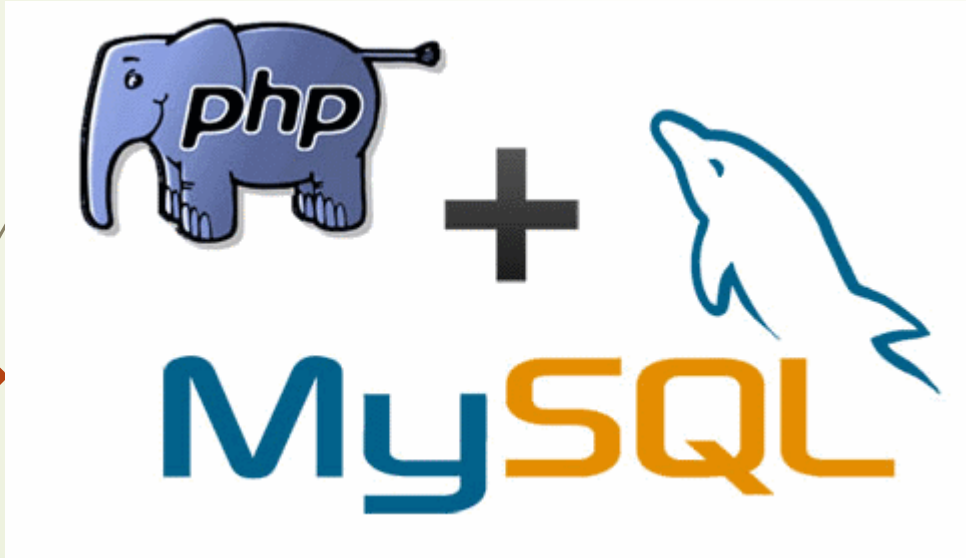
PHP / MySql



Master **IMSD**

IFRISSE

Pierre Claver OUEDRAOGO

Tables de matières

Aperçu du cours

Objectifs du cours

Dans ce cours, nous allons étudier de façon pratique les différentes fonctionnalités du PHP et de MySQL et voir comment les utiliser ensemble pour exploiter tout leur potentiel.

Ce cours aborde l'ensemble des fonctionnalités de base et utiles du PHP et du MySql. L'objectif étant de compléter vos connaissances en développement web.

Ce cours va ainsi vous permettre d'écrire , de comprendre du code PHP et de manipuler les données avec dans MySql.

Prérequis

Pour suivre ce cours dans de bonnes conditions, il est essentiel que vous possédiez des bases en HTML et en CSS. Ce qui est le cas puisque vous avez déjà eu le cours d'HTML et CSS.

Unité 0 : Définitions et rôles du PHP et du MySQL

Définition et rôle du PHP

- Le terme PHP est l'acronyme de « PHP Hypertext Preprocessor ». Le premier « P » de PHP est en effet lui-même l'abréviation de « PHP », une curiosité qui ne va pas présenter une grande importance pour nous.
- Ce langage a été créé en 1994. Sa version stable la plus récente (au 15 juillet 2019) est la version 7.3.7
- Le PHP va nous permettre de créer des pages qui vont être générées dynamiquement. En d'autres mots, grâce au PHP, nous allons pouvoir afficher des contenus différents sur une même page en fonction de certaines variables : l'heure de la journée, le fait que l'utilisateur soit connu et connecté ou pas, etc
- De plus, notez que le PHP va s'exécuter côté serveur. Il fait ainsi partie des langages qu'on nomme « server side » en opposition aux langages « client side » qui s'exécutent côté client

Unité 0 : Définitions et rôles du PHP et du MySQL

Définition et rôle du MySQL

Le MySQL est un système de gestion de bases de données relationnelles. Une base de données est un ensemble structuré de données. Les données vont pouvoir être des informations clients (nom, adresse, mot de passe, etc.), la liste des commentaires de notre blog, le texte de nos articles, etc.

Le problème ici est qu'on ne va pas directement pouvoir interagir avec les bases de données car les données sont stockées d'une manière illisible pour un humain. Pour manipuler les données stockées dans les bases de données, nous allons devoir utiliser un langage de bases de données.

Le langage de bases de données le plus célèbre est le SQL. SQL est l'acronyme de Structured Quercy Language (Langage de Requêtes Structurées).

Le système de gestion de bases de données MySQL utilise le langage SQL pour la manipulation des données des bases de données.

Unité 0 : Définitions et rôles du PHP et du MySQL

Définition et rôle du MySQL

Les avantages du MySQL sont sa simplicité d'utilisation, sa fiabilité et ses performances en plus du fait qu'on va pouvoir gérer plusieurs types de bases de données différentes si besoin avec MySQL et qu'on va pouvoir l'utiliser conjointement avec PHP.

Par exemple , nous allons pouvoir créer nos formulaires d'inscription en HTML et allons ensuite récupérer les données des formulaires en PHP. Ici, nous allons vouloir enregistrer ces données dans une base de données.

Une **base de données** (En abrégé **BD**, en anglais DB) **est** une entité dans laquelle il **est** possible de stocker des **données** de façon structurée.

Pourquoi utiliser le couple PHP/MySQL ?

Concrètement, il n'y a pas de raison « absolue ». Cependant, si le couple PHP / MySQL reste de loin le plus célèbre et le choix de référence lorsqu'on veut créer des sites dynamiques et stocker des données, c'est pour de bonnes raisons. On note par exemple :

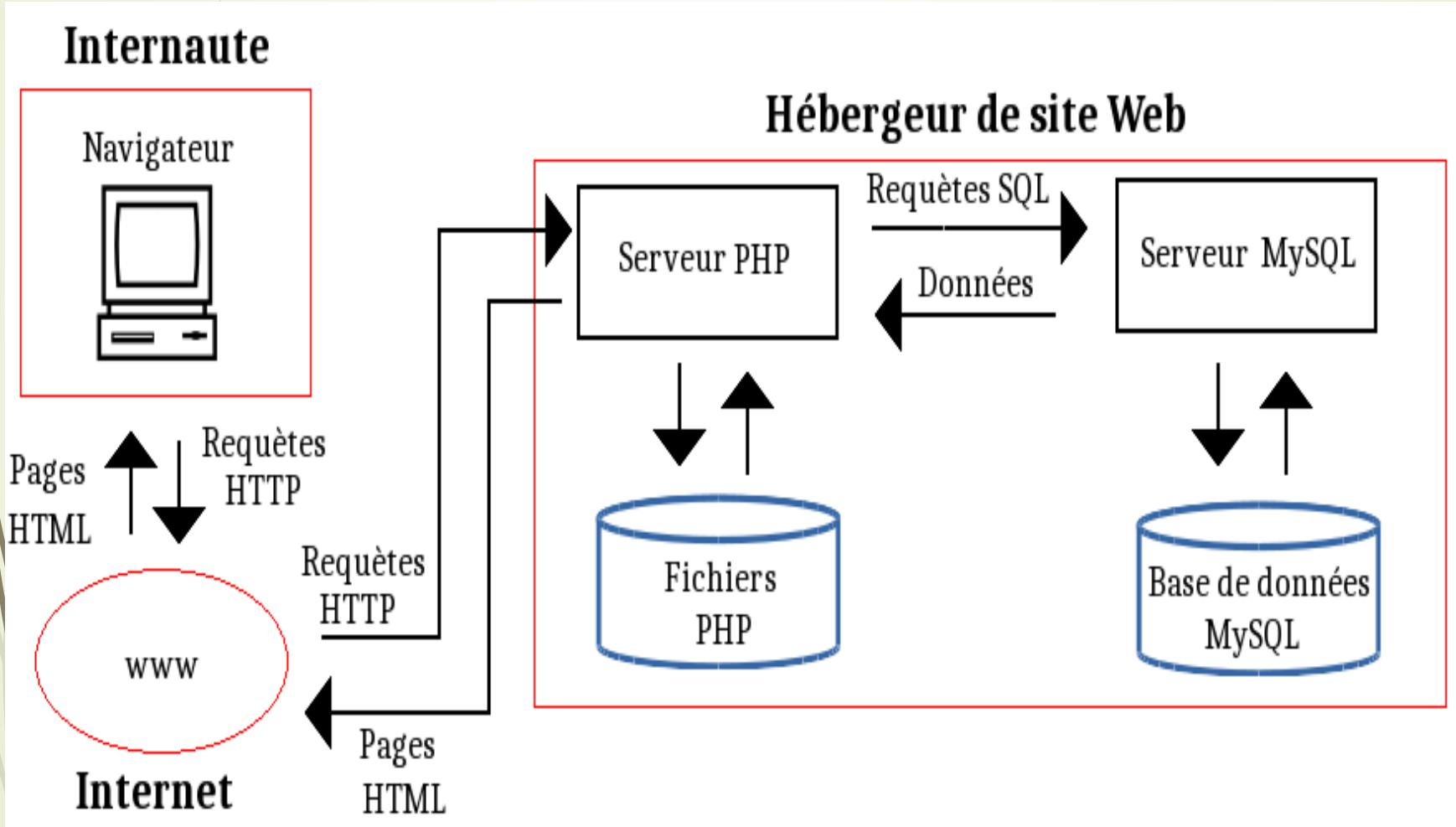
Unité 0 : Définitions et rôles du PHP et du MySQL

Pourquoi utiliser le couple PHP/MySQL ?

- La structure du langage PHP : Simple d'accès pour les débutants.
- PHP est un langage Open Source et donc gratuit.
- Le PHP se distingue par ses performances et sa solidité : Comme le langage est Open Source, n'importe qui peut contribuer à son évolution.
- Une totale compatibilité entre MySql et PHP.
- MySql utilise le langage SQL qui a une syntaxe standard.
- MySQL est à la fois simple d'utilisation, très robuste et offre d'excellente performances que cela soit pour une petite ou pour une grosse structure.

Unité 0 : Définitions et rôles du PHP et du MySQL

Notions client serveur



Unité 0 : Définitions et rôles du PHP et du MySQL

Environnement de travail

Comme on le sait maintenant, un serveur dispose de différents programmes lui permettant de lire et de comprendre certains langages informatiques que des ordinateurs « normaux » ne peuvent pas lire.

Pour pouvoir écrie nos codes PHP, nous allons avoir besoin d'un éditeur de code tout comme dans le cours HTLM et surtout, nous allons avoir besoin d'un serveur afin de pouvoir tester nos codes PHP et MySQL. La bonne nouvelle ici est qu'il existe des logiciels regroupant tous les programmes nécessaires pour faire cela.

Dans ce cours nous allons utiliser WAMP Sever, MAMP ou XAMPP en fonction de système d'exploitation selon qu'il est Windows, Mac Os ou Linux.

Unité 0 : Définitions et rôles du PHP et du MySQL

Où écrire le code PHP ?

Nous allons pouvoir écrire nos scripts PHP soit dans des fichiers dédiés, c'est-à-dire des fichiers qui ne vont contenir que du PHP, soit intégrer le PHP au sein de nos fichiers HTML.
Les fichiers qui contiennent du PHP vont devoir être enregistrés avec l'extension **.php**. Dans le cas où on intègre du code PHP dans un fichier HTML, il faudra également changer son extension en .php.

La balise PHP

Le serveur, pour être en mesure d'exécuter le code PHP, va devoir le reconnaitre. Pour lui indiquer qu'un script ou que telle partie d'un code est écrit en PHP, nous allons entourer ce code avec une balise PHP qui a la forme suivante : **<?php ?>**.
Lorsqu'on intègre du PHP dans du code HTML, on va pouvoir placer cette balise et du code PHP à n'importe quel endroit dans notre fichier. On va même pouvoir placer la balise PHP en dehors de notre élément html. De plus, on va pouvoir déclarer plusieurs balises PHP à différents endroits dans un fichier.

Unité 0 : Définitions et rôles du PHP et du MySQL

Exemple de code PHP?

Les structures de langage **echo** et **print** vont nous permettre d'afficher un résultat en PHP. Nous allons privilégier dans ce cours le mot clef **echo**

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <title>Cours PHP & MySQL</title>
5.          <meta charset="utf-8">
6.          <link rel="stylesheet" href="cours.css">
7.      </head>
8.
9.      <body>
10.         <h1>Titre principal</h1>
11.         <?php
12.             //Affiche "Hello World" avec un retour à la ligne
13.             echo 'Hello World <br>'; //Ceci est un commentaire
    PHP
14.
15.             /*Affiche
16.               "Bonjour le Monde
17.             */
18.             echo "Bonjour le Monde"; /*Ceci est un commentaire
    PHP*/
19.         ?>
20.         <p>Un paragraphe</p>
21.     </body>
22. </html>
```

Unité 1 : Les variables et types de données PHP

Une variable est un conteneur servant à stocker des informations de manière temporaire, comme une chaine de caractères (un texte) ou un nombre par exemple. En PHP, une variable ne va exister que durant le temps de l'exécution du script l'utilisant.

Les règles de déclaration des variables en PHP

il y a quelques règles à respecter et à connaitre lors de la déclaration d'une nouvelle variable :

- Toute variable en PHP doit commencer par le signe $ qui sera suivi du nom de la variable ;
- Le nom d'une variable doit obligatoirement commencer par une lettre ou un underscore (_) et ne doit pas commencer par un chiffre ;
- Le nom d'une variable ne doit contenir que des lettres, des chiffres et des underscores mais pas de caractères spéciaux ;
- Le nom d'une variable ne doit pas contenir d'espace.

Pour affecter une valeur à une variable, on utilise **l'opérateur =**

Unité 1 : Les variables et types de données PHP

Les règles de déclaration des variables en PHP

De plus, notez que le nom des variables est sensible à la casse en PHP. Cela signifie que l'usage de majuscules ou de minuscules va créer des variables différentes. Par exemple, les variables **$texte**, **$TEXTE** et **$tEXTe** vont être des variables différentes.

Déclarer une variable PHP

```
<!DOCTYPE html>
<html>
    <head>
        <title>Cours PHP & MySQL</title>
        <meta charset="utf-8">
        <meta name="viewport"
          content="width=device-width, initial-scale=1, user-scalable=no">
        <link rel="stylesheet" href="cours.css">
    </head>

    <body>
        <h1>Titre principal</h1>
        <?php
            $prenom = "Pierre";
            $age = 28;
        ?>
        <p>Un paragraphe</p>
    </body>
</html>
```

Unité 1 : Les variables et types de données PHP

Déclarer une variable PHP

Notez qu'il va falloir utiliser des guillemets ou des apostrophes pour stocker une chaine de caractères dans une variable. En revanche, nous n'en utiliserons pas pour assigner un nombre à une variable. Pour afficher le contenu d'une variable, nous allons utiliser une instruction **echo** et on peut changer la valeur d'une variable en faisant une nous affection.

```
1.  <!DOCTYPE html>
2.  <html>
3.      <head>
4.          <title>Cours PHP & MySQL</title>
5.          <meta charset="utf-8">
6.          <link rel="stylesheet" href="cours.css">
7.      </head>
8.
9.      <body>
10.         <h1>Titre principal</h1>
11.         <?php
12.             $prenom = "Pierre";
13.             $age = 28; //$age stocke le nombre 28
14.
15.             echo "La variable \$age contient : ";
16.             echo $age;
17.             echo "<br>";
18.
19.             $age = 29; //$age stocke le nombre 29
20.             echo "La variable \$age contient : ";
21.             echo $age;
22.         ?>
23.         <p>Un paragraphe</p>
24.     </body>
25. </html>
```

Unité 1 : Les variables et types de données PHP

Les types de données en PHP

Les variables PHP vont pouvoir stocker différents types de valeurs, comme du texte ou un nombre par exemple. Les variables en PHP vont pouvoir stocker 8 grands types de données différents :

- Le type « chaine de caractères » ou String en anglais ;
- Le type « nombre entier » ou Integer en anglais ;
- Le type « nombre décimal » ou Float en anglais ;
- Le type « booléen » ou Boolean en anglais ;
- Le type « tableau » ou Array en anglais ;
- Le type « objet » ou Object en anglais ;
- Le type « NULL » qui se dit également NULL en anglais ;
- Le type « ressource » ou Resource en anglais ;

En PHP, contrairement à d'autres langages de programmation, nous n'avons pas besoin de préciser à priori le type de valeur qu'une variable va pouvoir stocker. Le PHP va en effet automatiquement détecter quel est aleur stockée dans telle ou telle variable

Unité 1 : Les variables et types de données PHP

Le PHP va en effet automatiquement détecter quel est le type de la valeur stockée dans telle ou telle variable.

Opérateurs et concaténation

La documentation officielle de PHP classe les différents opérateurs qu'on va pouvoir utiliser selon les groupes suivants :

- Les opérateurs arithmétiques ;
- Les opérateurs d'affectation ;
- Opérateurs sur les bits ;
- Opérateurs de comparaison ;
- Opérateur de contrôle d'erreur ;
- Opérateur d'exécution ;
- Opérateurs d'incrémentation et décrémentation ;
- Les opérateurs logiques ;
- Opérateurs de chaînes ;

Unité 1 : Les variables et types de données PHP

Opérateurs et concaténation

- Opérateurs de tableaux ;
- Opérateurs de types ;

Dans ce cours, nous allons nous concentrer sur les opérateurs arithmétiques, les opérateurs de chaines et les opérateurs d'affectation.

Concaténer signifie littéralement « mettre bout à bout ». L'opérateur de concaténation qui est le **point** (**.**) va donc nous permettre de mettre bout à bout deux chaines de caractères. Cet opérateur va nous permettre d'afficher le contenu d'une variable concaténer avec du texte en utilisant **echo** . Par exemple:

$prenom = "Pierre";

$age = 28;

echo "Je m'appelle " .**$prenom**. " et j'ai " .**$age**. " ans
"; sera affiché comme suit : Je m'appelle **Pierre** et j'ai **28** ans

Unité 2 : Les Structures de contrôles PHP

Les Structures de contrôles PHP

Elles vont nous permettre d'exécuter une série d'instructions si une condition donnée est vérifiée ou (éventuellement) une autre série d'instructions si elle ne l'est pas.

Nous avons accès aux structures conditionnelles suivantes en PHP :

- La condition if (si) ;
- La condition if… else (si… sinon) ;
- La condition if… elseif… else (si… sinon si… sinon).

Les opérateurs logiques

Les opérateurs logiques vont être principalement utilisés avec les conditions puisqu'ils vont nous permettre d'écrire plusieurs comparaisons au sein d'une même condition ou encore d'inverser la valeur logique d'un test. En PHP, nous pouvons utiliser les opérateurs logiques suivants :

Unité 2 : Les Structures de contrôles PHP

Les Structures de contrôles PHP

Opérateur	Définition
AND	Renvoie true si toutes les comparaisons valent true
&&	Renvoie true si toutes les comparaisons valent true
OR	Renvoie true si une des comparaisons vaut true
||	Renvoie true si une des comparaisons vaut true
XOR	Renvoie true si une des comparaisons exactement vaut true
!	Renvoie true si la comparaison vaut false (et inversement)

Comme vous pouvez le constater, les opérateurs logiques « **ET** » et « **OU** » peuvent s'écrire de deux façons différentes : soit avec les mots clefs **AND** et **OR**, soit avec les signes **&&** et **| |**.
Ces deux écritures **ne sont pas tout à fait équivalentes** : la différence réside dans **l'ordre des priorités** de traitement des opérations. En effet, l'écriture avec des signes a une priorité plus importante que l'écriture avec des mots clefs.

Les Structures de contrôles PHP

Attention également à ne pas confondre les opérateurs logiques **OR** et **XOR** en PHP : si on utilise l'opérateur **OR**, le PHP va renvoyer true si au **moins une des comparaisons vaut true et renverra donc true si plusieurs comparaisons valent true** tandis qu'en utilisant l'opérateur **XOR** le PHP ne renverra **true que si une seule comparaison vaut true (et reverra false si plusieurs comparaisons valent true).**

Ordre de priorité des opérateurs

La tableau suivant liste les différents opérateurs vus jusqu'ici ainsi que les opérateurs d'incrémentation et de décrémentation et le ternaire qu'on étudiera plus tard selon leur priorité (la première ligne du tableau contient les opérateurs avec la plus grande priorité et etc. jusqu'à la dernière ligne contenant les opérateurs avec la plus petite priorité).

Unité 2 : Les Structures de contrôles PHP

Les Structures de contrôles PHP

Opérateurs
**
++ (incrémentation), -- (décrémentation)
!
*, /, %
+, -, .
<, <=, >, >=
==, ===, !=, !==, <>, <=>
&&
||
??
? : (ternaire)
=, +=, -=, *=, /=, %=, **=, .=
AND
XOR
OR

Unité 2 : Les Structures de contrôles PHP

Instruction switch

Elle va nous permettre d'exécuter un code en fonction de la valeur d'une variable. On va pouvoir gérer autant de situations ou de « cas » que l'on souhaite.

```
switch (expression) {
   case x:
       // execute case x code block
       break;
   case y:
       // execute case y code block
       break;
   default:
       // execute default code block
}
```

Exemple :

Unité 2 : Les Structures de contrôles PHP

Instruction switch

```
<?php
    $x = 2;
    switch($x){
        case 0:
            echo '$x stocke la valeur 0';
            break;
        case 1:
            echo '$x stocke la valeur 1';
            break;
        case 2:
            echo '$x stocke la valeur 2';
            break;
        default:
            echo '$x ne stocke pas de valeur entre 0 et 2';
    }
?>
```

Unité 2 : Les Structures de contrôles PHP

Les boucles

Les boucles vont nous permettre d'exécuter plusieurs fois un bloc de code, c'est-à-dire d'exécuter un code « en boucle » tant qu'une condition donnée est vérifiée.

Nous disposons de quatre boucles différentes en PHP :

- La boucle **while** (« tant que ») ;
- La boucle **do… while** (« faire… tant que ») ;
- La boucle **for** (« pour ») ;
- La boucle **foreach** (« pour chaque ») ;

Le fonctionnement général des boucles sera toujours le même : on pose une condition qui sera généralement liée à la valeur d'une variable et on exécute le code de la boucle « en boucle » tant que la condition est vérifiée.

Unité 2 : Les Structures de contrôles PHP

Les boucles

boucle While

La boucle while (« tant que » en français) va nous permettre de répéter une série d'instructions tant qu'une condition donnée est vraie c'est-à-dire tant que la condition de sortie n'est pas vérifiée. Sa syntaxe est :

```
while (condition) {
  instructions ;
}
```

boucle do….while

La boucle do… while (« faire… tant que ») est relativement semblable à la boucle while dans sa syntaxe. La grande différence entre les boucles while et do… while va résider dans l'ordre dans lequel vont se faire les opérations.

```
do{
Instructions;
} while(b < 10);
```

Unité 2 : Les Structures de contrôles PHP

Les boucles

Boucle for

La boucle for (« pour » en français) est structurellement différente des boucles while et do… while puisqu'on va cette fois-ci initialiser notre variable à l'intérieur de la boucle.
La boucle for utilise une syntaxe relativement condensée et est relativement puissante ce qui en fait la condition la plus utilisée en PHP. Sa syntaxe est :

```
<?php
for($x = 0; $x <= 5; $x++){
 instructions;
}
?>
```

- **Boucle foreach :** La boucle PHP **foreach** est un peu particulière puisqu'elle a été créée pour fonctionner avec des variables tableaux. Nous étudierons la cette fonction quand nous verrons les tableaux PHP.

Unité 2 : Les Structures de contrôles PHP

include et require

Les instructions PHP **include** et **require** vont nous permettre toutes deux d'inclure des fichiers de code (ou plus exactement le contenu de ces fichiers) à l'intérieur d'autres fichiers de code.

si l'inclusion a été tentée avec **include**, le PHP renverra **un simple avertissement et le reste du script s'exécutera quand même** tandis que si la même chose se produit avec **require**, **une erreur fatale sera retournée par PHP et l'exécution du script s'arrêtera immédiatement**.

```
<?php
        echo '<h2>Menu inclus avec include</h2> <br>';
        include 'menu.php';
        echo '<h2>Menu inclus avec require</h2> <br>';
        require 'menu.php';
?>
```

Unité 2 : Les Structures de contrôles PHP

Exercice 1

```
<?php
    $x = 4; //On affecte la valeur 4 à $x
    $y = -12; //On affecte la valeur -12 à $y
    if($x >= 0 AND $x <= 5){
        echo '$x contient une valeur entre 0 et 5 <br>';
    }
    if($x >= 0 AND $y >= 0){
        echo '$x et $y contiennent une valeur positive <br>';
    }
    if($x >= 0 OR $y >= 0){
        echo '$x ou $y (ou les deux) contient une valeur positive <br>';
    }
     if($x >= 0 XOR $y >= 0){
        echo '$x ou $y contient une valeur positive mais pas les deux';
    }
?>
```

Unité 2 : Les Structures de contrôles PHP

Exercice 2

1- Ecrire un programme en PHP qui affiche la somme, la multiplication et la division de deux variables a et **b**.

2- Ensuite si **a** est supérieur à **b**, afficher le modulo de **a** par **b**.

3- Affecter à la variable **a** la valeur 5. En utilisant l'instruction switch, afficher que la variable a ne stocke pas une valeur compris entre 0 et 3.

4- Ecrire un programme qui affiche la table de multiplication par 5. Multiplication allant de zéro à 10.

Unité 3 : Les Fonctions en PHP

Les Fonctions en PHP

Une fonction correspond à un bloc de code nommé et réutilisable et dont le but est d'effectuer une tâche précise. En PHP tout comme dans plusieurs langages, on a des **fonctions internes** au langage et des **fonctions personnalisés** (définies par l'utilisateur). Intéressons nous au dernier type :

Fonctions personnalisés

Pour déclarer une fonction, il faut déjà commencer par préciser le mot clef **function** qui indique au PHP qu'on va définir une fonction personnalisée. Ensuite, nous allons devoir préciser le nom de notre fonction (sauf dans le cas des fonctions anonymes que nous étudierons plus tard). Le nom des fonctions va suivre les mêmes règles que celui des variables

Unité 3 : Les Fonctions en PHP

Les Fonctions en PHP

Notez qu'à la différence des variables le nom des **fonctions est insensible à la casse.** Après le nom de notre fonction, nous devons obligatoirement mentionner un couple de parenthèses puis ensuite placer les instructions que la fonction devra exécuter lors de son appel entre des crochets. On pourra ajouter des paramètres à la fonction.

Nous allons pouvoir utiliser la structure de contrôle **return** qui va nous permettre de demander à une fonction de retourner un résultat qu'on va ensuite pouvoir stocker dans une variable ou autre pour le manipuler.

Portée des variables

En PHP, nous pouvons déclarer des variables n'importe où dans notre script : au début du script, à l'intérieur de boucles, au sein de nos fonctions, etc. Toute variable définie en dehors d'une fonction a une portée globale. Par définition, une variable qui a une portée globale est accessible « globalement », c'est-à-dire dans tout le script sauf dans les espaces locaux d'un script.

Unité 3 : Les Fonctions en PHP

Portée des variables

Au contraire, toute variable définie à l'intérieur d'une fonction va avoir une portée locale à la fonction. Cela signifie que la variable ne sera accessible qu'au sein de la fonction et notre variable sera par ailleurs par défaut détruite dès la fin de l'exécution de la fonction.

Parfois, nous voudrons nous servir de variables possédant une portée globale (c'est-à-dire définies en dehors d'une fonction) à l'intérieur d'une fonction.

Pour cela, on va pouvoir utiliser **le mot clef global** avant la déclaration des variables qu'on souhaite utiliser dans notre fonction. Cela va nous permettre d'indiquer que les variables déclarées dans la fonction sont en fait nos variables globales. Pour être tout à fait précis, on dit que les variables globales sont importées dans le contexte local par référence.

Une variable définie localement va être supprimée ou détruite dès la fin de l'exécution de la fonction dans laquelle elle a été définie.

Unité 3 : Les Fonctions en PHP

Portée des variables

Pour qu'une fonction de « souvienne » de la dernière valeur d'une variable définie dans la fonction, nous allons pouvoir utiliser le **mot clef static** devant la déclaration initiale de la variable.

La portée de la variable sera toujours statique, mais la variable ne sera pas détruite lors de la fin de l'exécution de la fonction mais plutôt conservée pour pouvoir être réutilisée lors d'une prochaine exécution.

Notez par ailleurs que lorsque nous initialisons une variable en utilisant static, la variable ne sera initialisée que lors du premier appel de la fonction (si ce n'était pas le cas, le mot clef static n'aurait pas grand intérêt).

Les constantes PHP

Une constante est un identifiant ou un nom qui représente une valeur simple. Les constantes, tout comme les variables, vont donc être des conteneurs qui vont nous servir à stocker une valeur.

Unité 3 : Les Fonctions en PHP

Les constantes PHP

Pour définir une constante en PHP, nous allons pouvoir utiliser la fonction **define()** ou le mot clef **const**.

Exemple :

```
<?php
// Définir une constante (dont le nom est par défaut sensible à la casse).
define('SITEWEB', 'OPENTUTO');
// Afficher la valeur de SITEWEB.
echo 'Mon site web préféré est = ',SITEWEB,'<br />';
// Utilisation du mot-clé const (depuis la version 5.3)
const COURS = 'PHP 5.5';
echo 'Mon cours préféré est = ', COURS;
?>
```

Unité 3 : Les Fonctions en PHP

Les Fonctions en PHP

Exemple :

```
<?php
        $prenom = 'Pierre';
        $x = 4;
        $y = 5;
        function bonjour($p){
            echo 'Bonjour ' .$p. '<br>';
        }
    function addition($p1, $p2){
            echo $p1. ' + ' .$p2. ' = ' .($p1 + $p2). '<br>';
        }
    bonjour($prenom);
        bonjour('Mathilde');
        addition($x, $y);
        addition(1, 1);
?>
```

Unité 3 : Les Fonctions en PHP

Exercice

Reprendre l'exercice 2 de l'unité 2 en utilisant des fonction pour la résolution.

Unité 4 : Les tableaux

Les tableaux

Les tableaux en PHP sont des variables spéciales qui peuvent stocker plusieurs valeurs en même temps. Dans un tableau, chaque valeur va être associée à une clef unique. Cette clef va nous permettre notamment de récupérer la valeur associée. Nous allons pouvoir définir les différentes clefs ou laisser le PHP générer automatiquement les clefs pour les différentes valeurs d'un tableau.

On va pouvoir créer trois types de tableaux différents en PHP :

- Des tableaux numérotés ou indexés (les clefs vont être des nombres) ;
- Des tableaux associatifs (nous allons définir la valeur que l'on souhaite pour chaque clef) ;
- Des tableaux multidimensionnels (tableaux qui stockent d'autres tableaux en valeur).

Nous allons nous intéresser au premier type : tableaux numérotés ou indexés. Pour créer un tableau, on peut soit utiliser la structure de langage **array()**, soit le nouvelle syntaxe plus courte **[]**.

Unité 4 : Les tableaux

Les tableaux numérotés ou indexés

Les tableaux numérotés sont le type de tableaux le plus simple à créer en PHP puisque les clefs vont être générées automatiquement par le PHP.

Pour créer un tableau numéroté en PHP, il suffit en fait d'indiquer une série de valeurs et le PHP associera automatiquement une clef unique à chaque valeur, en commençant avec la clef 0 pour la première valeur, la clef 1 pour la deuxième valeur, la clef 2 pour la troisième valeur, etc.

On peut tout à fait créer des tableaux qui vont stocker des chaines de caractères, des nombres, des booléens, etc. Comme d'habitude, seules les chaines de caractères doivent être entourées d'apostrophes ou de guillemets droits.

```
<?php
        $prenoms = array('Mathilde', 'Pierre', 'Amandine', 'Florian');
        $ages = [27, 29, 21, 29];
?>
```

Unité 4 : Les tableaux

Les tableaux numérotés ou indexés

Ici, on crée deux variables tableau **$prenoms** et **$ages**.

PHP va automatiquement créer les clefs qu'on appelle également indices ou index et les associer aux différentes valeurs. La valeur Mathilde de notre premier tableau va avoir la clef 0, la valeur Pierre la clef 1, etc. De même, la clef 0 va être associée à la valeur 27 de notre deuxième tableau, la deuxième valeur (le premier 29) va être associée à la clef 1 et etc.

On va également pouvoir créer un tableau indice par indice et valeur par valeur en précisant ici les indices à associer à chaque valeur. Dans ce cas-là, il faudra utiliser la syntaxe suivante :

Unité 4 : Les tableaux

Les tableaux numérotés ou indexés

```
<?php
        $prenoms[0] = 'Mathilde';
        $prenoms[1] = 'Pierre';
        $prenoms[2] = 'Amandine';
        $prenoms[3] = 'Florian';

        $ages[0] = 27;
        $ages[1] = 29;
        $ages[2] = 21;
        $ages[3] = 29;
 ?>
```

Unité 4 : Les tableaux

Afficher les valeurs d'un tableau numéroté

Pour afficher les valeurs d'un tableau numéroté une à une, il suffit d'echo notre variable tableau en précisant l'indice (entre crochets) correspondant à la valeur que l'on souhaite afficher.

```
<?php
        $prenoms = ['Mathilde', 'Pierre', 'Amandine', 'Florian'];
        echo $prenoms[0]. '<br>';
        echo $prenoms[2];
?>
```

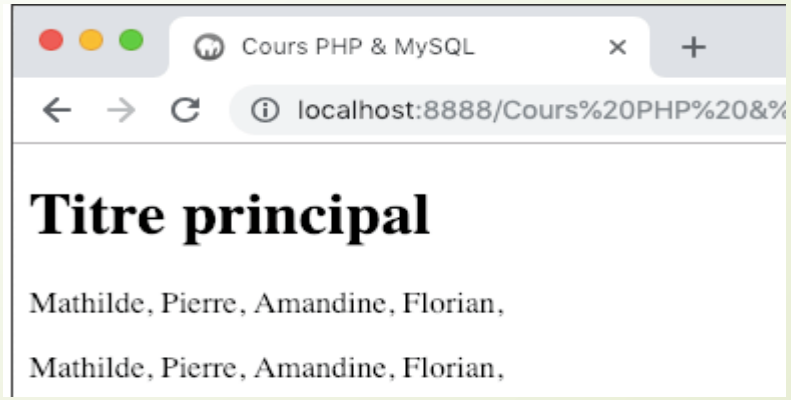
Pour afficher toutes les valeurs d'un tableau numéroté d'un coup, nous allons cette fois-ci devoir utiliser une **boucle for**.Il faudra dans ce cas connaitre la taille du tableau avec la fonction **count ().**
Une fois qu'on a déterminé la taille de notre tableau, il va être très simple de créer notre boucle for.

Exemple :

Unité 4 : Les tableaux

Afficher les valeurs d'un tableau numéroté

```php
<?php
    $prenoms = ['Mathilde', 'Pierre', 'Amandine', 'Florian'];
  //On récupère la taille du tableau et on la stocke dans $taille
    $taille = count($prenoms);
    //On peut soit parcourir le tableau et afficher les valeurs une à une
    for($i = 0; $i < $taille; $i++){
        echo $prenoms[$i]. ', ';
    }
    echo '<br><br>';
    //...soit les stocker dans une autre variable et echo cette variable
    for($i = 0; $i < $taille; $i++){
        $p .= $prenoms[$i]. ', ';
    }
    echo $p;
?>
```



Unité 4 : Les tableaux

La boucle foreach

La boucle **foreach**, au contraire, va nous permettre de parcourir des tableaux élément par élément. A chaque nouveau passage dans la boucle, la valeur d'un élément du tableau va être placée dans la variable qu'on a ici appelé **$valeurs** et la boucle va aller chercher la valeur suivante jusqu'à arriver à la fin du tableau.

Syntaxe :

```
foreach (array_expression as $value){
    //instructions
}
```

Unité 5 : Introduction aux bases de données, au SQL et à MySQL

Qu'est-ce qu'une base de données ?

Une base de données est un conteneur qui va servir à stocker toutes sortes de données : des dates, chiffres, mots, etc. de façon organisée et sans date d'expiration.

Pourquoi utiliser les bases de données ?

Pourquoi créer des bases de données et stocker des données dedans plutôt que simplement utiliser un fichier quelconque ?

Les bases de données possèdent deux grands avantages par rapport aux autres méthodes de stockage :

- Nous allons pouvoir stocker de très grandes quantités de données ;
- Nous allons pouvoir récupérer certaines données en particulier simplement et rapidement.

Qu'est-ce que le SQL ?

SQL est l'abréviation de Structured Query Language ou langage de requêtes structuré en français.

Unité 5 : Introduction aux bases de données, au SQL et à MySQL

Qu'est-ce que le SQL ?

SQL est l'abréviation de *Structured Query Language* ou langage de requêtes structuré en français.

- Le SQL est le langage principal utilisé pour accéder aux bases de données et les manipuler. Nous allons utiliser ce langage pour exécuter toutes sortes de requêtes dans une base de données : récupérer des données, les mettre à jour, en insérer de nouvelles ou même créer de nouvelles bases de données.
- Le SQL est un langage à part entière : il possède sa propre syntaxe que nous allons découvrir dans les chapitres suivants.

Qu'est-ce que le MySQL ?

Le MySQL est ce qu'on appelle un **système de gestion de bases de données**. (**SGBD**). De manière très schématique, c'est un programme qui va nous permettre de manipuler simplement nos bases de données.
En effet, les bases de données sont des systèmes très complexes. Nous utilisons un système de gestion de bases de données pour cacher cette complexité et effectuer simplement les opérations dont nous avons besoin sur nos bases de données.

Unité 5 : Introduction aux bases de données, au SQL et à MySQL

Qu'est-ce que le MySQL ?

Le MySQL est loin d'être le seul système de gestion de bases de données ; il en existe bien d'autres. Parmi les plus connus, on peut ici notamment citer SQL Server, MS Access ou encore Oracle. Chacun de ces systèmes de gestion de bases de données fonctionne de manière similaire (ils permettent d'envoyer des instructions SQL) et propose des fonctionnalités relativement équivalentes.

Nous allons donc pouvoir utiliser le MySQL en PHP pour passer des ordres à nos bases de données : le MySQL va nous servir à envoyer nos requêtes écrites en SQL standard à nos bases de données.

Qu'est-ce que phpMyAdmin ?

Nous allons avoir deux moyens d'interagir avec nos bases de données MySQL : soit en envoyant nos requêtes à partir de nos fichiers de code PHP, soit directement via l'interface phpMyAdmin.

phpMyAdmin est un logiciel gratuit code en PHP qui sert à gérer directement nos bases de données MySQL.

Unité 5 : Introduction aux bases de données, au SQL et à MySQL

Qu'est-ce que phpMyAdmin ?

Dans phpMyAdmin, nous allons par exemple pouvoir directement créer une nouvelle base de données ou envoyer toutes sortes de requêtes SQL à nos bases de données.

phpMyAdmin est un logiciel qui nous permet donc d'accéder directement aux données de nos bases de données et à gérer nos bases de données.

Notez que phpMyAdmin est embarqué et proposé sur tous les serveurs utilisant les bases de données MySQL. Vous allez également pouvoir y accéder en local et ceci que vous utilisiez Wamp, Mamp ou Xamp.

Structure d'une base de données

Une **base de données** est généralement constituée de **tables**. Une table est une collection cohérente de données. Par exemple, dans le cas d'un site e-commerce, vous aurez certainement une table « Clients », une autre table « Commandes », etc.

Unité 5 : Introduction aux bases de données, au SQL et à MySQL

Structure d'une base de données

On représente habituellement une table sous forme de tableau. Une table va ainsi être constituée de lignes qu'on appelle également entrées et de colonnes.

L'intersection entre une ligne et une colonne est ce qu'on appelle un champ. Un champ est l'équivalent d'une cellule dans un tableau et va contenir une donnée particulière (un ID, le nom d'un utilisateur, un numéro de téléphone, etc.).

Dans une table, chacun des champs d'une même ligne ou entrée va généralement être relatif à un même sujet. Si votre base de données possède une table « Clients » par exemple, la première ligne va regrouper des informations relatives à un client en particulier.

En colonne, nous allons trouver des informations de même type. Une table « Clients » par exemple pourra contenir des colonnes comme « Id du client », « nom du client », « adresse mail », « numéro de téléphone », etc.

Exemple

Unité 5 : Introduction aux bases de données, au SQL et à MySQL

Structure d'une base de données

Exemple : Table Clients

IdClient	NomClient	Adresse	Ville	CodePostal	Pays	Mail
1	Pierre Giraud	30 avenue des Acacias	Toulon	83000	France	pierre.giraud@edhec.com
2	Victor Durand	50 boulevard Jean Jaurès	Lille	59000	France	victor.durand@gmail.com
3	Julia Palaz	113 avenue de Versailles	Paris	75016	France	ju.palaz@gmail.com
4	Chloé Joly	28 rue Sainte Catherine	Bordeaux	33000	France	cjoly@outlook.fr
5	Florian Buisson	88 allée des sportifs	Lyon	69002	France	florian.b@gmail.com

Notre table « Clients » possède 7 colonnes et 5 entrées.

Unité 6 : Manipuler des données dans des bases MySQL

Créer une base de données

Pour créer une base de données, il faut dans un premier temps disposer d'un nom d'utilisateur ayant les droits de création. Ensuite utiliser la commande SQL suivante pour créer la base de données :

CREATE DATABASE nom_base;

Ex : **CREATE DATABASE masterimsd;**

Création d'une table

Une base de données est constituée de tables. Les tables sont les «casiers » dans lesquelles nous allons stocker nos données. Mais avant de pouvoir stocker des données, il va déjà falloir apprendre à créer des tables dans notre base de données !

Pour créer une nouvelle table dans une base de données, nous allons utiliser la requête SQL **CREATE TABLE** suivie du nom que l'on souhaite donner à notre table et nous allons également pouvoir préciser entre parenthèse le nom des colonnes de notre table ainsi que le type de données qui doit être stocké dans chaque colonne.

Unité 6 : Manipuler des données dans des bases MySQL

Création d'une table

Le MySQL nous offre beaucoup de choix de **types de données** différent nous permettant de créer des tables de manière vraiment précise. Il existe plusieurs types de données en MySQL dont les principaux sont: les données de **type texte**, les données de **type nombre**, les données de **type date**. Les sous types de valeurs les plus courants et les plus utilisés sont :

- **INT** : accepte un nombre entier de 4 octets. La fourchette pour les entiers relatifs est [-2 147 483 648, 2 147 483 647], celle pour les entiers positifs est [0, 4 294 967 295] ;
- **VARCHAR** : accepte une chaine de longueur variable (entre 0 et 65 535 caractères). La longueur effective réelle de la chaine dépend de la taille maximum d'une ligne ;
- **DATE** : accepte une date se situant entre le 1er janvier de l'an 1000 et le 31 décembre de l'an 9999.

En plus de cela, nous allons également pouvoir spécifier des attributs ou contraintes pour chacune des colonnes de notre table. Ces attributs ou contraintes vont venir apporter des contraintes supplémentaires sur les données attendues (non nulle, etc.) ou vont définir des comportements.

Unité 6 : Manipuler des données dans des bases MySQL

Création d'une table

Voici les attributs qu'on va pouvoir ajouter à nos colonnes durant la création de notre table :

- NOT NULL – Signifie que chaque entrée doit contenir une valeur pour cette colonne. La valeur null n'est pas acceptée ;
- UNIQUE – Chacune des valeurs dans la colonne doit être unique (est utile par exemple lorsqu'on reçoit des adresses mail, cela évite qu'un utilisateur s'inscrive deux fois sur notre site entre autres) ;
- PRIMARY KEY – Est utilisé pour identifier de manière unique chaque nouvelle entrée dans une table. C'est une combinaison de NOT NULL et de UNIQUE. PRIMARY KEY ne doit s'appliquer qu'à une colonne dans une table mais chaque table doit obligatoirement posséder une colonne avec une PRIMARY KEY. La colonne avec PRIMARY KEY est souvent une colonne d'ID (nombres) qui s'auto-incrémentent ;
- FOREIGN KEY – Utilisée pour empêcher des actions qui pourraient détruire les liens entre des tables. La FOREIGN KEY sert à identifier une colonne qui est identique à une colonne portant une PRIMARY KEY dans une autre table ;

Unité 6 : Manipuler des données dans des bases MySQL

Création d'une table

- CHECK – Sert à s'assurer que toutes les valeurs dans une colonne satisfont à une certaine condition ou se trouve dans un certain intervalle spécifié ;
- DEFAULT value – Sert à définir une valeur par défaut qui va être renseignée si aucune valeur n'est fournie ;
- AUTO_INCREMENT – MySQL va automatiquement incrémenter (c'est-à-dire ajouter 1) au champ pour chaque nouvelle entrée ;
- UNSIGNED – Utilisé pour les données de type nombre, cette contrainte permet de limiter les données reçues aux nombres positifs (0 inclus).

La syntaxe de création d'une table est la suivante :

```
CREATE TABLE nom_de_la_table
(
   colonne1 type_donnees,
   colonne2 type_donnees,
   colonne3 type_donnees,
   colonne4 type_donnees
);
```

Unité 6 : Manipuler des données dans des bases MySQL

Exemple de création d'une table

```
CREATE TABLE utilisateur
(
   id INT PRIMARY KEY NOT NULL,
   nom VARCHAR(100),
   prenom VARCHAR(100),
   email VARCHAR(255),
   date_naissance DATE,
   pays VARCHAR(255),
   ville VARCHAR(255),
   code_postal VARCHAR(5),
   nombre_achat INT
);
```

Une fois notre base de données et nos premières tables créées, nous allons pouvoir commencer à insérer des données dans ces dernières.

Unité 6 : Manipuler des données dans des bases MySQL

Insérer des données dans une table

Pour insérer des données dans une table, nous allons cette fois-ci utiliser l'instruction SQL **INSERT INTO** suivie du nom de la table dans laquelle on souhaite insérer une nouvelle entrée avec sa structure puis le mot clef **VALUES** avec les différentes valeurs à insérer. La structure de la requête est la suivante :

INSERT INTO **nom_de_table** (nom_colonne1, nom_colonne2, nom_colonne3, …)

VALUES (valeur1, valeur2, valeur3, …);

Il y a cependant quelques règles de syntaxe à respecter afin que cette requête fonctionne :

- Les valeurs de type chaine de caractère (String) doivent être placées entre apostrophes ;
- La valeur NULL ne doit pas être placée entre apostrophes ;
- Les valeurs de type numérique ne doivent pas être placées entre apostrophes.

Unité 6 : Manipuler des données dans des bases MySQL

Modifier les données d'une table MySQL

Nous allons utiliser l'instruction SQL **UPDATE** suivie du nom de la table pour mettre à jour des données dans une table.

Cette instruction va toujours être accompagnée de **SET** qui va nous servir à préciser la colonne à mettre à jour ainsi que la nouvelle valeur pour la colonne.

En s'arrêtant là, en effet, nous allons mettre à jour toutes les valeurs d'une colonne d'un coup ! Ce sera très rarement ce que nous voudrons faire en pratique, et c'est pour cela que nous allons généralement également utiliser **la clause WHERE** pour spécifier quelles entrées doivent être mises à jour. La syntaxe basique est :

```
UPDATE table
SET nom_colonne_1 = 'nouvelle valeur'
WHERE condition ;
```

Unité 6 : Manipuler des données dans des bases MySQL

Modifier la structure d'une table

Pour modifier la structure d'une table en soi, nous allons utiliser l'instruction SQL **ALTER TABLE**.
Cette commande va nous permettre d'ajouter, de supprimer ou de modifier une colonne dans une table.

Ajouter une colonne dans une table

Pour ajouter une colonne, nous allons également devoir utiliser **ADD** avec le **nom de la colonne** à ajouter et le **type de données** attendu.

```
ALTER TABLE Utilisateurs
ADD DateInscription TIMESTAMP
```

Supprimer une colonne dans une table

Pour maintenant supprimer une colonne dans une table, nous allons cette fois-ci utiliser **ALTER TABLE** de concert avec l'instruction **DROP COLUMN**.

```
ALTER TABLE Utilisateurs
DROP COLUMN DateInscription TIMESTAMP
```

Unité 6 : Manipuler des données dans des bases MySQL

Modifier une colonne dans une table

Pour finalement modifier le type de donnée d'une colonne dans une table, il faudra utiliser **ALTER TABLE** avec l'instruction **MODIFY COLUMN** si vous évoluez dans un environnement MySQL (la syntaxe de cette commande n'est pas encore standardisée et peut changer selon le système de bases de données utilisé).

```
ALTER TABLE Utilisateurs
MODIFY COLUMN prenom VARCHAR(50)
```

Renommer une colonne dans une table

```
ALTER TABLE nom_table
CHANGE colonne_ancien_nom colonne_nouveau_nom type_donnees
```

Supprimer des données d'une table

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

Unité 6 : Manipuler des données dans des bases MySQL

Supprimer des données d'une table

Pour préciser quelles entrées doivent être supprimées, nous allons accompagner **DELETE FROM** d'une clause **WHERE** nous permettant de cibler des données en particulier dans notre table.

DELETE FROM Utilisateurs **WHERE** nom='OUEDRAOGO';

Supprimer une table

Pour supprimer complètement une table, nous allons cette fois-ci utiliser l'instruction SQL **DROP TABLE** suivie du nom de la table que l'on souhaite supprimer.

DROP TABLE Utilisateurs;

Supprimer une base de données

Pour supprimer une base de données, nous utiliserons l'instruction SQL **DROP DATABASE** suivie du nom de la base de données que l'on souhaite supprimer.

DROP DATABASE masterimsd;

Unité 6 : Manipuler des données dans des bases MySQL

La sélection de données dans une base de données

Pour sélectionner des données dans une base de données, nous allons utiliser l'instruction SQL **SELECT… FROM**

SELECT nom_du_champ FROM nom_du_table;

Pour sélectionner tous les champs de la table on utilisera le caractère *****

SELECT ***** FROM Utilisateurs;

Plusieurs types de sélection peuvent être faits dans une table. Nous verrons plus de détails dans le cours de base de données.

Unité 6 : Manipuler des données dans des bases MySQL

Prise en main phpMyAdmin

phpMyAdmin (PMA) est une application Web de gestion pour les systèmes de gestion de base de données MySQL réalisée principalement en PHP.

il s'agit de l'une des plus célèbres interfaces pour gérer une base de données MySQL sur un serveur PHP.

Cette interface pratique permet d'exécuter, très facilement et sans grandes connaissances en bases de données, des requêtes comme les créations de table de données, insertions, mises à jour, suppressions et modifications de structure de la base de données, ainsi que l'attribution et la révocation de droits et l'import/export. Ce système permet de sauvegarder commodément une base de données sous forme de fichier .sql et d'y transférer ses données, même sans connaître SQL.

TP phpMyAdmin

Unité 7 : Gestion des formulaires HTML avec PHP

Gestion des formulaires HTML avec PHP

Nous avons vu que les formulaires HTML vont nous permettre de recueillir des données envoyées par nos utilisateurs. Nous allons donc pouvoir stocker ces données dans une table de notre base de données puis les manipuler.

Rappel :

Pour créer un formulaire, nous allons utiliser l'élément HTML **form**. Cet élément **form** va avoir besoin de deux attributs pour fonctionner normalement : les attributs **method** et **action**.
L'attribut **method** va indiquer comment doivent être envoyées les données saisies par l'utilisateur. Cet attribut peut prendre deux valeurs : **get** et **post**.

Le HTML nous permet de créer nos formulaires. Pour récupérer et manipuler les données envoyées, cependant, nous allons devoir utiliser du PHP.

Unité 7 : Gestion des formulaires HTML avec PHP

TP1 (Affichage des données du formulaire)

Créons un formulaire **formulaire.html** avec les attributs suivants : **method**="**post**" et **action**="**formulaire.php**". on va pouvoir très simplement afficher les données reçues à l'utilisateur. Pour cela, nous allons **echo** les valeurs contenues dans **$_POST** via notre page d'action **formulaire.php**

TP2 (Redirection de l'utilisateur)

En pratique, cependant, nous n'allons pas créer des formulaires pour afficher les données aux utilisateurs mais bien pour utiliser les données de notre côté. Généralement, donc, l'utilisateur ne verra pas la page de traitement des données et nous le redirigerons plutôt immédiatement vers une page pertinente.

Pour renvoyer un utilisateur vers une autre page, on peut utiliser la fonction PHP **header()** à laquelle on va passer la page où l'utilisateur doit être renvoyé sous la forme **Location : adresse de ma page**.

Pour illustrer cela, on va créer deux nouvelles pages **formulaire2.php** et **form-merci.html**. Notre page **formulaire2.php** va être notre nouvelle page d'action, pensez donc bien à modifier la valeur de l'attribut action.

Unité 7 : Gestion des formulaires HTML avec PHP

Se connecter à une base de données MySql avec PHP

Pour pouvoir manipuler nos bases de données MySQL en PHP , nous allons déjà devoir nous connecter à MySQL.

Pour cela, le PHP met à notre disposition deux API (Application Programming Interface) :

- L'extension MySQLi ;
- L'extension PDO (PHP Data Objects).

l'extension MySQLi ne va fonctionner qu'avec les bases de données MySQL. L'extension PDO peut être utilisé avec plusieurs SGBD différents.

Utilisons l'extension PDO.

Connexion au serveur avec PDO

Unité 7 : Gestion des formulaires HTML avec PHP

Connexion au serveur avec PDO

```
<?php
        $servername = 'localhost';
        $username = 'root';
        $password = 'root';
       //On essaie de se connecter
        try{
            $conn = new PDO("mysql:host=$servername;dbname=bddtest", $username, $password);
            //On définit le mode d'erreur de PDO sur Exception
            $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
            echo 'Connexion réussie';
        }
       /*On capture les exceptions si une exception est lancée et on affiche
         *les informations relatives à celle-ci*/
        catch(PDOException $e){
          echo "Erreur : " . $e->getMessage();
        }
                    // on ferme la connexion
                    $conn=null;
```

Unité 7 : Gestion des formulaires HTML avec PHP

Insérer des données avec PDO

```
<?php
        $servname = 'localhost';
        $dbname = 'marsterimsd';
        $user = 'root';
        $pass = 'root';
        try{
            $dbco = new
PDO("mysql:host=$servname;dbname=$dbname", $user, $pass);
            $dbco->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
            $dbco->beginTransaction();
            $sql1 = "INSERT INTO
Clients(Nom,Prenom,Adresse,Ville,Codepostal,Pays,Mail)
                VALUES('Doe','John','Rue des Lys','Nantes',
44000,'France','j.doe@gmail.com')";
```

Unité 7 : Gestion des formulaires HTML avec PHP

Insérer des données avec PDO

```
            $dbco->exec($sql1);
        $sql2 = "INSERT INTO
Clients(Nom,Prenom,Adresse,Ville,Codepostal,Pays,Mail)
                    VALUES('Dupont','Jean','Bvd Original','Bordeaux',
33000,'France','jd@gmail.com')";
            $dbco->exec($sql2);
            $dbco->commit();
            echo 'Entrées ajoutées dans la table';
        }
        catch(PDOException $e){
            $dbco->rollBack();
          echo "Erreur : " . $e->getMessage();
        }
?>
```

Unité 7 : Gestion des formulaires HTML avec PHP

Insérer des données avec PDO

La méthode **beginTransaction()** permet de démarrer ce qu'on appelle une transaction. Concrètement, cela signifie que toutes les manipulations faites sur la base de données ne seront pas appliquées tant qu'on ne mettra pas fin à la transaction en appelant **commit().**

La méthode **commit()** sert donc à valider une transaction, c'est-à-dire à valider l'application d'une ou d'un ensemble de requêtes SQL.

La méthode **rollBack()** sert à annuler une transaction si l'on s'aperçoit d'une erreur.

Unité 7 : Gestion des formulaires HTML avec PHP

Afficher des données avec PDO

Si le temps permet sinon voir en tp

Unité 7 : Gestion des formulaires HTML avec PHP

TP3 (Manipulation et stockage des données)

Dans ce TP, nous allons :

1- Créer notre base de données

2- Créer notre table

3- Dans notre formulaire2.php, nous allons nous connecter sur à notre base de données puis y insérer les données du formulaire.

4- Dans une autre page php, nous allons afficher le contenu de notre table

Conclusion