# JDBC

## 一、JDBC

JDBC(Java DataBase Connectivity)：Java代码与数据库相连接。JDBC是连接java应用程序和数据库之间的桥梁，是java语言访问数据库的一种规范。在JDBC中，导包一般导入的是**java.sql**中的包，而不是java.msql的包。

**jdbc的好处：可以实现java代码和数据库之间的交互，同时也能屏蔽不同数据库之间的差异，使java代码不需要考虑不同数据库之间有什么差异，而直接通过jdbc就能连接数据库**。

DriverManager是一个类，而Connection、Statement、PreparedStatement和ResultSet都是接口。

JDBC的四大物件：**DriverManager、Connection、Statement、ResultSet**。

**JDBC开发的简单步骤：**

**①加载驱动程序：Class.forName(“com.mysql.jdbc.Driver”)；**

**②建立与DB的连接：Connection con=DriverManager. getConnection(url，user,password);**url中包含主机名、端口号、数据库名。

**③写SQL语句，并创建用于向DB中发送SQL的对象。**如Statement s=con.createStatement(sql)。

**④将SQL语句送到DB中执行，并获得执行的结果。**如ResultSet rs=st.executeQuery(sql);

**⑤关闭数据库的资源。**

### 1.加载驱动并获得连接对象

此处不是导java.mysql.中的包，牢记。**将mysql的jar包复制到项目中，然后右键通过BuildPath导入**。

①加载Mysql驱动时要加载**com.mysql.jdbc.Driver**这个类。注意，若要加载Oracle驱动，则要加载“**oracle.jdbc.driver.OracleDriver**”这个类。

②通过驱动管理员获得Connection连接对象，**DriverManager.getConnection(String url，String user，String password)**：返回值为Connection类型，Connection也是一个接口。其中url中包括主机名，端口号和数据库名。括号内一般的参数一般为：(**"jdbc:mysql://localhost:3306/school","root","LQ3217103676"**)。3306为端口号，school为要连接的数据库名，用户名一般为root，密码为mysql自己设置的密码。jdbc相当于主协议，mysql则相当于子协议。这里的localhost是指数据库服务端的IP地址，localhost也可用127.0.0.1替换，表示的是本机的ip地址。若想连接其它电脑上的数据库，则需要替换成相应的ip地址。

### 2.Connection接口

jdbc中的Connection对象用于代表数据库的连接，客户端与数据库所有交互都是通过connection对象完成的，这个对象的常用方法如下：

①Connection**.createStatement()**：该方法用于创建向数据库发送SQL的Statement对象，返回值为Statement类型。

②Connection**.prepareStatement(sql)**：该方法用于创建向数据库发送预编译SQL的PreparedStatement对象，返回值为PreparedStatement类型。

③Connection**.prepareCall(sql)**：该方法用于创建执行存储过程的CallableStatement对象。

④Connection**.setAutoCommit(boolean auotCommit)**：用于设置事物是否自动提交。

⑤Connection**.commit()**：用于手动提交事物。

⑥Connection**.rollback()**：回滚事物。

### 3.Statement接口

Statement是一个接口。其对应的executeQuery(sql)方法和executeUpdate(sql)方法都是抽象方法。其部分源码如下：

**public** **interface** Statement **extends** Wrapper, AutoCloseable {...}

ResultSet executeQuery(String sql) **throws** SQLException;

**int** executeUpdate(String sql) **throws** SQLException;

**boolean** execute(String sql) **throws** SQLException;

①Statement**.executeUpdate(String sql)**：返回值为int类型。括号内的参数为向数据库中发送的sql语句。该sql语句只能是DML中的**insert、update、delete**操作。**该方法返回的实际上就是对数据库中表的修改的行数**。如若成功进行了一次insert操作，则返回值为1。

②Statement**.executeQuery(String sql)**：返回值为**ResultSet**类型，即返回的是查询之后得到的结果集。该方法用于向DB中发送查询(select)语句。

小结：**st.executeUpdate(sql)用于insert、update、delete操作，返回值为int类型。而st.executeQuery(sql)用于select操作，返回值为ResultSet类型**。

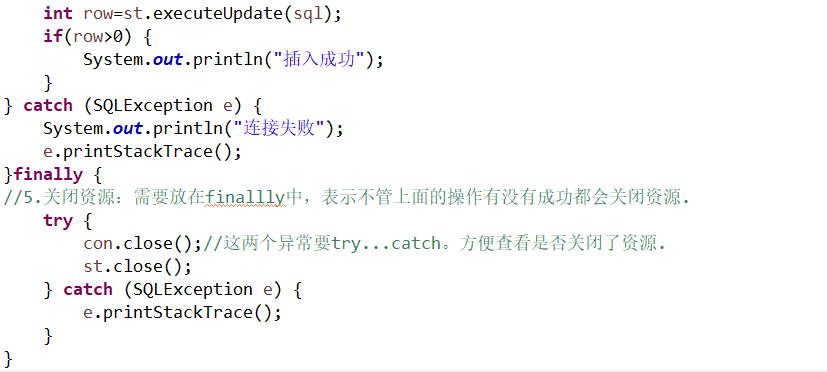
③Statement**.execute(String sql)**：该方法用于向数据库中发送任意的sql语句。**该方法的返回值是boolean类型，如果执行的结果为ResultSet，表示执行的是查询语句，返回值为true；如果执行的结果为行数或者没有结果，表示执行的是insert、delete、update的操作，返回值为false。**其中，执行的结果可以通过**getResultSet()**方法和**getUpdateCount**()方法获得。

④Statement**.addBatch(String sql)**：该方法用于把多条sql语句放到一个批处理中。

⑤Statement**.executeBatch()**：该方法用于向数据库中发送一批sql语句并执行。

**java代码与数据库相连的例子：插入数据**

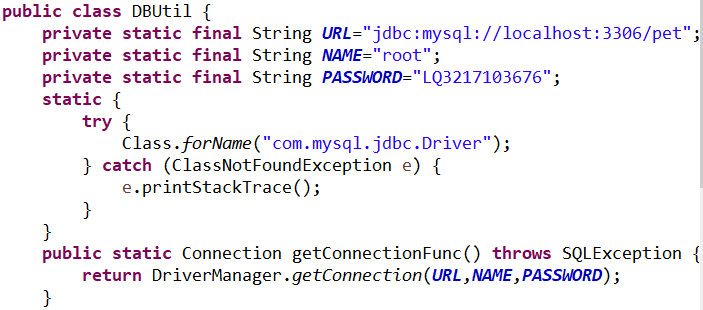




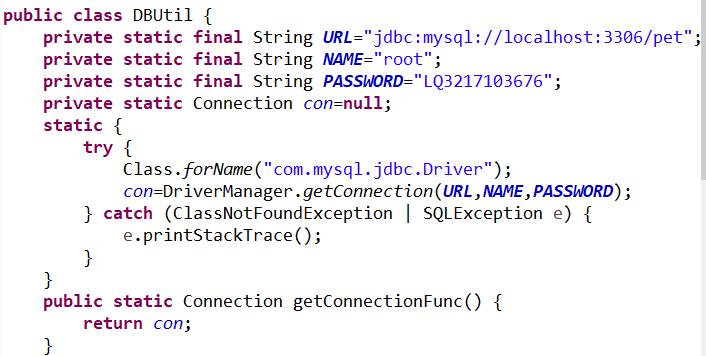
**上面代码写的不规范，写成下面这中规范的形式：**



**也可写成下面这个形式：**

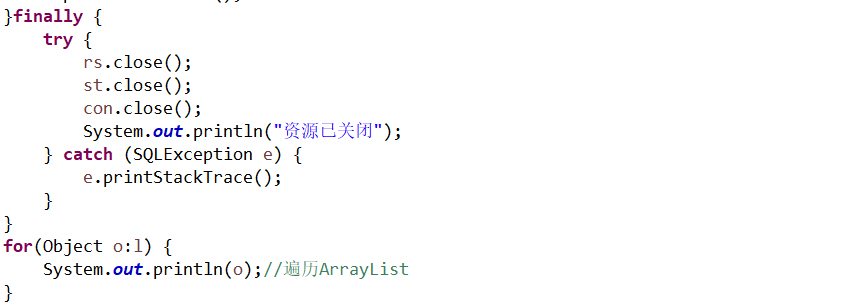


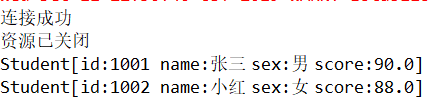
考虑一下，是写成上面那个种形式还是下面这种形式。



**java代码与数据库相连的例子：查询数据**





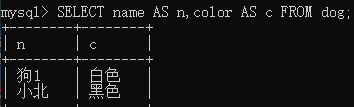


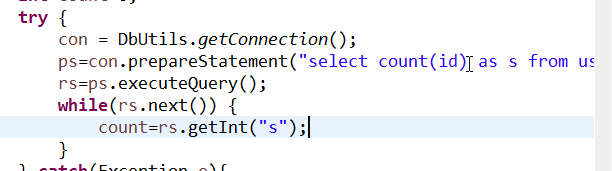
### 4.ResultSet接口

ResultSet中的rs.next()：判断下面是否还有数据，若有数据，则将该数据保存在一个地方，并继续往下走。要获得ResultSet集合中的具体内容，则一般用rs.getInt(String columnLable)、rs.getString(String columnLable)方法等。rs.next()类似将迭代器中的hasNext()和next()方法结合起来。

ResultSet**.getString(String columnLable)**：ResultSet的方法举例，括号内的参数一般填String类型的表的列标签。该方法返回的是一个String类型的数据。

**特别注意：ResultSet中的getXxx()方法是从查询结果表中获取数据。**若查询语句给表的列名等起了别名，则ResultSet中的getString()括号中填的是别名，如getString(n)和getString(c)而不是getString(name)等。





迭代器中，先用hasNext()判断下面是否有元素，如果有，则调用next()返回该元素，并继续往下走。

实体类一般放在model包中。工具和方法一般放在util包中，该方法一般是重复调用的方法。接口一般放在dao包中，对于表中数据的操作一般放在dao包中，dao包中放的接口，因此dao包中的方法一般是public abstract类型的。daoimplements包放实现dao里面的具体方法的内容，即重写dao包中的方法。测试一般放在Test包中。

### 5.PreparedStatement接口

PreparedStatement 是Statement的子类，也是一个接口。其部分源码如下：

**public** **interface** PreparedStatement **extends** Statement {...}

ResultSet executeQuery() **throws** SQLException;

**int** executeUpdate() **throws** SQLException;

**boolean** execute() **throws** SQLException;

PreparedStatement是预编译的sql执行对象，它会使用 ? 占位符 来进行占位，之后依次对这其中的占位符进行赋值，这期间会过滤掉一个不合法的字符，防止sql注入的问题。

**①**Connection**.prepareStatement(String sql)**：返回值为**PreparedStatement**类型。要特别注意这个单词的写法。括号内的sql语句可以包含问号？。注意Connection获得prepareStatement时，不用create，而是直接con.prepareStatement。而Connection获得Statement时，用的是con.createStatement**。**PrepareStatement对象先将sql语句送到数据库中编译,可以防止sql注入，推荐使用。

**②**PreparedStatement**.setInt(int parameterIndex, int x)**：parameterIndex表示填入的第一个参数为**1**，第二个为2........这个不是索引号，不是从0开始。x表示参数所对应的值。无返回值。

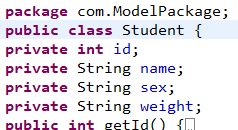
注意当出现下面这个异常时，一般是右边这个jar文件没有放入，或没有Build path-->Add to Build path：。



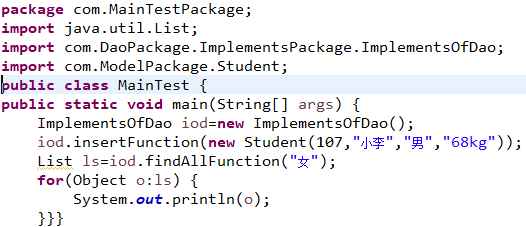
**③**ps.executeUpdate()和ps.executeQuery：该方法是PreparedStatement的。而括号内多一个参数sql时，如：st.executeUpdate(sql)则是Statement的。

**java代码与数据库相连的例子：**

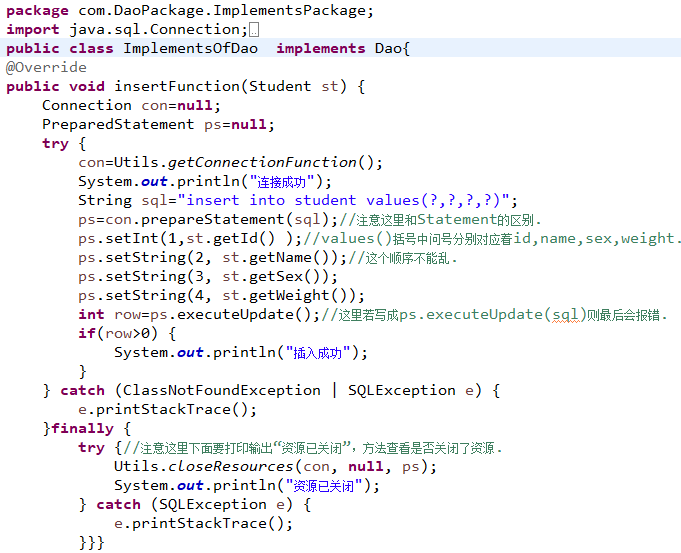
**student类**



**测试类**

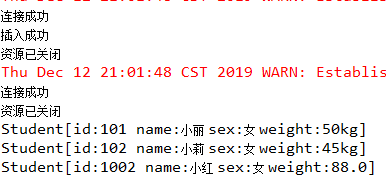


**对数据库操作的类**

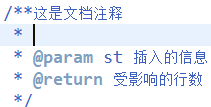




上面代码执行后的输出结果如下：



下面这个是文档注释：



在使用PreparedStatement时，它的SQL语句不在是采用字符串拼接的方式，而是采用占位符(？)的方式。如：



#### ⑴Statement和PreparedStatement的区别

**相同点**：两者都可用来执行SQL语句。都有executeQuery()、executeUpdate()、execute()方法。其中PreparedStatement继承Statement。

**不同点：**Statement接口是通过Connection接口的createStatement()方法创建的，而PreparedStatement是通过Connection接口的prepareStatement(String sql)方法和SQL语句作为该方法的参数创建的。PreparedStatement中的execute()、executeQuery()、executeUpdate()方法括号内都没有参数，而Statement的这几个方法都将SQL语句作为参数。**Statement 在进行批处理的时候可以执行不同类型的sql语句，而preparedStatement 在执行批处理的时候不可以执行不同类型的sql 语句**。

**优缺点：**

**①**PreparedStatement的SQL语句可以使用占位符，然后采用参数设置，可读性好。而Statement的SQL语句采用字符串拼接，可读性和维护性较差。

**②**PreparedStatement具有预编译机制，性能比Statement更快

**③**PreparedStatemenet可以有效的防止SQL注入，而Statement不能防止SQL注入。

#### ⑵execute()和executeQuery()和executeUpdate()的区别

**注意Statement的这几个方法都将SQL语句作为方法的参数。而PreparedStatement的这几个方法没有参数**。

**executeQuery()方法：**只能执行select的SQL语句，无法执行增删改，该方法的返回类型是结果集ResultSet。

**executeUpdate()方法：**只能执行CUD(insert、update 、delete)的SQL语句，它的返回值是影响数据库记录的行数。

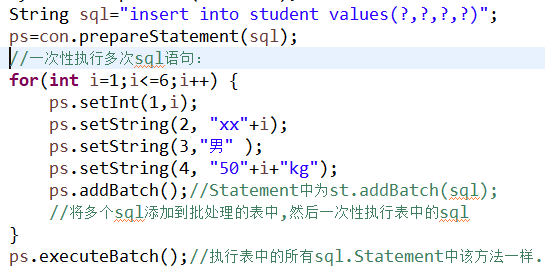
**execute()方法：**它能执行CRUD(insert、insert、update 、delete)中的任意一种语句，。返回值为boolean类型，表示是否有结果集。若为true，表示有结果集，执行的是select操作；若为false表示没有结果集，执行的是CUD操作。结果可通过getResultSet()和getUpdateCount()获得。

### 6.批处理

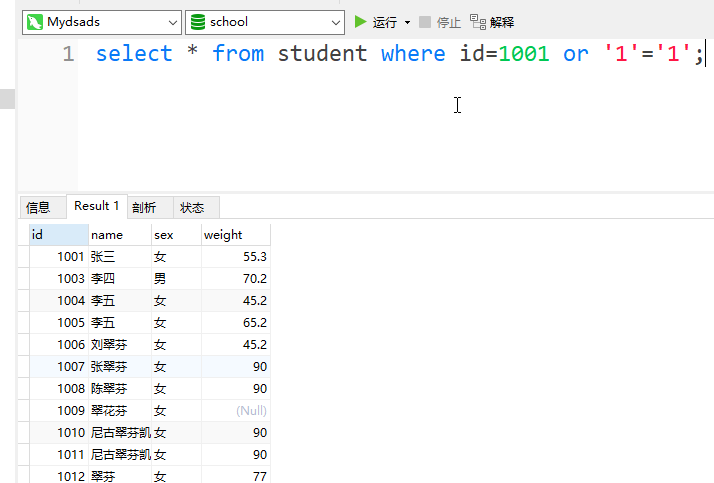
PreparedStatement**.addBatch()**：无返回值，当PreparedStatement使用该方法时，括号内无参数。而Statement使用该方法时括号内的参数为String sql。该方法表示将多个sql语句添加到批处理的表中,然后一次性执行表中的sql语句，即一次性执行多次向表中的如插入等操作。一般情况下，不使用addBatch()方法时，对数据库表中的数据操作每运行一次执行同一个sql语句一次。而使用addBatch()方法，则可以运行一次，就可以对表中的数据进行多次操作。

PreparedStatement**.executeBatch()**：括号中无参数，该方法是Statement的。该方法的返回值为一个int类型的数组，数组中的内容是什么还不太明白。ps**.executeBatch()**和st**.executeBatch()**表示执行表中的所有sql语句。有了executeBatch()则不需要再写executeUpdate()方法了。

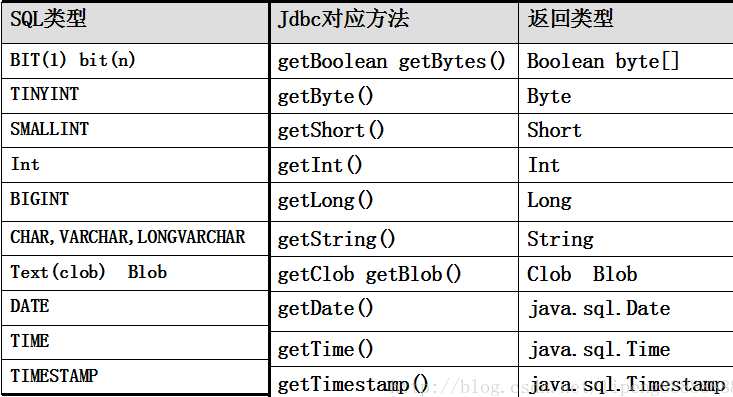
**注意JDBC的批处理不能加入select语句，否则会抛出异常**。



**sql注入攻击**是指，在使用字符串拼接sql语句时，恶意拼接带sql语言关键词，使得sql语句的原意改变，达到恶意攻击的目的。如：下面使用or关键字拼接了一个‘1’=‘1’的字符串，该条件是直接成立的，同时原来的条件id=1001则会失效，则相当于直接执行select \* from student语句。



### 7.数据类型转换



## 二、事物(Transaction)

### 1.事物的概念及属性

**事物是作为单个逻辑工作单元执行的一系列操作，这些操作作为一个整体一起向系统提交，要么都执行、要么都不执行**。事务是一个不可分割的工作逻辑单元

#### ⑴事务的四个属性，简称ACID 属性：

①**原子性**（Atomicity）：事务是一个完整的操作。不能对它进行再分割，是最小的一个单元。

②**一致性**（Consistency）：当事务完成时，数据必须处于一致状态。

③**隔离性**（Isolation）：对数据进行修改的所有并发事务是彼此隔离的。表明事务必须是独立的，它不应以任何方式依赖于或影响其他事务。

④**永久性（**Durability）：事务完成后，它对数据库的修改被永久保持，事务日志能够保持事务的永久性。

### 2.实现事物管理

通过提交commit()或回退rollback()来管理事物的操作。当事物完成时，通过commit()将事物提交到数据库中，然后我们的数据就会被永久保存。若采用rollback()，事物回滚，如插入的数据、更新的数据都会变成原来没有插入、更新的样子。

在JDBC中事物操作是默认提交的，即一条对数据库的操作表达式(如插入、更新)代表一项事物操作。操作成功后，系统将自动调用commit()来提交，否则调用rollback()来回退。在JDBC中，通过调用setAutoCommit(false)来禁止自动提交。之后把多个对数据库操作的表达式作为一个事物，操作完成后调用commit()来进行整体提交。

Connection**.setAutoCommit(boolean autoCommit)**：无返回值，当括号内的参数为true时，表示为自动提交模式，为false时表示不是自动提交，则要设置手动提交。

Connection**.commit()**：括号内无参数，该方法为手动提交，只有当con.setAutoCommit(false)中的参数为false才能使用，否则会报错。

Connection**.rollback()**：回滚，回到之前的状态，和con.commit()配套使用，该方法也只能用在con.setAutoCommit(false)中的参数为false的情况。一般系统默认为con.setAutoCommit(true)，即系统默认为自动提交状态。

为了让多个 SQL 语句作为一个事务执行，需要做如下事情：

①**执行sql语句前调用 Connection 对象的 setAutoCommit(false);以取消自动提交事务。**

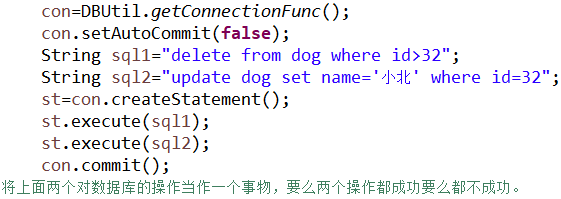
**②在所有的 sql语句都执行后，调用Connection对象的 commit()方法提交事务。**

**③在出现异常的地方，调用Connection对象的 rollback()方法回滚事务。**

**最好将批处理以及事物两者结合起来使用**。在使用批处理时可以在获取Connection对象中添加**rewriteBatchedStatements=true参数**。

DriverManager.getConnection("jdbc:mysql://localhost:3306/school(数据库名)?rewriteBatchedStatements=true","root", "LQ32....")。





### 3.不考虑隔离性会产生3个问题

[..\5.框架\12.MySQL高级.docx](../5.框架/12.MySQL高级.docx)。

### 4.解决办法

[..\5.框架\12.MySQL高级.docx](../5.框架/12.MySQL高级.docx)