Utiliser PDO avec PHP

Par Yohann JOURDAIN • Publié le 22/01/2018 à 11:42:18 • Noter cet article: ☆☆☆☆☆ (0 votes)
Avis favorable du comité de lecture

Yohann JOURDAIN

Promotion: M.Sc. 2
Campus de Paris

Introduction

Lors de cet article, nous allons voir comment utiliser PDO avec PHP. L'objectif sera d'apprendre comment insérer, sélectionner, mettre à jour et supprimer des données dans une base de données par des formulaires et des actions utilisateurs.
Au cours de cet article nous utiliserons l'éditeur PHPStorm et d'une application de gestion de bases de données comme PHPMyAdmin.
Avant de poursuivre, nous allons revoir la signification de certains termes dont nous allons avoir besoin durant cet article :

- **PHP :** De sa signification « Hypertext Preprocessor » est un langage de programmation qui permet de créer des applications ou des sites web et de traiter des données sur cette application.
- **PDO :** De sa signification « PHP Data Object » est une interface dont l'objectif est d'accéder à une base de données et ainsi accéder à son contenu.
- **PHPMyAdmin :** Application sur laquelle sera créée notre base de données.

Connexion à la base de données en PHP

Grâce au langage PHP, nous pouvons manipuler des données insérées dans une base de données. L'objectif est de pouvoir effectuer des actions sur ces données depuis une application web gérée et consultée par des utilisateurs via un formulaire (pour l'ajout ou la modification de données par exemple), de liste à travers de tableaux (dans le but de consulter les données) .

Création de la base de données avec PHPMyAdmin

Dans un premier temps, dans l'application PHPMyAdmin, nous allons créer notre table **« member »** qui va stocker nos données durant la suite de cet article. Le but étant de créer une base de données permettant de gérer différents membres d'une association par exemple.

Nom	Type	Taille/Valeurs*	Valeur par défaut	Interclassement	Attributs
id	INT	11	Aucun(e)		
lastname	VARCHAR	255	Aucun(e)	latin1_swedish_ci	
firstname	VARCHAR	255	Aucun(e)	latin1_swedish_ci	
email	VARCHAR	255	Aucun(e)	latin1_swedish_ci	
city	VARCHAR	255	Aucun(e)	latin1_swedish_ci	

Pour cela nous allons créer 5 champs qui sont les suivants :

- Un champ **« id »** de type « int » qui sera auto-incrémenté lors de l'insertion d'une donnée.
- Un champ **« lastname »** de type « varchar » de taille 255 qui contiendra le nom de notre membre.

Comment se connecter à la base de données ?

Comme dis précédemment, grâce au langage PHP, l'outil PDO permet de nous connecter à de nombreuses bases de données. Pour réussir à se connecter, nous avons besoins de plusieurs éléments :

- **Le nom de l'hôte :** Il correspond à l'adresse de l'ordinateur ou du serveur où l'application web est installée. Si vous êtes en local (sur votre pc) la valeur est souvent « localhost ». Cependant, il peut contenir aussi une adresse IP.
- **Le nom de la base de données :** C'est le nom correspondant à la base de données qui va contenir nos tables.
- **Le login permettant d'accéder à la base :** C'est l'identifiant qui va vous permettre de vous connecter à votre base afin de pouvoir la gérer en créant des tables, des colonnes...
- **Le mot de passe de la base :** C'est le mot de passe qui permet l'accès à la base.

```php
class database
{
    private $_host = 'mysql:host=localhost;port=3306;dbname=pdo;charset=utf8';
    private $_username = 'root';
    private $_password = '';
    private $_bdd;

    public function getHost()
    {
        return $this->_host;
    }
    public function setHost($host)
    {
        $this->_host = $host;
    }
```

Pour notre exemple, nous avons créé une classe **« database »** qui va contenir notre variable **« host »** qui correspond au nom de l'hôte et à de la base de données. On peut aussi y ajouter un port si on a changé celui par défaut qui est le 3306. Une variable **« username »** qui est le login et une variable **« password »**. Dans notre exemple ci-dessus, nous pouvons voir que notre base de données se nomme « pdo » qui a comme login « root » et sans mot de passe. Enfin, une variable **« bdd »** avec laquelle on va pouvoir nous connecter à notre base.
Nous devons créer également nos « getter » et nos « setter » pour nos variables.
Pour rappel, les « getters » (ou accesseurs en français) vont nous permettre de récupérer la valeur assignée à la variable, tandis que les « setters » (ou mutateur en français) vont permettre de modifier la valeur de la variable.

```php
//Connexion à la base de données
function connexion()
{
    try {
        $this->setBdd(new PDO($this->getHost(), $this->getUsername(), $this->getPassword()));
    } catch (Exception $e) {
        die('Erreur : ' . $e->getMessage());
    }
}
```

Dans notre même classe, nous allons implémenter une méthode « connexion », qui va nous permettre de nous connecter à notre base de données. Pour cela, on va utiliser un **« try ... catch ... »**. Lorsqu'on va appeler notre fonction, on va tout d'abord tenter de se connecter avec nos identifiants grâce au **« try »**. Si on n'arrive pas à s'y connecter, alors la partie **« catch »** va nous afficher une erreur. Dans la première partie, nous allons faire appel au « setter » de la variable **« bdd »**, que l'on a créé au début de notre classe. On lui passe ensuite un paramètre, l'outil PDO qui va contenir le nom de l'hôte, le login et le mot de passe de notre base. Ces derniers seront récupérés par nos « getters ».

Insertion des données en PHP

Dans cette deuxième partie, nous allons apprendre comment utiliser PDO pour insérer des données grâce à un formulaire.

HTML - Création du formulaire

```html
<div class="insert">
    <h2>Ajouter un membre</h2>
    <form action="member/insert.php" method="post">
        <input type="text" name="lastname" placeholder="Nom" required/><br/><br/>
        <input type="text" name="firstname" placeholder="Prénom" required/><br/><br/>
        <input type="text" name="email" placeholder="E-mail" required/><br/><br/>
        <input type="text" name="city" placeholder="Ville" required/><br/><br/>
        <input type="submit" name="form_insert" value="Insérer" required/><br/><br/>
    </form>
</div>
```

Pour notre exemple, nous allons créer un formulaire ayant comme action notre fichier **« insert.php »** situé dans notre dossier **« member »** et la méthode **« post »** qui permet de ne pas rendre visible les données à l'utilisateur par l'URL par exemple.

- Un champ « **Prénom** » de type « text » et comme nom « **firstname** ».
- Un champ **« E-mail »** de type « text » et comme nom **« email »**.
- Un champ **« Ville »** de type « text » et comme nom **« city »**.
- Un champ **« Insérer »** de type « submit » et comme nom **« form_insert »**.

Traitement PHP

Dans notre classe « database », on va ajouter une méthode (ici « addMember ») qui va retourner une requête SQL, afin de nous permettre d'insérer des données.

```
//Requête permettant d'ajouter un membre.
function addMember()
{
    return 'INSERT INTO member(lastname, firstname, email, city) VALUES (:lastname, :firstname, :email, :city)';
}
```

Explication de la requête :

- On va insérer dans la table **« member »** toutes les valeurs « lastname », « firstname », « email » et « city » dont les valeurs seront données via le formulaire.

Dans le fichier « index.php » :

```
require_once '../../pdo/database/database.php';

function insertMember()
{
    $bdd = new database();
    $bdd->connexion();
    $lastname = $_POST['lastname'];
    $firstname = $_POST['firstname'];
    $email = $_POST['email'];
    $city = $_POST['city'];
    $query = $bdd->getBdd()->prepare($bdd->addMember());
    $array = array(
        'lastname' => $lastname,
        'firstname' => $firstname,
        'email' => $email,
        'city' => $city
    );
    $query->execute($array);

}

if (isset($_POST['form_insert']))
{
    insertMember();
    header( string: 'Location: ../index.php');
}
```

La première étape pour insérer des données dans notre base est d'appeler notre fichier qui va contenir les éléments de connexion. Pour cela, on va utiliser la méthode **« require_once <chemin_du_fichier> ».**
Ensuite, on va créer notre méthode (ici « insertMember »). On doit se connecter à notre base de données, pour cela, on va appeler la méthode **« connexion »** de notre classe **« database »**.
On va récupérer nos données dans des variables grâce à **« $_POST[<nom_du_champ>] »**. Dans notre variable **« $query »** on appel notre variable **« $bdd »**, grâce à son **« getter »** puis on prépare notre requête, en passant en paramètre la méthode **« addMember »**, qui est notre requête SQL créée dans la classe **« database »**.
Pour terminer, on crée un tableau dans lequel on va assigner les valeurs à insérer pour chaque champs de notre base, par la syntaxe : **« champ_base_de_données »** => **« valeur_du_formulaire »**. Enfin, on exécute notre requête.
Pour appeler notre fonction, lors d'une action sur le formulaire, on doit vérifier si notre bouton « Insérer » a bien été actionné, en renvoyant une valeur booléenne, grâce à la méthode **« isset »**. Une fois exécutée, on redirige l'utilisateur vers la page **« index.php »**.

Résultat

Table 1.

Sélection des données en PHP

Dans ce troisième chapitre, nous allons étudier comment sélectionner des données présentent dans notre base avec et sans conditions.

Afficher les données

Dans notre classe « database », on va ajouter une fonction (ici « selectAllMember ») qui va nous permettre de sélectionner des données.

```php
//Requête qui affiche tous les membres
function selectAllMember()
{
    return 'SELECT * FROM member';
}
```

Explication de la requête :

- On sélectionne tous les champs de la table **« member »**.

Dans le fichier « liste.php » :

```php
<?php
    require '../pdo/default/header.html';
    require_once '../pdo/database/database.php';
    include '../pdo/member/select.php';
?>
<div class="container">
    <?php
        listAllMember();
    ?>
</div>
<?php
    require '../pdo/default/footer.html';
?>
```

Ensuite, dans notre fichier principal, on va tout d'abord avoir besoin de notre classe « database » que l'on va appeler par la méthode PHP **« require_once »**. Puis, on va inclure le fichier PHP ou sera traiter notre requête (ici « select.php »). Enfin, on va faire appel à notre fonction **« listAllMember »** qui se trouve dans notre fichier que nous avons inclus.
Dans le fichier « select.php » :

```php
    $bdd = new database();
    $bdd->connexion();
    $query = $bdd->getBdd()->query($bdd->selectAllMember());

    echo '<table>';
    echo '<tr>';
    echo '<th>'."Id".'</th>';
    echo '<th>'."Nom".'</th>';
    echo '<th>'."Prénom".'</th>';
    echo '<th>'."E-mail".'</th>';
    echo '<th>'."Ville".'</th>';
    echo '</tr>';

    while ($data = $query->fetch())
    {
        echo '<tr>';
        echo '<th>'.$data['id'].'</th>';
        echo '<th>'.$data['lastname'].'</th>';
        echo '<th>'.$data['firstname'].'</th>';
        echo '<th>'.$data['email'].'</th>';
        echo '<th>'.$data['city'].'</th>';
        echo '<th>'.'<a id="link_update_member" href="edit.php?id='.$data['id'].'">'."Modifier".'</a>'.'</th>';
        echo '<th>'.'<a id="link_delete" href="member/delete.php?id='.$data['id'].'">'."Supprimer".'</a>'.'</th>'
        echo '</tr>';
    }
    echo '</table>';
    $query->closeCursor();
}
```

Dans notre fichier, nous allons créer la fonction **« listAllMember »** qui va dans un premier temps ce connecter à la base de données grâce à la classe **« database »**. Puis, on exécute la requête par la méthode **« query »**. Enfin, on crée une table avec les balises HTML dans laquelle on va afficher nos données. Elle aura pour en-tête : « Id », « Nom », « Prénom », « E-mail » et « Ville ».
Pour terminer, on va parcourir notre base de données tant qu'il y a un résultat pour l'afficher. Les liens « Modifier » et « Supprimer » seront utilisés dans les parties suivantes. Ne pas oublier de fermer la requête par le **« closeCursor »**.

Utiliser PDO avec PHP

Id	Nom	Prénom	E-mail	Ville		
7	Doe	John	john.doe@gmail.com	New York	Modifier	Supprimer
6	Dupont	Jean	jean.dupont@gmail.com	Paris	Modifier	Supprimer

Mise à jour des données en PHP

Dans cette nouvelle partie, on va apprendre comment modifier des données par le biais d'un formulaire.

Préparation du formulaire

Dans un premier temps on va ajouter une fonction **« selectMemberId »** dans la classe **« database »**.

```php
//Requête qui affiche un membre en fonction de son id
function selectMemberId()
{
    return 'SELECT * FROM member WHERE id = :id';
}
```

Explication de la requête :

- On sélectionne toutes les données de la table **« member »** dont le champ « id » est celui donné.

Dans notre fichier « edit.php » :

```
?>
<div class="container">
    <?php
        displayInfoMember();
    ?>
</div>
<?php
    require '../pdo/default/footer.html';
?>
```

Dans le même principe que pour le fichier précédent « liste.php », on va tout d'abord appeler notre classe « database » par le **« require_once »**, puis on va inclure le fichier « form_update.php » qui va contenir la méthode **« displayInfoMember »**.

Dans notre fichier « form_update.php » :

```
<?php
function displayInfoMember()
{
    $bdd = new database();
    $bdd->connexion();
    $id = $_GET['id'];

    $query = $bdd->getBdd()->prepare($bdd->selectMemberId());
    $array = array(
        'id' => $id
    );
    $query->execute($array);
    if ($data = $query->fetch())
    {
        echo '<form action="member/update.php?id='.$data['id'].'" method="post">';
        echo '<input type="text" name="lastname_update" value="'.$data['lastname'].'" required/>'.'</br>'.'</br>';
        echo '<input type="text" name="firstname_update" value="'.$data['firstname'].'" required/>'.'</br>'.'</br>';
        echo '<input type="email" name="email_update" value="'.$data['email'].'" required/>'.'</br>'.'</br>';
        echo '<input type="text" name="city_update" value="'.$data['city'].'" required/>'.'</br>'.'</br>';
        echo '<input type="submit" name="form_update" value="Modifier" />';
        echo '</form>';
    }
    else
    {
        echo '<p>'."Aucun résultat n'a été trouvé...".'</p>';
    }
    $query->closeCursor();

}
```

Dans la fonction **« displayMemberId »**, on va se connecter à la base de données dans un premier temps, grâce à la classe « database ».

On va récupérer l'id par la méthode **« $_GET »** dont le paramètre se trouve dans l'URL de la page :

localhost/pdo/edit.php?id=7

On prépare ensuite la requête avec l'id récupéré dans l'URL et on exécute la requête.

On vérifie s'il y a une donnée de trouvée, si c'est le cas, on affiche un formulaire ayant pour action le fichier **« update.php »** avec comme paramètre l'id de la donnée à modifier et en méthode **« post »**.

Pour notre exemple, on reprend le même type de formulaire que pour l'ajout d'un membre, à l'exception que maintenant ces champs auront comme valeur les données correspondant à l'id de l'URL. En revanche, si le paramètre de l'id n'est pas trouvé dans l'URL, alors on affiche un message à l'utilisateur. Pour terminer, on ferme la requête.

Table 2.

```php
//Requête permettant de mettre à jour un membre en fonction de son id
function updateMemberId()
{
    return 'UPDATE member SET lastname = :lastname, firstname = :firstname, email = :email, city = :city WHERE id
}
```

Explication de la requête :

- On met à jour la table **« member »** avec les champs « lastname », « firstname », « email », « city » envoyés par le formulaire dont l'id correspond.

Dans le fichier « update.php » :

```php
<?php
    require_once '../../pdo/database/database.php';

function updateInfoMember()
{
    $bdd = new database();
    $bdd->connexion();
    $lastname = $_POST['lastname_update'];
    $firstname = $_POST['firstname_update'];
    $email = $_POST['email_update'];
    $city = $_POST['city_update'];
    $id = $_GET['id'];

    $query = $bdd->getBdd()->prepare($bdd->updateMemberId());
    $array = array(
      'lastname' => $lastname,
      'firstname' => $firstname,
      'email' => $email,
      'city' => $city,
      'id' => $id
    );
    $query->execute($array);

    $query->closeCursor();
}
if (isset($_POST['form_update']))
{
    updateInfoMember();
    header( string: 'Location: ../liste.php');
}
```

Pour commencer, on appel le fichier contenant les informations de connexion à notre base de données. Ensuite, dans la fonction **« updateInfoMember »** on se connecte à la base, puis on récupère nos différentes informations par la méthode **« $_POST »** et notre « Id » en paramètre.
On prépare la requête avec nos différentes données, afin de l'exécuter et ne pas oublier de fermer la requête.
Lors de la soumission du formulaire, on va vérifier par sécurité si le formulaire a bien été soumis. Si c'est le cas on exécute la fonction et l'utilisateur est redirigé vers notre fichier « liste.php ».

Utiliser PDO avec PHP

Id	Nom	Prénom	E-mail	Ville		
7	Doe	John	john.doe@gmail.com	Los Angeles	Modifier	Supprimer
6	Dupont	Jean	jean.dupont@gmail.com	Paris	Modifier	Supprimer

Suppression des données en PHP

Dans cette dernière partie, nous allons voir comment supprimer une donnée avec le langage PHP.
Toujours dans notre classe « database », nous devons ajouter une fonction **« deleteMemberId »** qui retournera une requête SQL.

```php
//Requête permettant de supprimer un membre en fonction de son id
function deleteMemberId()
{
    return 'DELETE FROM member WHERE id = :id';
}
```

Explication de la requête :

```php
require_once '../../pdo/database/database.php';
function deleteMember()
{
    $bdd = new database();
    $bdd->connexion();
    $id = $_GET['id'];
    $query = $bdd->getBdd()->prepare($bdd->deleteMemberId());
    $array = array(
        'id' => $id
    );
    $query->execute($array);
    $query->closeCursor();
    header( string: 'Location: ../liste.php');

}
deleteMember();
```

Dans ce fichier, on va comme d'habitude appeler notre fichier qui contient nos informations de connexion. Dans la fonction **« deleteMember »** on se connecte à la base et on récupère l'id passé en paramètre de l'URL. On prépare la requête SQL avec le paramètre « id ». On exécute cette dernière et on ferme la requête. Pour terminer, l'utilisateur est directement redirigé vers la page « liste.php » où sont listés nos données.

Exemple avec le tableau du fichier « liste.php » rempli :

Utiliser PDO avec PHP

Id	Nom	Prénom	E-mail	Ville		
7	Doe	John	john.doe@gmail.com	Los Angeles	Modifier	Supprimer
6	Dupont	Jean	jean.dupont@gmail.com	Paris	Modifier	Supprimer

Suppression de la première donnée :

Utiliser PDO avec PHP

Id	Nom	Prénom	E-mail	Ville		
7	Doe	John	john.doe@gmail.com	Los Angeles	Modifier	Supprimer

Tableau vide après suppression de toutes les données :

Utiliser PDO avec PHP

Id	Nom	Prénom	E-mail	Ville

Conclusion

Comme nous avons pu le voir durant cet article, il est très facile de gérer les données avec l'outil PDO et le langage PHP. Nous pouvons avoir besoin de manipuler des données, lors de la création de différents site internet ou applications et ainsi manipuler des données diverses et variées. En effet, ici nous avons vu un aperçu de la manipulation des données.

Nous avons aussi survolé l'utilisation de l'application PHPMyAdmin, qui permet de créer des bases de données ainsi que de nombreuses tables personnalisées, dont le but premier est de stocker des données en local.

Nous avons pu voir qu'il est également important d'avoir des notions du langage SQL (Structured Query Language) afin d'améliorer et de comprendre l'intégration des données dans vos futurs développements. A travers différentes requêtes personnalisées, ce langage permet de traiter des données en les affichants avec plus ou moins de conditions, de les mettre à jour directement par l'utilisateur et de les supprimer.