

TUTORIEL : Exécuter des requêtes SQL en Java avec Eclipse

Table des matières

	TUTORIEL : Exécuter des requêtes SQL en Java avec Eclipse (WAMP).....	1
	PRÉREQUIS.....	2
	1) Vérifier / créer la base de données.....	3
	2) Créer un projet Java dans Eclipse.....	4
	3) Code complet et commenté.....	5
	4) Explication simple du fonctionnement.....	8
	Résultat attendu dans la console.....	8

PRÉREQUIS

Avant de commencer ce tutoriel et d'aller plus bas, il faut avoir :

- **WAMP installé et lancé** (icône verte).
- **Base MySQL présente dans phpMyAdmin** (ou prête à être créée).
- **Java JDK installé** (au moins Java 8) :
 [Télécharger Java JDK](#)
- **Eclipse installé** :
 [Télécharger Eclipse](#)
- **MySQL Connector/J** (fichier `mysql-connector-j-8.x.x.jar`) téléchargé :
 [Télécharger Connector/J](#)

⚠ Sur **Windows** → Choisir **Platform Independent (ZIP)**.

➡ Dézipper → récupérer le fichier `mysql-connector-j-8.x.x.jar`.

Dans Eclipse :

- Clic droit sur le projet > **Build Path** > **Configure Build Path**
 - Onglet **Libraries** > **Add External JARs...**
 - Ajouter `mysql-connector-j-8.x.x.jar`
 - Appliquer et fermer → le driver est bien ajouté au projet ☒
-

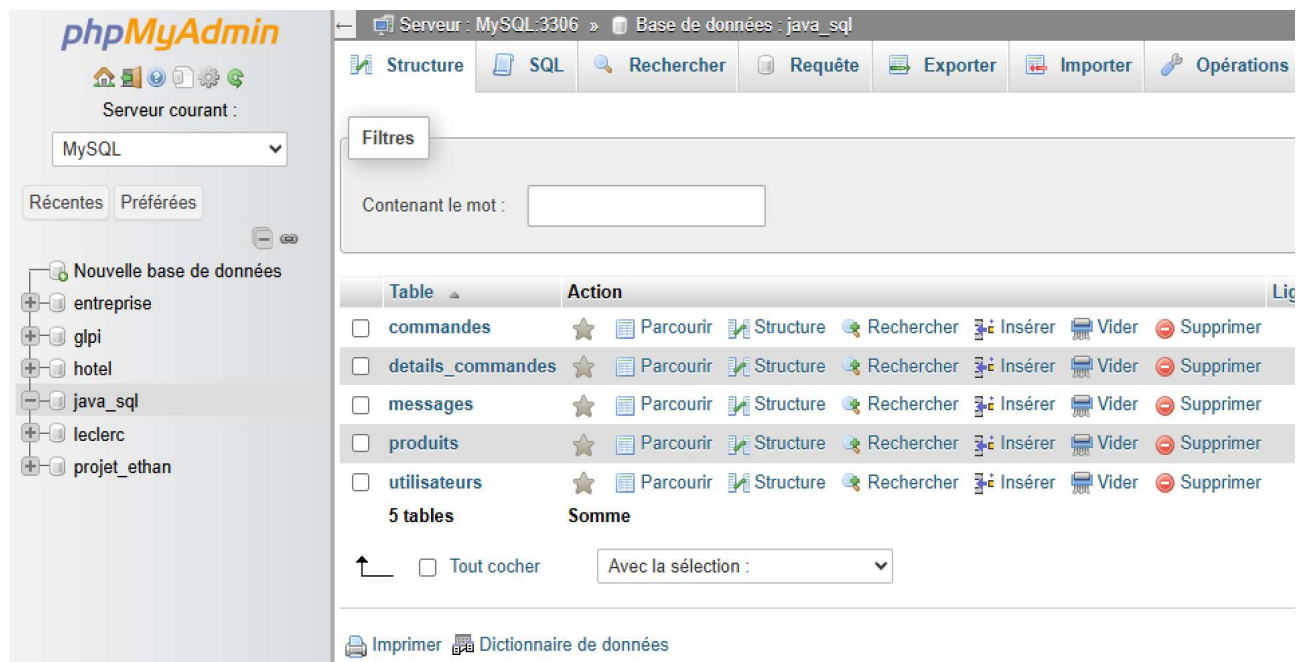
1) Vérifier / créer la base de données

On crée une base simple pour les tests :

```
CREATE DATABASE IF NOT EXISTS java_sql;  
USE java_sql;
```

```
CREATE TABLE IF NOT EXISTS utilisateurs (  
  id INT AUTO_INCREMENT PRIMARY KEY,  
  nom VARCHAR(100) NOT NULL,  
  email VARCHAR(150) UNIQUE NOT NULL,  
  mot_de_passe VARCHAR(255) NOT NULL,  
  date_creation TIMESTAMP DEFAULT CURRENT_TIMESTAMP  
);
```

```
-- Exemple de données  
INSERT INTO utilisateurs (nom, email, mot_de_passe) VALUES  
( 'Alice Dupont', 'alice@example.com', 'pass123'),  
( 'Bob Martin', 'bob@example.com', 'pass123');
```



The screenshot shows the phpMyAdmin interface. On the left, the database 'java_sql' is selected. The main panel displays the 'utilisateurs' table with 5 tables listed. The table structure is as follows:

Table	Action
☐ commandes	★ Parcourir Structure Rechercher Insérer Vider Supprimer
☐ details_commandes	★ Parcourir Structure Rechercher Insérer Vider Supprimer
☐ messages	★ Parcourir Structure Rechercher Insérer Vider Supprimer
☐ produits	★ Parcourir Structure Rechercher Insérer Vider Supprimer
☐ utilisateurs	★ Parcourir Structure Rechercher Insérer Vider Supprimer

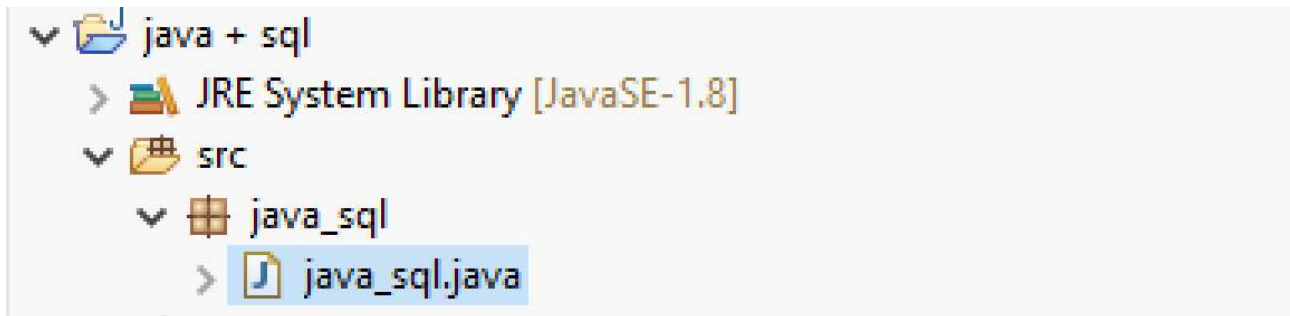
5 tables Somme

↑ ☐ Tout cocher Avec la sélection : ▼

Imprimer Dictionnaire de données

2) Créer un projet Java dans Eclipse

1. Ouvrir Eclipse → **File** > **New** > **Java Project** → nom : Ce que vous voulez.
2. Clic droit sur src → **New** > **Package** → votre_package.
3. Clic droit sur mon_package → **New** > **Class** → VotreClasse.





3) Code complet et commenté

Voici le code Java avec beaucoup de commentaires explicatifs :

```

package java_sql; // Le package du projet (mettez le vôtre)

import java.sql.*; // Importation des classes SQL (Connection, Statement, ResultSet, etc.)

// Classe principale
public class java_sql {

    // --- CONFIGURATION DE LA BDD ---
    // URL JDBC : indique le SGBD (MySQL), l'hôte (localhost), le port (3306) et la base (java_sql)
    private static final String URL = "jdbc:mysql://localhost:3306/java_sql?useSSL=false&serverTimezone=UTC";
    // Nom d'utilisateur MySQL (par défaut sous WAMP c'est root)
    private static final String USER = "root";
    // Mot de passe MySQL (par défaut sous WAMP il est vide "")
    private static final String PASSWORD = "";

    // === MÉTHODE DE CONNEXION ===
    public static Connection getConnection() throws SQLException {
        // DriverManager.getConnection utilise l'URL + user + password pour se connecter à MySQL
        return DriverManager.getConnection(URL, USER, PASSWORD);
    }

    // === SELECT : Afficher tous les utilisateurs ===
    public static void printAllUsers() {
        String sql = "SELECT id, nom, email, date_creation FROM utilisateurs ORDER BY id";

        // try-with-resources : ferme automatiquement la connexion après utilisation
        try (Connection conn = getConnection();
            Statement stmt = conn.createStatement(); // Statement = permet d'envoyer une requête
            ResultSet rs = stmt.executeQuery(sql)) { // ResultSet = contient les résultats du SELECT

            // Affichage de l'entête
            System.out.printf("%-4s %-20s %-30s %-20s\n", "ID", "NOM", "EMAIL", "DATE_CREATION");

            // Boucle sur chaque ligne du résultat
            while (rs.next()) {
                int id = rs.getInt("id"); // récupère la colonne "id"
                String nom = rs.getString("nom"); // récupère la colonne "nom"
                String email = rs.getString("email"); // récupère la colonne "email"
                Timestamp date = rs.getTimestamp("date_creation"); // récupère la date

                System.out.printf("%-4d %-20s %-30s %-20s\n", id, nom, email, date);
            }
        } catch (SQLException e) {
            System.err.println("X Erreur SELECT : " + e.getMessage());
        }
    }

    // === INSERT : Ajouter un utilisateur ===
    public static void insertUser(String nom, String email, String mdp) {
        String sql = "INSERT INTO utilisateurs (nom, email, mot_de_passe) VALUES (?, ?, ?)";

        try (Connection conn = getConnection();
            PreparedStatement ps = conn.prepareStatement(sql)) {

            // Remplace les ? par les vraies valeurs
            ps.setString(1, nom);
            ps.setString(2, email);
            ps.setString(3, mdp);

            int lignes = ps.executeUpdate(); // Exécute l'INSERT
            System.out.println("■ " + lignes + " utilisateur ajouté");
        } catch (SQLException e) {
            System.err.println("X Erreur INSERT : " + e.getMessage());
        }
    }

    // === UPDATE : Modifier le nom d'un utilisateur via son email ===
    public static void updateUser(String email, String nouveauNom) {
        String sql = "UPDATE utilisateurs SET nom = ? WHERE email = ?";

        try (Connection conn = getConnection();
            PreparedStatement ps = conn.prepareStatement(sql)) {

            ps.setString(1, nouveauNom);
            ps.setString(2, email);

            int lignes = ps.executeUpdate(); // Exécute l'UPDATE
            System.out.println("■ " + lignes + " utilisateur modifié");
        } catch (SQLException e) {
            System.err.println("X Erreur UPDATE : " + e.getMessage());
        }
    }
}

```

```

87 // === DELETE : Supprimer un utilisateur via son email ===
88 public static void deleteUser(String email) {
89     String sql = "DELETE FROM utilisateurs WHERE email = ?";
90
91     try (Connection conn = getConnection();
92         PreparedStatement ps = conn.prepareStatement(sql)) {
93
94         ps.setString(1, email);
95
96         int lignes = ps.executeUpdate(); // Exécute le DELETE
97         System.out.println("■ " + lignes + " utilisateur supprimé");
98
99     } catch (SQLException e) {
100         System.err.println("X Erreur DELETE : " + e.getMessage());
101     }
102 }
103
104 // === MAIN : Programme de test ===
105 public static void main(String[] args) {
106     System.out.println("=== Liste initiale ===");
107     printAllUsers();
108
109     System.out.println("\n=== Ajout d'un utilisateur ===");
110     insertUser("Utilisateur Test", "testjava@example.com", "pass123");
111
112     System.out.println("\n=== Liste après INSERT ===");
113     printAllUsers();
114
115     System.out.println("\n=== Mise à jour ===");
116     updateUser("testjava@example.com", "Utilisateur Modifié");
117
118     System.out.println("\n=== Liste après UPDATE ===");
119     printAllUsers();
120
121     System.out.println("\n=== Suppression ===");
122     deleteUser("testjava@example.com");
123
124     System.out.println("\n=== Liste finale ===");
125     printAllUsers();
126 }
127 }
128

```



4) Explication simple du fonctionnement

1. **Connexion** → `getConnection()` se connecte à ta base `java_sql`.
 2. **SELECT** → `printAllUsers()` affiche tous les utilisateurs.
 3. **INSERT** → `insertUser(...)` ajoute un utilisateur.
 4. **UPDATE** → `updateUser(...)` modifie le nom d'un utilisateur.
 5. **DELETE** → `deleteUser(...)` supprime un utilisateur.
 6. **main()** → exécute ces méthodes dans l'ordre pour montrer le CRUD complet.
-



Résultat attendu dans la console

=== Liste initiale ===

ID	NOM	EMAIL	DATE_CREATION
1	Alice Dupont	alice@example.com	2025-10-01 14:32:10
2	Bob Martin	bob@example.com	2025-10-01 14:32:10

=== Ajout d'un utilisateur ===

✓ 1 utilisateur ajouté

=== Liste après INSERT ===

ID	NOM	EMAIL	DATE_CREATION
1	Alice Dupont	alice@example.com	...
2	Bob Martin	bob@example.com	...
3	Utilisateur Test	testjava@example.com	...

=== Mise à jour ===

✓ 1 utilisateur modifié

=== Liste après UPDATE ===

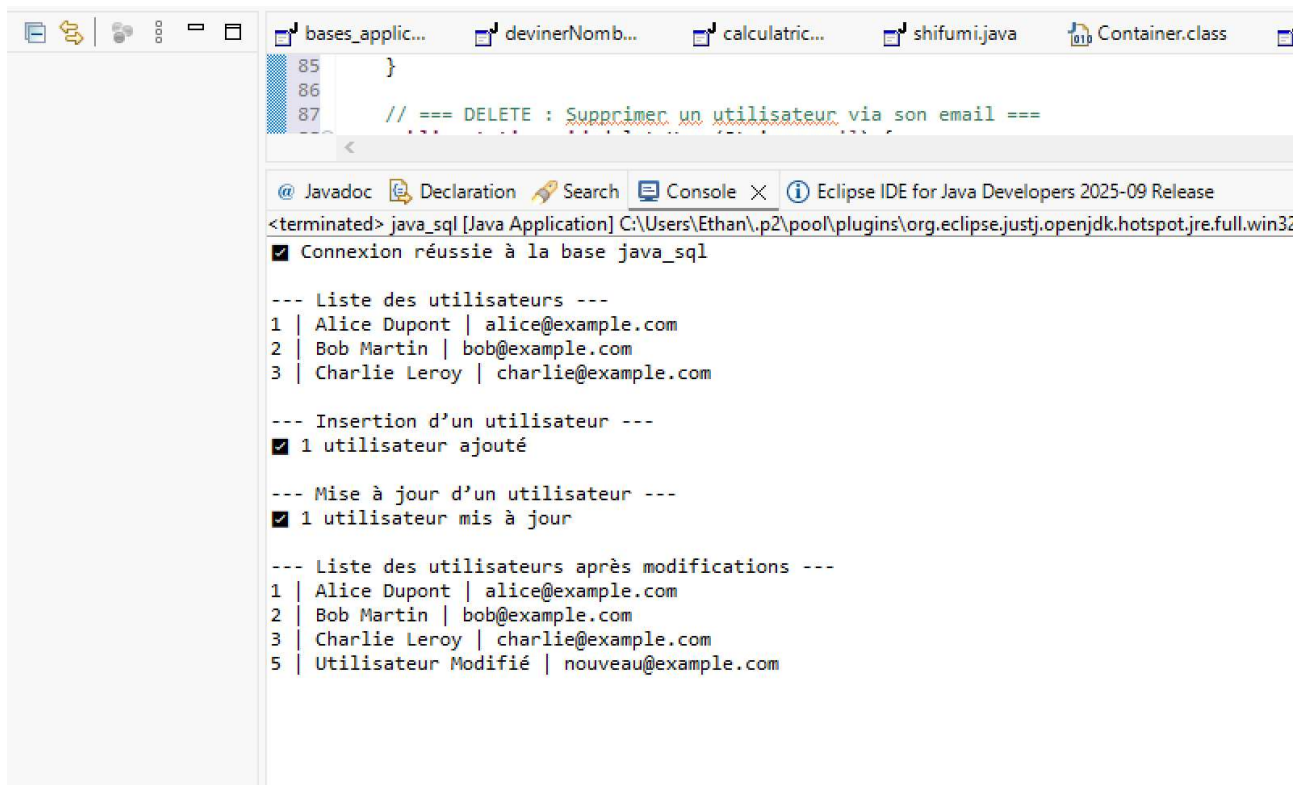
ID	NOM	EMAIL	DATE_CREATION
1	Alice Dupont	alice@example.com	...
2	Bob Martin	bob@example.com	...
3	Utilisateur Modifié	testjava@example.com	...

=== Suppression ===

✓ 1 utilisateur supprimé

=== Liste finale ===

ID	NOM	EMAIL	DATE_CREATION
1	Alice Dupont	alice@example.com	...
2	Bob Martin	bob@example.com	...



The screenshot shows the Eclipse IDE interface. The top part displays a Java code editor with a file named `Container.class`. The code includes a comment: `// === DELETE : Supprimer un utilisateur via son email ===`. Below the code editor, the console window is active, showing the output of a Java application named `java_sql`. The console output includes a successful database connection message, followed by three sections of user data: a list of existing users, a confirmation of a new user being added, and a confirmation of an existing user being updated. The final list shows the updated user information.

```
85 }
86
87 // === DELETE : Supprimer un utilisateur via son email ===

@ Javadoc Declaration Search Console X Eclipse IDE for Java Developers 2025-09 Release
<terminated> java_sql [Java Application] C:\Users\Ethan\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32
[checked] Connexion réussie à la base java_sql

--- Liste des utilisateurs ---
1 | Alice Dupont | alice@example.com
2 | Bob Martin | bob@example.com
3 | Charlie Leroy | charlie@example.com

--- Insertion d'un utilisateur ---
[checked] 1 utilisateur ajouté

--- Mise à jour d'un utilisateur ---
[checked] 1 utilisateur mis à jour

--- Liste des utilisateurs après modifications ---
1 | Alice Dupont | alice@example.com
2 | Bob Martin | bob@example.com
3 | Charlie Leroy | charlie@example.com
5 | Utilisateur Modifié | nouveau@example.com
```

C'est bon, vous savez maintenant comment faire des requêtes SQL en java sur Eclipse !

Fin