Technologies du WEB

PHP

Tarek AIT BAHA

t.aitbaha@uiz.ac.ma

ESEF AGADIR - Université IBN ZOHR

Plan du cours

- Introduction
- Syntaxe & Caractéristiques générales
- Chaînes de caractères
- Fonctions & Procédures
- Les fichiers
- Les formulaires
- Les Cookies
- Les Sessions
- Bases de données

Plan du cours

- **Introduction**
- Syntaxe & Caractéristiques générales
- Chaînes de caractères
- Fonctions & Procédures
- Les fichiers
- Les formulaires
- Les Cookies
- Les Sessions
- Bases de données

Introduction

Qu’est ce que le PHP ?

- ***PHP***, signifie "***Hypertext Preprocessor***" ***(Préprocesseur HyperTexte)*** et est ***un langage de script***.
- L'essentiel de sa syntaxe est emprunté aux langages C, Java et Perl, mais y ajoute plusieurs fonctionnalités uniques.
- Le ***but*** de ce langage est de permettre aux développeurs web de concevoir rapidement des sites aux contenus dynamiques.
- C’est un langage ***incrusté au HTML*** et ***interprété côté serveur***.

Introduction

Que peut faire le PHP ?

- Le langage ***PHP*** possède les mêmes fonctionnalités que les autres langages permettant d'écrire des scripts, comme collecter des données, générer dynamiquement des pages web … .
- La ***plus grande qualité*** et le ***plus important avantage*** du ***langage PHP*** est ***le support d'un grand nombre de bases de données***.
- Réaliser une page web dynamique interfaçant une base de données est extrêmement simple en PHP.

Notion du serveur Web

- PHP est ***un langage côté serveur***, pour pouvoir voir les résultats de vos pages PHP vous devez ***avoir un environnement dans lequel se trouve un serveur web***.

- Un ***serveur HTTP*** ou tout simplement ***serveur Web***, est un logiciel servant des requêtes respectant le protocole de communication client-serveur ***HyperText Transfer Protocol (HTTP)***, qui a été développé pour le World Wide Web.

Notion du serveur Web

- Un ordinateur sur lequel fonctionne un ***serveur HTTP*** est appelé ***serveur Web***. Le terme «***serveur Web***» peut aussi désigner le serveur HTTP (le logiciel) lui-même.
- Les serveurs HTTP les plus utilisés sont :
  - ✓ Apache HTTP Server de la Apache Software Foundation.
  - ✓ Internet Information Services de Microsoft (IIS)
  - ✓ Sun ONE de Sun Microsystems

Le plus populaire est ***Apache HTTP Server*** qui sert environ 70% des sites Web.

Notion du serveur Web

***WampServer* :**

- installe et configure automatiquement un environnement de travail complet permettant de mettre en œuvre toute la puissance et la souplesse qu'offrent le langage dynamique PHP et son support efficace des bases de données.
- Regroupe un serveur Apache, une base de données MySQL, le langage PHP ainsi que des outils facilitant le développement de vos sites ou de vos applications.

Plan du cours

- Introduction
- **Syntaxe & Caractéristiques générales**
- Chaînes de caractères
- Fonctions & Procédures
- Les fichiers
- Les formulaires
- Les Cookies
- Les Sessions
- Bases de données

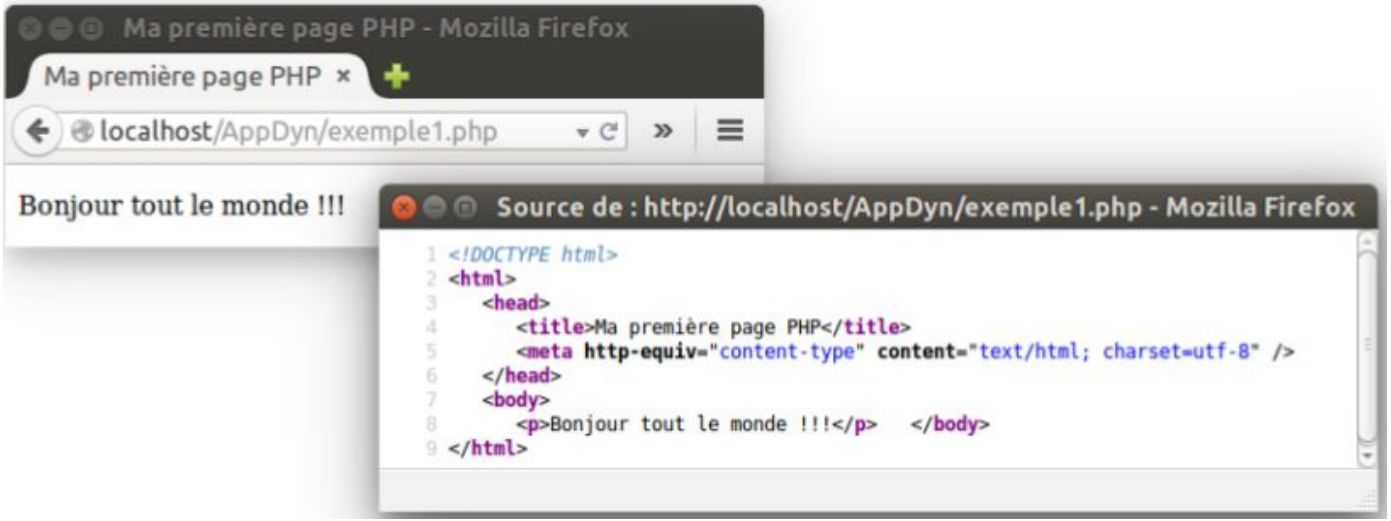
Syntaxe & Caractéristiques générales - Inclusion d'un code PHP

- Les *pages web* sont au format ***html***. Les *pages web dynamiques* générées avec **PHP** sont au format ***php***.
- Le code source PHP est directement insérer dans le fichier html grâce au conteneur de la forme `<? Php Code ?>`
- Autres syntaxes d'intégration du code PHP dans du HTML :
  - `<? Code ?>`
  - `<script language= "php"> Code </script>`
  - `<% Code %>`

La notation la plus utilisée et la plus conseillée est bien sur `<? Php Code ?>`

Syntaxe & Caractéristiques générales - Premier script PHP

```php
<!DOCTYPE html>
<html>
  <head>
    <title>Page PHP</title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
  </head>
  <body>
    <?php echo "<p>Bonjour le monde !!</p>\n"; ?>
  </body>
</html>
```



Syntaxe & Caractéristiques générales - Syntaxe de base

- La fin de chaque ligne d'instructions est marquée obligatoirement par un point-virgule « ; ».

```
<?php instruction1; instruction2; ... ?>
```

- Un bloc d'instructions est encadré par { }
- Les commentaires : Syntaxe à la C, C++ ou Shell

```
/* ..... */

// .....

# .....
```

Syntaxe & Caractéristiques générales - Les variables

Le typage des variables est *dynamique*

Syntaxe : `$NomDeVariable[=val];`

- Règles de nommage : `$[a-zA-Z\_]([a-zA-Z0-9\_])*`
- Sensibilité à la casse.
- Assignation par :
  - valeur : `$var1=$var2;`
  - référence : `$var1=&$var2;`
- La variable est ***locale*** à la fonction où elle est déclarée

Exemple:

```php
<?php
  $var1=10;
  $var2=&$var1;
  $var1=20;
  echo "<p>".$var1." ".
 $var2."</p>";
?>
```

Affichera : 20 20

Syntaxe & Caractéristiques générales - Les variables

Variable Globale :

- Déclaration de variable globale : `global $var;`

Exemple:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page PHP</title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
  </head>
  <body>
    <?php
      function affiche() {
        global $var1;
        echo "<p>Hello ".$var1." !!!</p>\n";
      }
    ?>
    <?php
        $var1="John";
        affiche();
    ?>
  </body>
</html>
```

Syntaxe & Caractéristiques générales - Les constantes

Syntaxe : `define("NOM_DE_LA_CONSTANTE", valeur)`

Les constantes :

- ne commencent pas par **$**.
- sont définies et accessibles globalement dans tout le code.
- ne peuvent pas être redéfinies.
- ne peuvent contenir que des booléens, des entiers, des flottants et des chaînes de caractères.

Exemple:

```
define("PHP","PHP Hypertext Preprocessor");
echo PHP;
```

Syntaxe & Caractéristiques générales - Les types

Il existe :

4 types simples :

- entiers : `integer`
- réels : `float` , `double`
- booléens : `boolean` (`TRUE` ou `FALSE`)
- chaînes de caractères : `string`

2 types complexes :

- tableaux : `array`
- objets : `object`

2 types spéciaux :

- ressources : `resource` (ex : connexion BD)
- absence de valeur : `null`

Syntaxe & Caractéristiques générales - Les tableaux

Principe : Associations ordonnées de type `clef` ⇒ `valeur`

Déclaration : `array( [clef =>] valeur, ...)`

- La clef est facultative, elle est de type entier ou chaîne de caractères ; en cas d'omission, la valeur sera associée à la clef d'indice max+1.
- La valeur peut être de n'importe quel type.

Exemple:

```
$tab=array("fruit"=>"pomme",42,"légume"=>"salade",1.5e3);
foreach($tab as $cle => $valeur) {
  echo "<p>".$cle."=>".$valeur."</p>";
}
echo "<p>tab[1]=".$tab[1]."</p>";
$tab[]="peu importe";
echo "<p>tab[2]=".$tab[2]."</p>";
echo "<p>tab['fruit']=".$tab["fruit"]."</p>";
```

Syntaxe & Caractéristiques générales - Opérations sur les tableaux

Attention, un tableau est toujours une référence !

- `count($array)` : nombre d'éléments.
- `sort($array)` : trie le tableau.
- `array_pop($array)` : récupère et supprime le dernier élément d'une liste.
- `array_push($array, $elem1, ...)` : ajoute des éléments à la fin d'une liste.
- `array_shift($array)` : récupère et supprime le premier élément d'une liste.
- `array_unshift($array, $elem1, ...)` : ajout d'éléments en début de liste.
- `array_merge($array1, $array2, ...)` : fusionne plusieurs tableaux.
- `in_array($elem, $array)` : recherche d'un élément dans un tableau.
- `array_key_exists($key, $array)` : recherche une clef dans un tableau.
- `array_flip($array)` : inverse les clef et les valeurs d'un tableau

Syntaxe & Caractéristiques générales - Détermination du type d'une variable

Type d'une variable : `string gettype($var);`

Test : `is_integer($var); is_double($var); is_array($var);` …

Conversion dynamique : `$result = (type-désiré)$var;`

Instructions de vérification d'existence (formulaires) :

- `boolean isset($var);` retourne `FALSE` si `$var` n'est pas initialisée ou a la valeur `NULL`, `TRUE` sinon.
- `boolean empty($var);` retourne `TRUE` si `$var` n'est pas initialisée, a la valeur 0, "0", ou `NULL`, `FALSE` sinon.
- `boolean is_null($var);` retourne `TRUE` si `$var` n'est pas initialisée ou vaut `NULL`, `FALSE` sinon (inverse de `isset`).

Syntaxe & Caractéristiques générales - Exemple

val	gettype()	empty()	is_null()	isSet()	(bool)
$x = "";	string	true	false	true	false
$x = null;	NULL	true	true	false	false
var $x ; (not set)	NULL	true	true	false	false
$x = array();	Array	true	false	true	false
$x = false;	boolean	true	false	true	false
$x = 15;	integer	false	false	true	true
$x = 1;	integer	false	false	true	true
$x = 0;	integer	true	false	true	false
$x = -1;	integer	false	false	true	true !
$x = "15";	string	false	false	true	true
$x = "1";	string	false	false	true	true
$x = "0";	string	true	false	true	false !
$x = "-1";	string	false	false	true	true
$x = "foo";	string	false	false	true	true
$x = "true";	string	false	false	true	true
$x = "false";	string	false	false	true	true !

Syntaxe & Caractéristiques générales - Syntaxe de base

- Les opérateurs sont identiques à ceux du C/C++/Java :
  - Arithmétiques : `+ - * / %`
  - De comparaison : `== != <= >= < >`
  - D'affectation : `= += -= *=`
  - Logiques : `&& || !`
  - in/décrémentation : `var++ var-- ++var --var`
  - Concaténation de chaînes de caractères : `.`
- Opérateurs spécifiques :
  - `‘commande shell‘` (ex : `$a = ‘ls -ul‘`)
  - `===` : Teste la valeur et le type.

Syntaxe & Caractéristiques générales - Instructions de branchement

Proches du C/C++/Java :

Si-sinon-alors :

```
if(condition) {
  instructions
}
[elseif(condition) {
  instructions
}]
[else {
  instructions
}]
```

Switch-case :

```
switch(expression) {
  case 'valeur1':
    Instructions
    break;
  ...
  default:
    Instructions
    break;
}
```

Syntaxe & Caractéristiques générales - Les boucles

Proches du C/C++/Java :

Boucles for :

```
for($i=0; $i<N; $i++) {
   Instructions
}

foreach($tab as $val) {
   Instructions
}

foreach($tab as $cle=>$val) {
   Instructions
}
```

Boucles while :

```
while(condition) {
   Instructions
}

do {
   Instructions
} while(condition);
```

Syntaxe & Caractéristiques générales - Exemple

Utilisation des boucles pour répéter du code HTML :

Entrelacement code PHP / code HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
    <title>Page PHP</title>
  </head>
  <body>
    <?php
      $tab=array("premier","second","troisième","...");
    ?>
    <ul>
      <?php foreach($tab as $elem) { ?>
        <li><?php echo $elem; ?></li>
      <?php } ?>
    </ul>
  </body>
</html>
```

Plan du cours

- Introduction
- Syntaxe & Caractéristiques générales
- **Chaînes de caractères**
- Fonctions & Procédures
- Les fichiers
- Les formulaires
- Les Cookies
- Les Sessions
- Bases de données

Chaînes de caractères - Déclaration et fonctionnement

- Les chaînes peuvent être déclarées avec :
  - Simples quotes : `$t='texte';`
  - Doubles quotes : `$t="texte";`
- Fonctionnement différent : entre doubles quotes, les variables et les caractères échappatoires sont interprétés.

Exemple :

Pour `t="Mot";`

Double cotes	Résultat	Simple cote	Résultat
"Texte"	Texte	'Texte'	Texte
"Texte $t"	Texte Mot	'Texte $t'	Texte $t
"Texte \n Suite"	Texte Suite	'Texte \n Suite'	Texte \n Suite
"Texte \"guillemets\""	Texte "guillemets"	'Texte \"guillemets"\'	Texte \"guillemets"\
"L'heure"	L'heure	'L\'heure'	L'heure

Chaînes de caractères - Opérations sur les chaînes de caractères

- Longueur : `int strlen(string $ch)`
- Répétition : `string str_repeat(string $cr, int $nb)`
- Minuscules : `string strtolower(string $ch)`
- Majuscules : `string strtoupper(string $ch)`
- Initiales en majuscules : `string ucwords(string $ch)`
- 1 ère lettre en majuscule : `string ucfirst(string $ch)`
- Suppression de caractères en début de chaîne : `string ltrim(string $ch, string liste)`
- Suppression de caractères en fin de chaîne : `string rtrim(string $ch, string liste)`
- Suppression de caractères en début et fin de chaîne : `string trim(string $ch, string liste)`

Chaînes de caractères - Exemple

```
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
    <title>Page PHP</title>
  </head>
  <body>
    <?php
      $prenom = "...JEan__";
      $nom = "  BONNEAu";
      $adresse = "10 rue abraham lincoln";
      $email = "jean-BONNEAU@asi.insa-rouen.fr";

      $complet = ucfirst(strtolower(trim($prenom,'._')));
      $complet .= " ".strtoupper(ltrim($nom,' '));
      $espaces = strlen($complet)+3;
      echo $complet." : ".ucwords($adresse)."<br />";
      echo str_repeat(".", $espaces).strtolower($email);
    ?>
  </body>
</html>
```

Chaînes de caractères - Sous-chaînes de caractères

- Recherche sensible à la casse (retourne tous les caractères de `$ch` depuis la 1ère occurence de `$ch2` jusqu'à la fin) : `string strstr(string $ch, string $ch2)`
- Recherche insensible à la casse : `string stristr(string $ch, string $ch2)`
- Extraction de chaînes de caractères : `string substr(string $chr, int indice, int N)`
- Décompte du nombre d'occurrences d'une sous-chaîne : `int substr_count(string $ch, string $ssch)`
- Remplacement : `string str_replace(string $oldssch, string $newssch, string $ch)`
- Position : `int strpos(string $ch, string $ssch)`

Chaînes de caractères - Exemple

```
    <?php
      $ch = "Un pot de lait et un pot de miel";

      echo strstr($ch, "pot")."<br />";
            // affiche "pot de lait et un pot de miel"

      echo substr($ch, 18, 6)."<br />";
            // affiche "un pot"

      echo substr_count($ch, "pot")."<br />";
            // affiche "2"

      echo str_replace("pot", "broc", $ch)."<br />";
            // affiche "Un broc de lait et un broc de miel"

      echo strpos($ch, "un pot")."<br />";
            // affiche "18"
    ?>
  </body>
</html>
```

Chaînes de caractères - Les expressions rationnelles

Une ***expression rationnelle*** (RegEx) permet de définir un motif de caractères, représentatif d'un ensemble de chaînes de caractères.

- Caractère(s) : `""` ou `' '` (ex : **`"a", "ab"`**)
- Caractères spéciaux : **`\., \$, \^ , \?, \\, \[, \], \(, \), \+ et \*`**
- Classe de caractères : `[]` (ex : **`[xyz], [a-z]`**)
- Classes de caractères prédéfinies :
    - **`[[:alnum:]]`** : caractères alphanumériques.
    - **`[[:alpha:]]`** : caractères alphabétiques
    - **`[[:ctrl:]]`** : caractères de contrôle
    - **`[[:digit:]]`** : chiffres
    - **`[[:punct:]]`** : caractères de ponctuation
    - **`[[:upper:]]`** : majuscules

Chaînes de caractères - Modèles généraux

- N'importe quel caractère : `.`
- 0 ou 1 fois : `?` (ex : `"https?"`)
- Au moins une fois : `+`
- 0, 1 ou plusieurs fois : `*` (ex : `"mat.*"`)
- Exactement n fois : `"{n}"`
- Au moins n fois : `"{n,}"`
- Entre n et m fois : `"{n,m}"`
- Groupements : `()` (ex : `"(ma)*"`)
- Alternative : `|` (ex : `"(\.net)|(\.com)"`)

Exemple:

```
"[[:digit:]]{2}/[[:digit:]]{2}/[[:digit:]]{4}"
"[[:alnum:]]*\.[[:alnum:]]*@asi\.insa-rouen\.fr
```

Chaînes de caractères - Fonctions de recherche et de remplacement

- Fonctions de recherche :

```
int preg_match ( string $modeleregex , string $chaine [ , array & $matches ] )
```

- Fonctions de remplacement :

```
mixed preg_replace ( mixed $modeleregex , mixed $replacement , mixed $chaine )
```

Chaînes de caractères - Exemple

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page PHP</title>
    <meta http-equiv="content-type" content="text/html;charset=utf-8" />
  </head>
  <body>
    <?php
      $chaine = "La dernière version est :\nPHP 4 (4ème version)\nVive le PHP 4
    ";
      $chaine = preg_replace("/4/", "5", $chaine);
      $chaine = preg_replace("/\n/", "<br />", $chaine);
      echo $chaine;
    ?>
  </body>
</html>
```

Plan du cours

- Introduction
- Syntaxe & Caractéristiques générales
- Chaînes de caractères
- **Fonctions & Procédures**
- Les fichiers
- Les formulaires
- Les Cookies
- Les Sessions
- Bases de données

Fonctions & Procédures

- Les fonctions et les procédures regroupent un ensemble d'instructions permettant d'accomplir un ensemble d'opérations.
- Elle peuvent accepter des valeurs appelées ***arguments*** ou ***paramètres***.
- Une ***fonction*** réalise une succession d'instructions et ***renvoie une seule valeur issue d'un calcul*** (en utilisant ***return***).
- Une ***procédure*** réalise une succession d'instructions ***sans avoir fournir une valeur de retour***.

Fonctions & Procédures - Création d'une fonction ou d'une procédure

- La ***création d'une fonction ou d'une procédure***, se fait par l'indication du ***nom de la fonction*** et/ou ***de la procédure*** est précédé du mot clé `function`.
- Les instructions qui composent le corps de la fonction/procédure sont encadrées par des ***accolades*** (`{` et `}`).

Syntaxe :

```
function NomDeLaFonction([$arg1, ..., $argn])
{
    instructions; ...
}
```

Fonctions & Procédures - Exemples fonction et valeur retournée

```php
<?php
/*
appel à la fonction calcul de superficie d'un disque
qui prend en paramètre la taille du rayon
*/
$rayon = 15;
$aire = calcul_superficie($rayon);
echo "L'aire du disque de rayon $rayon est : $aire";
?>
```

Pour déclarer cette fonction, la construction est la suivante :

```php
<?php
// fonction de calcul de la superficie d'un disque
function calcul_superficie($rayon) {
    $sup = $rayon * $rayon * 3.14159;
    return $sup;
}
?>
```

Nous remarquons qu'il nous faut préciser la valeur renvoyée par la fonction à l'aide de la commande *return*. De plus, il faut interroger directement la valeur renvoyée par la fonction, ce qui est classique.

Le résultat sera le suivant : `L'aire du disque de rayon 15 est : 706.85775`

Fonctions & Procédures - Exemples fonction sans argument ou paramètre

Voici le cas d'une fonction qui ne prend pas de paramètre en entrée et retourne la dizaine inférieure des minutes de l'heure courante.

```php
<?php
function dizaine() {
    $date = new DateTime(); // objet DateTime (programmation orientée objet)
    $minutes = $date->format("i");
    $min = floor($minutes/10)*10;
    return $min;
}
echo dizaine();
?>
```

Sans argument on laisse les parenthèses vides

Fonctions & Procédures - Exemples fonction avec un argument facultatif

Il suffit de préciser dans la déclaration de la fonction l'argument en lui assignant une valeur.

```php
<?php
function dizaineh($heure = 0) {
    $date = new DateTime(); // objet DateTime (programmation orientée objet)
    $minutes = $date->format("i");
    $min = floor($minutes/10)*10;
    if ($heure)
        $min = $date->format("H").":".$min;
    return $min;
}
echo dizaineh()."<br />".dizaineh(1);
?>
```

Le résultat :

```
30
22:30
```

Fonctions & Procédures - Exemples fonction qui retourne plusieurs valeurs

Dans ce cas, il faut retourner un tableau que l'on exploitera ensuite comme tel.

```php
<?php
function calcul($n1) {
    $resultat[] = $n1 * 3;
    $resultat[] = $n1 + 3;
    return $resultat;
}
$res = calcul(5);
echo "produit = $res[0] et somme = $res[1]<br />";
list($produit, $somme) = calcul(7);
echo "produit = $produit et somme = $somme";
?>
```

Le résultat sera : `Avec le chiffre 5, produit = 15 et somme = 8`
`Avec le chiffre 7, produit = 21 et somme = 10`

Plan du cours

- Introduction
- Syntaxe & Caractéristiques générales
- Chaînes de caractères
- Fonctions & Procédures
- **Les fichiers**
- Les formulaires
- Les Cookies
- Les Sessions
- Bases de données

Les fichiers - Ouverture des fichiers

- La ***manipulation des fichiers*** en **PHP** se fait grâce à un ensemble de fonctions prédéfinie par le langage.
- ***Ouverture*** : `$fichier = fopen("NOM", "MODE");` ouvre le fichier `NOM` avec *MODE* valant :
  - `r, r+` : lecture et lecture/écriture, pointeur au début.
  - `w, w+` : écriture et lecture/écriture, avec création ou effacement, pointeur au début.
  - `a, a+` : écriture et lecture/écriture, pointeur à la fin, avec création.
  - `x, x+` : création en écriture et lecture/écriture, pointeur au début, erreur en cas d'existence du fichier.
  - `c, c+` : création en écriture et lecture/écriture, pointeur au début, sans erreur.
- ***Fermeture*** : `fclose($fichier);` ferme le fichier identifié par `$fichier`
- ***Présence*** : `file_exists($fichier);`

Les fichiers - Gestion des fichiers

- ***Lecture d'une ligne :***

  `string fgets($fichier, $nbCar)` : Lit une ligne de `$MaxCar` caractères.

- ***Lecture d'un caractère :***

  `string fgetc($fichier)` : Lit un seul caractère et fait avancer le pointeur d'un caractère.

  `fread($fichier,"nombre")` : Lit un nombre donné d'octets (***nombre***) du fichier.

- ***Ecriture d'une ligne :***

  `integer fputs($fichier, $str)` : Écrit la chaîne `$str` dans le fichier identifié par `$fichier`.

  `fwrite($fichier,"chaine")` : Écrit une chaîne de caractères (***chaîne***) dans le fichier.

- ***Test de fin de fichier :***

  `boolean feof($fichier)` : Teste la fin du fichier. Elle retourne true si la fin du fichier est atteinte.

Les fichiers - Exemple

```
<!DOCTYPE html>
<html>
  <head>
        <title>Lecture/Ecriture dans un fichier</title>
        <meta http-equiv="Content-type" content="text/html; charset=utf-8"/>
  </head>
     <?php
        $fichier=fopen("fichier.txt", "a");
        fputs($fichier, "Une phrase\n");
        fclose($fichier);

        $fichier=fopen("fichier.txt", "r");
        echo "<p>Dans le fichier fichier.txt :</p>";
        while(!feof($fichier)){
          echo fgetc($fichier);
        }
        fclose($fichier);
     ?>
</html>
```

Plan du cours

- Introduction
- Syntaxe & Caractéristiques générales
- Chaînes de caractères
- Fonctions & Procédures
- Les fichiers
- **Les formulaires**
- Les Cookies
- Les Sessions
- Bases de données

Les formulaires - Fonctionnement

- L'un des intérêts majeurs de **PHP** est sa capacité à ***exploiter les données*** qui sont fournies par un internaute ***via les formulaires HTML*** bien connus.

- Un formulaire est constitué de champs et de contrôles : zones de texte, boutons radio, cases à cocher, listes déroulantes ...
- Chacun de ces éléments se caractérise par un attribut "***name***" ***qui définira tout simplement le nom de la variable à récupérer en PHP*** avec la valeur rentrée par l'utilisateur.
- Ainsi, rappelons qu'un formulaire HTML se définit avec les balises **<`form`>** et **</`form`>**. L'attribut "`action`" de cette balise permet de ***spécifier la page qui traitera les données fournies*** dans le formulaire par l'internaute ; l'attribut "`method`" la méthode de transmission (`GET` ou `POST`).

Les formulaires - Exemple

```
<form action="resultat.php" method="post">
     Entrez votre prénom : <input type="text" name="prenom" />
     <input type="submit" value="valider" />
</form>
```

Ce qui donnera sur votre navigateur, si vous rentrez : Tarek

- Sur cet exemple, l'utilisateur a la possibilité de spécifier une seule information : son prénom, cette donnée sera ***exploitable dans la page resultat.php*** sous la forme d'une variable `$_POST["prenom"]` *(composée de l'attribut "**name**" de la zone de texte à remplir)* dont la valeur sera égale à la chaîne de caractères entrée.

Les formulaires - Exemple

Ainsi on aura par exemple dans la page ***resultat.php*** :

```php
<?php
echo "Prénom tapé par l'utilisateur : ".$_POST['prenom'];
?>
```

```
Prénom tapé par l'utilisateur : Tarek
```

Les formulaires - Exemple

- ***Une variable dite superglobale est alors automatiquement créée*** et exploitable dans le fichier ***resultat.php*** et uniquement dans celui-ci.
- Tous les contrôles de formulaire qui possèdent un attribut "`name`" engendrent la création d'une ***variable superglobale*** dont le nom est `$_POST["nom de la variable"]` et la valeur fonction des renseignements saisis par l'utilisateur.
- Nous remarquerons l'emploi de la syntaxe `$_POST` qui rappelle que la variable a été construite à partir de la méthode `POST`.
- Notons enfin que l'on peut écrire indistinctement : `$_POST['prenom']`, `$_POST["prenom"]` ou `$_POST[prenom]`.

Les formulaires - La méthode `POST`

- Deux méthodes peuvent être utilisées dans l'envoi de données via les formulaires, la méthode `GET` et `POST`.
- Nous préconisons l'emploi de la méthode `POST` car elle cache les informations transmises (qui transiteraient par l'URL dans le cas de la méthode `GET`).
- De plus, elle permet d'envoyer des données importantes en taille (la méthode `GET` se limite à 255 caractères) et assure la gestion de l'envoi de fichiers.
- La méthode `GET` est utilisée par défaut lorsque l'on utilise l'URL pour faire passer des variables, ce que nous aborderons plus loin sur cette page

Les formulaires - Exemples de contrôles et de champs de formulaires

```
<form method="POST" action="traitement.php">
    Votre adresse e-mail
    <input type="text" name="mail" value="adresse e-mail" size="30" maxlength="55">
    <input type="checkbox" NAME="mailing" value="oui" checked> abonnement gratuit |
    <br>
    <textarea name="description" rows="3" cols="60"></textarea>
    <br> Type d'échange souhaité :
    <select name="demande">
        <option selected="selected" value="echange de liens">Echange de liens</option>
        <option value="partenariat">Partenariat</option>
    </select>
    Dans quelle rubrique souhaitez-vous être présent ?
    <br>
    <input type="radio" name="rubrique" value="environnement" />Environnement / Ecologie
    <br>
    <input type="radio" name="rubrique" value="geographie" />Géographie
    <br>
    <input type="radio" name="rubrique" />Photos
</form>
```

- Prenons l'exemple ci-dessus de traitement des variables qui émanent des différents contrôles et champs renseignés par un internaute sur un formulaire d'échange de liens.

Les formulaires - Exemples de contrôles et de champs de formulaires

Ce qui donne sur votre navigateur, une fois qu'un demandeur a rempli le formulaire :

Les formulaires - Exemples de contrôles et de champs de formulaires

- Nous remarquons que l'appel à la page de traitement des données est "***traitement.php***", on peut tout à fait appeler la même page en utilisant une variable globale PHP intitulée `$_SERVER['PHP_SELF']` qui signifie la page en cours.
- Ainsi, *l'affichage du formulaire* et *l'exploitation de ses résultats* sera opéré sur la même page.
- Ceci implique que dans le code **PHP**, *un test soit effectué* pour *savoir si le formulaire a été rempli ou non* par exemple en testant si une des variables du formulaire a été déclarée avec la fonction `isset()`.

Les formulaires - Exemples de contrôles et de champs de formulaires

- Le code qui permet de récupérer les différentes valeurs saisies par l'utilisateur est le suivant :

```php
<?php
$mail = $_POST["mail"];
echo "votre adresse e-mail est : $mail<br />";
$mailing = (isset($_POST["mailing"])? "oui" : "non");
echo "Avez-vous souhaité vous inscrire sur la lettre d'informations ? $mailing<br />";
$description = htmlentities($_POST["description"],ENT_QUOTES);
echo "et sa description est la suivante : <br />".nl2br($description)."<br />";
$rubrique =  $_POST["rubrique"];
echo "pour la rubrique ".$rubrique;
?>
```

Les formulaires - Exemples de contrôles et de champs de formulaires

Remarques :

- Pour la case à cocher la variable prend la valeur ***true*** ou ***false*** si la case a été cochée ou non.
- L'attribut "`name`" des boutons radios est le même car ils appartiennent à une même liste de choix. Le bouton radio sélectionné crée une variable nommée `$_POST["rubrique"]` qui est égale à la valeur notée dans l'attribut "`value`".
- la liste d'options fonctionne de la même façon pour la valeur de la variable `$_POST["demande"]`, l'attribut "`name`" étant déclaré dans la balise `<select>` .
- Nous utilisons la fonction `htmlentities()` qui transforme certains caractères du texte entré par l'utilisateur dans la zone de texte en d'autres qui ne permettront pas l'exécution de code HTML par une personne malintentionnée.
- nous employons la fonction `nl2br()` qui permet de conserver les sauts de ligne effectués dans la zone de texte

Les formulaires - Exemples de contrôles et de champs de formulaires

Ces données seront ensuite insérées dans une base de données pour y être conservées et exploitées par la suite.

```
votre adresse e-mail est : demandeur@fai.com
Avez-vous souhaité vous inscrire sur la lettre d'informations ? oui
et sa description est la suivante :
Site sur les insectes tropicaux.
Découvrez les plus impressionnants insectes visibles sur notre planète en toute saison : l'été
comme l'hiver.
pour la rubrique photos
```

Plan du cours

- Introduction
- Syntaxe & Caractéristiques générales
- Chaînes de caractères
- Fonctions & Procédures
- Les fichiers
- Les formulaires
- **Les Cookies**
- Les Sessions
- Bases de données

Cookies

- Les cookies sont des fichiers texte qui conservent certains paramètres propres à votre navigation sur le web, de sorte que vous puissiez bénéficier d'une personnalisation lors de votre visite d'un site web.
- Lorsque vous visitez des sites web, vous avez parfois à entrer des informations personnelles, ou à modifier certains paramètres du site afin que celui-ci vous soit plus efficace. Ainsi, lorsque vous revenez sur le site web par la suite, les formulaires peuvent être pré-remplis ou l'interface déjà adaptée à vos besoins définis auparavant. Toutes ces informations "personnelles" sont stockées dans des petits fichiers texte, sur votre ordinateur et plus précisément via votre navigateur qui leur alloue une place, si vous les acceptez.

Cookies

- PHP nous permet de placer des cookies sur l'ordinateur de la personne qui visite votre site web.
- Un cookie peut perdurer sur l'ordinateur client pendant plusieurs jours, mois, années ou seulement le temps de la visite sur le site, mais sont limités à 20 par domaine (ce qui est amplement suffisant), avec un poids unitaire de 4 ko.

  Nous prendrons l'exemple ici d'un cookie que l'on installe pour faciliter la saisie d'informations dans un formulaire.

Cookies - Installer un cookie

- Pour déclarer un cookie sur le poste client on utilise la fonction `setcookie()`.
- **Attention !** Cette fonction doit obligatoirement être placée avant tout contenu HTML (généré ou pas par le PHP), car les cookies se gèrent dans les en-têtes envoyées avant la page web.

Cookies - Installer un cookie

Nous prendrons l'exemple ici d'un cookie que l'on installe pour faciliter la saisie d'informations dans un formulaire.

Exemple :

ici, un formulaire vient d'être validé et l'on récupère le nom qui a été rentré pour l'intégrer dans un cookie dont le nom est 'nom' et la valeur `$_POST['nom']`.

```php
<?php
setcookie('nom', $_POST['nom']);
?>
```

Par la suite, on pourra faire appel à ce cookie pour pré-remplir un formulaire qui réclame le nom de l'internaute :

```php
<?php
echo '<input type="text" name="nom" value="'.$_COOKIE['nom']'." />';
?>
```

Cookies - Installer un cookie

Nous prendrons l'exemple ici d'un cookie que l'on installe pour faciliter la saisie d'informations dans un formulaire.

Exemple :

ici, un formulaire vient d'être validé et l'on récupère le nom qui a été rentré pour l'intégrer dans un cookie dont le nom est 'nom' et la valeur `$_POST['nom']`.

```php
<?php
setcookie('nom', $_POST['nom']);
?>
```

Par la suite, on pourra faire appel à ce cookie pour pré-remplir un formulaire qui réclame le nom de l'internaute :

```php
<?php
echo '<input type="text" name="nom" value="'.$_COOKIE['nom']'." />';
?>
```

Cookies - Installer un cookie

`$_COOKIE[]` est un tableau associatif et une variable superglobale qui contient tous les cookies déclarés sur l'ordinateur client, uniquement pour le site en cours : on ne peut pas récupérer le cookie d'un autre site.

Ce qui nous donnera :

```
Karim
```

Ainsi, l'internaute n'aura pas à rentrer son nom à chaque fois qu'un formulaire sur le site lui demande.

Notons bien que le cookie est envoyé uniquement sur le poste du client qui a effectué une action et que ***Karim*** n'apparaîtra bien sûr pas sur les autres ordinateurs des internautes qui visitent le site...

Cookies - Tester la présence d'un cookie

Il peut arriver qu'on ait besoin de savoir si un cookie est présent sur l'ordinateur de l'internaute pour agir en conséquence.

Par exemple, un compteur de lectures *(pour une actualité)* ne sera pas incrémenté si l'internaute a déjà visité la page en question : un cookie est donc placé à la première visite de la page et on vérifiera sa présence pour les visites ultérieures.

On place un cookie à la première lecture de la page, si celui-ci n'existe pas :

```php
<?php
//incrémentation du compteur de lecture
if (!isset($_COOKIE["news45"])) {
    setcookie ("news45", '1');
}
?>
```

Cookies - Supprimer un cookie

- **Suppression d'un cookie :** On peut abréger la durée de vie d'un cookie en le supprimant. Pour ce faire, on lui donne une durée de vie qui a déjà expiré (ici une heure de moins) :

```php
<?php
setcookie ('nom', '', time() - 3600);
?>
```

Plan du cours

- Introduction
- Syntaxe & Caractéristiques générales
- Chaînes de caractères
- Fonctions & Procédures
- Les fichiers
- Les formulaires
- Les Cookies
- **Les Sessions**
- Bases de données

Sessions

- Une ***session*** en **PHP** correspond à une façon de stocker des données différentes pour chaque utilisateur en utilisant un identifiant de session unique.
- Les identifiants de session vont généralement être envoyés au navigateur via des cookies de session et vont être utilisés pour récupérer les données existantes de la session.
- Un des ***grands intérêts*** des sessions est qu'on va pouvoir conserver des informations pour un utilisateur lorsqu'il navigue d'une page à une autre.
- De plus, les informations de session ne vont cette fois-ci pas être stockées sur les ordinateurs de vos visiteurs à la différence des ***cookies*** mais plutôt côté serveur ce qui fait que les sessions vont pouvoir être beaucoup plus sûres que les cookies.

Sessions - Commencer une session

- **Commencer une session : `session_start()`**
- Cette fonction devra être placée en tête de page (car envoyé dans le header) sur chaque page qui doit utiliser les sessions.
- **Attention !** Aucun caractère en devra être affiché avant cette fonction.

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

Sessions - Stocker dans une session

- **Stocker des objets dans la session : `$_SESSION`**
- ***Exemples :***

```php
<?php
    session_start();
    // enregistrer une variable session
    $_SESSION['user']="Toto";
?>
<html> <body>
<?php
    //Récupérer la valeurs de la variable session
    echo "Bienvenue ". $_SESSION['user'];
?>
</body> </html>
```

Sessions - Supprimer une variable stocker dans une session

- Pour supprimer une variable stockée dans la session : `unset()`
- ***Exemples :***

```php
<?php
    unset($_SESSION['views']);
?>
```

- Pour supprimer la session : `session_destroy()`
- ***Exemples :***

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

Plan du cours

- Introduction
- Syntaxe & Caractéristiques générales
- Chaînes de caractères
- Fonctions & Procédures
- Les fichiers
- Les formulaires
- Les Cookies
- Les Sessions
- **Bases de données**

Bases de données - Introduction

- L'intérêt majeur de **PHP** est son interfaçage avec un grand nombre de bases de données d'une manière relativement simple et efficace.
- Nous avons vu en ***introduction*** que pratiquement tous les **SGBD** sont pris en charge, mais **PHP** s'utilise bien souvent avec **MySQL**, un **SGBD** rapide et qui satisfait à la plupart des sites Internet, même si il n'a pas encore toutes les potentialités d'autres.
- Nous développerons dans cette partie les fonctions **PHP** qui permettent d'exploiter des bases de données **MySQL**.
- Pour la visualisation de vos tables et bases de données **MySQL,** l'emploi de l'excellent **phpMyAdmin** ,qui est une interface en ligne très simple d'utilisation et redoutable dans ses possibilités, est conseillé.

Bases de données - MySQL & PHP

- Pour se connecter à **MySQL** via **PHP** on utilise une **API** (**A**pplication **P**rogramming **I**nterface, ou interface de programmation d'application) qui définit les classes, méthodes, fonctions et variables dont votre application va faire usage pour exécuter différentes tâches.

- Les API peuvent être ***procédurale*** ou ***orientée objet***.
- Avec une ***API procédurale***, vous pouvez appeler les fonctions pour exécuter des commandes.
- Avec une ***API orientée objet***, vous devez créer des objets et appeler les méthodes de ces objets.

Bases de données - MySQL & PHP

Il existe trois API principales pour se connecter à **MySQL** :

- L'extension `mysql` (pour les versions PHP < 4.1.3).
- L'extension `mysqli` (mysql improved pour les versions PHP > 5)
- **PHP Data Objects** (PDO) qui a des fonctionnalités moins avancées que mysqli mais permet de travailler avec différentes bases de données en conservant le même code.

Nous utiliserons l'extention `mysqli` avec une syntaxe orientée objet.

Bases de données - MySQL & PHP

Étapes pour se connecter à une base de données :

L'exploitation de MySQL avec PHP s'effectue en plusieurs étapes :

1. Ouverture d'une connexion à MySQL et sélection de la base de données.
2. Requête sur la base de données.
3. Exploitation des résultats de la requête.
4. Fermeture de la connexion à MySQL.

Bases de données - MySQL & PHP

Ouvrir une connexion à la base de données MySQL :

- Avant d'exploiter les données qui se trouvent sur notre base, il nous faut ouvrir un canal de communication qui ne sera fermé qu'une fois toutes nos opérations effectuées.
- Notre base de données se trouve sur un serveur désigné avec un nom de domaine, un adresse IP ou un alias. Pour nous y connecter, il nous faut un login et un mot de passe.
- *Tous ces paramètres sont fournis par votre hébergeur lorsque vous ouvrez un hébergement.*
- Nous choisirons pour nos exemples les variables suivantes :

```php
<?php
$serveur = "mysql.monserveursql.com";
$base = "mabase";
$user = "monuser";
$pass = "monpass";
?>
```

Bases de données -

Pour se connecter, on utilise la classe **`mysqli`**.

Nous effectuons un test pour savoir si la connexion est effective ou pas avec l'alias **`die()`** (de *exit()*) : en cas d'échec, le code s'interrompra ici en affichant l'erreur rencontrée par MySQL et donnée par la méthode **`connect_error`**.

```php
<?php
$serveur = "mysql.monserveursql.com";
$base = "mabase";
$user = "monuser";
$pass = "monpass";

/*
$mysqli est une nouvelle instance de la classe mysqli
prédéfinie dans php et hérite donc de ses propriétés et méthodes
connexion à la base de données
*/
$mysqli = new mysqli($serveur, $user, $pass, $base);
// si la connexion se fait en UTF-8, sinon ne rien indiquer
$mysqli->set_charset("utf8");
/*
utilisation de la méthode connect_error
qui renvoie un message d'erreur si la connexion échoue
*/
if ($mysqli->connect_error) {
    die('Erreur de connexion ('.$mysqli->connect_errno.')'. $mysqli->connect_error);
}
?>
```

Bases de données - MySQL & PHP

On pourrait également faire un test plus classique comme ci-dessous :

```php
<?php
if ($mysqli->connect_error) {
    echo 'connexion impossible... :'.$mysqli->connect_error;
}
else {
    echo 'connexion réussie : '.$mysqli->host_info;
}
?>
```

Afin d'éviter de définir vos identifiants de connexion à chaque fois que vous exploitez votre base de données, il est utile de les préciser dans un fichier qui sera appelé à chaque fois via l'instruction `include()`. Ainsi, vous pourrez facilement les mettre à jour si besoin et renforcerez la sécurité de votre site web.

Bases de données - MySQL & PHP

Fermer la connexion :

Normalement, la fermeture de la connexion se fait automatiquement, mais afin de s'en assurer, on peut utiliser la méthode `close()`

```php
<?php
$mysqli->close();
?>
```

Bases de données - Les requêtes MySQL en PHP

Requête : cas de la clause SELECT :

Une fois que nous sommes connectés à MySQL et à une base de données, nous pouvons effectuer des tâches classiques dans un SGBD : *afficher une table*, *supprimer un enregistrement*, *mettre à jour un enregistrement*, *afficher le résultat d'une requête* …

Il est donc indispensable d'avoir au préalable une bonne connaissance du langage SQL

Toutes ces actions sont directement effectuées en **SQL** en exploitant les méthodes de la classe ***mysqli***. Ici on utilisera la méthode PHP `query()`.

Bases de données - Les requêtes MySQL en PHP

Requête : cas de la clause SELECT :

Admettons que nous soyons connectés à notre base de données comme vu précédemment.

Nous souhaitons par exemple afficher tous les enregistrements de la table nommée ***'user'*** pour lesquels le champ intitulé ***'nom'*** est égal à ***'Samia'*** :

```php
<?php
$requete = "SELECT * FROM user WHERE nom = 'Samia'";
$resultat = $mysqli->query ($requete);
?>
```

Cet exemple suffit pour comprendre comment PHP s'interface avec MySQL : une requête SQL est passée en argument de la fonction **`query()`**, cette requête est ici déclarée avant dans une variable de type chaîne de caractères. Le raccourci suivant est bien sûr possible :

```php
$resultat = $mysqli->query ("SELECT * FROM user WHERE nom = 'Samia'");
```

Bases de données - Les requêtes MySQL en PHP

Récupérer un enregistrement avec la clause SELECT :

Notons que le résultat de la requête se trouve dans la classe **MySQLi_Result** nommée **`$resultat`**

Ce résultat peut ensuite être exploité via différentes méthodes / fonctions.

Nous utiliserons ici la méthode **`fetch_assoc()`** comme en témoigne l'exemple ci-dessous :

```php
<?php
$ligne = $resultat->fetch_assoc();
echo 'le premier enregistrement a pour utilisateur '.$ligne["user"].'
et pour score '.$ligne["score"];
?>
```

Ce qui donnera: **`le premier enregistrement a pour utilisateur Samia et pour score 72`**

Pour récupérer les valeurs des champs d'un enregistrement, on utilise donc la fonction **`fetch_assoc()`** qui retourne l'enregistrement en cours sous la forme d'un ***tableau associatif***.

Bases de données - Les requêtes MySQL en PHP

Exploiter plusieurs enregistrements avec la clause SELECT :

En effet, si notre requête retourne plusieurs enregistrements, il suffit d'effectuer une boucle **while** qui lira tous les enregistrements jusqu'au dernier pour affichage, exploitation…

```
$req = "SELECT * FROM personne";
if($res = $mysqli -> query($req)) {
    while($ligne = mysql_fetch_row($res)) {
        $code = $ligne[0];
        $nom = $ligne[1];
        $adresse = $ligne[2];
        echo "$code - $nom , $adresse <br>";
    }
}
```

Bases de données - Les requêtes MySQL en PHP

Insérer un enregistrement avec la clause INSERT :

Pour la mise à jour de la base nous avons besoin de deux fichiers :

- Un premier fichier qui doit contenir un formulaire HTML dans lequel l'internaute peut saisir les données à insérer dans la base.
- Un deuxième fichier PHP qui permette de récupérer les données du formulaire pour les insérer dans la base.

Bases de données - Les requêtes MySQL en PHP

Insérer un enregistrement avec la clause INSERT :

- Ici, le fichier peut être un simple fichier HTML, puisque toutes les éléments de la page sont statiques. Nous l'appelons "*saisie.html*".
- **Notez que la mise en page HTML est des plus sommaires, ceci afin d'améliorer la lisibilité et se concentrer sur les éléments importants du code.**

```
<html>

<body>
    <h1>Mise à jour de la base</h1>
    <form method="post" action="maj.php">
        Référence :
        <input type="text" name="ref">
        <br> Nom :
        <input type="text" name="nom">
        <br> Prix :
        <input type="text" name="prix">
        <br>
        <input type=submit value="Insérer">
    </form>
</body>

</html>
```

Bases de données - Les requêtes MySQL en PHP

Insérer un enregistrement avec la clause INSERT :

- Nous voulons maintenant récupérer les données du formulaires pour les insérer dans la base.
- Avant de l'écrire, il est d'abord nécessaire de savoir comment sont transmises les données du formulaire à notre nouveau fichier (celui désigné par l'attribut **action** dans le formulaire précédent), et comment on peut récupérer ces valeurs en PHP.
- PHP récupère ces valeurs postées dans un tableau qu'il nomme **`$_POST`**. C'est un tableau associatif, dont les index de tableaux sont les noms des variables.Il est donc facile de retrouver la valeur associée à une variable. Par exemple, **$_POST['prix']** désigne la valeur associée au champ HTML prix dans le formulaire posté.

Bases de données - Les requêtes MySQL en PHP

Insérer un enregistrement avec la clause INSERT :

- Maintenant que nous savons récupérer les valeurs des variables du formulaire, il faut les sauvegarder dans la base de données.
- Nous écrivons un deuxième fichier, que nous appelons "*insert.php*". Sur le principe seule la requête SQL change, avec un **`INSERT`** au lieu d'un **`SELECT`**.

```
<?php
  // recuperation des valeurs du formulaire
  $nom = $_POST['nom'];
  $prix= $_POST['prix'];

  // insertion des valeurs dans la base
  $query = "INSERT INTO produit (nom,prix) VALUES ('$nom','$prix')";
  $result = mysqli->query($query);
  mysqli->close();
?>
    Mise a jour faite
```

- Ici, on suppose toujours que ref est une clé primaire de la table produit, et qu'elle possède en plus la propriété **`autoincrement`**. Ainsi, nous n'avons pas à gérer l'unicité de la référence, puisque MySQL inserera automatiquement une référence entière supérieure aux autres références trouvées.

Bases de données - Les requêtes MySQL en PHP

Modifier un enregistrement avec la clause Update :

- La modification de données *(on dit encore la mise à jour, ou le remplacement de valeur)* peut porter sur une ou plusieurs lignes, d'une ou plusieurs tables.
- La modification en SQL se dit **`UPDATE`**.
- On indique les noms des colonnes concernées et les valeurs à y mettre *(**`SET`** col1=valeur1, col2=valeur2, ...)* et à quelles lignes cela s'applique *(**`WHERE`** condition)*.
- ***Exemple:*** le nom d'un domaine informatique change de '*esefa.com*' à '*esefa.uiz.ac.ma*'. Je veux mettre à jour la colonne '***nondomaine***' de ma table ***'adresses'*** en conséquence. La commande suivante modifiera toutes les lignes qui ont '*esefa.com*''.

```
UPDATE adresses SET nomdomaine='esefa.uiz.ac.ma' WHERE nomdomaine='esefa.com'
```

Bases de données - Les requêtes MySQL en PHP

Modifier un enregistrement avec la clause Update :

- Reprenons notre exemple précédent. On veut modifier le prix d'un article dont on connait la référence, stockée dans la variable **`$ref`**. La référence est la clé primaire *(colonne ref)* de notre table 'lecteurs'. Le remplacement va être opéré si la valeur présente dans **`$ref`** existe déjà dans la table.
- MySQL introduit une extension (ne faisant pas partie du standard SQL) permettant de modifier toute une ligne: **`REPLACE`**.

```
$query = "REPLACE INTO produit (ref,marque,modele,annee,prix)
          VALUES ('$ref','$marque','$modele','$annee','$nouveau_prix')";
$ok = $mysqli->query( $query );
if ($ok)
   echo "Le prix de l'article $ref est modifié ...";
```

Bases de données - Les requêtes MySQL en PHP

Supprimer un enregistrement avec la clause DELETE :

Pour supprimer des données d'une table, nous allons utiliser l'instruction SQL **DELETE**.

```
<?php

  // Suppression de lignes dans une table
  $query = "DELETE FROM produit WHERE nom='chaise'";
  $result = $mysqli->query($query);
  $mysqli->close();
?>
    Mise a jour faite
```