什么是JDBC

JDBC：Java DataBase Connectivity（Java连接数据库技术）

在Java代码中，使用JDBC提供的方法，可以`发送字符串类型的SQL语句到数据库管理软件（MySQL）`，`并且获取语句执行结果`，进而实现数据库数据CURD操作的技术。

通俗点说，就是类似navicat。

引入mysql-jdbc驱动包

在IDEA下创建lib文件夹，将mysql-connector-java-8.0.27-bin.jar和mysql-connector-java-8.0.27-src.zip复制到该文件夹下，右键jar包Add as jar..即可。

JDBC基本使用步骤分析（6步）

`1.注册驱动`

`2.创建连接`

`3.创建发送SQL语句对象`

`4.发送SQL语句并且获取结果`

`5.结果解析`

`6.释放资源`

基于Statement演示查询

`1.注册驱动：DriverManager.registerDriver(new Driver())`

`2.获取连接：Connection connection = DriverManager.getConnection`

`3.创建语句对象(Statement)：Statement statement = connection.createStatement()`

`4.发送SQL获取结果：ResultSet resultSet = statement.executeQuery(sql)`

`5.结果解析：while (resultSet.next()) {}`

`6.关闭资源`

```
1   package com.atguigu.api.statement;
2
3   import com.mysql.cj.jdbc.Driver;
4
5   import java.sql.*;
6
7   /**
8    * Project:  jdbcTutorial
9    * Create date:  2023/12/26
10   * Created by lwPigKing
11   */
12
13  public class StatementQueryPart {
```

```
14          public static void main(String[] args) throws SQLException {
15              // 1.注册驱动
16              //依赖：驱动版本8+ com.mysql.jc.jdbc.Driver
17              //       驱动版本5+ com.mysql.jdbc.Driver
18              DriverManager.registerDriver(new Driver());
19
20              // 2.获取连接
21              // 和数据库创建连接：1.数据库ip 2.端口好 3.账号 4.密码 5.数据库名称
22              // ip: jdbc:数据库厂商名://ip地址:port/数据库名
23              Connection connection = DriverManager.getConnection(
24                      "jdbc:mysql://localhost:3306/atguigu",
25                      "root",
26                      "123456");
27
28              // 3.创建statement
29              Statement statement = connection.createStatement();
30
31              // 4.发送sql语句，并且获取返回结果
32              String sql = "select * from t_user";
33              ResultSet resultSet = statement.executeQuery(sql);
34
35              // 5.进行结果集解析
36              // 看看有没有下一行数据，有就可以获取
37              while (resultSet.next()) {
38                  int id = resultSet.getInt("id");
39                  String account = resultSet.getString("account");
40                  String password = resultSet.getString("PASSWORD");
41                  String nickname = resultSet.getString("nickname");
42                  System.out.println(id + "--" + account + "--" + password + "--" + nickname);
43              }
44
45              // 6.关闭资源
46              // 从内往外关
47              resultSet.close();
48              statement.close();
49              connection.close();
50
51          }
52      }
53
```

基于Statement方式问题

Statement详解

```
1.注册驱动
方案1：DriverManager.registerDriver(new Driver())
问题：
    1.注册两次驱动DriverManager.registerDriver()方法本身会注册一次。
    2.Driver.static{DriverManager.registerDriver()}静态代码块，也会注册一次。
解决：只注册一次（只触发静态代码块）

方案2：new Driver() 该写法太固定了，如果切换了数据库，就需要该代码。
方案3：反射 字符串的Driver全限定符，可以引导外部的配置文件 -> xx.properties，如果切换数据库，只需修改配置文件即可。
Class.forName("com.mysql.cj.jdbc.Driver");
```

```
2.获取数据库连接
getConnection是一个重载方法。
允许开发者用不同的形式传入数据库连接的核心参数。
核心属性：
    1.数据库软件所在的主机的IP地址
    2.数据库软件所在的主机的端口号
    3.连接的具体数据库
    4.连接的账号
    5.连接的密码
String url = "jdbc:mysql://127.0.0.1:3306/atguigu";
Properties properties = new Properties();
properties.put("user", "root");
properties.put("password", "123456");
Connection connection = DriverManager.getConnection(url, properties);
```

```
3.创建发送sql语句的statement对象
statement：可以发送sql语句到数据库，并且获取返回结果
Statement statement = connection.createStatement();
```

```
4.发送sql语句
SQL分类：DDL(数据定义语言)、DML(数据操纵语言)、DQL(数据查询语言)、DCL(数据控制语言)、TPL

executeUpdate
参数：非DQL
返回：int
情况1：DML（返回影响的行数，例如删除了三条数据 return 3）
情况2：非DML return 0

executeQuery
参数：DQL
返回：ResultSet对象

int i = statement.executeUpdate(sql);
ResultSet resultSet = statement.executeQuery(sql);
```

```
5.查询结果集解析 resultSet
resultSet -> 逐行获取数据
想要进行数据解析，我们需要进行两件事情：1.移动游标指定获取数据行 2.获取指定数据行的列数据即可
游标移动问题
ResultSet内部包含一个游标，指定当前行数据
默认游标指定的是第一行数据之前
我们可以调用next方法向后移动一行移动
如果有很多行数据，我们可以使用while(next){获取每一行的数据}
boolean = next() true:有更多行数据，并且向下移动一行
                false:没有更多行数据
```

Statement问题

1.SQL语句需要字符串拼接，比较麻烦

2.只能拼接字符串类型，其他的数据库类型无法处理

3.可能发生注入攻击（动态值充当了SQL语句结构，影响了原有的查询结果！）

基于preparedStatement演示curd

```java
package com.atguigu.api.preparedStatement;

import com.mysql.cj.jdbc.Driver;
import org.junit.Test;

import java.sql.*;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

/**
 * Project:  jdbcTutorial
 * Create date:  2023/12/28
 * Created by lwPigKing
 */
public class PSCURDPart {

    /**
     * t_user插入一条数据
     *        account test
     *        password test
     *        nickname 二狗子
     */
    @Test
    public void testInsert() throws ClassNotFoundException, SQLException {
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection connection = DriverManager.getConnection(
                "jdbc:mysql:///atguigu",
                "root",
                "123456"
        );
        String sql = "insert into t_user (account, password, nickname) values(?,?,?);";
        PreparedStatement preparedStatement = connection.prepareStatement(sql);
        preparedStatement.setObject(1, "test");
        preparedStatement.setObject(2, "test");
        preparedStatement.setObject(3, "二狗子");
        int rows = preparedStatement.executeUpdate();
        if (rows > 0) {
            System.out.println("输入插入成功！");
        } else {
            System.out.println("数据插入失败！");
        }
        preparedStatement.close();
        connection.close();
    }

```

```
47          /**
48           * 修改id = 3的用户nickname = 三狗子
49           */
50          @Test
51          public void testUpdate() throws ClassNotFoundException, SQLException {
52                Class.forName("com.mysql.cj.jdbc.Driver");
53                Connection connection = DriverManager.getConnection(
54                            "jdbc:mysql:///atguigu",
55                            "root",
56                            "123456"
57                      );
58                String sql = "update t_user set nickname=? where id=?;";
59                PreparedStatement preparedStatement = connection.prepareStatement(sql);
60                preparedStatement.setObject(1, "三狗子");
61                preparedStatement.setObject(2, 3);
62                int rows = preparedStatement.executeUpdate();
63                if (rows > 0) {
64                      System.out.println("修改成功！");
65                } else {
66                      System.out.println("修改失败！");
67                }
68                preparedStatement.close();
69                connection.close();
70          }
71
72          /**
73           * 删除id=3的用户数据
74           */
75          @Test
76          public void testDelete() throws ClassNotFoundException, SQLException {
77                Class.forName("com.mysql.cj.jdbc.Driver");
78                Connection connection = DriverManager.getConnection(
79                            "jdbc:mysql:///atguigu",
80                            "root",
81                            "123456"
82                      );
83                String sql = "delete from t_user where id=?;";
84                PreparedStatement preparedStatement = connection.prepareStatement(sql);
85                preparedStatement.setObject(1, 3);
86                int rows = preparedStatement.executeUpdate();
87                if (rows > 0) {
88                      System.out.println("删除成功！");
89                } else {
90                      System.out.println("删除失败！");
91                }
92                preparedStatement.close();
93                connection.close();
94          }
95
96          /**
97           * 目标：查询所有用户数据，并且封装到一个List<Map>集合中
98           */
99          @Test
100         public void testSelect() throws ClassNotFoundException, SQLException {
101               Class.forName("com.mysql.cj.jdbc.Driver");
102               Connection connection = DriverManager.getConnection(
103                           "jdbc:mysql:///atguigu",
104                           "root",
```

```
                        "123456"
                );
                String sql = "select id, account, password, nickname from t_user;";
                PreparedStatement preparedStatement = connection.prepareStatement(sql);
                ResultSet resultSet = preparedStatement.executeQuery();
                ArrayList<Map> list = new ArrayList<>();

                // 获取列的信息对象
                // metaData装的当前结果集列的信息对象（可以获取列的名称，也可以获取列的数量）
                ResultSetMetaData metaData = resultSet.getMetaData();
                // 水平遍历列
                int columnCount = metaData.getColumnCount();


                while (resultSet.next()) {
                        Map map = new HashMap();
                        // 纯手动取值！
//                        map.put("id", resultSet.getInt("id"));
//                        map.put("account", resultSet.getString("account"));
//                        map.put("password", resultSet.getString("password"));
//                        map.put("nickname", resultSet.getString("nickname"));

                        // 自动遍历列：要从1开始，并且小于等于总列数
                        for (int i = 1; i <= columnCount; i++) {
                                // 获取指定列下角标的值！
                                Object value = resultSet.getObject(i);
                                map.put(metaData.getColumnLabel(i), value);
                        }


                        // 一行数据的所有列全部存放到了Map中
                        // 将Map存储到集合中即可
                        list.add(map);


                }
        }

}
```

preparedStatement使用方式总结

1. 加载驱动（类反射）

2. 获取连接对象

3. 编写SQL结构语句（用?表示动态值）

4. 利用连接对象创建PreparedStatement对象

5. 给动态值（?）赋值

6. 关闭连接