TP : Introduction à JDBC avec MySQL

Objectifs pédagogiques

À la fin de ce TP, vous serez capable de :
- Comprendre le rôle de JDBC (Java Database Connectivity).
- Établir une connexion entre une application Java et une base de données MySQL.
- Exécuter des requêtes SQL (SELECT, INSERT, UPDATE, DELETE) depuis Java.
- Gérer les résultats et les exceptions.

1. Pré-requis

- Java (JDK 8 ou supérieur) installé.
- MySQL installé et en fonctionnement.
- Le connecteur JDBC pour MySQL (fichier `.jar` nommé souvent `mysql-connector-j-x.x.xx.jar`).
Placez ce fichier dans le classpath de votre projet ou dans le dossier lib/ de votre IDE.

2. Création de la base de données

Connectez-vous à MySQL et exécutez :

```
CREATE DATABASE jdbc_auteur;
USE jdbc_auteur;

CREATE TABLE auteur (
   id INT AUTO_INCREMENT PRIMARY KEY,
   nom VARCHAR(50),
   prenom VARCHAR(50),

   adresse VARCHAR(50),
   age INT
);

INSERT INTO auteur (nom, prenom,adresse, age)
VALUES
('Ali', 'Mohamed', 'quartier6', 22),
('Hassan', 'Abdi', 'balbala 5',21),
('Fatouma', 'Ahmed','quartier7', 23);
```

Nb: Faites les modifications des classes java en conséquence.

3. Programme Java pour la connexion JDBC

Fichier : ConnexionMySQL.java

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class ConnexionMySQL {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/jdbc_auteur";
        String utilisateur = "root";
        String motDePasse = ""; // ou votre mot de passe

        try {
            Connection connexion = DriverManager.getConnection(url, utilisateur,
motDePasse);
            System.out.println("Connexion réussie à la base de données !");
            connexion.close();
        } catch (SQLException e) {
            System.out.println("Erreur de connexion : " + e.getMessage());
        }
    }
}
```

4. Lire les données (SELECT)

```
import java.sql.*;

public class AfficherAuteurs {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/jdbc_auteur";
        String utilisateur = "root";
        String motDePasse = "";

        try (Connection connexion = DriverManager.getConnection(url, utilisateur,
motDePasse);
             Statement stmt = connexion.createStatement();
             ResultSet rs = stmt.executeQuery("SELECT * FROM auteur")) {

            while (rs.next()) {
```

```
            System.out.println(
               rs.getInt("id") + " - " +
               rs.getString("nom") + " " +
               rs.getString("prenom") +
               rs.getString("adresse") +", âge : " + rs.getInt("age")
            );
         }

      } catch (SQLException e) {
         e.printStackTrace();
      }
   }
}
```

5. Insérer des données (INSERT)

```
import java.sql.*;
import java.util.Scanner;

public class AjouterAuteur{
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      System.out.print("Nom : ");
      String nom = sc.nextLine();
      System.out.print("Prénom : ");
      String prenom = sc.nextLine();
      System.out.print("Adresse : ");
      String prenom = sc.nextLine();
      System.out.print("Âge : ");
      int age = sc.nextInt();

      String url = "jdbc:mysql://localhost:3306/jdbc_auteur";
      String utilisateur = "root";
      String motDePasse = "";

      String sql = "INSERT INTO auteur (nom, prenom, age) VALUES (?, ?, ?)";

      try (Connection connexion = DriverManager.getConnection(url, utilisateur,
motDePasse);
          PreparedStatement pstmt = connexion.prepareStatement(sql)) {
         pstmt.setString(1, nom);
         pstmt.setString(2, prenom);
```

```
            pstmt.setString(3, adresse);
            pstmt.setInt(4, age);
            pstmt.executeUpdate();

            System.out.println("auteur ajouté avec succès !");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

6. Mettre à jour et supprimer (UPDATE / DELETE)

```
// Exemple de mise à jour
String sql = "UPDATE auteur SET age = ? WHERE id = ?";
PreparedStatement pstmt = connexion.prepareStatement(sql);
pstmt.setInt(1, 25);
pstmt.setInt(2, 1);
pstmt.executeUpdate();

// Exemple de suppression
String sql = "DELETE FROM auteur WHERE id = ?";
PreparedStatement pstmt = connexion.prepareStatement(sql);
pstmt.setInt(1, 2);
pstmt.executeUpdate();
```

7. Exercices proposés

1. Modifier le programme pour afficher uniquement les auteurs de plus de 22 ans.
2. Créer une méthode ajouterauteur() dans une classe Gestionauteur.
3. Ajouter une méthode rechercherauteurParNom(String nom).
4. Créer un menu interactif permettant à l'utilisateur de :
   - Afficher tous les auteurs
   - Ajouter un auteur
   - Mettre à jour un auteur
   - Supprimer un auteur
   - Quitter

8. À retenir

- JDBC est une API Java pour interagir avec des bases de données relationnelles.
- Le DriverManager permet d'obtenir une connexion.
- Les classes Statement et PreparedStatement permettent d'exécuter des requêtes SQL.
- Toujours fermer la connexion à la fin (try-with-resources recommandé).