Projet : Créer et administrer une base de données

Table des matières

I. Contexte

Durée : 60 minutes

Environnement de travail : Visual Studio Code[1]

Prérequis *:*

- Avoir installé Mysql sur sa machine.
- Avoir installé PHP sur sa machine.

Contexte

Dans ce projet, nous allons créer un mini site permettant de se connecter. De la base de données au site web, nous allons tout créer à partir de zéro. L'objectif est de comprendre comment créer une base de données, et surtout comment l'appeler depuis un site en PHP avec un utilisateur dédié.

II. Création de la base de données

A. Introduction

Objectifs

- Apprendre à modéliser une base de données
- Apprendre à se connecter à une base de données depuis PHP
- Apprendre à créer le contenu d'un site

B. Modéliser la base de données

Méthode

Pour modéliser la base de données, nous devons développer un module d'authentification. Donc nous avons besoin d'une table utilisateur. Pour le moment, notre application gérera uniquement l'authentification, donc une seule table nous suffira. Lorsque l'application évoluera, nous pourrons alors faire évoluer la base de données.

Nous obtenons alors ce diagramme de base de données :

DiagramBDD.png

C. Créer la base de données

Méthode

Depuis ce diagramme, nous allons pouvoir créer notre base de données, en exécutant les commandes SQL suivantes :

```sql
-- Création de la base de données
CREATE DATABASE ProjetAuthentificationBDD;

-- Utilisation de la base de données
USE ProjetAuthentificationBDD;

-- Création de la table "users" pour stocker les informations des utilisateurs
CREATE TABLE users (
  idUser integer AUTO_INCREMENT,
```

1. https://code.visualstudio.com/

```
  pseudo varchar(50),
  name varchar(100),
  surname varchar(100),
  email varchar(255) NOT NULL UNIQUE,
  password varchar(255) NOT NULL,
  PRIMARY KEY (idUser)
);
```

D. Créer l'utilisateur de base de données pour le site

Méthode

Lorsque nous créons un site internet qui se connecte à une base de données, nous devons au préalable créer un utilisateur MySQL (si on utilise une base de données MySQL), avec des droits restreints. C'est avec cet utilisateur de base de données que nous allons exécuter les requêtes depuis le PHP.

Dans notre projet, nous allons donc créer cet utilisateur, avec des droits uniquement sur la table users. Ainsi, si les identifiants de cet utilisateur nous échappent, la personne malveillante ne pourra pas modifier d'autres choses que les données de la table Users. Voici le SQL à exécuter pour créer cet utilisateur. Pensez à noter le nom d'utilisateur et le mot de passe, nous allons nous en servir plus tard dans le cours.

```
-- Création de l'utilisateur
CREATE USER 'user_php'@'localhost' IDENTIFIED BY '3f7zhhRn4NH69R';

-- Attribution des droits sur la table "users"
GRANT SELECT, INSERT, UPDATE, DELETE ON ProjetAuthentificationBDD.users TO
'user_php'@'localhost';
```

E. Créer des utilisateurs de base de données pour le site

Méthode

Pour pouvoir faire quelques tests, nous pouvons ajouter différents utilisateurs dans la table users. Voici le script :

```
-- Insertion des utilisateurs en BDD
--Le mot de passe de tous les utilisateurs est password123
INSERT INTO users (pseudo, name, surname, email, password)
VALUES ('john_doe', 'John', 'Doe', 'john.doe@example.com',
'$2y$10$hv2m6oFnpMs6sZmpyNK1r.iWEJO/CU96h7b95VjYCC5Msw.lGdn8G');

INSERT INTO users (pseudo, name, surname, email, password)
VALUES ('jane_smith', 'Jane', 'Smith', 'jane.smith@example.com',
'$2y$10$hv2m6oFnpMs6sZmpyNK1r.iWEJO/CU96h7b95VjYCC5Msw.lGdn8G');

INSERT INTO users (pseudo, name, surname, email, password)
VALUES ('test_user', 'Test', 'User', 'test@example.com',
'$2y$10$hv2m6oFnpMs6sZmpyNK1r.iWEJO/CU96h7b95VjYCC5Msw.lGdn8G');
```

III. Création du site en PHP

A. Introduction

Objectifs

- Tester la connexion à la base de données depuis PHP
- Créer un module de connexion
- Créer un module d'inscription

B. Tester la connexion à la base de données depuis PHP

Méthode

Pour vérifier que la connexion avec notre utilisateur fonctionne, nous pouvons créer un fichier test.php, qui récupère les utilisateurs et les affiche. Si nous voyons les utilisateurs, la connexion est valide, sinon le message d'erreur sera affiché sur la page.

```php
<?php
$dsn = 'mysql:host=localhost;dbname=ProjetAuthentificationBDD';
$username = 'user_php';
$password = '3f7zhhRn4NH69R';

try{
    $pdo = new PDO($dsn, $username, $password);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    //Récupérer les utilisateurs
    $query = "SELECT * FROM users";
    $stmt = $pdo->query($query);
    $users = $stmt->fetchAll(PDO::FETCH_ASSOC);

    //Afficher les utilisateurs
    foreach($users as $user){
        echo "ID : " . $user['idUser'] . "<br>";
        echo "Nom : " . $user['name'] . "<br>";
        echo "Prenom : " . $user['surname'] . "<br>";
        echo "email : " . $user['email'] . "<br>";
        echo "<br>";
    }
}
catch (PDOException $e){
    echo "Erreur de connexion à la base de données : ". $e->getMessage();
}

?>
```

C. Module de connexion

Méthode **Créer la vue en HTML**

Maintenant que notre base de données est prête et que nous savons nous connecter notre code PHP à la base de données, nous allons pouvoir créer un formulaire de connexion. Pour commencer, nous allons donc créer le formulaire de connexion en HTML. Ce formulaire de connexion nous redirigera vers une page PHP qui vérifiera les données que nous avons entrées dans le formulaire.

Voici le formulaire HTML :

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Connexion</title>
</head>
<body>
    <h1>Connexion</h1>
    <form action="loginPost.php" method="POST">
        <!-- EMAIL -->
```

```
        <label for="email">Adresse email :</label>
        <input type="email" name="email" required /> <br><br>

        <!-- PASSWORD -->
        <label for="password">Mot de passe :</label>
        <input type="password" name="password" required /> <br><br>

        <!-- BUTTON -->
        <input type="submit" value="Se connecter"/>
    </form>
</body>
</html>
```

Méthode

Lorsque nous validons le formulaire, nous sommes redirigés vers la page login.php. Depuis cette page, nous pouvons récupérer les informations du formulaire grâce à la variable super globale $_POST.

L'objectif est donc de récupérer le mail de l'utilisateur, de récupérer son profil en base de données et de vérifier si le mot de passe est bien le même qu'en base de données. Le mot de passe est encrypté en base de données, donc il faut encrypter la proposition de l'utilisateur pour comparer le mot de passe encrypté en Base de données avec le mot de passe du formulaire encrypté. Si les deux sont égaux, nous pourrions afficher un message pour dire que la connexion a fonctionné. Voici le code de cette page :

```php
<?php

$dsn = 'mysql:host=localhost;dbname=ProjetAuthentificationBDD';
$username = 'user_php';
$password = '3f7zhhRn4NH69R';

try{
    $pdo = new PDO($dsn, $username, $password);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    //Récupérer les données du formulaire de connexion
    $emailForm = $_POST['email'];
    $passwordForm = $_POST['password'];

    //Récupérer les utilisateurs
    $query = "SELECT * FROM users WHERE email = :email";
    $stmt = $pdo->prepare($query);
    $stmt->bindParam(':email', $emailForm);
    $stmt->execute();

    //Est-ce que l'utilisateur (mail) existe ?
    if($stmt->rowCount() == 1){
        $monUser = $stmt->fetch(PDO::FETCH_ASSOC);
        if(password_verify($passwordForm, $monUser['password'])){
            echo "Connexion réussie ! Bienvenue " . $monUser['name'] . $monUser['surname'];
        }else{
            echo "Mot de passe incorrect";
        }
    }
    else{
        echo "Utilisateur introuvable, êtes-vous sûr de votre mail ?";
    }
}
catch (PDOException $e){
```

```php
    echo "Erreur de connexion à la base de données : ". $e->getMessage();
}

?>
```

D. Module d'inscription

Méthode **Créer la vue en HTML**

Nous voulons maintenant coder le module d'inscription, nous pouvons donc commencer par le formulaire d'inscription.

```html
<!DOCTYPE html>
<html>
<head>
  <title>Inscription</title>
</head>
<body>
  <h2>Inscription</h2>
  <form action="register.php" method="POST">
  <label for="name">Nom :</label>
  <input type="text" name="name" required><br><br>
  <label for="surname">Prenom :</label>
  <input type="text" name="surname" required><br><br>
  <label for="pseudo">Pseudo :</label>
  <input type="text" name="pseudo" required><br><br>
  <label for="email">Adresse email :</label>
  <input type="email" name="email" required><br><br>
  <label for="password">Mot de passe :</label>
  <input type="password" name="password" required><br><br>
  <input type="submit" value="S'inscrire">
</form>
</body>
</html>
```

Méthode

Lorsque nous validons le formulaire, nous sommes redirigés vers la page register.php. Depuis cette page, nous pouvons récupérer les informations du formulaire grâce à la variable superglobale $_POST, tout comme lors de la connexion.

Nous voulons alors suivre différentes étapes. D'abord, nous voulons vérifier si l'adresse email est déjà présente en base de données. Si elle n'est pas présente, alors nous voulons encrypter le mot de passe et enregistrer toutes les informations utilisateur en base de données, grâce à la commande SQL `'INSERT INTO'`.

```php
<?php

$dsn = 'mysql:host=localhost;dbname=ProjetAuthentificationBDD';
$username = 'user_php';
$password = '3f7zhhRn4NH69R';

try{
    $pdo = new PDO($dsn, $username, $password);
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

    //Récupérer les données du formulaire d'inscription
    $nameForm = $_POST['name'];
    $surnameForm = $_POST['surname'];
```

```php
    $pseudoForm = $_POST['pseudo'];
    $emailForm = $_POST['email'];
    $passwordForm = $_POST['password'];

    //Vérifier l'unicité de l'adresse mail
    $query = "SELECT * FROM users WHERE email = :email";
    $stmt = $pdo->prepare($query);
    $stmt->bindParam(':email', $emailForm);
    $stmt->execute();

    //Est-ce que l'utilisateur (mail) existe ?
    if($stmt->rowCount() > 0){
        die("Cette adresse email est déjà utilisée");
    }

    // Hashage(encryptage)
    $hashedPassword = password_hash($passwordForm, PASSWORD_DEFAULT);

    //Insérer les données dans la base
    $insertQuery = "INSERT INTO users (pseudo, name, surname, email, password) VALUES (:pseudo, :name, :surname, :email, :password)";
    $stmt = $pdo->prepare($insertQuery);
    $stmt->bindParam(':pseudo', $pseudoForm);
    $stmt->bindParam(':name', $nameForm);
    $stmt->bindParam(':surname', $surnameForm);
    $stmt->bindParam(':email', $emailForm);
    $stmt->bindParam(':password', $hashedPassword);
    $stmt->execute();

    echo "Inscription réussie";

}
catch (PDOException $e){
    echo "Erreur lors de l'inscription : ". $e->getMessage();
}

?>
```

IV. L'essentiel

Nous avons pu créer une application qui communique avec une base de données. Nous avons vu une notion fondamentale : la création d'un utilisateur MySql. Certains développeurs se connectent depuis leurs sites vers la base de données avec un utilisateur admin, ce qui est très dangereux. Nous avons pu voir comment fonctionne la connexion PDO, nous permettant en PHP de communiquer avec notre base de données. Cet exemple simpliste nous a permis de voir à petite échelle les premiers moments de développement d'une application en PHP. On commence par la construction de la base de données, on essaye de s'y connecter pour vérifier que tout fonctionne et on implémente des fonctionnalités.

V. Pour aller plus loin

Pour aller plus loin, vous pouvez ajouter différentes fonctionnalités à votre site. Vous pourriez développer certaines fonctionnalités ou en ajouter des nouvelles. Pour vous aider, voici quelques idées :

- **Gérer la connexion et la déconnexion en session**

  Vous pouvez utiliser $_SESSION (php[2]) pour stocker les données utilisateurs en session pour gérer une session utilisateur, avec connexion et déconnexion : au final, ce site est uniquement un module de vérification de données de connexion, sans gestion de session.

- **Ajouter du style au site**

  Vous pouvez ajouter du CSS pour rendre le site plus joli.

- **Faire un header de site**

  Si vous faites un header de site avec le module d'authentification dans le menu, vous serez prêt à avoir un vrai site.

2. https://www.php.net/manual/fr/reserved.variables.session.php