

Connexion base de données en JAVA

Introduction et concepts de base

- **JDBC** (Java Database Connectivity) est une API Java standardisée qui permet de se connecter à une base de données, d'exécuter des requêtes et d'extraire des résultats.
- **Pilote JDBC** : Chaque type de base de données (MySQL, PostgreSQL, Oracle, etc.) nécessite un pilote JDBC spécifique.

2. Installation et configuration des outils nécessaires

Outils requis :

1. **JDK** (Java Development Kit) - Pour écrire et exécuter le code Java.
2. **IDE** (IntelliJ IDEA, Eclipse, NetBeans, etc.) ou un éditeur de texte (VS Code).
3. **Serveur de base de données** (MySQL, PostgreSQL, Oracle, etc.).
4. **Pilote JDBC** - Téléchargez le fichier JAR correspondant à votre SGBD (par exemple, mysql-connector-java.jar pour MySQL).

Configuration :

- Installez le serveur de base de données et créez une base. Exemple pour MySQL :

```
CREATE DATABASE exemple_java;  
USE exemple_java;  
CREATE TABLE utilisateurs (  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    nom VARCHAR(50),  
    email VARCHAR(100)  
);  
INSERT INTO utilisateurs (nom, email) VALUES ('Alice', 'alice@example.com'), ('Bob', 'bob@example.com');
```

3. Connexion à la base de données (JDBC)

Étape 1 : Importer le pilote JDBC

Téléchargez le fichier JAR du pilote JDBC et ajoutez-le au projet.

Étape 2 : Code de connexion

Exemple pour MySQL :

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

public class ConnexionBD {

    public static void main(String[] args) {

        String url = "jdbc:mysql://localhost:3306/exemple_java";

        String utilisateur = "root"; // Votre nom d'utilisateur MySQL

        String motDePasse = "votre_mot_de_passe"; // Votre mot de passe MySQL

        try {

            Connection connexion = DriverManager.getConnection(url, utilisateur, motDePasse);

            System.out.println("Connexion établie avec succès !");

            connexion.close();

        } catch (SQLException e) {

            System.err.println("Erreur de connexion : " + e.getMessage());

        }

    }

}
```

4. Exécution de requêtes SQL

Étape 1 : Importer les classes nécessaires

Ajoutez les classes nécessaires pour exécuter des requêtes : Statement et ResultSet.

Étape 2 : Exemple de requêtes

```
import java.sql.*;

public class RequetesSQL {
```

```
public static void main(String[] args) {

    String url = "jdbc:mysql://localhost:3306/exemple_java";

    String utilisateur = "root";

    String motDePasse = "votre_mot_de_passe";


    try (Connection connexion = DriverManager.getConnection(url, utilisateur,
motDePasse)) {

        Statement statement = connexion.createStatement();


        // Lecture des données

        ResultSet resultSet = statement.executeQuery("SELECT * FROM utilisateurs");
        while (resultSet.next()) {

            System.out.println("ID: " + resultSet.getInt("id"));

            System.out.println("Nom: " + resultSet.getString("nom"));

            System.out.println("Email: " + resultSet.getString("email"));

            System.out.println("---");

        }


        // Insertion de données

        String insertionSQL = "INSERT INTO utilisateurs (nom, email) VALUES ('Charlie',
'charlie@example.com')";

        int lignesAffectees = statement.executeUpdate(insertionSQL);

        System.out.println("Lignes insérées : " + lignesAffectees);


    } catch (SQLException e) {

        e.printStackTrace();

    }

}
```

5. Gestion des exceptions et bonnes pratiques

1. **Utilisez des blocs try-with-resources** : Cela garantit que les ressources comme Connection, Statement, et ResultSet sont fermées automatiquement.
2. **Utilisez des requêtes paramétrées avec PreparedStatement** pour éviter les injections SQL :

```
String requete = "SELECT * FROM utilisateurs WHERE email = ?";  
PreparedStatement preparedStatement = connexion.prepareStatement(requete);  
preparedStatement.setString(1, "alice@example.com");  
ResultSet rs = preparedStatement.executeQuery();
```

3. **Journalisation** : Loggez les erreurs pour faciliter le débogage.

6. Exemples et exercices pratiques

Exercice 1 :

- Créez une table pour gérer des produits (ID, nom, prix).
- Implémentez un programme Java pour insérer des produits et afficher leur liste.

Exercice 2 :

- Créez une fonctionnalité pour mettre à jour un produit en fonction de son ID.

Exercice 3 :

- Implémentez une méthode pour rechercher des utilisateurs par leur nom.

Solution

1. Création de la table SQL :

```
CREATE TABLE produits (  
    id INT PRIMARY KEY AUTO_INCREMENT,  
    nom VARCHAR(100),  
    prix DECIMAL(10, 2)  
);  
  
INSERT INTO produits (nom, prix) VALUES ('Produit A', 100.50), ('Produit B', 250.00);
```

2. Code Java pour insérer et afficher des produits :

```
import java.sql.*;

public class GestionProduits {

    public static void main(String[] args) {

        String url = "jdbc:mysql://localhost:3306/exemple_java";

        String utilisateur = "root";

        String motDePasse = "votre_mot_de_passe";

        try (Connection connexion = DriverManager.getConnection(url, utilisateur,
motDePasse)) {

            // Insertion de produits

            String insertionSQL = "INSERT INTO produits (nom, prix) VALUES (?, ?)";

            PreparedStatement preparedStatement =
connexion.prepareStatement(insertionSQL);

            preparedStatement.setString(1, "Produit C");

            preparedStatement.setDouble(2, 150.75);

            int lignesInserees = preparedStatement.executeUpdate();

            System.out.println("Lignes insérées : " + lignesInserees);

            // Affichage des produits

            String lectureSQL = "SELECT * FROM produits";

            Statement statement = connexion.createStatement();

            ResultSet resultSet = statement.executeQuery(lectureSQL);

            System.out.println("Liste des produits :");

            while (resultSet.next()) {

                System.out.println("ID: " + resultSet.getInt("id"));

                System.out.println("Nom: " + resultSet.getString("nom"));

                System.out.println("Prix: " + resultSet.getDouble("prix"));

                System.out.println("---");

            }

        }

    }

}
```

```
}

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

Exercice 2 : Mise à jour d'un produit

Énoncé

Créer une fonctionnalité pour mettre à jour un produit en fonction de son ID.

Solution

1. **SQL de base** : Aucun changement à faire à la base. Les produits sont déjà dans la table.
2. **Code Java pour la mise à jour** :

```
import java.sql.*;

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

        try (Connection connexion = DriverManager.getConnection(url, utilisateur,
motDePasse)) {
            // Mise à jour du produit

            String miseAJourSQL = "UPDATE produits SET prix = ? WHERE id = ?";

            PreparedStatement preparedStatement =
connexion.prepareStatement(miseAJourSQL);

            preparedStatement.setDouble(1, 300.00); // Nouveau prix
```

```
preparedStatement.setInt(2, 2); // ID du produit à mettre à jour

int lignesModifiees = preparedStatement.executeUpdate();

System.out.println("Lignes mises à jour : " + lignesModifiees);

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

Exercice 3 : Recherche d'utilisateurs

Énoncé

Implémenter une méthode pour rechercher des utilisateurs par leur nom.

Solution

1. **SQL de base** : Aucun changement à faire à la base. Les utilisateurs sont déjà dans la table.
2. **Code Java pour la recherche** :

```
import java.sql.*;

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

        try (Connection connexion = DriverManager.getConnection(url, utilisateur,
motDePasse)) {
            // Recherche par nom

            String rechercheSQL = "SELECT * FROM utilisateurs WHERE nom = ?";
```

```
PreparedStatement preparedStatement =
connexion.prepareStatement(rechercheSQL);

preparedStatement.setString(1, "Alice");

ResultSet resultSet = preparedStatement.executeQuery();


System.out.println("Résultat de la recherche :");
if (resultSet.next()) {
    System.out.println("ID: " + resultSet.getInt("id"));
    System.out.println("Nom: " + resultSet.getString("nom"));
    System.out.println("Email: " + resultSet.getString("email"));
} else {
    System.out.println("Aucun utilisateur trouvé.");
}

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

}
```