使用 JDBC (Java 資料庫連接) 驅動程式

要在 Java 中使用 JDBC（Java Database Connectivity）驅動程序，您需要遵循一系列步驟來建立與資料庫的連接、執行查詢並處理結果。以下，我將逐步引導您完成這個過程，並以清晰和實用的方式進行說明。

第 1 步：將 JDBC 驅動程序添加到您的項目中

首先，您需要適當的 JDBC 驅動程序來與您正在使用的資料庫（例如 MySQL、PostgreSQL、Oracle 等）進行通信。每個資料庫供應商都提供其自己的驅動程序。

- **對於 Maven 項目**：將依賴項添加到您的 `pom.xml`。例如，對於 MySQL：

  ```
  <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>8.0.33</version> <!-- 檢查最新版本 -->
  </dependency>
  ```

  對於 PostgreSQL：

  ```
  <dependency>
      <groupId>org.postgresql</groupId>
      <artifactId>postgresql</artifactId>
      <version>42.7.3</version> <!-- 檢查最新版本 -->
  </dependency>
  ```

- **對於非 Maven 項目**：從供應商的網站（例如從 mysql.com 下載 MySQL Connector/J）下載 JDBC 驅動程序 JAR 文件，並手動將其添加到項目的類路徑中。

第 2 步：加載 JDBC 驅動程序

在較舊版本的 Java（JDBC 4.0 之前），您必須使用 `Class.forName()` 明確加載驅動程序。在現代 JDBC 驅動程序（JDBC 4.0+）中，這一步是可選的，因為驅動程序會通過服務提供者機制自動註冊。然而，為了清楚起見，這是如何工作的：

```
try {
    Class.forName("com.mysql.cj.jdbc.Driver"); // MySQL 示例
    // 對於 PostgreSQL：Class.forName("org.postgresql.Driver");
} catch (ClassNotFoundException e) {
    e.printStackTrace();
}
```

第 3 步：建立連接

使用 `DriverManager` 類通過提供連接 URL、用戶名和密碼來連接資料庫。URL 格式取決於資料庫。

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

public class JdbcExample {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/mydatabase"; // MySQL 示例
        String username = "your_username";
        String password = "your_password";

        try (Connection connection = DriverManager.getConnection(url, username, password)) {
            System.out.println(" 已連接到資料庫！");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

- **URL 示例：**
  - MySQL：`jdbc:mysql://localhost:3306/mydatabase?useSSL=false`
  - PostgreSQL：`jdbc:postgresql://localhost:5432/mydatabase`
  - Oracle：`jdbc:oracle:thin:@localhost:1521:xe`

第 4 步：創建語句並執行查詢

連接後，您可以使用 `Statement`、`PreparedStatement`（適用於參數化查詢）或 `CallableStatement`（適用於存儲過程）來執行 SQL 查詢。

使用簡單查詢的示例：

```
import java.sql.*;

public class JdbcExample {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/mydatabase";
        String username = "your_username";
```

```
        String password = "your_password";

        try (Connection connection = DriverManager.getConnection(url, username, password)) {
            Statement statement = connection.createStatement();
            ResultSet resultSet = statement.executeQuery("SELECT * FROM users");

            while (resultSet.next()) {
                System.out.println("ID: " + resultSet.getInt("id") + ", Name: " + resultSet.getString("name"))
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

使用 PreparedStatement（對用戶輸入更安全）：

```
try (Connection connection = DriverManager.getConnection(url, username, password)) {
    String sql = "SELECT * FROM users WHERE id = ?";
    PreparedStatement preparedStatement = connection.prepareStatement(sql);
    preparedStatement.setInt(1, 5); // 設置參數值
    ResultSet resultSet = preparedStatement.executeQuery();

    while (resultSet.next()) {
        System.out.println("ID: " + resultSet.getInt("id") + ", Name: " + resultSet.getString("name"));
    }
} catch (SQLException e) {
    e.printStackTrace();
}
```

第 5 步：關閉資源

始終關閉 `ResultSet`、`Statement` 和 `Connection` 對象以釋放資源。使用 try-with-resources（如上所示）會自動處理這一點。

完整工作示例（MySQL）

假設有一個 `users` 表，其中包含 `id`（int）和 `name`（varchar）列：

```
import java.sql.*;
```

```
public class JdbcExample {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/mydatabase?useSSL=false";
        String username = "root";
        String password = "your_password";

        try (Connection connection = DriverManager.getConnection(url, username, password)) {
            String sql = "SELECT * FROM users WHERE id = ?";
            PreparedStatement preparedStatement = connection.prepareStatement(sql);
            preparedStatement.setInt(1, 1);
            ResultSet resultSet = preparedStatement.executeQuery();

            while (resultSet.next()) {
                System.out.println("ID: " + resultSet.getInt("id") + ", Name: " + resultSet.getString("name"))
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

關鍵提示

- **錯誤處理**：始終捕捉 `SQLException` 以處理資料庫相關錯誤。
- **依賴項**：確保 JDBC 驅動程序與您的資料庫版本匹配。
- **安全性**：對於包含用戶輸入的查詢，使用 `PreparedStatement` 以防止 SQL 注入。
- **連接池**：對於生產應用程序，考慮使用連接池（例如 HikariCP）而不是 `DriverManager`。

如果您需要幫助處理特定資料庫或更高級用例，請告訴我！