GALIMATECH
L'EXPERIENCE DIGITALE

Le SGBD MySQL

***

Plan du cours

- Introduction
- Installer MySQL
- Installer PHPMyAdmin
- Utiliser MySQL avec PHP
- Sélectionner des données
- Afficher les données
- Ordonner les données
- Limiter les données retournées
- Insérer des données dans la base
- Mettre à jour
- Supprimer des données
- Les requêtes préparées

Une base de données permet d'enregistrer des informations de manière structurée et de les récupérer facilement.

MySQL est un système de gestion de base de données open source. Il utilise une base de données relationnelle et le langage SQL (Structured query langage) pour gérer ses données.

Dans ce cours nous allons voir comment installer MySQL sur Ubuntu puis nous verrons comment connecter la base de données à une application Php.

Installation

L'installation de MySQL sur Ubuntu se fait avec les commandes suivantes:

sudo apt update
sudo apt install mysql-server
Avec ces commandes nous avons d'abord mis à jour nos paquets, puis nous avons installé le paquet mysql-server. Nous devons maintenant configurer notre installation avec la commande suivante.
sudo mysql_secure_installation
Cette commande va nous permettre d'ajouter un mot de passe à notre installation.
Nous pouvons ensuite tester l'état de MySQL en faisant:
systemctl status mysql.service

Installation de PHPMyAdmin

PHPMyAdmin a été créé pour permettre d'utiliser MySQL via une interface Web. Sur un serveur Apache, il va avoir besoin de modules php pour fonctionner. Nous allons donc installer ces modules en même temps que phpMyadmin.

```
sudo update
sudo apt install phpmyadmin php-mbstring php-gettext
```

Lors de l'installation il faut choisir Apache comme serveur en appuyant sur la touche espace.

Nous pouvons ensuite activer le module mbstring et redémarrer le serveur Apache

```
sudo phpenmod mbstring
sudo systemctl restart apache2
```

Configuration de PHPMyAdmin

Par défaut on accède à MySQL avec le user root. Mais nous pouvons changer cela en créant un utilisateur dédié à la connexion à phpmyadmin. D'abord nous devons nous connecter à MySQL: `sudo mysql` (ou sudo mysql -u root -p si le mot de passe root est définie)

Nous pouvons alors créer un nouvel utilisateur avec mot de passe:

```
CREATE USER 'name'@'localhost' IDENTIFIED BY 'motDePasse';
```

Vous remplacerez *name* par un nom d'utilisateur et motDePasse par un mot de passe.

Puis on attribue tous les droits au nouvel utilisateur.

```
GRANT ALL PRIVILEGES ON *.* TO 'name'@'localhost' WITH GRANT OPTION;
```

Lier MySQL à notre projet PHP

Nous allons utiliser PDO pour connecter PHP à MySql.

PDO est l'acronyme pour PHP Data Objets. C'est un moyen simple et cohérent d'accéder aux bases de données. Cela signifie que les développeurs peuvent écrire du code portable beaucoup plus facilement. L'extension PDO est activé par défaut sur Php.

Pour connecter MySQL utilisant PDO nous allons créer un objet de la class PDO. Cette classe prend 3 paramètres:

- Le DSN (domain source name): ce paramètre comprend le host (souvent localhost), le nom de la base de données et l'encoding.
- le login de l'utilisateur de la base
- le mot de passe de l'utilisateur

```
$bdd = new PDO( 'mysql:host=localhost;dbname=basename;charset=utf8 ', 'root', 'root');
```

Il faudra inclure cette instruction dans un bloc try catch pour gérer les éventuelles erreurs de connexion de manière sécurisée.

Sélectionner des données

Pour récupérer les données de notre base de données nous allons construire une requête avec le mot-clé “SELECT”: SELECT * from nomTable.

Pour lancer des requêtes nous allons utiliser l'objet de la connexion et appeler la méthode query() de la classe PDO dessus. La requête avec PDO:

```
$connexion->query('SELECT * FROM nomTable');
```

La valeur * va récupérer tous les champs de la table.

```
$connexion->query('SELECT field1, field2,field3 FROM nomTable');
```

Afficher les données

Le retour de la requête avec la méthode **query()** est un gros objet. Pour afficher le contenu nous allons parcourir l'objet avec la méthode **fetch()** et la boucle *while* ou *foreach*. Nous devons donc enregistrer la requête dans une variable puis parcourir le résultat avec cette variable.

```
// On récupère tout le contenu d'une table
$requete = $bdd->query('SELECT * FROM nom_ma_table');

while ($donnees = $requete->fetch()){
    echo $donnees['collone_name'];
}
```

Et après la loupe il faut fermer le traitement avec la méthode **closeCursor()**;

```
$requete->closeCursor();
```

Ajouter une condition à la requête

Nous pouvons récupérer une liste de données qui correspondent à une condition avec la closure WHERE.

```
SELECT * FROM nom_ma_table WHERE field = 1
```

Ordonner les données

Nous pouvons définir l'ordre de retour des données avec le mot-clé ORDER.
Ce mot-clé prend à paramètre un champ et optionnellement la direction dans laquelle il doit ordonner les données.
Les directions possibles sont ascendantes ou descendantes; l'ordre par défaut est ascendant.

```
SELECT * FROM nom_ma_table ORDER field = 1
```

```
SELECT * FROM nom_ma_table ORDER field = 1 DESC
```

Limiter les données retournées

Au lieu de retourner tous les éléments nous pouvons limiter le nombre de retour avec le mot-clé LIMIT.
LIMIT prend deux valeurs numériques:On spécifie à partir de quelle entrée on commence à récupérer les données. Puis avec la deuxième valeur on précise le nombre de données à récupérer.

```
SELECT * FROM nom_ma_table LIMIT 0, 20
```

Insérer des données dans la base

Pour insérer des données dans la base de données nous allons utiliser la requête INSERT.

Cette requête se construit comme suite:

```
INSERT INTO nom_ma_table(field1,field2,field3) VALUES('valeur field1','valeur field2','valeur field3');
```

Nous remarquons deux partis dans cette requête. La première partie liste les colonnes ou insérer les données. La seconde partie liste les valeurs à insérer dans les colonnes. Il faut lister les valeurs dans le même ordre que les colonnes.

Après construction de notre requête nous pouvons utiliser PDO pour enregistrer dans la BD. Cette fois, à la place de la méthode query() nous allons utiliser la méthode exec().

```
$bdd->exec('INSERT INTO nom_ma_table(field1,field2,field3) VALUES('valeur field1','valeur field2','valeur field3');')
```

Mettre à jour

Après l'insertion de nouvelles valeurs, nous allons voir comment les éditer dans notre BD. La requête SQL pour mettre à jour les données est :

```
UPDATE nom_ma_table SET field1 = valeurs, field2 = valeur WHERE CONDITIONS
```

Cette requête s'exécute avec la méthode PDO exec()

```
$bdd->exec('UPDATE nom_ma_table SET field1 = valeurs, field2 = valeur WHERE CONDITIONS');
```

La closure `WHERE CONDITIONS'` est optionnelle. Mais si elle n'est pas renseignée, les modifications s'appliqueront sur tous les enregistrements.

Supprimer des données

Pour supprimer un enregistrement nous pouvons utiliser la requête `DELETE FROM nom_ma_table WHERE CONDITIONS'` Cette requête s'exécute avec la méthode exec().

```
$bdd->exec('DELETE FROM nom_ma_table WHERE CONDITIONS');
```

Là aussi la condition `WHERE CONDITIONS` est optionnelle mais si elle n'est pas renseignée on risque de supprimer tous les enregistrements dans la table sélectionnée.

Les requêtes préparées

Nous pouvons utiliser des variables pour renseigner des valeurs dans nos requêtes.

```
$id = 12;
$valeur = "Famara";
$bdd->exec("UPDATE users SET name = $valeur WHERE id = $id");
$bdd->query("SELECT id,name FROM users WHERE id = $id");
```

Le souci avec les requêtes de nos exemples est qu'elles présentent une faille. En effet nous nous exposons aux injections SQL. Pour sécuriser les requêtes nous allons donc utiliser les requêtes préparées. Le concept est d'utiliser en premier la méthode prepare() pour écrire la requête et remplacer les variables par '?'.

```
$req = $bdd->prepare("UPDATE users SET name = ? WHERE id = ?");
```

Et ensuite exécuter la requête avec la méthode execute() qui prend en paramètre un array avec les variables devant remplacer les '?' de la requête.

```
$req->execute([$valeur, $id]);
```

Les requêtes préparées sont impératives si les données à enregistrer proviennent d'un formulaire.

Les requêtes préparées

Une autre méthode d'écrire les requêtes préparées est d'utiliser des références à la place des '?'. Cette écriture est plus lisible.

```
$bdd->prepare("UPDATE users SET name = :valeur WHERE id = :id");
```

On peut alors utiliser les références dans un array pour appliquer les valeurs.

```
$bdd->execute(["valeur"=>$valeur,"id"=>$id]);
```

Liens utiles

- https://openclassrooms.com/fr/courses/918836-concevez-votre-site-web-avec-php-et-mysql/913655-quest-ce-quune-base-de-donnees
- https://phpdelusions.net/pdo
- https://www.digitalocean.com/community/tutorials/how-to-install-and-secure-phpmyadmin-on-ubuntu-18-04