**一、statement对象介绍**

　　Jdbc中的statement对象用于向数据库发送SQL语句，想完成对数据库的增删改查，只需要通过这个对象向数据库发送增删改查语句即可。  
　　Statement对象的executeUpdate方法，用于向数据库发送增、删、改的sql语句，executeUpdate执行完后，将会返回一个整数(即增删改语句导致了数据库几行数据发生了变化)。  
　　Statement.executeQuery方法用于向数据库发送查询语句，executeQuery方法返回代表查询结果的ResultSet对象。

**1.1、CRUD操作-create**

　　使用executeUpdate(String sql)方法完成数据添加操作，示例操作：

Statement st = conn.createStatement();

String sql = "insert into user(id,name,password,email,birthday) values(4,'zhangxt','123456','zs@sina.com','1993-08-23') ";

int num = st.executeUpdate(sql);

if(num>0){

System.out.println("插入成功！！！");

}

### 1.2、CRUD操作-update

　　使用executeUpdate(String sql)方法完成数据修改操作，示例操作：

Statement st = conn.createStatement();

String sql = “update user set name=‘zhangxt’ where name=‘zhang\_xiao\_tong’";

int num = st.executeUpdate(sql);

if(num>0){

System.out.println(“修改成功！！！");

}

### 1.3、CRUD操作-delete

　　使用executeUpdate(String sql)方法完成数据删除操作，示例操作：

Statement st = conn.createStatement();

String sql = “delete from user where id=1;

int num = st.executeUpdate(sql);

if(num>0){

System.out.println(“删除成功！！！");

}

### 1.4、CRUD操作-read

　　使用executeQuery(String sql)方法完成数据查询操作，示例操作：

Statement st = conn.createStatement();

String sql = “select \* from user where id=1;

ResultSet rs = st.executeUpdate(sql);

while(rs.next()){

//根据获取列的数据类型，分别调用rs的相应方法映射到java对象中

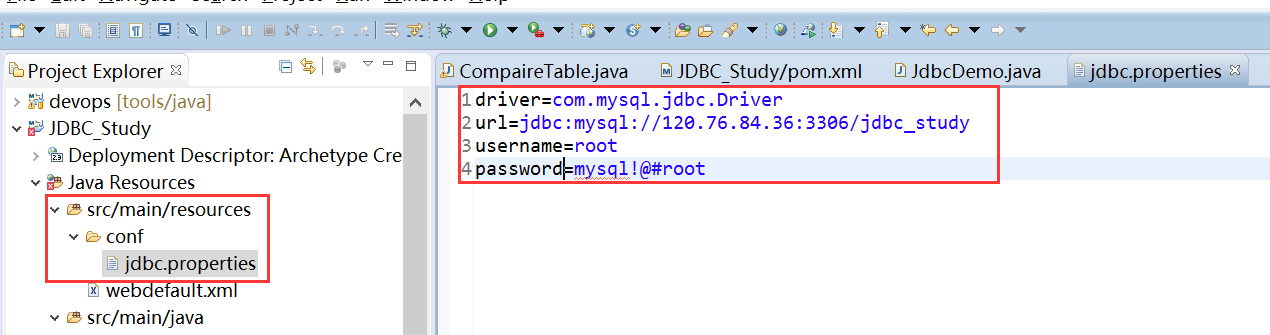
}

## 二、使用jdbc对数据库增删改查

### 2.1、搭建实验环境

继续使用《jdbc介绍.docx》文档中的试验环境，工程名：JDBC\_Study

1、在src/main/resources/下创建一个conf/jdbc.properties文件



2、编写一个JdbcUtils工具类，用于连接数据库，获取数据库连接和释放数据库连接，代码如下：

代码如下：

**package** com.test.utils;

**import** java.io.InputStream;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**import** java.util.Properties;

**public** **class** JdbcUtils {

**private** **static** String *driver* = **null**;

**private** **static** String *url* = **null**;

**private** **static** String *username* = **null**;

**private** **static** String *password* = **null**;

**static**{

**try**{

//读取db.properties文件中的数据库连接信息

InputStream in = JdbcUtils.**class**.getClassLoader().getResourceAsStream("conf/jdbc.properties");

Properties prop = **new** Properties();

prop.load(in);

//获取数据库连接驱动

*driver* = prop.getProperty("driver");

//获取数据库连接URL地址

*url* = prop.getProperty("url");

//获取数据库连接用户名

*username* = prop.getProperty("username");

//获取数据库连接密码

*password* = prop.getProperty("password");

//加载数据库驱动

Class.*forName*(*driver*);

}**catch** (Exception e) {

**throw** **new** ExceptionInInitializerError(e);

}

}

/\*\*

\* **@Method**: getConnection

\* **@Description**: 获取数据库连接对象

\* **@Anthor**:孤傲苍狼

\*

\* **@return** Connection数据库连接对象

\* **@throws** SQLException

\*/

**public** **static** Connection getConnection() **throws** SQLException{

**return** DriverManager.*getConnection*(*url*, *username*,*password*);

}

/\*\*

\* **@Method**: release

\* **@Description**: 释放资源，

\* 要释放的资源包括Connection数据库连接对象，负责执行SQL命令的Statement对象，存储查询结果的ResultSet对象

\* **@Anthor**:孤傲苍狼

\*

\* **@param** conn

\* **@param** st

\* **@param** rs

\*/

**public** **static** **void** release(Connection conn,Statement st,ResultSet rs){

**if**(rs!=**null**){

**try**{

//关闭存储查询结果的ResultSet对象

rs.close();

}**catch** (Exception e) {

e.printStackTrace();

}

rs = **null**;

}

**if**(st!=**null**){

**try**{

//关闭负责执行SQL命令的Statement对象

st.close();

}**catch** (Exception e) {

e.printStackTrace();

}

}

**if**(conn!=**null**){

**try**{

//关闭Connection数据库连接对象

conn.close();

}**catch** (Exception e) {

e.printStackTrace();

}

}

}

}

### 2.2、使用statement对象完成对数据库的CRUD操作

**package** com.test;

**import** java.sql.Connection;

**import** java.sql.ResultSet;

**import** java.sql.Statement;

**import** com.test.utils.JdbcUtils;

**import** org.junit.Test;

/\*\*

\* **@ClassName**: JdbcCRUDByStatement

\* **@Description**: 通过Statement对象完成对数据库的CRUD操作

\* **@author**: zhangxt

\* **@date**: 2017-4-19 下午11:31:12

\*

\*/

**public** **class** JdbcCRUDByStatement {

@Test

**public** **void** insert(){

Connection conn = **null**;

Statement st = **null**;

ResultSet rs = **null**;

**try**{

//获取一个数据库连接

conn = JdbcUtils.*getConnection*();

//通过conn对象获取负责执行SQL命令的Statement对象

st = conn.createStatement();

//要执行的SQL命令

String sql = "insert into users(id,name,password,email,birthday) values(4,'白虎神皇','123','bhsh@sina.com','1980-09-09')";

//执行插入操作，executeUpdate方法返回成功的条数

**int** num = st.executeUpdate(sql);

**if**(num>0){

System.***out***.println("插入成功！！");

}

}**catch** (Exception e) {

e.printStackTrace();

}**finally**{

//SQL执行完成之后释放相关资源

JdbcUtils.*release*(conn, st, rs);

}

}

@Test

**public** **void** delete(){

Connection conn = **null**;

Statement st = **null**;

ResultSet rs = **null**;

**try**{

conn = JdbcUtils.*getConnection*();

String sql = "delete from users where id=3";

st = conn.createStatement();

**int** num = st.executeUpdate(sql);

**if**(num>0){

System.***out***.println("删除成功！！");

}

}**catch** (Exception e) {

e.printStackTrace();

}**finally**{

JdbcUtils.*release*(conn, st, rs);

}

}

@Test

**public** **void** update(){

Connection conn = **null**;

Statement st = **null**;

ResultSet rs = **null**;

**try**{

conn = JdbcUtils.*getConnection*();

String sql = "update users set name='孤傲苍狼',email='gacl@sina.com' where id=3";

st = conn.createStatement();

**int** num = st.executeUpdate(sql);

**if**(num>0){

System.***out***.println("更新成功！！");

}

}**catch** (Exception e) {

e.printStackTrace();

}**finally**{

JdbcUtils.*release*(conn, st, rs);

}

}

@Test

**public** **void** find(){

Connection conn = **null**;

Statement st = **null**;

ResultSet rs = **null**;

**try**{

conn = JdbcUtils.*getConnection*();

String sql = "select \* from users where id=3";

st = conn.createStatement();

rs = st.executeQuery(sql);

**if**(rs.next()){

System.***out***.println(rs.getString("name"));

}

}**catch** (Exception e) {

e.printStackTrace();

}**finally**{

JdbcUtils.*release*(conn, st, rs);

}

}

}

## 三、PreparedStatement对象介绍

PreperedStatement是Statement的子类，它的实例对象可以通过调用Connection.preparedStatement()方法获得，相对于Statement对象而言：PreperedStatement可以避免SQL注入的问题。  
　　Statement会使数据库频繁编译SQL，可能造成数据库缓冲区溢出。PreparedStatement可对SQL进行预编译，从而提高数据库的执行效率。并且PreperedStatement对于sql中的参数，允许使用占位符的形式进行替换，简化sql语句的编写。

### 3.1、使用PreparedStatement对象完成对数据库的CRUD操作

测试代码如下：

**package** com.test;

**import** java.sql.Connection;

**import** java.sql.PreparedStatement;

**import** java.sql.ResultSet;

**import** java.util.Date;

**import** com.test.utils.JdbcUtils;

**import** org.junit.Test;

/\*\*

\* **@ClassName**: JdbcCRUDByPreparedStatement

\* **@Description**: 通过PreparedStatement对象完成对数据库的CRUD操作

\* **@author**: zhangxt

\* **@date**: 2017-4-19 下午11:51:12

\*

\*/

**public** **class** JdbcCRUDByPreparedStatement {

@Test

**public** **void** insert(){

Connection conn = **null**;

PreparedStatement st = **null**;

ResultSet rs = **null**;

**try**{

//获取一个数据库连接

conn = JdbcUtils.*getConnection*();

//要执行的SQL命令，SQL中的参数使用?作为占位符

String sql = "insert into users(id,name,password,email,birthday) values(?,?,?,?,?)";

//通过conn对象获取负责执行SQL命令的prepareStatement对象

st = conn.prepareStatement(sql);

//为SQL语句中的参数赋值，注意，索引是从1开始的

/\*\*

\* SQL语句中各个字段的类型如下：

\* +----------+-------------+

| Field | Type |

+----------+-------------+

| id | int(11) |

| name | varchar(40) |

| password | varchar(40) |

| email | varchar(60) |

| birthday | date |

+----------+-------------+

\*/

st.setInt(1, 1);//id是int类型的

st.setString(2, "白虎神皇");//name是varchar(字符串类型)

st.setString(3, "123");//password是varchar(字符串类型)

st.setString(4, "bhsh@sina.com");//email是varchar(字符串类型)

st.setDate(5, **new** java.sql.Date(**new** Date().getTime()));//birthday是date类型

//执行插入操作，executeUpdate方法返回成功的条数

**int** num = st.executeUpdate();

**if**(num>0){

System.***out***.println("插入成功！！");

}

}**catch** (Exception e) {

e.printStackTrace();

}**finally**{

//SQL执行完成之后释放相关资源

JdbcUtils.*release*(conn, st, rs);

}

}

@Test

**public** **void** delete(){

Connection conn = **null**;

PreparedStatement st = **null**;

ResultSet rs = **null**;

**try**{

conn = JdbcUtils.*getConnection*();

String sql = "delete from users where id=?";

st = conn.prepareStatement(sql);

st.setInt(1, 1);

**int** num = st.executeUpdate();

**if**(num>0){

System.***out***.println("删除成功！！");

}

}**catch** (Exception e) {

e.printStackTrace();

}**finally**{

JdbcUtils.*release*(conn, st, rs);

}

}

@Test

**public** **void** update(){

Connection conn = **null**;

PreparedStatement st = **null**;

ResultSet rs = **null**;

**try**{

conn = JdbcUtils.*getConnection*();

String sql = "update users set name=?,email=? where id=?";

st = conn.prepareStatement(sql);

st.setString(1, "gacl");

st.setString(2, "gacl@sina.com");

st.setInt(3, 2);

**int** num = st.executeUpdate();

**if**(num>0){

System.***out***.println("更新成功！！");

}

}**catch** (Exception e) {

e.printStackTrace();

}**finally**{

JdbcUtils.*release*(conn, st, rs);

}

}

@Test

**public** **void** find(){

Connection conn = **null**;

PreparedStatement st = **null**;

ResultSet rs = **null**;

**try**{

conn = JdbcUtils.*getConnection*();

String sql = "select \* from users where id=?";

st = conn.prepareStatement(sql);

st.setInt(1, 1);

rs = st.executeQuery();

**if**(rs.next()){

System.***out***.println(rs.getString("name"));

}

}**catch** (Exception e) {

}**finally**{

JdbcUtils.*release*(conn, st, rs);

}

}

}