2016-2017学年第2学期《WEB程序设计》期末考试

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 专业 | 班级 | 姓名 | 学号 | 分数 |
| 计算机科学与技术 | 141 | 甘志强 | 6103114031 |  |

|  |  |
| --- | --- |
| 项目组编号 | Webgroup 6 |
| 项目名称 | 网上通讯录系统 |
| 同组成员 | 于坤龙，李荣福，钟梦城，黄志龙 |
| 项目分工 | 学生信息前端页面和前端与后台的数据交互 |

**考试说明：**

1、本课程考试采用开放式的课外编程+课内提问检查相结合的方式。

2、要求同学们运用所学知识，在十六周周五之前完成所要求项目的设计与开发，并按本文档要求完成相关报告。

3、第十六周的周二下午和周三上午携带打印好的报告到实验室完成项目检查及回答老师提问。

**项目要求：**

1、项目名称见 题目列表。

2、要求系统采用**MVC架构**来进行设计，尽可能使用**Struts2和Hibernate**框架技术。未能掌握框架技术的同学可采用**JSP+JavaBean+Servlet**来设计及实现，但总分只有85分。

3、除了上述技术要求之外，希望同学们尽可能地运用所学的WEB编程技术，JSTL、EL、过滤器或拦截器、分页、以及前端的CSS和JavaScript技术等。

4、要求所建项目名称为**webGroupX**，其中**GroupX**为分组的编号。

5、要求报告的电子档名称为：**2014计科n班-学号-姓名-系统分析与设计报告**。报告按下页所给格式完成。

《网上通讯录》系统分析及设计报告

一、需求分析

提示：主要分析本系统的用户有哪些，能用系统做什么，并最终分析出系统应该实现的功能模块及功能点。

**本系统的用户面向的应该是在校的整个人群，学生老师都可进入该系统查询到某人的信息。在用户部分，应该有注册登录模块，且应该是真实信息注册登录。对于用虚假信息注册登录的或者考虑到已经毕业的或离开学习的，后台管理员应该具备对信息的审核，删除已经拉入黑名单。在登录成功之后，学生应该能查看和修改自己的信息，以及查询到别人的信息。**

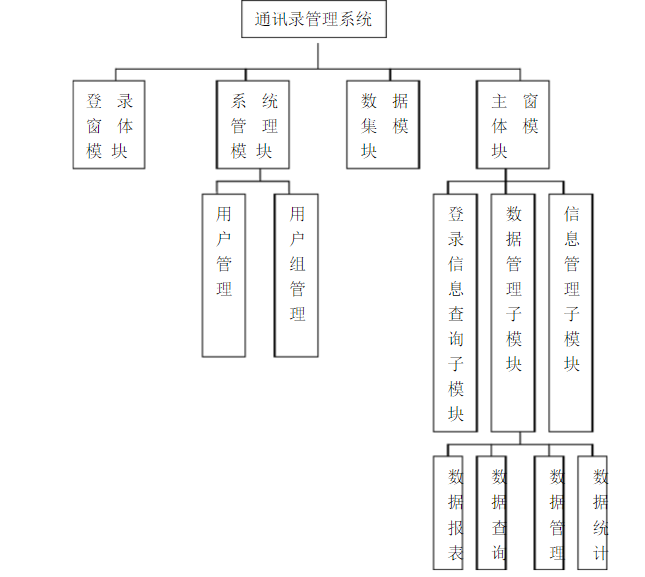
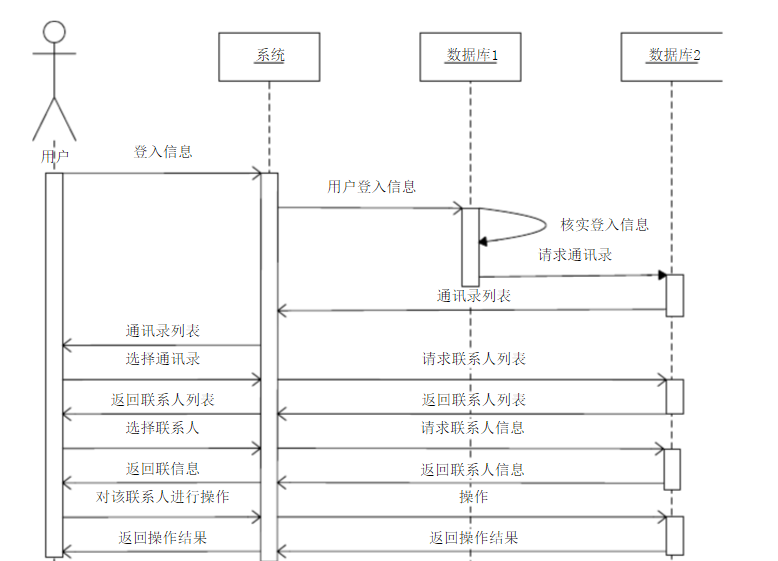






图 1 系统顶级用例图



二、数据库设计

提示：根据需求分析实现本系统功能所需的关系表，及表之间的关系。可画ER图，并给出最终的物理设计

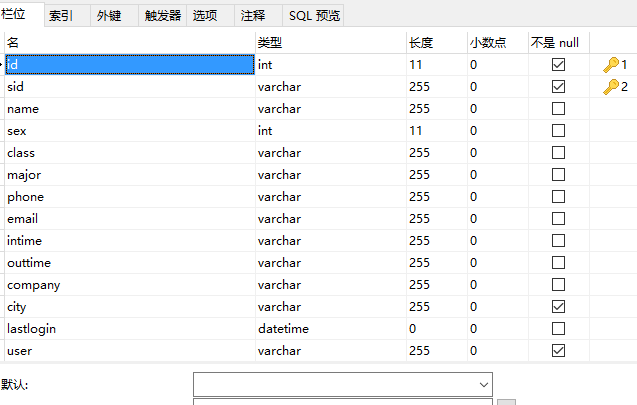
1、学生基本信息表 StuInfo

学号、姓名、年龄、专业、性别、籍贯、……

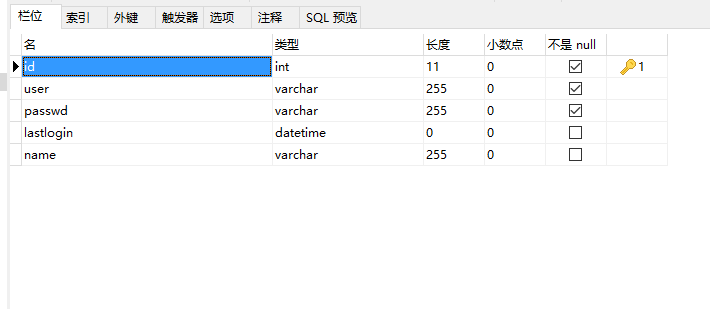
No、Name、Age、Major、Gender、

Id、String varchar（10）、string、int、string、boolean

学生信息表



2、\*\*\*\*\*\*\*\*表



三、架构设计

提示：阐述搭建项目的基本设计思路，阐述项目的分包依据及其中所设计类的作用等。

通讯录分为前端和后台两个部分，前端负责登录注册修改信息以及实现与数据库的交互，而后台负责对用户数据进行删除和拉入黑名单操作，所以在编写时，后台与前端分开进行。主要采用MVC架构，jsp和servlet，filter，ajax，js，html，css等技术的综合使用，前端和后端完全分离。

主要类的作用

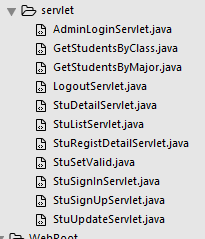
**LoginFilter.java** 过滤器实现对登陆后页面session的查验

**Admin.java** 管理员的类，存基本信息的get和set

**DatabaseConnection.java** 连接数据库

**Stu.java** 学生的类，存基本信息的get和set

接口类，管理员和学生的基本操作



Servlet类，每个servlet完成一个特定的功能

四、功能实现

提示：分功能介绍实现思路，并给出核心业务代码。比如登录验证，只需给出如何验证登录成功的那部分核心代码即可，不需要给出界面代码。

**管理员登陆验证核心代码**

public class AdminLoginServlet extends HttpServlet{

public void doGet(HttpServletRequest req, HttpServletResponse resp)

throws ServletException,IOException{

req.setCharacterEncoding("utf8");

String path = "login.jsp";

String user = req.getParameter("user");

String passwd = req.getParameter("passwd");

Map<String,String> adminInfo = new HashMap<String,String>();

List<String> info = new ArrayList<String>();

if(user == null || "".equals(user)){

info.add("用户名为空");

}

if(passwd == null || "".equals(passwd)){

info.add("密码为空");

}

if(info.size()==0){

Admin admin = new Admin();

admin.setUser(user);

admin.setPasswd(passwd);

try{

if(DAOFactory.getIAdminDAOInstanse().findLogin(admin)){

adminInfo.put("name", admin.getName());

adminInfo.put("lastlogin", admin.getLastLogin());

String[] value = new String[2];

if(DAOFactory.getIStuDAOInstanse().getTotalNum(value)){

adminInfo.put("usernum", value[0]);

}else{

System.out.println("something is wrong here ....");

}

req.setAttribute("adminInfo", adminInfo);

req.setAttribute("login", 1);

req.setAttribute("user",user);

}else{

req.setAttribute("fail","登录失败用户名或密码错误");

}

}catch(Exception e){

e.printStackTrace();

}

}

req.setAttribute("info", info);

req.getRequestDispatcher(path).forward(req, resp);

}

public void doPost(HttpServletRequest req, HttpServletResponse resp)

throws ServletException, IOException{

this.doGet(req, resp);

}

}

**管理员接口**

public interface IAdminDAO {

public boolean findLogin(Admin admin) throws Exception;

public boolean logout(Admin admin) throws Exception;

}

**学生接口**

public interface IStuDAO {

public boolean getStuList(String[] json,int isvalid) throws Exception;

public boolean getDetail(String[] json, String sid) throws Exception;

public boolean setValid(Stu stu,int i) throws Exception;

public boolean findLogin(Stu student) throws Exception;

public boolean getTotalNum(String[] value) throws Exception;

public boolean getStudentsByClass(String[] json, String c) throws Exception;

public boolean getStudentsByMajor(String[] json, String m) throws Exception;

public boolean regist(Stu stu)throws Exception;

public boolean registdetail(Stu stu) throws Exception;

public boolean update(Stu stu) throws Exception;

}

**管理员操作实现**

public class AdminDAOImpl implements IAdminDAO{

private Connection conn = null;

private PreparedStatement pstmt = null;

public AdminDAOImpl(Connection conn){

this.conn = conn;

}

public boolean findLogin(Admin admin) throws Exception{

boolean flag = false;

try{

String sql = "SELECT name,lastlogin FROM administrator WHERE user=? AND passwd=?";

this.pstmt = this.conn.prepareStatement(sql);

this.pstmt.setString(1,admin.getUser());

this.pstmt.setString(2,admin.getPasswd());

ResultSet rs = this.pstmt.executeQuery();

if(rs.next()){

admin.setName(rs.getString(1));

admin.setLastLogin(rs.getString(2));

flag = true;

}

}catch(Exception e){

throw e;

}finally{

if(this.pstmt!=null){

try{

this.pstmt.close();

}catch(Exception e){

throw e;

}

}

}

return flag;

}

public boolean logout(Admin admin) throws Exception{

boolean flag= false;

try{

String sql = "UPDATE `administrator` SET `lastlogin`=now() WHERE (user=?)";

this.pstmt = this.conn.prepareStatement(sql);

this.pstmt.setString(1,admin.getUser());

if(this.pstmt.execute())

flag = true;

}catch(Exception e){

e.printStackTrace();

}finally{

if(this.pstmt!=null){

try{

this.pstmt.close();

}catch(Exception e){

throw e;

}

}

}

return flag;

}

}

**数据库连接**

public class DatabaseConnection {

private static final String DBDRIVER = "com.mysql.jdbc.Driver";

private static final String DBNAME = "webgroup6";

private static final String USER = "admin";

private static final String PASSWD = "admin";

private static final String DBURL = "jdbc:mysql://115.159.86.195:3306/"+DBNAME+"?useUnicode=true&characterEncoding=UTF-8";

private Statement stmt;

private Connection conn = null;

private ResultSet rs = null;

public DatabaseConnection() {

try {

Class.forName(DBDRIVER).newInstance();

System.out.println("驱动连接错误");

} catch (IllegalAccessException e) {

e.printStackTrace();

} catch (InstantiationException e) {

e.printStackTrace();

} catch (ClassNotFoundException e) {

e.printStackTrace();

}

try {

conn = DriverManager.getConnection(DBURL, USER, PASSWD);

stmt = conn.createStatement();

System.out.println("连接地址错误");

} catch (SQLException e) {

System.out.println("用户名或密码错误");

e.printStackTrace();

}

}

public Statement getStat() {

return stmt;

}

public Connection getConn() {

return conn;

}

public ResultSet getRs() {

return rs;

}

public void close()throws Exception{

if(this.conn!= null){

try{

if(this.rs!=null)

this.rs.close();

if(this.stmt!=null)

this.stmt.close();

this.conn.close();

}catch(Exception e){

e.printStackTrace();

}

}

}

}

**学生操作实现**

public class StuDAOProxy implements IStuDAO{

private DatabaseConnection dbc = null;

private IStuDAO dao = null;

public StuDAOProxy(){

try{

this.dbc = new DatabaseConnection();

}catch(Exception e){

e.printStackTrace();

}

this.dao = new StuDAOImpl(this.dbc.getConn());

}

@Override

public boolean getStuList(String[] json, int isvalid) throws Exception {

// TODO Auto-generated method stub

boolean flag =false;

try{

flag = this.dao.getStuList(json, isvalid);

}catch(Exception e){

throw e;

}finally{

if(this.dbc!=null)

this.dbc.close();

}

return flag;

}

public boolean findLogin(Stu student) throws Exception{

boolean flag =false;

try{

flag = this.dao.findLogin(student);

}catch(Exception e){

throw e;

}finally{

this.dbc.close();

}

return flag;

}

@Override

public boolean getDetail(String[] json, String sid) throws Exception {

// TODO Auto-generated method stub

boolean flag =false;

try{

flag = this.dao.getDetail(json, sid);

}catch(Exception e){

throw e;

}finally{

if(this.dbc!=null)

this.dbc.close();

}

return flag;

}

@Override

public boolean setValid(Stu stu, int i) throws Exception {

// TODO Auto-generated method stub

boolean flag =false;

try{

flag = this.dao.setValid(stu, i);

}catch(Exception e){

throw e;

}finally{

if(this.dbc!=null)

this.dbc.close();

}

return flag;

}

@Override

public boolean getTotalNum(String[] value) throws Exception {

// TODO Auto-generated method stub

boolean flag =false;

try{

flag = this.dao.getTotalNum(value);

}catch(Exception e){

throw e;

}finally{

if(this.dbc!=null)

this.dbc.close();

}

return flag;

}

@Override

public boolean getStudentsByClass(String[] json, String c) throws Exception {

// TODO Auto-generated method stub

boolean flag =false;

try{

flag = this.dao.getStudentsByClass(json, c);

}catch(Exception e){

throw e;

}finally{

if(this.dbc!=null)

this.dbc.close();

}

return flag;

}

@Override

public boolean getStudentsByMajor(String[] json, String m) throws Exception {

// TODO Auto-generated method stub

boolean flag =false;

try{

flag = this.dao.getStudentsByMajor(json, m);

}catch(Exception e){

throw e;

}finally{

if(this.dbc!=null)

this.dbc.close();

}

return flag;

}

@Override

public boolean regist(Stu stu) throws Exception {

boolean flag =false;

try{

flag = this.dao.regist(stu);

}catch(Exception e){

throw e;

}finally{

if(this.dbc!=null)

this.dbc.close();

}

return flag;

}

@Override

public boolean registdetail(Stu stu) throws Exception {

// TODO Auto-generated method stub

boolean flag =false;

try{

flag = this.dao.registdetail(stu);

}catch(Exception e){

throw e;

}finally{

if(this.dbc!=null)

this.dbc.close();

}

return flag;

}

@Override

public boolean update(Stu stu) throws Exception {

// TODO Auto-generated method stub

boolean flag =false;

try{

flag = this.dao.update(stu);

}catch(Exception e){

throw e;

}finally{

if(this.dbc!=null)

this.dbc.close();

}

return flag;

}

}

分页实现文件：

Page.java：

package zmc.util;

/\*\*

\*

\* @author Administrator

\*/

public class Page {

private int everyPage; //每页显示数量

private int totalCount; //总记录数

private int totalPage; //总页数

private int currentPage; //当前页

private int beginIndex; //起始点

private boolean hasPrePage; //是否有上一页

private boolean hasNextPage; //是否有下一页

public Page(int everyPage, int totalCount, int totalPage, int currentPage,int beginIndex, boolean hasPrePage, boolean hasNextPage)

{

this.everyPage = everyPage;

this.totalCount = totalCount;

this.totalPage = totalPage;

this.currentPage = currentPage;

this.beginIndex = beginIndex;

this.hasPrePage = hasPrePage;

this.hasNextPage = hasNextPage;

}

public Page() {

}//构造函数，默认

public int getEveryPage() {

return everyPage;

}

public void setEveryPage(int everyPage) {

this.everyPage = everyPage;

}

public int getTotalCount() {

return totalCount;

}

public void setTotalCount(int totalCount) {

this.totalCount = totalCount;

}

public int getTotalPage() {

return totalPage;

}

public void setTotalPage(int totalPage) {

this.totalPage = totalPage;

}

public int getCurrentPage() {

return currentPage;

}

public void setCurrentPage(int currentPage) {

this.currentPage = currentPage;

}

public int getBeginIndex() {

return beginIndex;

}

public void setBeginIndex(int beginIndex) {

this.beginIndex = beginIndex;

}

public boolean isHasPrePage() {

return hasPrePage;

}

public void setHasPrePage(boolean hasPrePage) {

this.hasPrePage = hasPrePage;

}

public boolean isHasNextPage() {

return hasNextPage;

}

public void setHasNextPage(boolean hasNextPage) {

this.hasNextPage = hasNextPage;

}

}

PageUtil.java：

package zmc.util;

/\*\*

\*

\* @author Administrator 分页信息辅助类

\*/

public class PageUtil {

public static Page createPage(int everyPage, int totalCount, int currentPage) {//创建分页信息对象

everyPage = getEveryPage(everyPage);

currentPage = getCurrentPage(currentPage);

int totalPage = getTotalPage(everyPage, totalCount);

int beginIndex = getBeginIndex(everyPage, currentPage);

boolean hasPrePage = getHasPrePage(currentPage);

boolean hasNextPage = getHasNextPage(totalPage, currentPage);

return new Page(everyPage, totalCount, totalPage, currentPage,

beginIndex, hasPrePage, hasNextPage);

}

public static int getEveryPage(int everyPage) { //获得每页显示记录数

return everyPage == 0 ? 10 : everyPage;

}

public static int getCurrentPage(int currentPage) { //获得当前页

return currentPage == 0 ? 1 : currentPage;

}

public static int getTotalPage(int everyPage, int totalCount) {//获得总页数

int totalPage = 0;

if (totalCount % everyPage == 0) {

totalPage = totalCount / everyPage;

} else {

totalPage = totalCount / everyPage + 1;

}

return totalPage;

}

public static int getBeginIndex(int everyPage, int currentPage) {//获得起始位置

return (currentPage - 1) \* everyPage;

}

public static boolean getHasPrePage(int currentPage) {//获得是否有上一页

return currentPage == 1 ? false : true;

}

public static boolean getHasNextPage(int totalPage, int currentPage) { //获得是否有上一页

return currentPage == totalPage || totalPage == 0 ? false : true;

}

}

action文件：

ShowStudent.java：

package zmc.action;

/\*\*

\*

\* @author Administrator

\*/

import java.util.List;

import org.apache.struts2.ServletActionContext;

import com.opensymphony.xwork2.ActionSupport;

import zmc.bean.Student;

import zmc.dao.StudentDAO;

import zmc.factory.StudentDAOFactory;

import zmc.util.Page;

import zmc.util.PageUtil;

public class ShowStudentAction extends ActionSupport {

private int currentPage;//当前页

public int getCurrentPage() {//获得当前页

return currentPage;

}

public void setCurrentPage(int currentPage) {//设置当前页

this.currentPage = currentPage;

}

@Override

public String execute() throws Exception {

StudentDAO studentDAO = StudentDAOFactory.getStudentDAOInstance();// 通过DAO工厂类获得DAO实现类实例

List<Student> studentList = studentDAO.findAllStudent();// 查找所有商品记录

Page page = PageUtil.createPage(3, studentList.size(), currentPage);//得到分页信息对象

int endIndex = page.getBeginIndex() + page.getEveryPage();//终点

if ((page.getBeginIndex() + page.getEveryPage()) > page.getTotalCount()) {

endIndex = page.getTotalCount();

}

studentList = studentList.subList(page.getBeginIndex(), endIndex);//获得子记录

ServletActionContext.getRequest().setAttribute("studentList",studentList);//保存记录到request范围

ServletActionContext.getRequest().setAttribute("page",page);//保存分页信息到request范围

return this.SUCCESS;

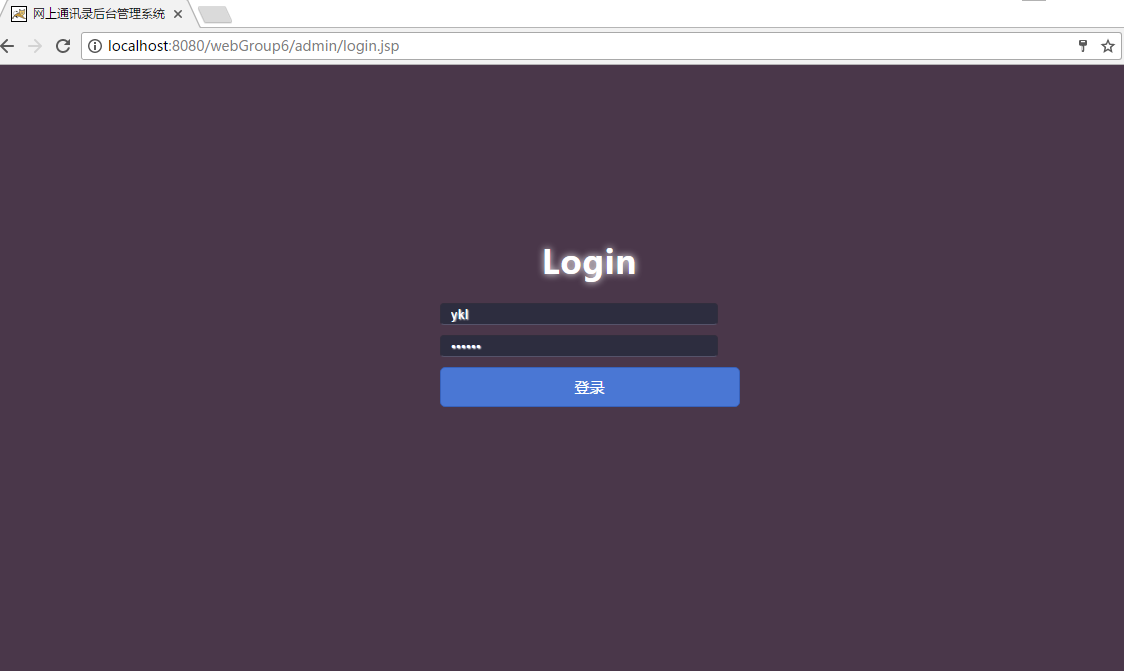
}

}

分页功能的实现是利用Struts2框架，从导入相应的jar包，配置好struts.xml和web.xml文件，在struts.xml写入相应的动作，在从数据库中调取数据，以此来实现分页。

用户注册功能校验器的实现：是利用JSTL标签库，主要用<c:>和EL表达式来对用户输入的文本进行校验，有相应的提示功能。

五、界面成果展示



管理员系统登陆页面



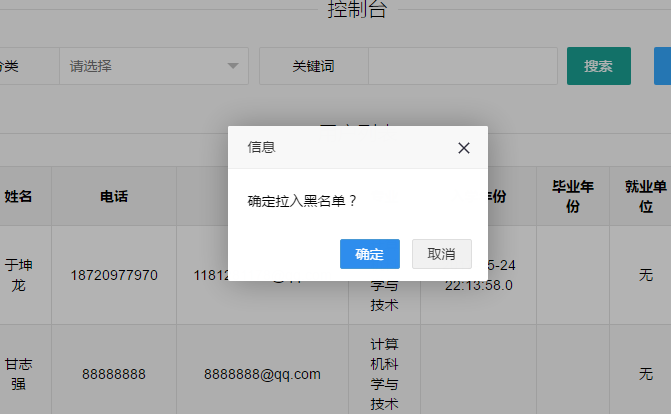
登陆成功后的页面



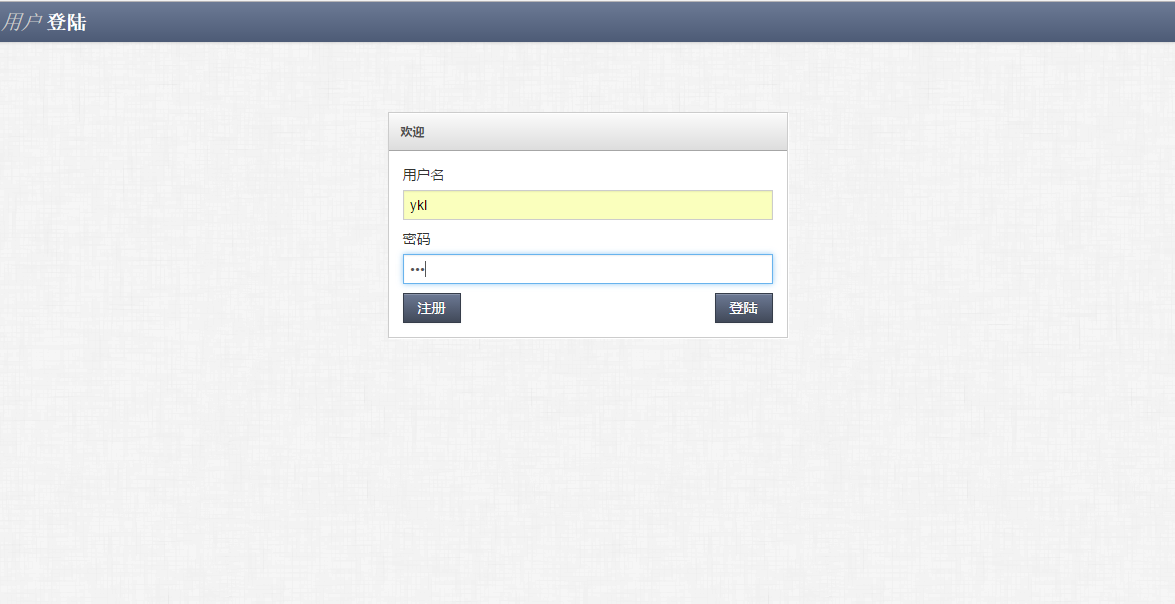
登陆失败后的温馨提示，细节处理







拉入黑名单时的一些细节处理



学生用户的登陆界面



登陆成功后的主页面



个人信息展示页面，用了session临时保存数据



个人信息修改页面，考虑到一些不能修改的字段。比如性别，学号等



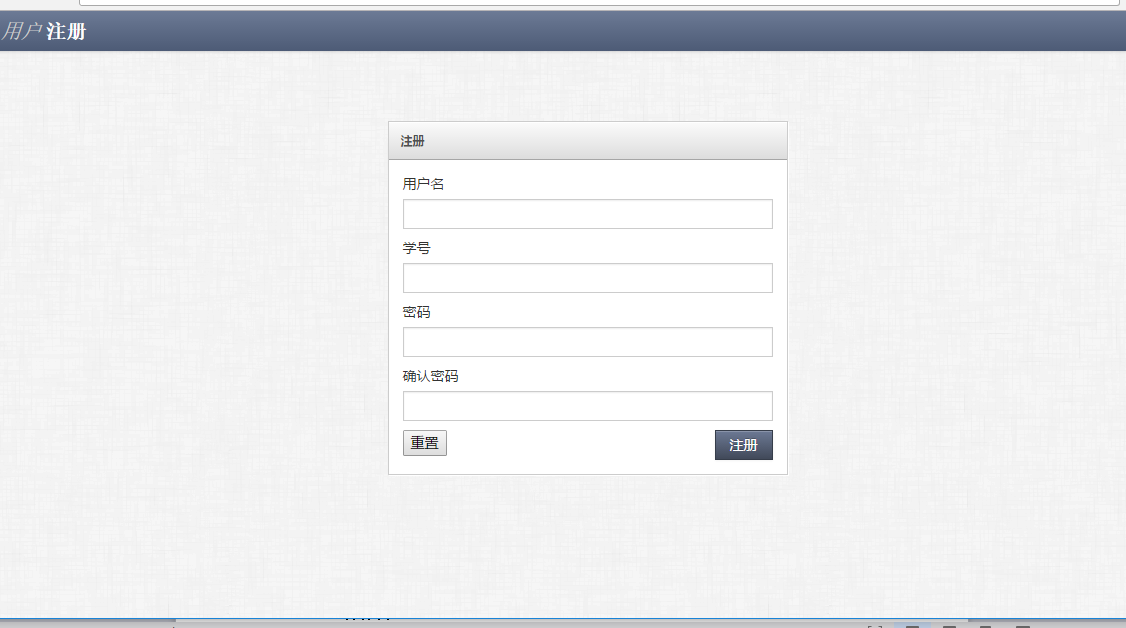
通讯录查询的两个接口，专业通讯录查询和班级通讯录查询







注销的时候要消除session



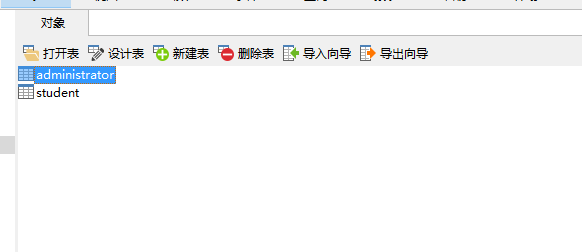
用户注册页面



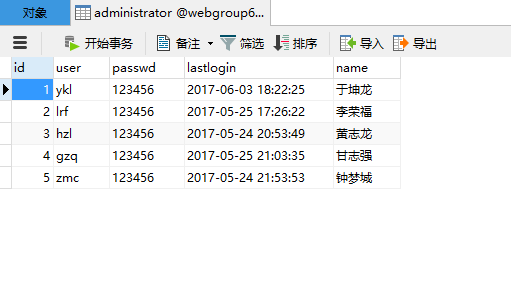
在注册成功的时候要跳转到登记信息页面

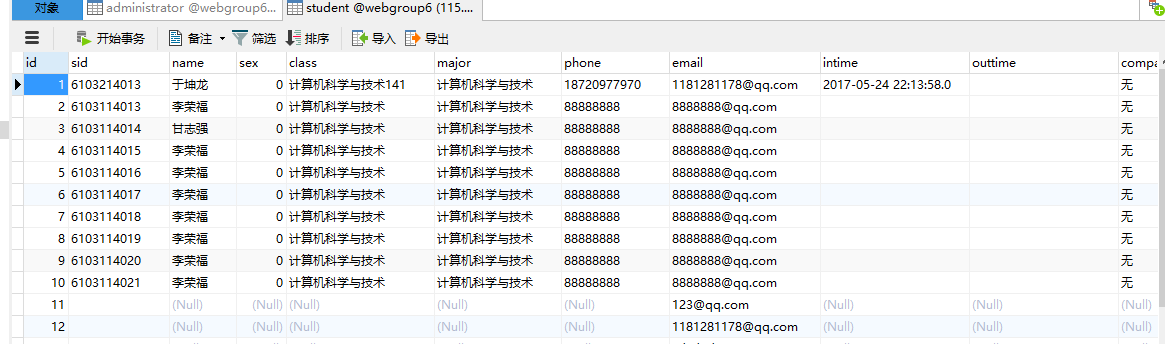


完成信息登记后跳转主页面（本打算跳转待审核页面）



我们建了两个表





五、界面成果展示

提示：展示所完成功能的界面。

六、心得体会

提示：不少于500字。可阐述遇到的问题，解决问题的思路、过程或办法等。**阅读后请删除**！

在此次JAVAWEB期末大作业中，我们小组选中的是网上通讯录。在得知题目后，我们就确定了组员的分工情况，在后台与前端的完成中，我选择的是前端页面的实现。由于存在后台审核功能，所以前端页面分为大部分的登录，注册，用户信息显示，用户信息修改，通讯录查询的用户页面，和审核，删除禁用的管理员页面，而我所做的为用户信息这一块。

在编写代码之前，我现在纸上将用户信息所需要页面以及页面该显示的信息都大致画了出来，然后确认的各个页面之间的联系，诸如各个按钮的功能跳转的页面。确定了页面内容之后我开始编写代码实现。

前端页面的编写大体上难度是不大的，考虑到有用户信息修改与完善这个功能，我在注册页面只让用户很简单的填写的信息。然后跳转到审核的页面，经过后台管理员的审核后用户进入详细的通讯录信息页面，在填写完专业班级学号电话等一系列信息之后，用户便可以登录通讯录。在主页面上，我设置了个人信息显示，个人信息修改和通讯录查询，在个人信息修改页面增添完善了用户信息，比如用户需要填写自己的专业班级电话地址等等，而个人信息显示这一页面是为了用户能查看自己的信息确认是否有错误。

在基本完成了页面的时候，我想用户在注册等填写信息的时候，添加一个重置按钮有的时候会更加方便快捷，所以我添加了<input type="reset" value="重置" width="" height="" > 重置按钮。而在信息显示页面，考虑到我们使用的是MVC框架，在显示数据库信息的时候，应该是将数据从数据访问层取出来，然后进行业务逻辑层封装，最后完成控制层和页面交互显示，所以我们采用了JSON格式给数据到前端，然后通过AJAX技术获取后端的数据，然后对所给的数据进行页面展示。AJAX的优势在于可以异步，在进行数据更新的时候不必每次都刷新页面。

最后，在此次大作业的实验中，我体会到团队合作的重要性，想要完成一个项目，单靠一个人的力量是很难的，而团队成员相互配合，相互分工，就能事半功倍。