dbname=ma-base';`
- Enfin, la connexion se fait par la création d'un objet de la classe PDO
 - `$dbh= new PDO($dsn, $user, $pass);`
- Il ne faut pas oublier de fermer la base
 - `$dbh=null;`

PHP et MySQL

■ Exemple:

connexion-PDO.php

```
<?php
$username = "root";
$pass= "root";
$host="localhost";
$db="lst_gi";
$dsn = "mysql:host=$host;dbname=$db";

try{
    $cnx = new PDO($dsn,$username,$pass);
    if($cnx)
    {
        echo "Connected to the $db database successfully!";
    }

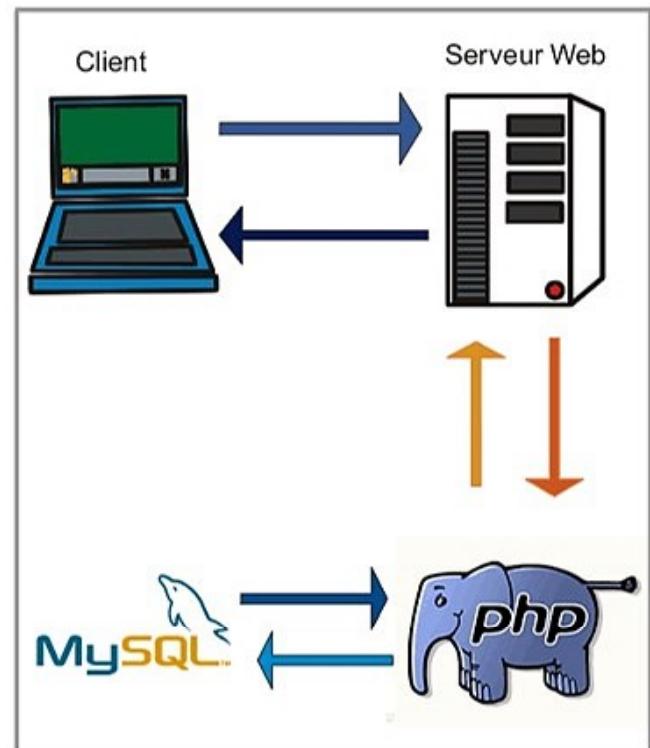
}catch(PDOException $e){
    $error_cnx = $e->getMessage();
    echo $error_cnx;
    exit();
}
?>
```

■ Effectuer une requête

- Une fois la connexion ouverte, on va pouvoir utiliser la BD pour lire, modifier...
- Pour cela, on utilise le langage **SQL**

■ Pour envoyer une requête au serveur

- On peut utiliser deux méthodes de la classe PDO
 - `query()` pour la sélection
 - `exec()` pour la mise à jour



■ Requête de sélection

- On utilise la méthode `query()`
- Les données ne sont pas affichées, elles sont mises en mémoire
- Il faut donc aller les chercher et les afficher
- La méthode `fetchAll()` retourne l'ensemble des données sous forme d'un tableau PHP et libère le SGBD
- La méthode `fetch()` permet une lecture séquentielle du résultat
- Le paramètre `fetch_style` détermine la façon dont PDO retourne les résultats (format des résultats)

■ Effectuer une requête: Créeer une table

creation-table.php

```
<?php
//Inclusion du fichier contenant la connexion à la base
include_once('connexion-PDO.php');

//Création de la table personne
$sql="CREATE TABLE personne ( id_personne INTEGER PRIMARY KEY , nom VARCHAR( 20 ) NOT
NULL ,prenom VARCHAR( 20 ) , depart INTEGER( 2 ))";

try{
    $sth = $cnx->query($sql);
    if($sth)
    {
        echo "<br> table bien crée !";
    }
}catch(PDOException $e){
    $error_cnx = $e->getMessage();
    echo "<br><br>".$error_cnx;
    exit();
}
?>
```

PHP et MySQL

■ Effectuer une requête: Lire tous les enregistrements -fetchAll

```
<?php  
//Inclusion du fichier contenant la connexion à la base  
include_once('connexion-PDO.php');  
//La requête SQL  
$sql = "SELECT * FROM `etudiant` LIMIT 0 , 30";  
//Recherche des données  
$sth = $cnx->query($sql);  
// On voudrait les résultats sous la forme d'un tableau associatif  
$result = $sth->fetchAll(PDO::FETCH_ASSOC);  
  
echo "<br><br>";  
  
//Affichage des résultats  
foreach ($result as $row){  
echo "Nom: ".$row['nom']; echo '<br/>';  
echo "Prenom: ".$row['prenom'];echo '<br/>';  
echo "Email: ".$row['email'];echo '<br/> -----<br/>';  
}  
$cnx=NULL;  
?>
```

lire-enregistrements.php

PHP et MySQL

■ Effectuer une requête: Lire tous les enregistrements

■ La table Etudiant

A screenshot of a MySQL query interface. At the top, a green bar displays the message "✓ Affichage des lignes 0 - 1 (total de 2, traitement en 0,0014 seconde(s))." Below this, a SQL query is shown: "SELECT * FROM `etudiant`". A toolbar below the query includes options: Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]. Further down are filtering options: "Tout afficher" (checkbox), "Nombre de lignes : 25", "Filtrer les lignes: Chercher dans cette tab", and "Trier par clé : Aucun(e)". The main area shows a table with two rows of data:

| | id | nom | prenom | email |
|---|----|-------|---------|-----------------|
| <input type="checkbox"/> Copier <input checked="" type="checkbox"/> Supprimer | 1 | Amine | Ahmed | amine@gmail.com |
| <input type="checkbox"/> Copier <input checked="" type="checkbox"/> Supprimer | 2 | Saber | Mohamed | saber@gmail.com |

■ Résultat

A screenshot of a web browser window. The address bar shows the URL: "localhost/PHPExemples/DBmysql/lire-enregistrements.php". The page content is as follows:

Connected to the lst_gi database successfully!

Nom: Amine
Prenom: Ahmed
Email: amine@gmail.com

Nom: Saber
Prenom: Mohamed
Email: saber@gmail.com

PHP et MySQL

■ Effectuer une requête: Lire tous les enregistrements - fetch

```
<?php  
//Inclusion du fichier contenant la connexion à la base  
include_once('connexion-PDO.php');  
//La requête SQL  
$sql = "SELECT * FROM `etudiant` ";  
//Recherche des données  
$sth = $cnx->query($sql);  
  
//Affichage des résultats  
  
while($rss = $sth->fetch())  
{  
echo nl2br($rss['id']." ,".$rss['nom']." ,".$rss['prenom']." ,".$rss['email']."\n");  
}  
$cnx=NULL;  
?>
```

lire-enregistrements.php

PHP et MySQL

■ Construire des requêtes en fonction de variables 1 – Marqueur « ? »

Au lieu d'exécuter la requête avec **query()** comme la dernière fois, on appelle ici **prepare()**. La requête est alors prête, sans sa partie variable. Maintenant, nous allons exécuter la requête en appelant **execute()** et en lui transmettant la liste des paramètres

```
<?php  
//Inclusion du fichier contenant la connexion à la base  
include_once('connexion-PDO.php');  
//La requête SQL  
$sql = "SELECT * FROM `etudiant` where id=?";  
$v = $_GET['id'];  
  
$sth = $cnx->prepare($sql);  
$rq = $sth->execute(array($v));  
$result = $sth->fetchAll(PDO::FETCH_ASSOC);  
foreach($result as $rss)  
{  
    echo nl2br($rss['id']." ,".$rss['nom']." ,".$rss['prenom']." ,".$rss['email'])."\n";  
}  
?  
 lire-enregistrements.php
```

La requête est alors exécutée à l'aide des paramètres que l'on a indiqués sous forme d'array.

PHP et MySQL

■ Construire des requêtes en fonction de variables 2 – Plusieurs marqueurs « ? »

Au lieu d'exécuter la requête avec **query()** comme la dernière fois, on appelle ici **prepare()**. La requête est alors prête, sans sa partie variable. Maintenant, nous allons exécuter la requête en appelant **execute()** et en lui transmettant la liste des paramètres

- ✓ S'il y a plusieurs marqueurs, il faut indiquer les paramètres dans le bon ordre

```
<?php  
//Inclusion du fichier contenant la connexion à la base  
include_once('connexion-PDO.php');  
//La requête SQL  
$sql = "SELECT * FROM `etudiant` where nom=? and prenom=? ";  
$nom = $_GET['nom'];  
$prenom = $_GET['prenom'];  
  
$sth = $cnx->prepare($sql);  
$rq = $sth->execute([$nom,$prenom]); // execute(array($nom,$prenom));  
$result = $sth->fetchAll(PDO::FETCH_ASSOC);  
foreach($result as $rss)  
{  
    echo nl2br($rss['id']." ,".$rss['nom']." ,".$rss['prenom']." ,".$rss['email'])."\n";  
}  
?>
```

lire-enregistrements.php

PHP et MySQL

■ Construire des requêtes en fonction de variables 3 – Marqueur nominatif

Au lieu d'exécuter la requête avec **query()** comme la dernière fois, on appelle ici **prepare()**. La requête est alors prête, sans sa partie variable. Maintenant, nous allons exécuter la requête en appelant **execute()** et en lui transmettant la liste des paramètres

- ✓ Si la requête contient beaucoup de parties variables, il peut être plus pratique de nommer les marqueurs plutôt que d'utiliser des points d'interrogation.

```
<?php  
//Inclusion du fichier contenant la connexion à la base  
include_once('connexion-PDO.php');  
//La requête SQL  
$sql = "SELECT * FROM `etudiant` where nom=:n and prenom=:pr";  
$sth = $cnx->prepare($sql);  
$rq = $sth->execute(array( "pr"=>$_GET['prenom'], "n"=> $_GET['nom']));  
// ce n'est pas obligatoire d'envoyer les variables dans le même ordre que la requête !  
$result = $sth->fetchAll(PDO::FETCH_ASSOC);  
foreach($result as $rss)  
{  
    echo nl2br($rss['id']." ,".$rss['nom']." ,".$rss['prenom']." ,".$rss['email'] ."\n");  
}  
?>
```

lire-enregistrements.php

- **Effectuer une requête: calculer le nombre d'enregistrements**
- **Deux manières :**
 - Créer une requête spécifique en utilisant la fonction `count()` de MySql
 - Compter le nombre d'éléments contenus dans le tableau renvoyé par la méthode `fetchAll()`

PHP et MySQL

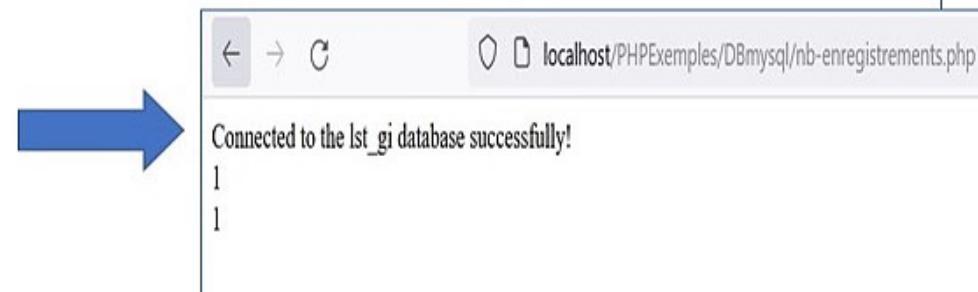
■ Exemple:

nb-enregistrements.php

```
<?php
//Inclusion du fichier contenant la connexion à la base
include_once('connexion-PDO.php');
echo "<br/>";
//1) En utilisant une requête particulière
$sql = "SELECT COUNT(*) as nbe FROM `etudiant` WHERE nom='Amine'";
$q = $cnx->query($sql);
$result = $q->fetchAll();
$nombre = $result[0]['nbe'];
echo $nombre;
echo "<br/>";

//2) En comptant le nombre d'éléments présents dans le tableau des
résultats
$sql = "SELECT * FROM `etudiant` WHERE nom='Amine'";
$qq = $cnx->query($sql);
$result = $qq->fetchAll();
$nombre = count($result);
echo $nombre;

$cnx=NULL;
?>
```



PHP et MySQL

- Effectuer une requête: **Requête d'insertion**
- On utilise la méthode **exec()** de la classe PDO

```
<?php
//Inclusion du fichier contenant la connexion à la base
include_once('connexion-PDO.php');
echo "<br/>";

$sql ="INSERT INTO `etudiant` (`id`, `nom`, `prenom`, `email`)
      VALUES (NULL, 'Saad', 'Karim', 'karim@gmail.com'); ";
$cnx->exec($sql);
$cnx=NULL;
?>
```

PHP et MySQL

▪ Effectuer une requête: Requête de modification « update »

```
<?php
//Inclusion du fichier contenant la connexion à la base
include_once('connexion-PDO.php');

$id = 5;      $nom = "mohammadi";      $prenom = "Amine";      $email = "mod_ali@gmail.com";

try{
    $sql ="update `etudiant` SET `nom` = :nom,`prenom` = :prenom, `email` = :email where id = :id";

    $req = $cnx->prepare($sql);
    // $req->execute(array("nom"=>$nom,"prenom"=>$prenom,"email"=>$email,"id"=>$id));

    $req->bindParam(":nom",$nom);
    $req->bindParam(":prenom",$prenom);
    $req->bindParam(":email",$email);
    $req->bindParam(":id",$id);
    $req->execute();
    echo "update success !!";
}catch( PDOException $e){
    echo $e->getMessage();
    exit;
}
$cnx=NULL; ?>
```

PHP et MySQL

- Effectuer une requête: Requête de suppression « delete »

```
<?php
//Inclusion du fichier contenant la connexion à la base
include_once('connexion-PDO.php');

$id = 5;

try{

    $sql ="delete from `etudiant` where id = :id";
    $req = $cnx->prepare($sql);
    // $req->execute(array("id"=>$id));

    $req->bindParam(":id",$id);
    $req->execute();
    echo "delete success !!";

}catch( PDOException $e){

    echo $e->getMessage();
    exit;
}
$cnx=NULL;
?>
```