

# 一、我们要将业务逻辑抽出来，制作成业务逻辑，不让代码只是显示在一个界面下，同时将常用的代码放到一个封装好的类中去

## 1.连接数据库的类 ConnDB

//从数据库中得到链接

import java.sql.\*;

public class ConnDB {

private Connection ct = null;

public Connection getConn(){

try{

class .forName("com.microsoft.jdbc.sqlserver.SQLServerDriver")

cn = DriverManager.getConnection("","","")

//创建一个Statement}

catch(Exception e){

e.printStackTrace(); }

return cn; }

## 2得到像Android中一样设置一个person用来存储user表中的数据

public class UserBean {

private String name;

private int userId;

public void serUserId(int userId){

this.userId = userId;

}

pubilc int getUserId(){

return this.userId;

}

}

## 3.创建一个userService类，这个类中有些方法用来验证用户是不是存在的

**public** **class** UserService {

Connection cn;

PreparedStatement ps;

ResultSet rs;

//验证用户是不是存在

**public** **boolean** CheckUse(User user){

**boolean** b = **false**;

**try** {

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

cn =DriverManager.*getConnection*("jdbc:mysql://localhost:3306/shouru","root","147094");

ps = cn.prepareStatement("select password from lu where user = ?");

//ps中给对象设计对象

ps.setObject(1, user.getUser());

ResultSet rs = ps.executeQuery();

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

b = **true**;

}

}

**catch** (Exception e) {

e.printStackTrace();

}finally{

//这里有东西我没有写

}

**return** b;

}

}

## 4. yangzheng中，现将我们的user实例出来，然后进行使用



## 5.我刚刚从网上下载了这个SqlHeper，它里面都是关于数据库的连接

**package** control;

**import** java.io.FileInputStream;

**import** java.io.IOException;

**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;

**import** java.sql.\*;

**public** **class** SqlHelper

{

//定义变量

**private** **static** Connection *ct* = **null**;

//大多数情况下用preparedstatement替代statement

**private** **static** PreparedStatement *ps* = **null**;

**private** **static** ResultSet *rs* = **null**;

//连接数据库的参数

**private** **static** String *url* = "";

**private** **static** String *username* = "";

**private** **static** String *driver* = "";

**private** **static** String *passwd* = "";

**private** **static** CallableStatement *cs* = **null**;

**public** **static** CallableStatement getCs()

{

**return** *cs*;

}

**private** **static** Properties *pp* = **null**;

**private** **static** InputStream *fis* = **null**;

//加载驱动，只需要一次，用静态代码块

**static**

{

**try**

{

//从dbinfo.properties

*pp* = **new** Properties();

*fis*=SqlHelper.**class**.getClassLoader().getResourceAsStream("mysql.properties");

//fis = new FileInputStream();

*pp*.load(*fis*);

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

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

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

*passwd* = *pp*.getProperty("passwd");

Class.*forName*(*driver*);

}

**catch** (Exception e)

{

e.printStackTrace();

}

**finally**

{

**try**

{ *fis*.close();}

**catch**(IOException e) {e.printStackTrace();}

*fis* = **null**;//垃圾回收站上收拾

}

}

//得到连接

**public** **static** Connection getConnection()

{

**try**

{*ct* = DriverManager.*getConnection*(*url*,*username*,*passwd*);}

**catch**(Exception e) {e.printStackTrace();}

**return** *ct*;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*callPro1存储过程函数1\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **static** CallableStatement callPro1(String sql,String[] parameters)

{

**try**{

*ct* = *getConnection*();

*cs* = *ct*.prepareCall(sql);

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

**for**(**int** i=0;i<parameters.length;i++){

*cs*.setObject(i+1,parameters[i]);

}

}

*cs*.execute();

}

**catch**(Exception e) { e.printStackTrace(); **throw** **new** RuntimeException(e.getMessage());}

**finally**

{ *close*(*rs*,*cs*,*ct*);}

**return** *cs*;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*callpro2存储过程2\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **static** CallableStatement callPro2(String sql,String[] inparameters,

Integer[] outparameters)

{

**try**

{

*ct* = *getConnection*();

*cs* = *ct*.prepareCall(sql);

**if**(inparameters!=**null**)

{

**for**(**int** i=0;i<inparameters.length;i++)

{

*cs*.setObject(i+1,inparameters[i]);

}

}

//cs.registerOutparameter(2,oracle.jdbc.OracleTypes.CURSOR);

**if**(outparameters!=**null**)

{

**for**(**int** i=0;i<outparameters.length;i++)

{

*cs*.registerOutParameter(inparameters.length+1+i,outparameters[i]);

}

}

*cs*.execute();

}

**catch**(Exception e) {

e.printStackTrace(); **throw** **new** RuntimeException(e.getMessage());

}

**finally**

{

}

**return** *cs*;

}

**public** **static** ResultSet executeQuery(String sql,String[] parameters)

{

**try**

{

*ct*=*getConnection*();

*ps*=*ct*.prepareStatement(sql);

**if**(parameters!=**null**)

{

**for**(**int** i=0;i<parameters.length;i++)

{

*ps*.setString(i+1,parameters[i]);

}

}

*rs* = *ps*.executeQuery();

}

**catch**(Exception e)

{

e.printStackTrace();

**throw** **new** RuntimeException(e.getMessage());

}

**finally**

{

}

**return** *rs*;

}

**public** **static** Connection getCt()

{

**return** *ct*;

}

**public** **static** PreparedStatement getPs()

{

**return** *ps*;

}

**public** **static** ResultSet getRs()

{

**return** *rs*;

}

**public** **static** **void** executeUpdate2(String[] sql,String[][] parameters)

{

**try**

{

*ct* = *getConnection*();

*ct*.setAutoCommit(**false**);

**for**(**int** i=0;i<sql.length;i++)

{

**if**(**null**!=parameters[i])

{

*ps* = *ct*.prepareStatement(sql[i]);

**for**(**int** j=0;j<parameters[i].length;j++)

{

*ps*.setString(j+1,parameters[i][j]);

}

*ps*.executeUpdate();

}

}

*ct*.commit();

}**catch** (Exception e)

{

e.printStackTrace();

**try**

{

*ct*.rollback();

}

**catch** (SQLException e1)

{

e1.printStackTrace();

}

**throw** **new** RuntimeException(e.getMessage());

}**finally**

{

*close*(*rs*,*ps*,*ct*);

}

}

//先写一个update、delete、insert

//sql格式：update 表名 set 字段名 =？where 字段=？

//parameter神应该是（”abc“,23）

**public** **static** **void** executeUpdate(String sql,String[] parameters)

{

**try**

{

*ct*=*getConnection*();

*ps* = *ct*.prepareStatement(sql);

**if**(parameters!=**null**)

{

**for**(**int** i=0;i<parameters.length;i++)

{

*ps*.setString(i+1,parameters[i]);

}

}

*ps*.executeUpdate();

}

**catch**(Exception e)

{

e.printStackTrace();//开发阶段

//抛出异常

//可以处理，也可以不处理

**throw** **new** RuntimeException(e.getMessage());

}

**finally**

{

*close*(*rs*,*ps*,*ct*);

}

}

**public** **static** **void** close(ResultSet rs,Statement ps,Connection ct)

{

//关闭资源(先开后关)

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

{

**try**

{

rs.close();

}

**catch**(SQLException e)

{

e.printStackTrace();

}

rs=**null**;

}

**if**(ps!=**null**)

{

**try**

{

ps.close();

}

**catch**(SQLException e)

{

e.printStackTrace();

}

ps=**null**;

}

**if**(**null**!=ct)

{

**try**

{

ct.close();

}

**catch**(SQLException e)

{

e.printStackTrace();

}

ct=**null**;

}

}

}

## 6.这个里面有好多的方法，我们在使用数据库的时候一定能够用到，现在我们修改下 3 中的 UserService 的方法，使用我们的SqlHelper类中的东西

**public** **class** UserService {

Connection cn;

PreparedStatement ps;

ResultSet rs;

//验证用户是不是合法，下面的and 语句完全避免了我们的

**public** **boolean** CheckUser(User user){

**boolean** b = **false**;

String sql = "select \* from lu where user=? and password = ?";

String parameters[ ] = {user.getUser(),user.getPassword()};

ResultSet rs = SqlHelper.*executeQuery*(sql, parameters) ;

//这样就是根据我们的rs 卡看是不是存在

**try** {

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

b = **true**;

}

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}**finally**{

//这里就是关闭我们的资源

SqlHelper.*close*(rs, SqlHelper.*getPs*(),SqlHelper.*getCt*());

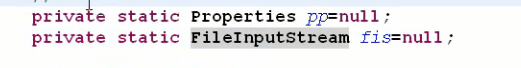
}

**return** b;

}

}

## 7.读取配置文件的



## 8.然后我们测试一下

**try**{

UserService userService = **new** UserService();

User user = **new** User();

user.setPassword("zhang");

user.setUser("zhanng");

**boolean** a = userService.CheckUser(user);

**if**(a){

out.println("正确");

}**else**{

out.println("错误");

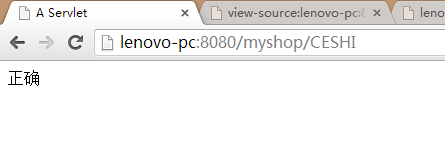
}

}**catch**(Exception e){

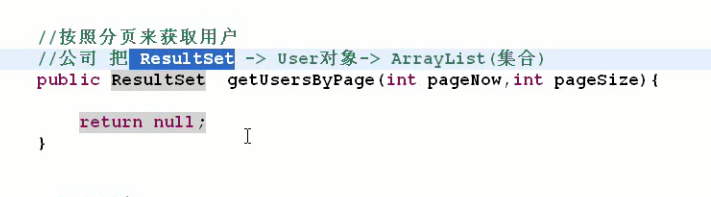
e.printStackTrace();

}

## 9。经过测试改进之后我们发现这个是正确的



# 10，ArrayList这样我们就可以用了，现在我们来进行一下大换血，这里使用UserBean来进行得到用户（下面的这个是不正确的，我们使用的是ArrayList）



10.1这里对上面的进行解释，我们使用ArrayList其实就是对上面的进行，修改，下面是代码

**public**　ArrayList getUserByPage(**int** pageNow,**int** pageSize){

ArrayList<User> arraylist = **new** ArrayList<User>();

ResultSet rs = **null**;

//先查询下

**try** {

String sql = "select \* from book limit "+((pageNow)\*3-3)+","+pageSize;

rs = SqlHelper.executeQuery(sql, **null**);

//进行二次封装

User user = **new** User();

**while** (rs.next()) {

user.setId(rs.getInt(1));

user.setUser(rs.getString(2));

user.setPassword(rs.getString(3));

user.setPhone(rs.getString(4));

user.setEmil(rs.getString(5));

user.setGrade(rs.getString(6));

arraylist.add(user);

}

} **catch** (Exception e) {

e.printStackTrace();

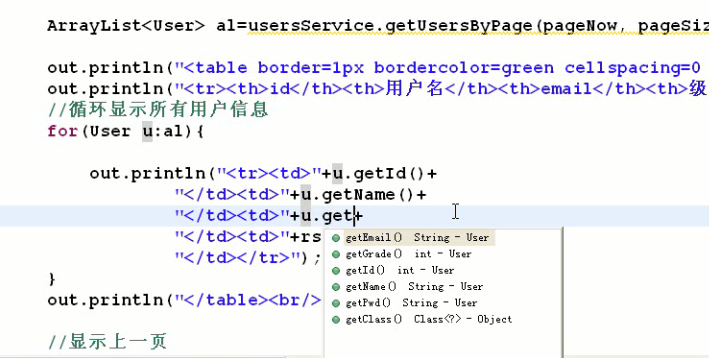
}**finally**{

SqlHelper.close(rs, SqlHelper.getPs(), SqlHelper.getCt());

}

**return** arraylist;

# 11.然后到了我们的分页的地方进行改动



## 12.上面做完了，我们要管理用户，所以我们添加一个删除用户，

//制作一个删除记录的方法，用来删除我们的东西

**public** **boolean** DelUser(User user){

**boolean** a = **true**;

String sql = "delete \* from where user = ?";

String parameters[] = {user.getUser()};

**try** {

SqlHelper.*executeUpdate*(sql, parameters);

} **catch** (Exception e) {

a = **false**;

}

**return** **true**;

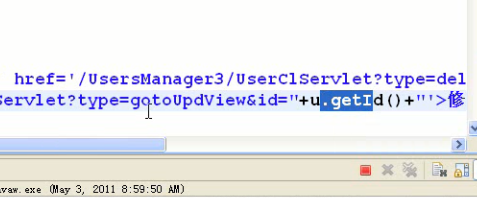
}

## 13.删除然后我们开始测试一下

request.getRequestDispatcher()是请求转发，前后页面共享一个request ;   
response.sendRedirect()是重新定向，前后页面不是一个request。

request.getRequestDispather();返回的是一个RequestDispatcher对象

request.getRequestDispatcher("/ok").forward(request, response);



## 14.现在我们开始制作修改用户了(删除就是现将id传过去就可以了)

1.这里有一个注意的地方就是，有时候我们经常会用到一些绑定的东西，比如删除用户，和修改用户，在发送他们的链接的时候，我们可以只是通过一个参数进行捕获 type



## 2.到了这里首先需要做的就是得到用户的user中的东西，我们创建 （通过id来进行删除东西）

**public** User getUser(**int** id){

User user = **null**;

ResultSet rs = **null**;

**try** {

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

String parameters[] = {""+id};

rs = SqlHelper.*executeQuery*(sql, parameters);

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

user.setId(rs.getInt(1));

user.setUser(rs.getString(2));

user.setPassword(rs.getString(3));

user.setPhone(rs.getString(4));

user.setEmil(rs.getString(5));

user.setGrade(rs.getString(6));

}

} catch (Exception e) {

e.printStackTrace();

}finally{

SqlHelper.*close*(rs, SqlHelper.*getPs*(), SqlHelper.*getCt*());

}

**return** user;

}

## 3.这样我们的User对象就做好了，然后我们就可以 进行了，这个时候我们看到前面看的那个

# 3.1这里重点讲下我们的修改，删除和修改的工呢过成，这个和yanzheng的界面其实是一样的

3.2我们在得到了user之后，但是跳转到另一个页面我们的参数

responceRedirect()

request.getRequestDispatcher()是请求转发，前后页面共享一个request ;



3.3我们上面的那个request.serAttribute(“userinfo”,user)

1.就是相当于创建了一个参数，然后使用下面进行传输

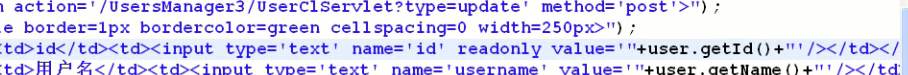
request.getRequestDispatcher("/ok").forward(request, response);

2.然后我们到了ok的界面进行接收，上面的那个是一个User对象，但是一定要进行强制转型



4.我们在修改用户的时候不希望它的id会被修改，所以我们需要它的id是不能被修改的，可以这么设置，这个id将来就是我们修改的时候必须用到的主键，有时我们可能不希望这个现实到桌面上去，这个时候我们就可以将它这个标签设置为影藏了

readonly





12.到了这里我们还需要学习一个按钮，就是 重新填写

<input type=” reset” rvalue=”重新修改”/>

13