Chapitre 3 : Accès objet à MySQL avec PHP

Une tendance évidente qui a prévalu dans le développement de PHP 5 est la programmation objet ; l'accès à MySQL n'y a pas échappé. Cette possibilité est liée à l'utilisation de l'extension mysqli, dite « mysql améliorée », qui comprend les trois classes suivantes :

- La classe mysqli qui permet de créer des objets de type mysqli_object. Cette dernière possède des méthodes et des propriétés pouvant remplacer ou compléter les fonctions de l'extension mysql comme la connexion à la base, l'envoi de requêtes, les transactions ou les requêtes préparées.
- La classe mysqli_result, permettant de gérer les résultats d'une requête SQL effectuée par l'objet précédent. Ses méthodes et propriétés sont également les équivalents des fonctions de l'extension mysql.
- La classe mysql_stmt qui représente une requête préparée. Il s'agit là d'une nouveauté par rapport à l'extension mysql.

Nous allons maintenant reprendre différents exemples du cours PHP (niv. 1) et voir comment obtenir les mêmes résultats via un accès objet.

I. Connexion au serveur MySQL

La première chose à faire est de se connecter au serveur MySQL. Pour cela nous créons un objet de la classe mysqli selon la syntaxe suivante :
$idcom est un objet mysqli et $host, $user et $pass sont le nom du serveur MySQL, le nom de l'utilisateur et le mot de passe. Le paramètre facultatif $base permet de choisir la base sur laquelle seront effectuées les commandes SQL.
L'objet $idcom représentant la connexion sera utilisé pour toutes les opérations à effectuer sur la base. Si la connexion n'est pas effectuée, la variable $idcom contiendra la valeur FALSE, permettant ainsi de tester si la connexion est bien réalisée.
Si le serveur comporte plusieurs bases de données et que le paramètre $base a été employé lors de la création de l'objet $idcom, il est possible de changer de base sans interrompre la connexion en cours en appelant la méthode select_db() avec la syntaxe suivante :
boolean $idcom–>select_db (string $base)
Cette méthode retourne un booléen qui permet de tester la bonne fin de l'opération. Si le paramètre $base n'a pas été précisé lors de la création de l'objet mysqli, c'est cette méthode qui permet de choisir la base.
La structure d'un script accédant à MySQL est donc la suivante :

```
<?php
//Connexion au serveur
$idcom = new mysqli("localhost","root","","ma_base");
//Affichage d'un message en cas d'erreurs
if(!$idcom)
{
echo "Connexion Impossible à la base";
}
//*******************************
//Requêtes SQL sur la base choisie
//Lecture des résultats
//*******************************
?>
```

Comme nous l'avons fait pour l'accès procédural à MySQL, nous avons intérêt à créer une fonction de connexion au serveur que nous réutiliserons systématiquement dans tous les exemples qui suivent. C'est l'objet de l'exemple 1 qui crée la fonction connexobjet() dont les paramètres sont les paramètres de connexion dans les variables $host,$user et $pass et le nom de la base dans la variable $base.
La fonction initialise d'abord les paramètres de connexion puis crée un objet mysqli ; elle vérifie ensuite que la connexion est bien réalisée, affiche un message d'errer et sort de la fonction en cas de problème. Si la connexion est bien réalisée elle retourne l'objet $idcom.

☛ **Exemple 1. Fonction de connexion au serveur**

```
<?php
function connexobjet($host,$user,$pass, $base)
{
$idcom = new mysqli($host,$user,$pass,$base);
if (!$idcom)
{
echo "Connexion Impossible à la base'";
exit();
}
return $idcom;
}
?>
```

Chacun de vos scripts d'accès à la base doit donc contenir les lignes suivantes :

```
include("connexobjet.php");
$idcom = connexobjet ("localhost","root","","nom_base");
```

II. Envoi de requêtes SQL au serveur

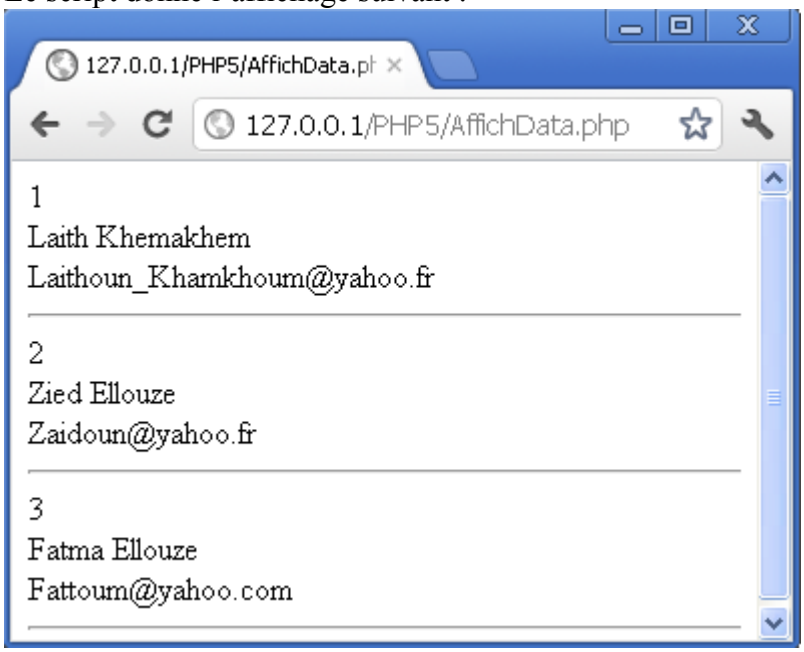
Les différentes opérations à réaliser sur la base MySQL impliquent l'envoi de requêtes SQL au serveur.
Pour envoyer une requête, il faut employer la méthode query() de l'objet mysqli dont la syntaxe est :
divers $idcom–>query (string $requete)
La requête est soit directement une chaîne de caractères, soit une variable de même type.
La méthode query() retourne TRUE en cas de réussite et FALSE sinon et un objet de type mysqli_result pour les commandes SQL de sélection comme SELECT. Nous pouvons donc tester la bonne exécution d'une requête. En résumé, un script d'envoi de requête a la forme suivante :

Exemple 2 : Envoi de requête type

```
<?php
include ("ConnexObjet.php");
$idcom=connexobjet("localhost","root","","basee");
$requete="SELECT * FROM liste ";
$result=$idcom->query($requete);
if(!$result)
{
echo "Lecture impossible";
}
else
{
//Lecture des résultats
while ($row = $result->fetch_array(MYSQLI_NUM))
{
foreach($row as $donn)
{
echo $donn."<br>";
```

```
}
echo "<hr />";
}
}
?>
```

Le script donne l'affichage suivant :



III. 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, la méthode query() retourne un objet de type mysqli_result, identifié dans nos exemples par la variable $result. La classe mysqli_result offre une grande variété de méthodes permettant de récupérer des données sous des formes diverses, la plus courante étant un tableau.
Chacune de ces méthodes ne récupérant qu'une ligne du tableau à la fois, il faut recourir à une ou plusieurs boucles pour lire l'ensemble des données.

III.1. Lecture à l'aide d'un tableau

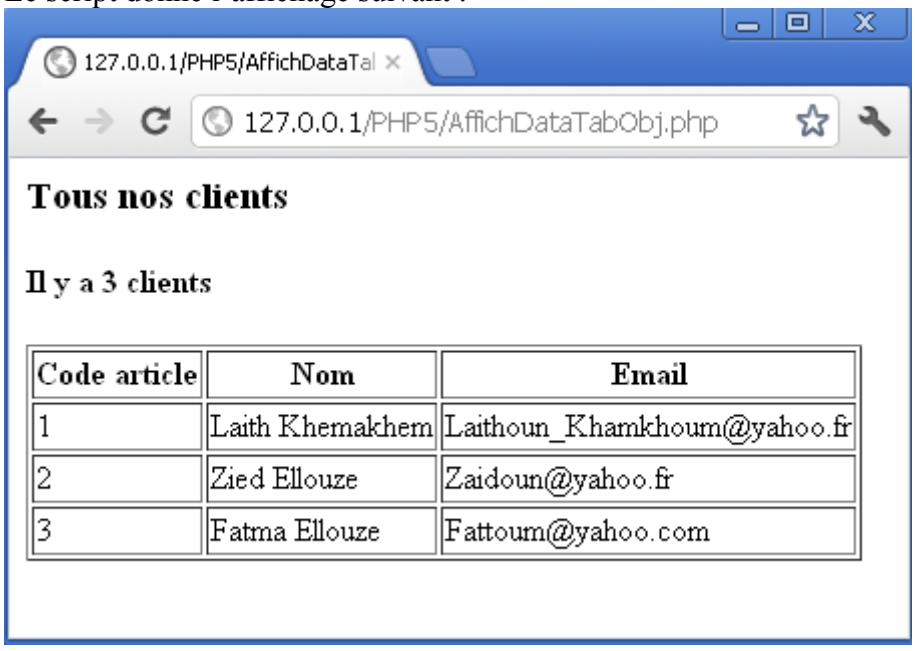
La méthode des objets instances de la classe mysqli_result la plus perfectionnée pour lire des données dans un tableau est fetch_array(), dont la syntaxe est :
array $result–>fetch_array (int type)
Elle retourne un tableau qui peut être indicé (si la constante type vaut MYSQLI_NUM), associatif (si type vaut MYSQLI_ASSOC), dont les clés sont les noms des colonnes ou les alias de la table interrogée, ou encore mixte, contenant à la fois les indices et les clés (si type vaut MYSQLI_BOTH). Pour lire toutes les lignes du résultat, il faut écrire une boucle (while par exemple) qui effectue un nouvel appel de la méthode fetch_array() pour chaque ligne.
Cette boucle teste s'il existe encore des lignes à lire, la méthode fetch_array() retournant la valeur NULL quand il n'y en a plus. Pour lire et afficher chaque ligne nous utilisons ensuite une boucle foreach. Notez encore une fois que si le tableau retourné est indicé, l'indice 0 correspond au premier attribut écrit dans la requête et ainsi de suite.
L'exemple 3 met cette méthode en pratique dans le but d'afficher le contenu de la table liste de la base "base" dans un tableau HTML.

Nous récupérons d'abord le nombre de clients dans la table grâce à la propriété num_rows de l'objet mysqli_result et affichons ce nombre. Une boucle while permet de lire une ligne à la fois dans un tableau indicé, puis une boucle foreach permet d'afficher chacune des valeurs du tableau dans un tableau HTML.

Exemple 3. Lecture de la table article

```
<?php
include ("ConnexObjet.php");
$idcom=connexobjet("localhost","root","","basee");
$requete="SELECT * FROM liste ";
$result=$idcom->query($requete);
if(!$result)
{
echo "Lecture impossible";
}
else
{
$nbcol=$result->field_count;
$nbart=$result->num_rows;
echo "<h3> Tous nos clients</h3>";
echo "<h4> Il y a $nbart clients </h4>";
echo "<table border=1>";
echo "<tr><th>Code article</th> <th>Nom</th> <th>Email</th></tr>";
while($ligne=$result->fetch_array(MYSQLI_NUM))
{
echo "<tr>";
foreach($ligne as $valeur)
{
echo "<td> $valeur </td>";
}
echo "</tr>";
}
echo "</table>";
}
?>
```

Le script donne l'affichage suivant :



IV. Insertion de données dans la base

Le formulaire HTML est l'outil privilégié pour saisir de données et les envoyer vers le serveur PHP/MySQL.
Nous disposons du formulaire suivant pour envoyer les données à la table liste de notre base.

```
<html>
<head>
<title>FormulaireInscription</title>
<head>
<body>
<H1>Formulaire d'inscription</H1>
<form method="post"
action="InsertDataObj.php">
Nom : <input type="text"
name="nom"><br>
Email : <input type="text"
name="email"><br>
<input type="submit" name="submit" value="Insérer">
</form>
</body>
</html>
```

La commande SQL INSERT est utilisée lors de l'opération d'insertion. Le script de l'exemple 4 réalise ce type d'opération en récupérant les données saisies par le client dans un formulaire. Nous commençons par vérifier
l'existence des saisies obligatoires correspondant aux variables $_POST['nom'] et $_POST['email'].
Le script récupère toutes les saisies. La colonne id de la table liste ayant été déclarée avec l'option AUTO_INCREMENT, il faut y insérer la valeur NULL en lui donnant la valeur "". Le résultat de la requête est ici un booléen permettant de vérifier la bonne insertion des données dans la table.
Le script doit communiquer au client son identifiant pour qu'il puisse modifier éventuellement ses données. La valeur de la colonne id est récupérable à l'aide de la propriété insert_id de l'objet mysqli. Ce nombre entier est affiché.

Exemple 4. Insertion de données

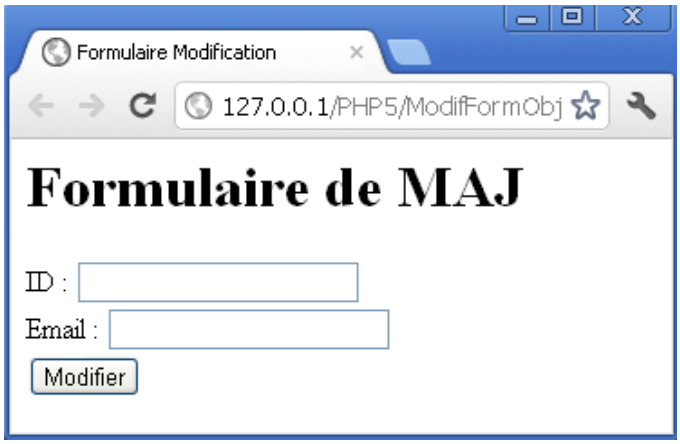
```
<?php
include ("ConnexObjet.php");
$idcom=connexobjet("localhost","root","","basee");
if(!empty($_POST['nom'])&& !empty($_POST['email']))
{
$id="";
$nom=$_POST['nom'];
$email=$_POST['email'];
//Requête SQL
$requete="INSERT INTO liste VALUES('$id','$nom','$email')";
$result=$idcom->query($requete);
if(!$result)
{
```

```
echo $idcom->errno;
echo $idcom->error;
echo "Erreur : ".$idcom->error;
}
else
{
echo "Vous êtes enregistré Votre numéro de client est : ".$idcom-
>insert_id;
}
}
else {echo "<h3>Formulaire à compléter!</h3>";}
?>
```

V. Mise à jour d'une table

Un client doit pouvoir modifier ses coordonnées, par exemple son e-mail. Nous disposons d'un formulaire qui permet la saisie de l'id client et du nouveau email.

```
<html>
<head>
<title>Formulaire Modification</title>
<head>
<body>
<H1>Formulaire de MAJ</H1>
<form method="post"
action="ModifDataObj.php">
ID : <input type="text" name="id"><br>
Email : <input type="text"
name="email"><br>
<input type="submit" name="submit"
value="Modifier">
</form>
</body>
</html>
```



La mise à jour des coordonnées du client est réalisée par le script de l'exemple 5.

```
<?php
include ("ConnexObjet.php");
$idcom=connexobjet("localhost","root","","base");
if(!empty($_POST['id'])&& !empty($_POST['email']))
{
//Requête SQL
$id=$_POST['id'];
$email=$_POST['email'];
$requete="UPDATE liste SET email='$email' WHERE id='$id'";
```

```
$result=$idcom->query($requete);
if(!$result)
{
echo $idcom->errno;
echo $idcom->error;
echo "Erreur : ".$idcom->error;
}
else
{
echo "Votre email a été modifié";
}
}
else {echo "<h3>Formulaire à compléter!</h3>";}
?>
```

VI. Suppression de lignes d'une table

Un client peut être supprimé. Nous disposons d'un formulaire qui permet la saisie de l'id client à supprimer.


```
<html>
<head>
<title>Formulaire
Suppresion</title>
<head>
<body>
<H1>Formulaire de Suppression</H1>
<form method="post"
action="SupDataObj.php">
ID : <input type="text"
name="id"><br>
<input type="submit" name="submit" value="Supprimer">
</form>
</body>
</html>
```

La suppression du client est réalisée par le script de l'exemple 6.

```
<?php
include ("ConnexObjet.php");
$idcom=connexobjet("localhost","root","","base");
if(!empty($_POST['id']))
{
//Requête SQL
$id=$_POST['id'];
```

```
$requete="DELETE FROM liste WHERE id='$id'";
$result=$idcom->query($requete);
if(!$result)
{
echo $idcom->errno;
echo $idcom->error;
echo "Erreur : ".$idcom->error;
}
else
{
echo "Votre enregistrement a été supprimé";
}
}
else {echo "<h3>Formulaire à compléter!</h3>";}
?>
```

VII. Recherche

Nous pouvons avoir besoin de chercher des informations dans la base de données relatives à un critère donné. Par exemple, nous voulons connaître les clients ayant un compte email sur le serveur donné. Pour se faire, nous disposons du formulaire suivant :

```
<html>
<head>
<title>Formulaire
Recherche</title>
<head>
<body>
<H2>Formulaire de Recherche
email</H2>
<form method="post"
action="Cherch1DataObj.php">
Serveur Email : <input
type="text" name="email"><br>
<input type="submit" name="submit" value="Chercher">
</form>
</body>
</html>
```

La recherche de clients ayant un compte email sur un serveur donné est réalisée par le script 7.

```
<?php
include ("ConnexObjet.php");
$idcom=connexobjet("localhost","root","","basee");
if(!empty($_POST['email']))
{
//Requête SQL
$email=$_POST['email'];
```

```
$requete="SELECT * FROM liste WHERE email LIKE'%$email%'";
$result=$idcom->query($requete);
if(!$result)
{
echo $idcom->errno;
echo $idcom->error;
echo "Erreur : ".$idcom->error;
}
else
{
$nbcol=$result->field_count;
$nbart=$result->num_rows;
$titres=$result->fetch_fields();
echo "<h3> Il y a $nbart clients ayant un email sur le serveur :


$email</h3>";
//Affichage des titres du tableau
echo "<table border=1>";
echo "<tr>";
foreach($titres as $nomcol=>$val)
{
echo "<th>", $titres[$nomcol]->name ,"</th>";
}
echo "</tr>";
//Affichage des valeurs du tableau
for($i=0;$i<$nbart;$i++)
{
$ligne=$result->fetch_row();
echo "<tr>";
for($j=0;$j<$nbcol;$j++)
{
echo "<td>",$ligne[$j],"</td>";
}
echo "</tr>";
}
echo "</table>";


}
}
else {echo "<h3>Formulaire à compléter!</h3>";}
?>
```

Un exemple d'exécution se présente comme suit :