Module 18 :
Développement WEB -2-

PHP

Accès à une base de donnée MySQL

Benkhayat Yassine

yasben11@gmail.com

L’accès à une base MySQL

L’accès à une base MySQL et son utilisation, qu’il s’agisse d’insérer, de modifier ou de lire des données, suit les étapes ci-dessous :

1. Connexion au serveur MySQL.
2. Envoi de diverses requêtes SQL au serveur (insertion, lecture, suppression ou mise à jour des données).
3. Récupération du résultat d’une requête.
4. Fermeture de la connexion au serveur.

Connexion au serveur MySQL

❖Avant toute chose, le script doit permettre de se connecter au serveur MySQL. La fonction essentielle de ce script est mysql_connect() , dont la syntaxe est la suivante :

```
mysql_connect ($host , $user , $pass)
```

❑**$host** est une chaîne contenant le nom du serveur

❑**$user** est le nom sous lequel l'utilisateur est autorisé à accéder au serveur

❑**$pass** est le mot de passe associé à l'utilisateur

Exemple : $id_con=mysql_connect("localhost " , "root", "")

L'exemple montre que la fonction mysql_connect() retourne un identifiant de connexion ($id_con dans l'exemple) sont on aura besoin par la suite.

❖La connexion au serveur peut également être établie à l'aide de la fonction **mysql_pconnect(),** qui requiert les mêmes paramètres que mysql_connect() mais établit une connexion persistante.

Connexion au serveur MySQL

- Pour mettre fin à la connexion, vous appelez la fonction mysql_close() , dont la syntaxe est la suivante :

```
mysql_close([$idcom])
```

Le paramètre **$idcon** est l'identifiant de connexion

- Si le serveur comporte plusieurs bases de données, le script précise la base désirée au moyen de la fonction mysql_select_db() dont la syntaxe est :

```
mysql_select_db($nom_base [,$idcom])
```

Cette fonction retourne TRUE si la base existe et FALSE dans le cas contraire. Après l'appel de cette fonction, toutes les requêtes SQL envoyées au serveur MySQL sont effectuées sur la base choisie, sans qu'il soit besoin de le préciser

Connexion au serveur MySQL

❖ **Exemple : connexion à une base de données nommé base**

```php
<?php
$idcon = @mysql_connect("localhost", "root", "");
$base = "base";
$idbase = @mysql_select_db($base);
if (!$idcon | !$idbase) {
    echo "<script type='text/javascript'>";
    echo "alert('Connexion Impossible à la base $base')</script>";
} else {
    echo "connexion etablie";
}
//Interaction avec une base MySql via des requêtes ;
mysql_close($idcon);
?>
```

L'exécution d'une requête SQL

- Pour exécuter une requête SQL, vous utilisez d'abord la fonction **mysql_query()**, dont la syntaxe est la suivante :

```
mysql_query($requete[,$idcom])
```

- La chaîne **$requete** contient le code de la requête SQL. Elle ne doit pas se terminer par un point-virgule.
- La fonction retourne un identifiant de résultat ,noté systématiquement $result
  Si la requête contient des commandes SELECT, cet identifiant permet d'accéder aux données fournies par la requête en utilisant certaines fonctions PHP.
- Pour les autres requêtes (suppression, modification, mise à jour), la fonction retourne **TRUE** si la requête est bien exécutée. Si une requête quelconque n'est pas exécutée, la fonction **mysql_query()** retourne **FALSE**

L'exécution d'une requête SQL

- **Exemple : exécution de la requête $req**

```php
<?php
@mysql_connect("localhost", "root", "");
mysql_select_db("base");
$req = "select * from tache";
$result = mysql_query($req);
if(!$result){
    echo "Lecture impossible"; }
else{
    echo "pret à parcourir le resultat";
    //Parcourir le résultat de la requête
}
?>
```

Lecture du résultat d'une requête

- Pour les opérations d'insertion, de suppression ou de mise à jour de données dans une base, il est simplement utile de vérifier si la requête a bien été exécutée.
- Par contre, lorsqu'il s'agit de lire le résultat d'une requête contenant la commande SELECT, il est indispensable de recueillir les données. PHP offre une grande variété de fonctions qui permettent de récupérer des données sous des formes diverses, la plus courante étant un tableau.
- Chacune de ces fonctions ne récupérant qu'une ligne du tableau à la fois, il faut recourir à une ou plusieurs boucles pour lire l'ensemble des données.

Lecture à l'aide d'un tableau

❖ La fonction la plus perfectionnée pour lire les données dans un tableau est **mysql_fetch_array().** Sa syntaxe est la suivante :

```
mysql_fetch_array($result ,typetab)
```

- Cette fonction retourne un tableau contenant autant d'éléments qu'il y a de colonnes précisées dans la requête SELECT.
- Le paramètre **$result** est celui qui a été retourné par la fonction **mysql_query()** .
- Le paramètre **typetab** une constante entière précisant si le tableau retourné doit être associatif (valeur MYSQL_ASSOC), indicé (valeur MYSQL_NUM) ou les deux à la fois (valeur MYSQL_BOTH, qui est la valeur par défaut).
- Si le tableau est associatif, les clés du tableau sont les noms des colonnes de la table interrogée ou des alias, si vous en avez définis dans la requête. Il n'est donc pas nécessaire de connaître l'ordre des colonnes dans la table.
- La fonction **mysql_num_fields($result)** permet de déterminer le nombre de colonnes du résultat et la fonction **mysql_num_rows($result)** le nombre de lignes

Lecture à l'aide d'un tableau

❖ **Exemple : Parcours de la table tache.**

```php
<?php
@mysql_connect("localhost", "root", "");
mysql_select_db("base");
$req = "select * from tache";
$result = mysql_query($req);
echo "<table border=1>";
    while ($ligne = mysql_fetch_array($result, MYSQL_NUM)) {
            echo "<tr>";
                foreach ($ligne as $valeur) {echo "<td> $valeur </td>";}
            echo "</tr>";}
echo "</table>";
?>
```

Lecture à l'aide d'un tableau

Les deux autres fonctions suivantes permettent de récupérer une ligne de résultat à la fois dans un tableau :

- ✓ **mysql_fetch_assoc(resource $result)** , qui retourne un tableau uniquement associatif dont les clés sont les noms des colonnes de la table interrogée.
- ✓ **mysql_fetch_row(resource $result)** , qui retourne un tableau uniquement indicé dont les indices sont les numéros des colonnes dans la table interrogée.

Ces fonctions peuvent s'utiliser en lieu et à la place de **mysql_fetch_array()** avec la même rapidité.

Lecture des noms de colonnes

1ére méthode :

Après l'envoi de la requête par la fonction **mysql_query()** ,un premier appel à la fonction **mysql_fetch_array()** lit la première ligne du résultat dans un tableau associatif . Une boucle foreach lit les clés du tableau obtenu et affiche les titres des colonnes du tableau .

Exemple:

```php
<?php
$idcon = @mysql_connect("localhost", "root", "");
$base = "base";
$idbase = @mysql_select_db($base);
$req="select * from tache";
$result=mysql_query($req);
$ligne=  mysql_fetch_array($result,MYSQL_ASSOC);
foreach ($ligne as $key => $value) {
   echo "$key||";
}
?>
```

Lecture des noms de colonnes

2éme méthode :

Il existe une autre méthode pour lire le nom des colonnes du résultat d'une requête SELECT. La fonction **mysql_field_name()** , dont la syntaxe est la suivante :
mysql_field_name($result, num_col)

retourne le nom de la colonne ou de l'alias dont le numéro est passé en second paramètre L'ordre des colonnes est celui de la requête et donc pas nécessairement celui de la table

Exemple:

```php
<?php
@mysql_connect("localhost", "root", "");
mysql_select_db("base");
$req = "select * from tache";
$result = mysql_query($req);
$nb_col = mysql_num_fields($result);
$nb_row = mysql_num_rows($result);
echo "nb col: $nb_col";
echo "nb row: $nb_row";
for ($i = 0; $i < $nb_col; $i++) {
    echo mysql_field_name($result, $i) . "<br>";
}
?>
```

Insertion de données dans la base

- Comme vous l'avez vu au chapitre consacré aux formulaires, l'outil de communication essentiel entre le poste client et le serveur est le formulaire HTML. Ce dernier permet la saisie de données et leur envoi vers le serveur, sur lequel un script PHP enregistre les valeurs saisies dans la base MySQL.
- Seule la requête qui contient la commande SQL INSERT, qui va être envoyée au serveur, distingue cette opération de celle de lecture des données.

Exemple :

```
$requete="INSERT INTO client VALUES('$_POST[‘id_client']',
                                    '$_POST['nom']',
                                     '$_POST['age ']') ')
```

- La requête SQL INSERT contenue dans la variable **$requete** permet l'insertion de toutes ces valeurs dans la table client. Dans le cas d'une commande INSERT, le résultat de la requête envoyée par la fonction mysql_query() n'est qu'une valeur booléenne TRUE ou FALSE selon que l'insertion a été réalisée ou non.

Insertion de données dans la base

Fonctions utiles :

- La fonction **mysql_insert_id()** permet de retourne la dernière valeur insérée dans une colonne ayant cette option AUTO_INCREMET.
- Dans le cas où une requête est formée en utilisant des saisies faites par l'utilisateur dans un formulaire, il est préférable d'utiliser un caractère d'échappement pour les caractères spéciaux des chaînes récupérées dans le tableau $_POST, en particulier les guillemets, qui peuvent poser problème. Vous disposez pour cela des fonctions **mysql_escape_string()**et **mysql_real_escape_string()** , dont les syntaxes respectives sont les suivantes :
  - **mysql_escape_string($chaine)** :Protège les caractères spéciaux de la chaîne $ch (sauf % et _) et retourne la chaîne protégée.
  - **mysql_real_escape_string($chaine)**: Échappe la chaîne précisée afin de l'inclure dans une requête SQL
  - **Exemple :**

```
$code=mysql_escape_string($_POST['code']);
```

Mise à jour d'une table

Seule la requête qui contient la commande SQL **UPDATE**, qui va être envoyée au serveur, distingue cette opération de celle de lecture des données.

Exemple : Mise à jour de la table client

```php
<?php
// étapes de connexion et de sélection de base sont préétablis
$nom=mysql_escape_string($_POST['nom']);
$adresse=mysql_escape_string($_POST['adresse']);
$mail=mysql_escape_string($_POST['mail']);
$code=mysql_escape_string($_POST['code']);
//Requête SQL
$requete="UPDATE client SET nom='$nom',adresse='$adresse', mail='$mail' WHERE id_client='$code'";
//execution de la requete update
$result=@mysql_query($requete);
?>
```

Suppression d'un enregistrement dans une table

Seule la requête qui contient la commande SQL **DELETE**, qui va être envoyée au serveur, distingue cette opération de celle de lecture des données.

Exemple : suppression d'un enregistrement dans la table client

```php
<?php
// étapes de connexion et de sélection de base sont préétablis
$nom=mysql_escape_string($_POST['nom']);
$adresse=mysql_escape_string($_POST['adresse']);
$mail=mysql_escape_string($_POST['mail']);
$code=mysql_escape_string($_POST['code']);
//Requête SQL
//$requete="DELETE FROM `utilisateur` WHERE `id` = 1";
$result=@mysql_query($requete);
?>
```