使用 JDBC (Java 数据库连接) 驱动程序

在 Java 中使用 JDBC（Java 数据库连接）驱动程序，您需要按照一系列步骤建立与数据库的连接、执行查询并处理结果。下面，我将逐步详细介绍这个过程。

第 1 步：将 JDBC 驱动程序添加到您的项目中

首先，您需要适合您正在使用的数据库的 JDBC 驱动程序（例如，MySQL、PostgreSQL、Oracle 等）。每个数据库供应商都提供自己的驱动程序。

- **对于 Maven 项目**：将依赖项添加到您的 `pom.xml` 中。例如，对于 MySQL：

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

  对于 PostgreSQL：

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

- **对于非 Maven 项目**：从供应商的网站（例如，从 mysql.com 下载 MySQL Connector/J）下载 JDBC 驱动程序 JAR 文件，并将其手动添加到项目的类路径中。

第 2 步：加载 JDBC 驱动程序

在较旧版本的 Java（JDBC 4.0 之前）中，您需要使用 `Class.forName()` 显式加载驱动程序。对于现代 JDBC 驱动程序（JDBC 4.0+），这个步骤是可选的，因为驱动程序通过服务提供者机制自动注册。不过，为了清楚起见，这里是它的工作原理：

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

第 3 步：建立连接

使用 `DriverManager` 类通过提供连接 URL、用户名和密码来连接到数据库。URL 格式取决于数据库。

```
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(" 已连接到数据库！ ");
        } 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：创建语句并执行查询

连接后，您可以使用 `Statement`、`PreparedStatement`（用于参数化查询，推荐使用）或 `CallableStatement`（用于存储过程）执行 SQL 查询。

使用简单查询的示例：

```
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（对于用户输入更安全）：

```
try (Connection connection = DriverManager.getConnection(url, username, password)) {
    String sql = "SELECT * FROM users WHERE id = ?";
    PreparedStatement preparedStatement = connection.prepareStatement(sql);
    preparedStatement.setInt(1, 5); // 设置参数值
    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：关闭资源

始终关闭 `ResultSet`、`Statement` 和 `Connection` 对象以释放资源。使用 try-with-resources（如上所示）可以自动处理这一点。

完整工作示例（MySQL）

假设有一个 `users` 表，其中包含 `id`（int）和 `name`（varchar）列：

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

```java
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();
        }
    }
}
```

关键提示

- **错误处理**：始终捕获 `SQLException` 以处理数据库相关错误。
- **依赖项**：确保 JDBC 驱动程序与您的数据库版本匹配。
- **安全性**：对于包含用户输入的查询，使用 `PreparedStatement` 以防止 SQL 注入。
- **连接池**：对于生产应用程序，考虑使用连接池（例如，HikariCP）而不是 `DriverManager`。

如果您需要帮助处理特定数据库或更高级用例，请告诉我！