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 | 计科141 | 于坤龙 | 6103214013 |  |

|  |  |
| --- | --- |
| 项目组编号 | WebGroup6 |
| 项目名称 | 网上通讯录 |
| 同组成员 | 李荣福、甘志强、黄志龙、钟梦城 |
| 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。

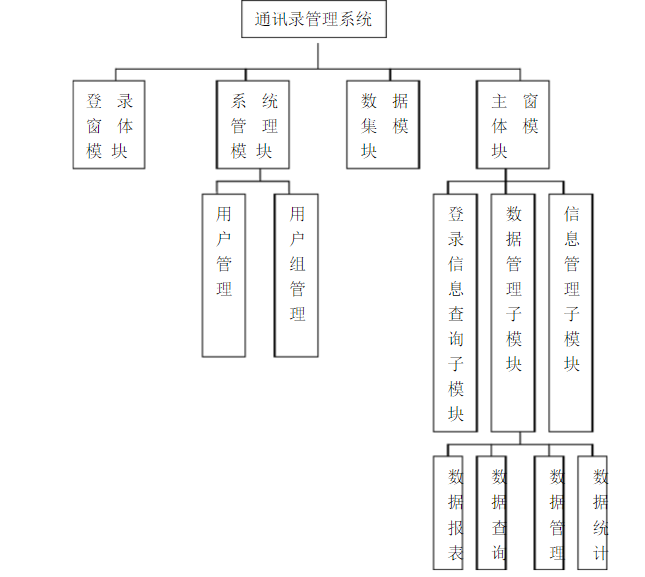
3、除了上述技术要求之外，希望同学们尽可能地运用所学的WEB编程技术，JSTL、EL、过滤器或拦截器、分页、以及前端的CSS和JavaScript技术等。

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



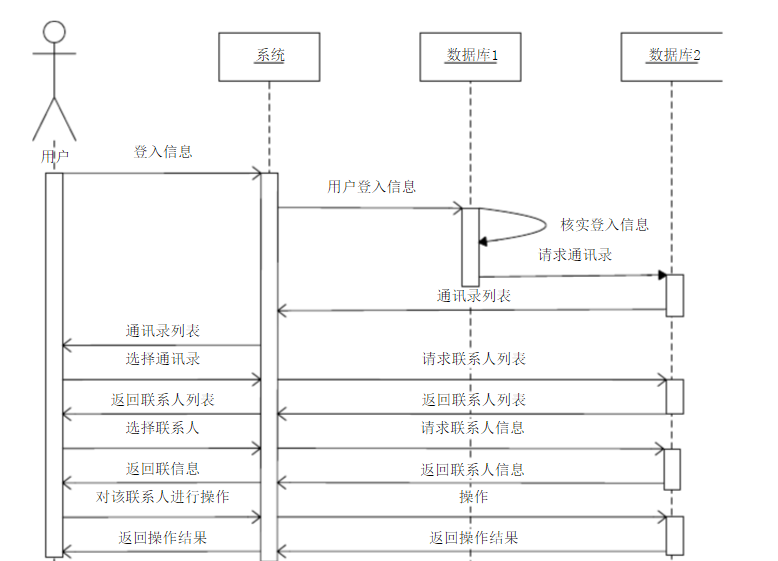
**本系统主要是为了更好的管理每个人的通讯录，给每个人提供一个井然有序的管理平台，防止手工管理混乱，造成不必要的麻烦。**

**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。

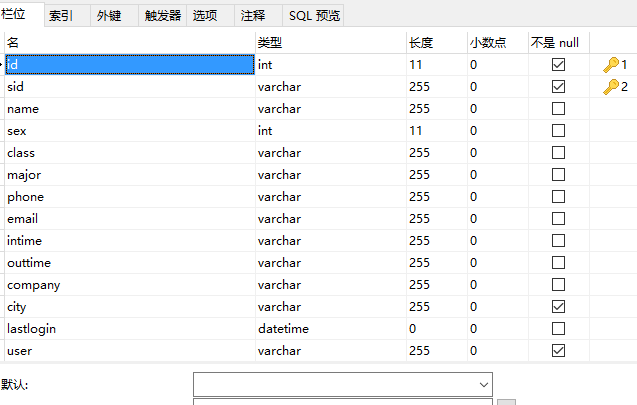
1. 学生基本信息表 student

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。

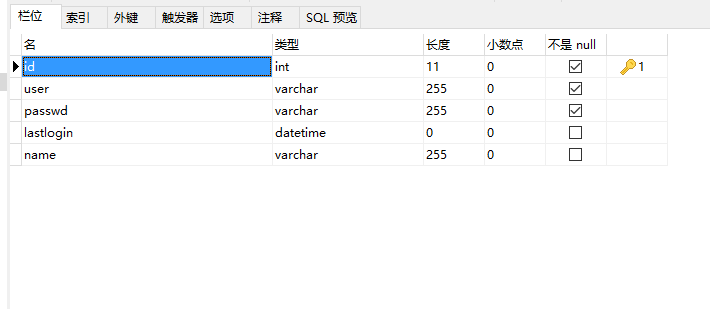
（1）对用户信息建立数据表，其中的数据项包括登录账号、登录密码、学号等。

（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

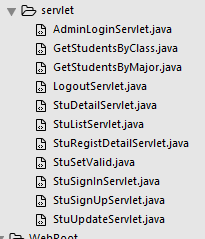
**LoginFilter.java** 过滤器实现对登陆后页面session的查验

**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("椹卞姩鍔犺浇鎴愬姛");

} 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("杩炴帴鎴愬姛");

} catch (SQLException e) {

System.out.println("杩炴帴澶辫触");

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;

}

}

**登陆追踪对session的过滤器filter实现**

public class LoginFilter implements Filter{

public void init(FilterConfig config) throws ServletException{

}

public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)

throws IOException, ServletException{

HttpServletRequest req = (HttpServletRequest)request;

HttpSession ses = req.getSession();

if(ses.getAttribute("user")!=null){

chain.doFilter(request,response);

}else{

request.getRequestDispatcher("/admin/login.jsp").forward(request,response);

}

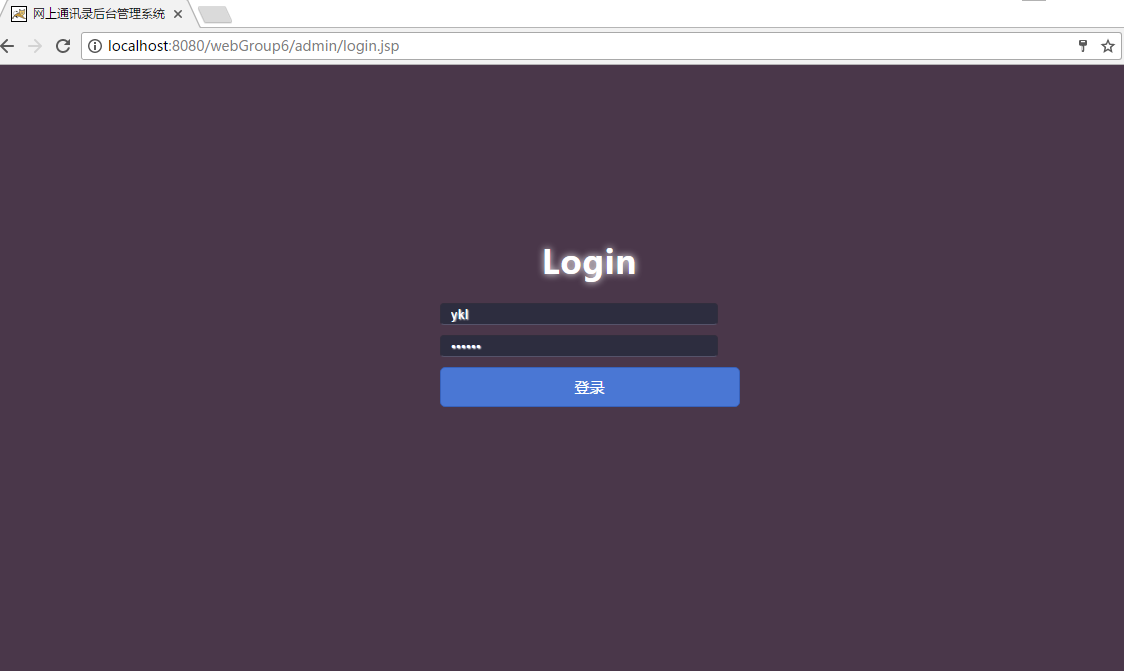
}

public void destroy(){

}

}

五、界面成果展示



管理员系统登陆页面



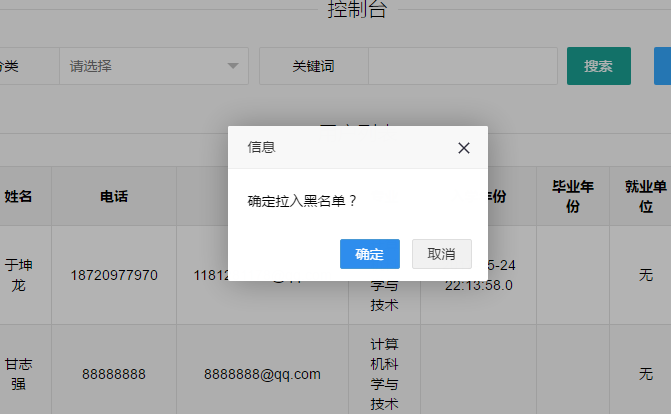
登陆成功后的页面



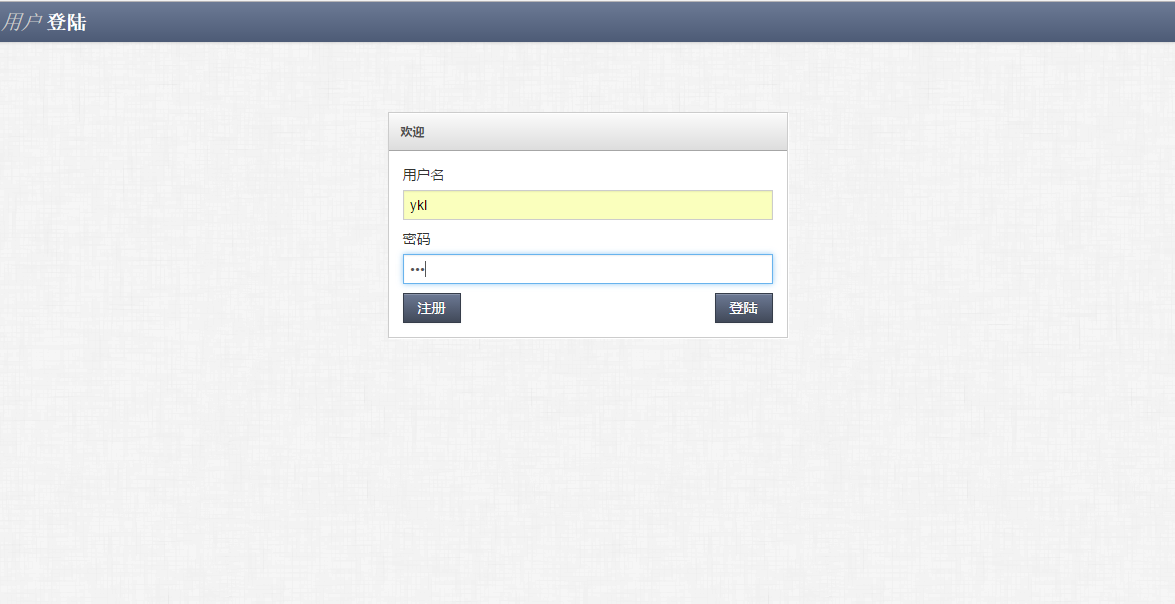
登陆失败后的温馨提示，细节处理







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



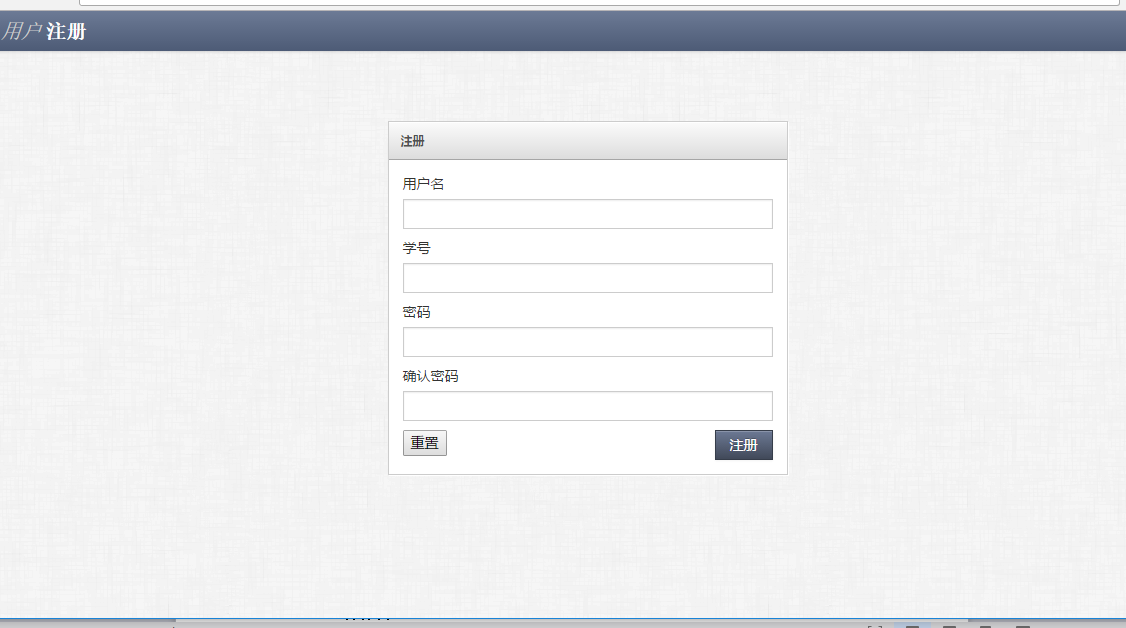
通讯录查询的两个接口，专业通讯录查询和班级通讯录查询







注销的时候要消除session



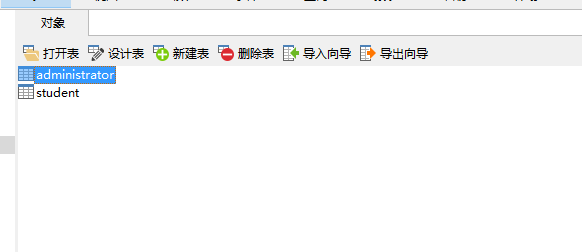
用户注册页面



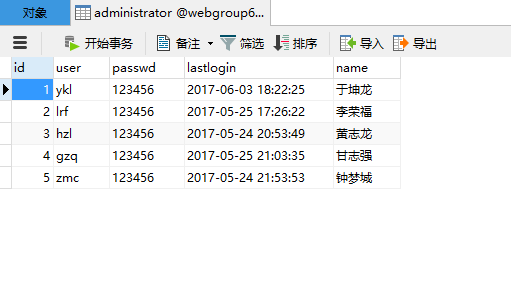
在注册成功的时候要跳转到登记信息页面

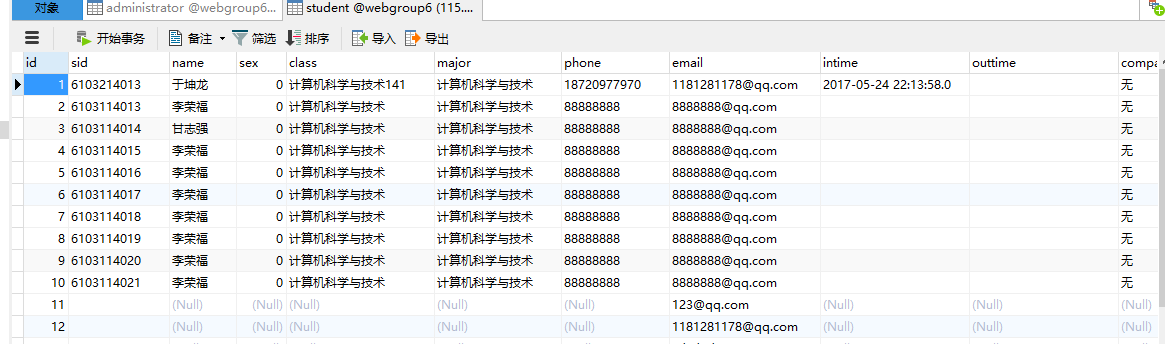


完成信息登记后跳转主页面（本打算跳转待审核页面）



我们建了两个表





六、心得体会

本次实验完成的是《网上通讯录项目开发》开发过程体会颇多。

首先，在进行项目需求分析的时候，要考虑用户人群，因为是小项目，所以我们把用户群体锁定在了计算机系的全体学生，考虑到管理员的身份，最终我们小组决定完成一套后台管理系统，和用户系统，这样，实现了用户和管理员的分离。

再者，在设计数据库的时候，考虑到大家都是线下开发，可能对数据库不同步影响开发进度，所以我们选择了一个线上的数据库作为我们的项目开发的数据库（远程服务器上的数据库）。

其次，框架设计上，我们经过讨论最终决定使用MVC框架体系，尽管没用使用成品的MVC框架，但是我们的设计思路和实现过程完全实现了模型，试图，和控制的分离，我们对数据库的操作也就是Struts2框架中的action进行了封装。在开发过程中，为了实现前端和后端的完全分离，我们采用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。

在开发过程中，会遇到各种各样的问题，比如说，当我们使用数据库的时候，会遇到编码的问题，其实编码的问题有很多，在Myeclipse中有页面的编码格式，我们采用的不是gbk中文编码而是统一使用了utf8编码格式，然后当我们把前端模板代码导入到Myeclipse项目当中的时候，IDE会报错误，其实这些错误不会影响项目的正常运行，后来百度之后发现是IDE对js等文件的纠错机制在搞鬼，所以我们通过exclude from validation 来忽略了这种报错。在进行管理员系统登录设计时，我们为了更好的用户体验，对form表单提交的内容进行查验并做出相应的反馈。查验过程放在了servlet当中，并通过request参数返回。一开始进行用户验证的时候在每个页面设置了session来 判断是否是登陆状态，后来发现这样代码重用率很低，所以我们设置了一个filter过滤器来对登陆后可以访问的目录进行session的过滤，这样实现了登陆的追踪。

考虑到用户数据的无价，我们小组决定在后台不进行数据库中的数据删除操作，即使有需要删除的我们也只是增添了一个字段来表示记录的有效性。这一点着重体现在管理员系统中的添加到黑名单和从黑名单中取消这两种操作。

在设计和实现student系统的时候需要一些个人信息展示和更改的页面，一开始我们采用的是请求页面的时候从数据库中提取数据，后来发现这样操作等待延时太高，然后我们就进行了相应的改进，我们在登陆的时候，如果用户登陆成功，在跳转页面的同时，我们设置用户基本信息的相应的session，这样在用到本用户的信息的地方直接可以通过session获取，十分方便快捷。

在进行代码实现部分，我们发现有好多功能会用到类似的代码，为了实现更好的代码重用，我们修改了原有的方法，简化了代码。在进行目录结构设计的时候，我们吃了大亏，也就是说filter在目录过滤的时候如何设置，所以经验告诉我们，最好把登陆前的页面和登陆后的页面放置在不同的目录结构下面，这样方便过滤器的过滤。提到目录结构，我们还遇到了一些问题，那就是html等前端页面中的url根目录和jsp默认页面中的根目录不同，所以每次都会出错。再者有些需要和服务器进行通讯的页面最好设置成jsp文件。

最后稍稍总结一下，一个完整的项目是需要团队共同完成的。所以在进行项目开发时，一定要分工明确，然后，衔接部分要沟通好，因为，每个人的开发进度是不一样的。虽然我们在共同努力下完成了这个大项目，但是还有很多细节需要我们处理，在时间上不允许，所以我们在相似的地方，选择了一部分进行了细节处理，体会一下处理个过程。

虽然说实现了用户的注册登陆，使用等等，但是从另一个角度考虑，我们的系统是否安全，所以这又是一门学问，我们还要对用户输入的密码进行加密，对敏感字符进行过滤，避免一些跨站脚本攻击等等，然后对数据库进行安全测试等等，这些我们都考虑到了，但是时间和精力不允许，所以我们没有实现。总结来说，这次大作业我们都收获很多。