Partie 3 : Lire / écrire dans une BDD

Partie 3 : Sommaire

1. Se connecter à la BDD
2. Lire des données
3. Ajouter / Modifier / Supprimer des données
4. Injections SQL

Partie 3 : Se connecter à la BDD

Pour accéder à une base de données il faut s'y connecter en renseignant des paramètres comme :

- L'adresse du serveur sur le quel est installée la base de données
- Le port de communication
- Le nom de la base de données
- Le nom d'utilisateur
- Le mot de passe
- L'extension utilisée

Nous utiliserons l'extension PDO pour nous connecter à notre base MYSQL car cette extension est la plus à jour et permet également de se connecter aux bases de données PostgreSQL et Oracle

Attention : l'extension php_pdo_mysql doit être activée sur votre serveur

Partie 3 : Se connecter à la BDD

Pour accéder à une base de données il faut s'y connecter en renseignant des paramètres comme :

- L'adresse du serveur sur le quel est installée la base de données
- Le port de communication
- Le nom de la base de données
- Le nom d'utilisateur
- Le mot de passe
- L'extension utilisée

Nous utiliserons l'extension PDO pour nous connecter à notre base MYSQL car cette extension est la plus à jour et permet également de se connecter aux bases de données PostgreSQL et Oracle

Attention : l'extension php_pdo_mysql doit être activée sur votre serveur

Partie 3 : Se connecter à la BDD

Connexion à la base de donnée :

```
$db = new PDO(‘mysql:host=localhost ;dbname=greta;charset=utf8’,’root’, ‘mdp’);
```

- localhost : adresse du serveur mysql ou l’on veut se connecter (domaine ou adresse ip)
- greta : nom de la base de données sur la quelle je veux travailler
- utf8 : charset utilisé
- root : identifiant
- mdp : mot de passe

Partie 3 : Lire des données

```php
/* Connexion au serveur localhost
 * Base de donnée carnet_adresses
 * Utilisateur root
 * Mot de passe root
 */
$db = new PDO('mysql:host=localhost;dbname=carnet_adresses;charset=utf8','root','root');


/* Requette SQL
 * Je selectionne tous les champs de la table contacts
 */
$sql = 'SELECT * FROM contacts';

/* Je stocke le resultat de ma requete dans
 * une variable $return
 */
$return = $db->query($sql);

/* Je boucle sur tous les enregistrements
 * J'affecte chaque enregistrement à la variable $ligne qui est un tableau
 */
while ($ligne = $return->fetch()) {

    // J'affiche le champ nom du tableau $ligne
    echo $ligne['nom'].'<br/>';
}

// Je ferme le cursseur
$return->closeCursor();
```

PhpMyAdmin :

1/ Créer une nouvelle base de données greta. UTF8_general_ci
2/ Créer une table eleves avec les champs : id(int autoincrement), nom (varchar 50), prenom (varchar 50)
3/ Ajouter quelques enregistrements

Php :

Créer un nouveaux fichier et adapter le code ci-contre pour lister les noms et prénoms de chaque élève.

Partie 3 : TP Créer et lire une BDD

Créer une base de données greta avec une table eleves

La table eleves sera composée des champs suivants :

- id (clee primaire + auto incrément)
- nom
- prenom
- genre
- email
- datenaissance
- telephone

Remplir cette table soit en saisie dans phpmyadmin soit en créant un script php permettant de créer des requêtes INSERT a coller directement dans phpmyadmin.

Enfin écrire le code permettant de lire tous les enregistrements de la table eleves.

Partie 3 : sélectionner des données

Quelques exemple de requêtes :

- SELECT * FROM matable WHERE nomduchamp = 1
  Sélectionne tous les champs de la table matable pour lesquels la valeur du champ nomduchamp = 1

- SELECT * FROM matable WHERE nomduchamp = ‘valeurtexte’
  Sélectionne tous les champs de la table matable pour lesquels la valeur du champ nomduchamp = ‘valeurtexte’
  attention lorsqu’onn recherches des valeurs de type texte il faut les entourer de quotes ‘’

- SELECT * FROM matable WHERE nomduchamp LIKE ‘a%’
  Sélectionne tous les champs de la table matable pour lesquels la valeur du champ nomduchamp commence par a

- SELECT * FROM matable WHERE nomduchamp LIKE ‘%chaine%’
  Sélectionne tous les champs de la table matable pour lesquels la valeur du champ nomduchamp contient la chaine

Partie 3 : sélectionner des données

- SELECT * FROM matable WHERE nomduchamp > 1
  Sélectionne tous les champs de la table matable pour lesquels la valeur du champ nomduchamp est supérieur à 1

- SELECT * FROM matable WHERE nomduchamp BETWEEN 1 AND 4
  Sélectionne tous les champs de la table matable pour lesquels la valeur du champ nomduchamp est compris entre 1 et 4

- SELECT * FROM matable LIMIT 10
  Sélectionne tous les champs de la table matable et retourne les 10 premiers enregistrements

- SELECT * FROM matable LIMIT 5 10
  Sélectionne tous les champs de la table matable et retourne les 10 enregistrements suivants depuis les 5 premiers

- SELECT * FROM matable ORDER BY nomduchamp ASC
  Sélectionne tous les champs de la table matable et retourne les enregistrements dans l'odre ASCendent (du plus petit au plus grand)

Partie 3 : sélectionner des données

- SELECT * FROM matable WHERE nomduchamp > 1 AND nomduchamp <4
  Sélectionne tous les champs de la table matable pour lesquels la valeur du champ nomduchamp est supérieur à 1 et inférieur à 4

- SELECT * FROM matable WHERE nomduchamp LIKE ‘a%’ OR nomchamp2 = ‘azerty’
  Sélectionne tous les champs de la table matable pour lesquels la valeur du champ nomduchamp commence par a ou le nomchamp2 = azerty

Partie 3 : ajouter modifier des données

INSERT ajouter des données :

INSERT INTO matable (champ1, champ2, champ3) VALUES ('valeur1','valeur2','valeur3')

Ajoute un enregistrement dans la table matable liste des champs à ajouter, valeurs à assigner dans chaque champ

UPDATE modifier des données :

UPDATE matable SET champ1 = 'valeur1', champ2 = 'valeur2' WHERE id = 1
Modifie l'enregistrement dont id=1 de la table matable et modifie les champs1 et champ2 avec les valeur1 et valeur2

Partie 3 : ajouter modifier des données

DELETE supprimer des données :

DELETE FROM matable WHERE id=10

Efface l'enregistrement dont le champ id = 10 dans la table matable.

ATTENTION : lorsque l'on execute une requete SQL sans attendre de données en retours (UPDATE, INSERT, DELETE) il convient d'utiliser la méthode exec() au lieu de query() qui est réservé à la récupération de données (SELECT)

Partie 3 : TP Gérér une base de contacts

En se basant sur la table **eleves** de la base de données **greta** créer un système permettant :

- la lecture,
- la modification,
- la suppression d'un élève
- ajoute un eleve

Bonus : Pour ceux qui sont à l'aise créer un masque de saisie pour la date de naissance au travers de 3 menus déroulants

Partie 3 : Sécuriser, débugger ses requêtes

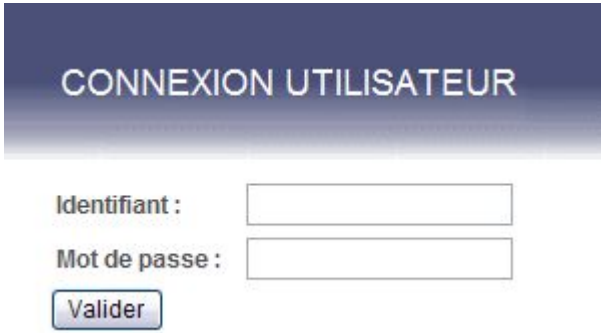
Injection SQL :

Cette faille apparaît quand il est possible d'injecter du code SQL dans les requêtes SQL qui sont faites dans une page web. Ce type de faille est en général facile à reconnaître, puisque si l'on injecte un simple guillemet ‘ dans le formulaire affiche une erreur.

Il est possible par ce moyen de contourner un formulaire d’authentification par exemple

Partie 3 : Sécuriser, débugger ses requêtes

Exemple d'injection SQL :



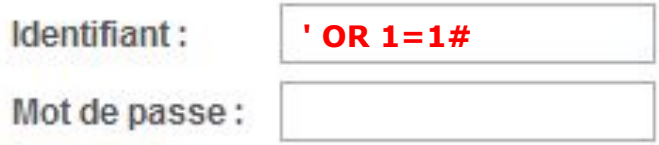

```
<form action="?" method="post">
    <input type="text" name="user" />
    <input type="text" name="pass" />
    <input type="submit" value="Valider" />
</form>
```

Partie 3 : Sécuriser, débugger ses requêtes

```php
// Traitement de l'authentification
if(isset($_POST['user'] && $_POST['pass'] )){

    $user = $_POST['user'];
    $pass = $_POST['pass'];


    $sql = "SELECT *
            FROM utilisateurs
            WHERE userName = '$user'
            AND pass = '$pass'
           ";
}
```



SELECT * FROM utilisateurs WHERE userName = '' OR 1=1 #' AND pass = '$pass'

étant le caractère de commentaire supprime tout ce qui le suit dans la requête ce qui est exécuté

SELECT * FROM utilisateurs WHERE userName = '' OR 1=1 #'

Soit tous les enregistrements de la table

Partie 3 : Sécuriser, débugger ses requêtes

Requêtes préparées :

```php
// Requette SQL Préparé
$sql = "SELECT *
        FROM utilisateurs
        WHERE userName = :user
        AND pass = :pass
       ";
$req = $bdd->prepare($sql);

// Attribution des valeurs
$values = array(
                'user' => $_POST['user'],
                'pass' => $_POST['pass'],
               );
$req->execute($values);
```

Partie 3 : Sécuriser, débugger ses requêtes

Gestion des erreurs :

```php
try{
    // On se connecte à MySQL
    $bdd = new PDO('mysql:host=localhost;dbname=test;charset=utf8', 'root', '');
}catch(Exception $e){
    // En cas d'erreur, on affiche un message et on arrête tout
    die('Erreur : '.$e->getMessage());
}
```

try : partie du code à tester
$e : variable contenant l'exception
catch : code exécuté quand une l'exception est soulevé

Partie 3 : Sécuriser, débugger ses requêtes

Affichage des erreurs MySql:

```php
try{
    // On se connecte à MySQL
    $bdd = new PDO('mysql:host=localhost;dbname=test;charset=utf8', 'root', '',
                   array(PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION));
}catch(Exception $e){
    // En cas d'erreur, on affiche un message et on arrête tout
    die('Erreur : '.$e->getMessage());
}
```

Cette chaîne de connexion permet d'activer un affichage explicite des erreurs SQL

Par exemple : Unknown column 'login' in 'field list'