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 |
| 计科 | 141班 | 钟梦城 | 6103114015 |  |

|  |  |
| --- | --- |
| 项目组编号 | webGroup6 |
| 项目名称 | 网上通讯录设计 |
| 同组成员 | 李荣福、钟梦城、甘志强、黄志龙 |
| 项目分工 | 管理员系统后台实现，和用户系统部分后台实现 |

**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。

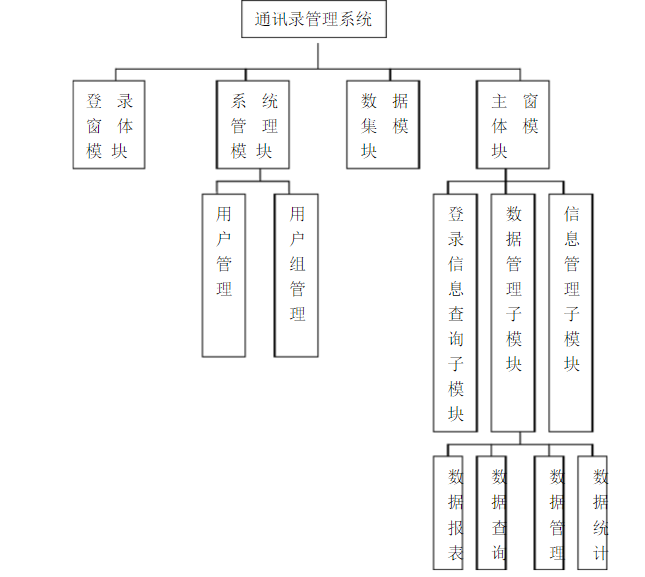
3、除了上述技术要求之外，希望同学们尽可能地运用所学的WEB编程技术，JSTL、EL、过滤器或拦截器、分页、以及前端的CSS和JavaScript技术等。

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



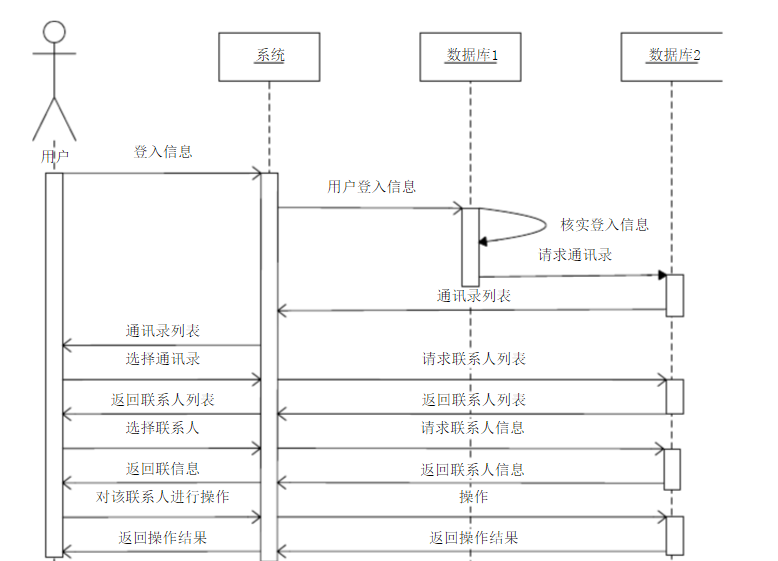
**本系统主要是为了更好的管理每个人的通讯录，给每个人提供一个井然有序的管理平台，防止手工管理混乱，造成不必要的麻烦。**

**本系统主要实现我们信息的管理，包括我们平时朋友信息的添加、修改、查询和删除，及一些辅助项目的管理，如登录系统用户管理和用户组管理。**

**使用本系统的用户分为两类：管理员和用户**

**经过对各种通讯录系统的考察、分析，要求本系统具有以下功能： （1）统一友好的操作界面，保证系统的易用性，方便用户的操作； （2）具备联系人信息的显示功能，方便用户及时查看联系人信息； （3）联系人信息管理功能，用户可以添加、删除联系人信息； （4）个性化的联系人分组，方便用户分类检索联系人信息； （5）全面的后台管理功能，以方便管理员对用户信息进行管理。**





系统前台功能需求

（1）首页信息 显示开发的信息，包括：系统开发开发环境、主要采用的技术、系统开发时间

（2）分组信息 可以查看在当前数据库通讯录系统中的总的分组情况。

（3）联系人信息 可以查看当前权限下通讯录中所有的联系人的全部信息，并且可以按照不同的条件（比如姓名、出生日期等），对联系人进行查找，并且可以查看联系人的具体信息（包括个人照片等）。

（4）后台登录 可以点击后台登录，对当前用户权限下的通讯录进行后台管理。

系统后台功能需求

（1）分组信息管理

包括添加分组信息和分组信息管理两个模块，可以添加分组和对已经建立的分组进行编辑和删除。

（2）联系人信息管理

包括添加联系人和联系人信息管理两个模块，可以新增加联系人和对已经记录的联系人信息进行编辑和删除。

（3）系统管理

包括修改密码和退出系统两个模块，可以完成对系统的密码进行修改和退出本系统两个操作系统后台功能需求。

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。

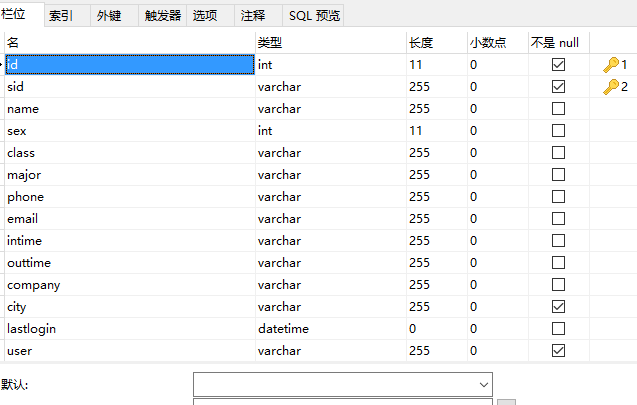
1. 学生基本信息表 student

需要分析该系统功能所隐含的对数据应用的需求，从而确定数据库的结构。

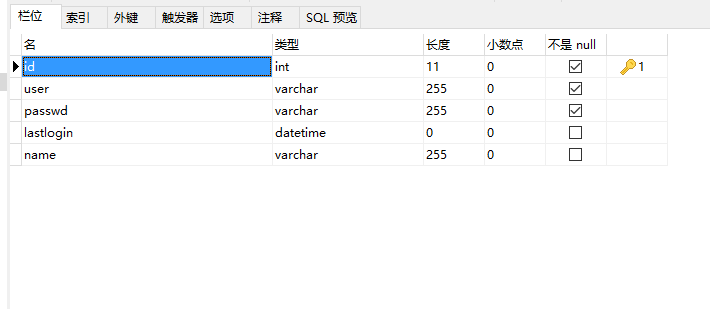
（1）对用户信息建立数据表，其中的数据项包括登录账号、登录密码、学号等。

（2）对分组信息建立数据表，其中的数据项包括分组编号、分组名等。

（3）对联系人信息建立数据表，其中的数据项包括所在分组、姓名、性别、专业、班级、电话、邮箱地址、住址、入学年份、毕业年份等



1. 管理员基本信息表 administrator



三、架构设计

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

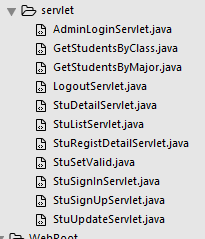
**LoginFilter.java** 过滤器实现对登陆后页面session的查验

**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

**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;

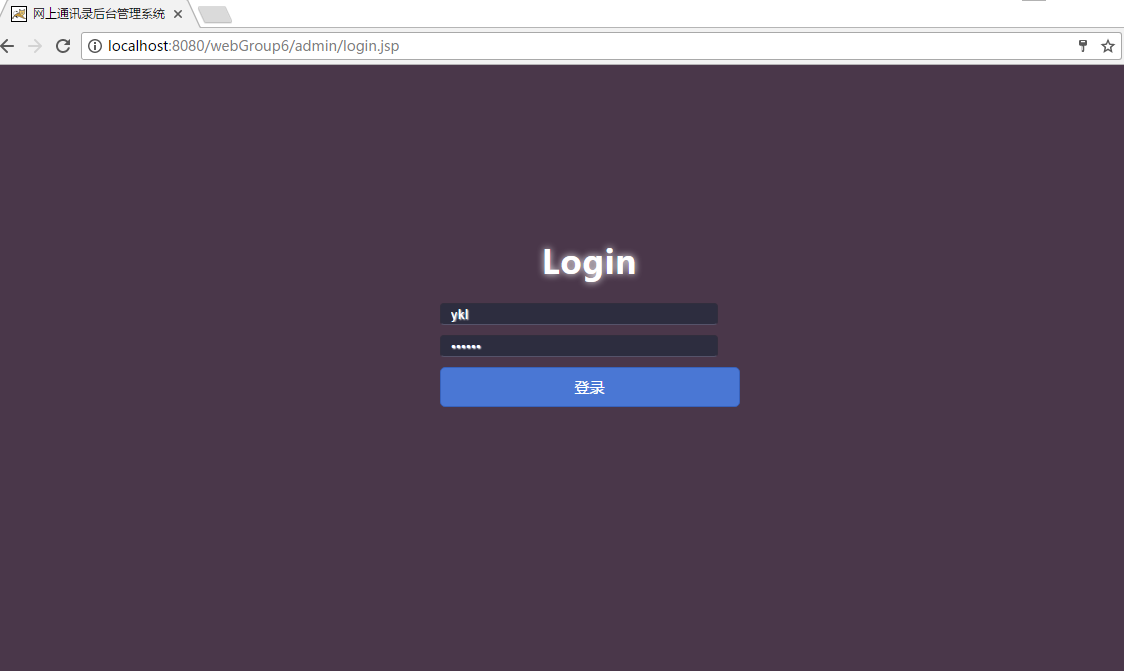
}

}

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



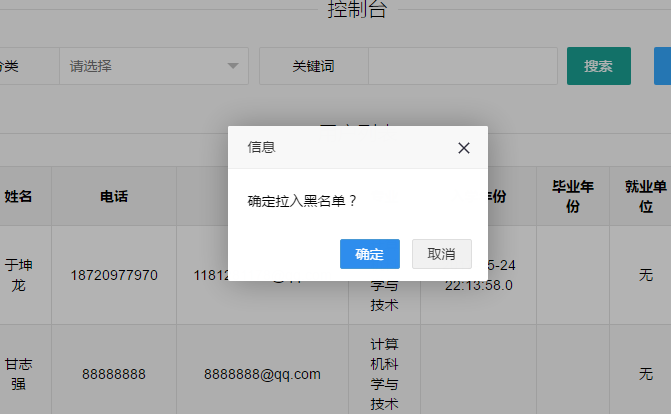
登陆成功后的页面



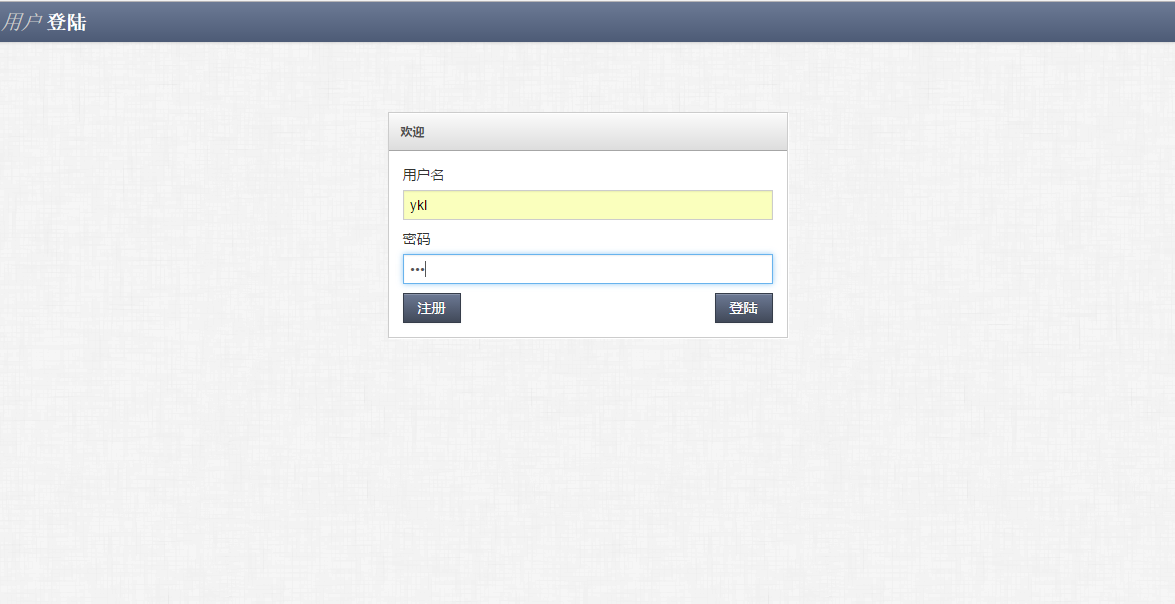
登陆失败后的温馨提示，细节处理







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



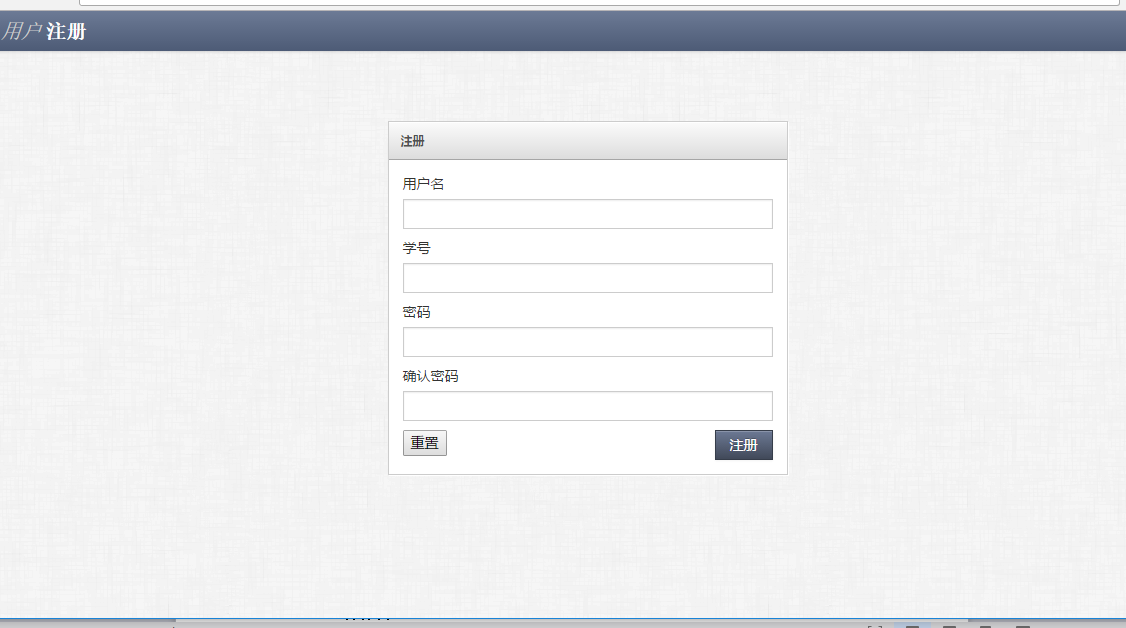
通讯录查询的两个接口，专业通讯录查询和班级通讯录查询







注销的时候要消除session



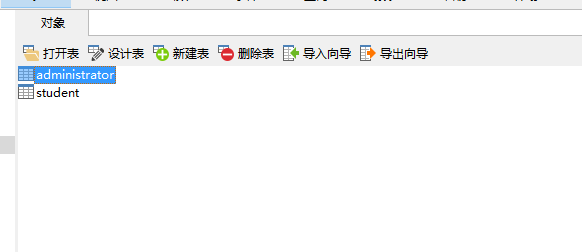
用户注册页面



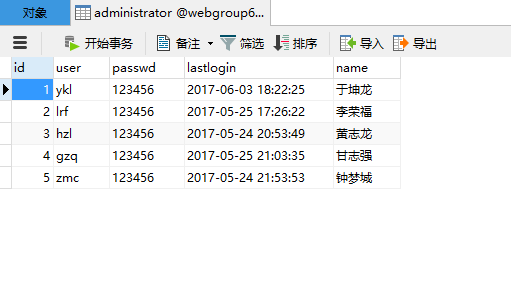
在注册成功的时候要跳转到登记信息页面

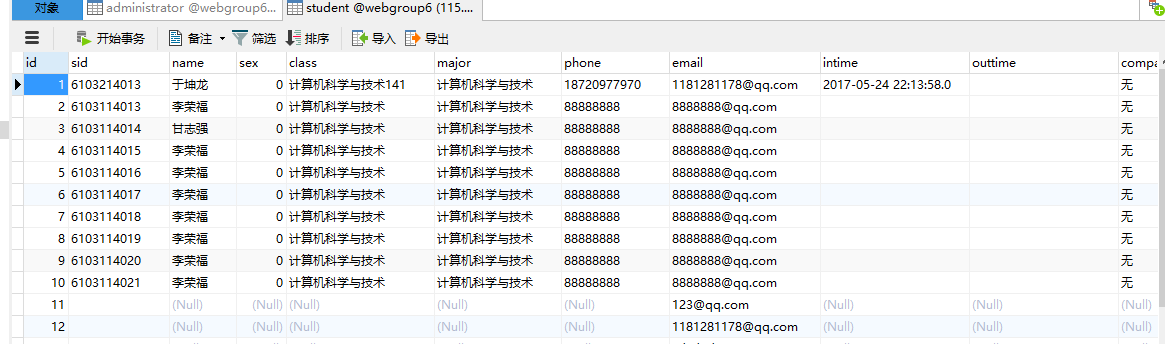


完成信息登记后跳转主页面（本打算跳转待审核页面）



我们建了两个表





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**！

《网上通讯录项目开发》的心得体会：

这次大作业设计让我受益匪浅，我从时间的远近上给老师讲讲我的感受吧。

这次实验我第一个收获是了解了一般项目的开发过程。面对一个项目时，首先要想清楚这个项目“是什么”，然后在大概的想清楚“做什么”，最后才是具体的实现——“怎样做”，如果这三点没有考虑清楚，面对一个项目就会感觉是一团乱麻，无从下手。在设计初期，我在得到老师对实验的要求时，给我的第一感觉是这个项目是一个信息管理系统，系统要分为两个模块，用户和管理员模块，用户模块上要实现用户的登录注册，用户自己信息的完善，用户对自己联系人要可以实现增删改查等一系列功能，管理员要做好用户的注册审核，用户信息的管理等。具体怎样做，我们就在网上找那些项目模版，找是找到了，但网上提供的模版给我的感觉是“太草率了”，这些模版只是实现了相关页面的显示效果和跳转功能，但在具体细节上，完全是一塌糊涂，比如：用户的联系人管理页面，看上去的确有一个表，表内有各种信息，也配置了两个按钮，下面还有分页数字，但真正的了解了这些模版后，我才知道，那个表真就是只有一个表的样子，里面的数据并不是通过调用，从数据库里面取得，而是在网页源代码上都有的“直接硬添上去的数据”，就只是有一个表的样子；按钮也有两个，但不实现相关的跳转功能；分页就更强了，直接就是一个链接，跳到下一个页面，而不是根据联系人数目来分的页，总的来说，就是华而不实，外强中干，只是有一个表现效果，没有功能实现，一想到这，就觉得这模版用与不用都差不多了。基本上还是要靠自己。

于是我们只能自己来解决系统的设计问题。首先，当然是考虑数据库的连接，开发框架的搭建。在数据库的问题上，考虑到我们是线下开发，数据库的不同步会影响开发进度，所以我们选择了一个远程服务器上的数据库作为我们的项目开发的数据库。我们部分的功能实现是通过JSP+Servlet+AJAX的方式实现的，部分功能采用了Struts2框架，在对数据库的操作上采用了Struts2框架来对action进行封装实现，在进行分页工作时采用了Struts2框架，因为书中对这两方面有十分详细的介绍，在开发时，我们也有使用JSON技术把数据传到前端，用AJAX技术获取后端的数据，然后对数据进行页面展示。

实验中期，最为头痛的功能实现时期。大作业前面的实验，老师说的实现的基本功能我都能够实现，有些突然提出的没有做到。所以我做了用户登录注册，还有联系人分页的工作，以此来熟悉我之前不懂的两块：校验器和分页。我先是用期中考试的表用来测试实现分页功能，遇到的问题有，在Struts2环境的搭建上时，会找不到配置struts.xml文件，一开始，我以为是我把配置文件放错了位置，然后就在不断的换在不同的文件夹中，不断的尝试，但他就是找不到，然后只能上网搜，网上提供的资料也是乱七八糟的，最后我只能靠对比配置成功的人的文件，才知道是在struts.xml文件的开头有个空格，而这个配置文件在这句<?xml version="1.0" encoding="UTF-8" ?>声明前面什么都不能有，包括空格。这问题我觉得就是来恶心我的，这样的错，让我十分无语。还有一个类似的错误是在做下拉列表来区分登录类型时遇到过一个JasperException的异常，原因是逗号是中文的逗号。在做分页时我建好了Page类，建好了Action后，按书上敲好代码，这次的错没有错误提示，就是没有表中数据，这样的错反而更麻烦，有异常提示，我至少能知道错在了哪方面，现在，他的意思就是代码本身并没有什么错，只是不知道哪里出了错，导致了这个你不想要的结果。我在想为什么时，想过“是不是相关的标签没有设置，或者难道是我的Struts2框架还是没有搭建好，又或者是哪里的方法写错，或者调错了”，这是我觉得在编程工作上最可怕的问题，他会让你对自己写的东西产生怀疑，甚至失去信心，没有错，又不能在网上搜索资料，最后还是在书中找到了答案，发现了一个细节，就是不是直接打开一个jsp页面，而是在action之后，才调出界面。对程序员来说，看见那些令人讨厌的一堆异常结果不可怕，可怕的是，没有异常，却得不到想要的结果。这更让人头疼。

实验后期就是对试验的总结和文档的编写。这里校考的是文字功底，只要熟悉编程实现过程，问题就不大。虽然我们在共同努力下完成了项目，但是还有很多细节我们并没有处理，像是没有考虑系统安全问题，没有敏感词侦测，用户的联系人列表没有实现分类等等。

最后的领悟就是，一个完整的项目是需要团队共同完成的。一个人难做到面面俱到，在团队合作时，要做好充分的沟通，最好能做到像计算机网络上的“无缝衔接”，齐心协力才能做好一件大事。还有就是在编程上面对错误要有耐心，对自己的代码要有信心。