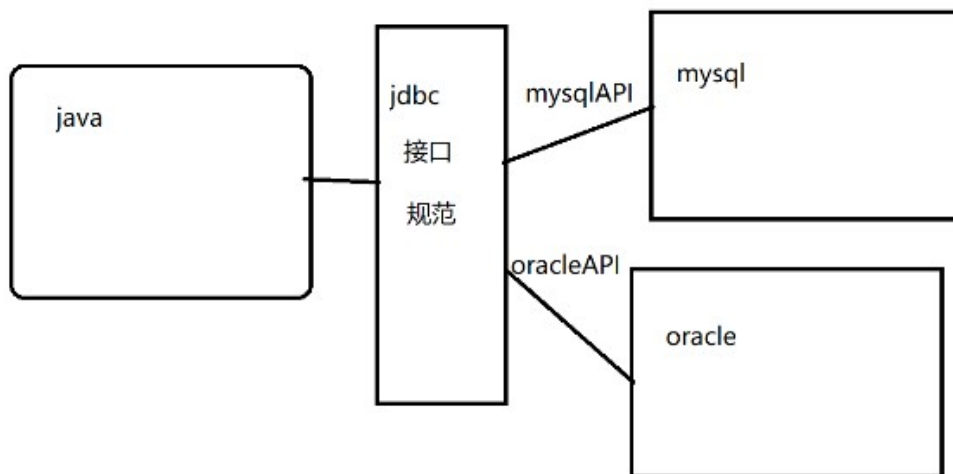


# JDBC

## 一 JDBC 简介

由于各大数据库厂商提供的驱动(操作数据库的 jar 包), 各不相同, 导致开发人员学习成本改哦, sun 公司为简化操作, 提供一套规范, 本质是一大堆接口, 要求各数据库厂商实现这套规范, 这个规范就是 JDBC, 只要学会 jdbc 这套接口, 即可操作各类数据库。



## 二 JAVA JDBC

Jdbc 由二个包组成, 分别是 java.sql 和 javax.sql, 目前已经集成到 javase 中  
但 jdbc 只是一套接口, 真正实现在具体的数据库的 jar 包中, 所以连接数据库需要导入具体的驱动 jar 包。  
JDBC 提供了对 Java 程序员, 数据库厂商及第三方中间件厂商的 API。

## 三 JDBC 相关类或接口

### 1 DriverManager 类

获取连接注册驱动

## 2 Connection 接口

代表数据库连接的对象，客户端与数据库的交互都是通过 Connection 对象完成的  
CreateStatement():创建传输器，用与发送 sql 和返回结果

## 3 Statement 接口

用于向数据库发送 SQL 语句的方法

### 3.A ExecuteQuery(String sql):

执行查询操作，返回 ResultSet 结果集

### 3.B ExecuteUpdate(String sql):

执行增删改操作，返回的时被影响的行数

### 3.C 子接口

#### 3.C.1 CallableStatement

用于执行 SQL 存储过程的接口

#### 3.C.2 PreparedStatement

注入攻击：用户提交一些特殊字符，后台在拼接用户名和密码形成 SQL 语句时，特殊字符有了特殊语义，使 SQL 语句语义改变，获得了不可预知的结果。

该接口为预编译，可解决注入攻击。

表示预编译的 SQL 语句的对象。

## 4 ResultSet 接口

ResultSet 对象具有指向其当前数据行的光标。最初，光标被置于第一行之前。next 方法将光标移动到下一行；因为该方法在 ResultSet 对象没有下一行时返回 false，所以可以在 while 循环中使用它来迭代结果集。

此对象用于封装查询结果，封装时类似于表格的方式，ResultSet 中维护了一个游标，初始位置在第一行之前，调用 next()方法，游标向下移动一行

### 4.A 操作游标的方法：

next(): 移动到下一行，有数据返回 true，没有返回 false

previous(): 移动到上一行  
absolute(int row): 移动到指定行  
beforeFirst(): 移动到第一行的前面  
afterLast(): 移动到最后一行的后面

#### 4.B 获取数据的方法:

getInt(String col\_name)  
getString(String col\_name)  
getDouble(String col\_name)  
getObject(String col\_name)

## 四 JAVA 连接数据库步骤

### 1 加载驱动

#### 1.A 代码:

```
Class.forName("com.mysql.jdbc.Driver");
```

#### 1.B 说明:

通过反射加载 com.mysql.jdbc.Driver 类，这个类就是在我们导入的 MySQL JAR 包中，该类中有静态代码块，一旦反射，static 代码块执行，构造该类

所以理论上可以直接

```
DriverManager.registerDriver(new com.mysql.jdbc.Driver());
```

但是这里 new 了一次，static 代码块 new 了一次，重复了  
编译时异常 *ClassNotFoundException*

```
public class Driver extends NonRegisteringDriver implements java.sql.Driver {  
    public Driver() throws SQLException {  
    }  
  
    static {  
        try {  
            DriverManager.registerDriver(new Driver());  
        } catch (SQLException var1) {  
            throw new RuntimeException("Can't register driver!");  
        }  
    }  
}
```

## 2 创建连接

### 2.A 代码

```
Connection conn=DriverManager.getConnection  
("jdbc:mysql://localhost:3306/mydb1?  
useSSL=false&serverTimezone=UTC","root","admin");
```

### 2.B 说明

2.B.1 URL(类似 httpurl) "jdbc:mysql://localhost:3306/mydb1"

协议: *jdbc:mysql*

地址: *localhost* 本机, 如果是本机可忽略不写

端口: *mysql* 为 *3306* 如果是默认的可以省略不写

数据库名: *mysb1*

参数: ?参数名 1=参数值 1&参数名 2.....和 URL 类似

2.B.2 用户名: *root*

2.B.3 密码: *admin*

2.B.4 注意:

编译时异常: *SQLException*

URL 省略后: *jdbc:mysql:///mydb1*

## 3 创建传输器

### 3.A 代码 1:

```
Statement statement=conn.createStatement();  
编译时异常: SQLException
```

### 3.B 代码 2:

```
PreparedStatement statement=conn.prepareStatement  
("select * from exam WHERE USER=?");  
编译时异常: SQLException
```

3.B.1 “?”为占位符

3.B.2 传值：

```
//注意序号从1开始
statement.setString(1,"123");
//表示第一个问号的值替换为123
最终语句：“select * from exam WHERE USER='123'”
```

3.B.3 特点(Statement 相反)

*预编译：效率高*

提前将 SQL 骨架编译

*防注入攻击：*

预编译：就是定死了，后面不管填什么,都当作字符串，不能当作 sql 语句

*SQL 语句固定，不灵活*

## 4 发送 SQL

4.A *Statement 发送 sql*

```
ResultSet resultSet= statement.executeQuery("select * from
exam;");
编译时异常：SQLException
```

4.B *PreparedStatement 发送 sql*

```
ResultSet resultSet= statement.executeQuery();
```

## 5 处理结果

5.A 代码：

```
while (resultSet.next()){
    System.out.println(resultSet.getString("name"));
    或者 getString(1) 第一列数据
    //注意序号从1开始
}
```

5.B 说明

初始时指针指向表头，调用 `next()` 方法移动指针，第一次就是第一行记录，如果没有记录，`next()` 返回 `false`

## 6 释放资源

### 6.A 代码

```
33     } finally {
34         if (resultSet!=null){
35             try {
36                 resultSet.close();
37             } catch (SQLException e) {
38                 e.printStackTrace();
39             }finally {
40                 resultSet=null;
41             }
42         }
43         if (statement!=null){
44             try {
45                 statement.close();
46             } catch (SQLException e) {
47                 e.printStackTrace();
48             }finally {
49                 statement=null;
50             }
51         }
52     }
53     if (conn!=null){
54         try {
55             conn.close();
56         } catch (SQLException e) {
57             e.printStackTrace();
58         }finally {
59             conn=null;
60         }
61     }
62 }
63
64
65
66 }
```

### 6.B 从后往前依次关闭

先关闭 ResultSet->Statement->Connection

Connecting 是非常稀有的资源，应尽量晚创建、早释放

当生成 ResultSet 对象的 Statement 对象关闭、重新执行或用来从多个结果的序列获取下一个结果时，ResultSet 对象将自动关闭。

## 五 数据库连接池(C3P0)


- 1 C3p0 常见连接池，性能优秀开源
- 2 会读取默认配置文件，配置文件：c3p0.properties 或 c3p0-config.xml
- 3 使用

### 3.A 配置文件内容

*Properties:*

```
c3p0.driverClass=com.mysql.jdbc.Driver  
c3p0.jdbcUrl=jdbc:mysql://localhost:3306/mydb1  
c3p0.user=root  
c3p0.password=admin
```

*XML:*



```
<?xml version="1.0" encoding="utf-8" ?>  
<c3p0-config>  
  <default-config>  
    <property name="driverClass">com.mysql.jdbc.Driver</property>  
    <property name="jdbcUrl">jdbc:mysql://localhost:3306/mydb1</property>  
    <property name="user">root</property>  
    <property name="password">admin</property>  
  </default-config>  
</c3p0-config>
```

### 3.B 创建连接池对象

```
ComboPooledDataSource pool = new ComboPooledDataSource();
```

### 3.C 获取 Connection 对象

```
Connection conn=pool.getConnection();
```

### 3.D 关闭连接

提问：Connection 使用完不是应该放回队列中吗，我看老师在 finally 中直接关闭了？

回答：在后面老师会讲设计模式，和代理模式有关？讲了后更好理解

4.22 已讲，装饰模式