目 录

[第一章 绪论 1](#_Toc481927055)

[第二章 相关技术介绍 1](#_Toc481927056)

[第三章 系统设计 6](#_Toc481927060)

[3.1功能需求 6](#_Toc481927061)

[3.2 可行性分析 6](#_Toc481927063)

[3.3分析模型 7](#_Toc481927064)

[第四章 系统功能实现 12](#_Toc481927066)

[4.1 系统架构 12](#_Toc481927067)

[4.2 系统设计 13](#_Toc481927068)

[4.3 数据库设计 15](#_Toc481927069)

[4.4前后台交互 17](#_Toc481927068)

[第五章 系统测试 21](#_Toc481927066)

[5.1用户模块功能测试 21](#_Toc481927080)

[5.2角色模块功能测试 23](#_Toc481927081)

[5.3权限模块功能测试 24](#_Toc481927082)

[第六章 总结 25](#_Toc481927066)

# 绪论

权限管理系统是大部分应用系统的重要组成部分，系统权限的有效控制也是应用系统实施成功的保证。首先，不同职责的人员，对系统操作的权限应该是不同的。优秀的业务系统，这是最基本的功能。可以对“组”进行权限分配。对于一个大企业的业务系统来说。如果要求管理员为其下员工逐一分配系统操作权限的话，是耗时且不够方便的事情。所以出现了以用户绑定角色（组）的模型，也就是基于RBAC的权限管理模型。

权限管理系统应该是可扩展的。它应该可以加入到任何带有权限管理功能的系统中。就像组件一样可以被不断的重用，可不是没开发一套管理系统，就要针对权限部分进行重新开发。

**第二章 相关技术介绍**

这个项目是一个简单的java web项目，项目采用的是IntelliJ IDEA +tomcat8+ bootstrap+JSP+ spring mvc +spring +mybatis+mysql开发的,采用MVC设计模式。以下分别介绍这次开发的主要技术方法和应用方式。

## 2.1 JAVA简介

Java是一门面向对象编程语言，不同与C++的面向过程编程语言，但是java中继承了许多C++中的许多优点，同时少了 C++里的多继承、指针，因此Java语言具有面向对象和单继承的特征。Java中面向对象编程语言是编程语言的代表，体现了面向对象理论可以使用java的面向对象和跨平台性多线程队程序进行编写和实现使程序的可拓展性更强。

## 2.2 JSP技术简介

JSP全名为Java Server Pages，中文名叫java服务器页面，其根本是一个简化的Servlet设计，它是由Sun Microsystems公司倡导、许多公司参与一起建立的一种动态网页技术标准。JSP技术有点类似ASP技术，它是在传统的网页HTML（标准通用标记语言的子集）文件(\*.htm,\*.html)中插入Java程序段(Scriptlet)和JSP标记(tag)，从而形成JSP文件，后缀名为(\*.jsp)。 用JSP开发的Web应用是跨平台的，既能在Linux下运行，也能在其他操作系统上运行。

它实现了Html语法中的java扩展（以 <%, %>形式）。JSP与Servlet一样，是在服务器端执行的。通常返回给客户端的就是一个HTML文本，因此客户端只要有浏览器就能浏览。

JSP技术使用Java编程语言编写类XML的tags和scriptlets，来封装产生动态网页的处理逻辑。网页还能通过tags和scriptlets访问存在于服务端的资源的应用逻辑。JSP将网页逻辑与网页设计的显示分离，支持可重用的基于组件的设计，使基于Web的应用程序的开发变得迅速和容易。 JSP(JavaServerPages)是一种动态页面技术，它的主要目的是将表示逻辑从Servlet中分离出来。Java Servlet是JSP的技术基础，而且大型的Web应用程序的开发需要Java Servlet和JSP配合才能完成。JSP具备了Java技术的简单易用，完全的面向对象，具有平台无关性且安全可靠，主要面向因特网的所有特点。

## 2.3 MySQL数据库简介

管理语言--结构化查询语言（SQL）进行数据库管理。由于MySQL是开放源代码的，因此任何人都可以在General Public License的许可下下载并根据个性化的需要对其进行修改。MySQL因为其速度、可靠性和适应性而备受关注。大多数人都认为在不需要事务化处理的情况下，MySQL是管理内容最好的选择。MySQL这个名字，起源不是很明确。一个比较有影响的说法是，基本指南和大量的库和工具带有前缀“my”已经有10年以上，而且不管怎样，MySQL AB创始人之一的Monty Widenius的女儿也叫My。这两个到底是哪一个给出了MySQL这个名字至今依然是个迷，包括开发者在内也不知道。MySQL的海豚标志的名字叫“sakila”，它是由MySQL AB的创始人从用户在“海豚命名”的竞赛中建议的大量的名字表中选出的。获胜的名字是由来自非洲斯威士兰的开源软件开发者Ambrose Twebaze提供。根据Ambrose所说，Sakila来自一种叫SiSwati的斯威士兰方言，也是在Ambrose的家乡乌干达附近的坦桑尼亚的Arusha的一个小镇的名字。MySQL，虽然功能未必很强大，但因为它的开源、广泛传播，导致很多人都了解到这个数据库。它的历史也富有传奇性。

## 2.4 JDBC与Mybatis

JDBC（Java DataBase Connectivity,java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问，它由一组用[Java语言](https://baike.baidu.com/item/Java%E8%AF%AD%E8%A8%80)编写的类和接口组成。JDBC提供了一种基准，据此可以构建更高级的工具和接口，使数据库开发人员能够编写数据库应用程序。有了JDBC，我们可以在java语言中实现对mysql的curd操作

MyBatis 是一款优秀的持久层框架，它支持定制化 SQL、存储过程以及高级映射。MyBatis 避免了几乎所有的 JDBC 代码和手动设置参数以及获取结果集。MyBatis 可以使用简单的 XML 或注解来配置和映射原生信息，将接口和 Java 的 POJOs(Plain Old Java Objects,普通的 Java对象)映射成数据库中的记录。有了Mybatis我们对数据库的操作将会更加方便。

## 2.5 Tomcat

Tomcat是由一些知名软件公司开发的，例如Apache和sun等等公司。首先要明确，tomcat的开发者很人性，不收取费用，而且还给我们提供了其制作原理。是属于服务器，而且是轻量级的对设备的要求低，我们可以从tomcat的实际原理进行参考，对自己分析研究服务器设计具有重大意义。为什么要选择tomcat是因为它在占用很少的资源的情况下能够独立运行一些应用，而且扩展性能也是非常好的。

## 2.6 IntelliJ IDEA简介

IntelliJ IDEA是一款功能强大的java开发工具，在代码自动提示、重构、J2EE支持、各类版本工具(如git、svn、github)、maven等方面都有很好的应用。

## 2.7 MVC设计模式

MVC是一种编程思想，是一种将代码中的输入部分，输出部分和程序的逻辑处理部分分开进行代码设计。这种编程方法主要将代码分成三部分：模型，视图和控制器三个核心部分。

视图

主要是指用户主管视觉上能够欣赏的界面。在以前的网站中，视图主要是有HTML来扮演主要的角色。但随着世界上计算机科学与技术水平的不断革新，新的技术水平不断出现，主要有xml,flash等等。

视图仅仅是允许数据输出和用户能够就行操作等简单界面，并没有进行真正的逻辑处理。

模型

模型表示系统的逻辑处理部分，主要指在数据库中处理数据。一个视图可以对应多个模型，因此大大减少了代码的数量和代码的重复量。

控制器

控制器是指在知晓用户输入的情况下，通过使用模型和调用视图来满足用户的具体需求。

MVC工作过程是：首先让用户输入信息并使用控制器来得到用户输入的结果。接下来根据用户的具体请求，用具体模块来处理用户的信息请求并返回最终结果。最后，控制器用具体的视图来显示来返回请求数据

## 2.8 srping mvc spring

spring框架是由于软件开发的复杂性而创建的。Spring使用的是基本的JavaBean来完成以前只可能由EJB完成的事情。然而，Spring的用途不仅仅限于服务器端的开发。从简单性、可测试性和松耦合性角度而言，绝大部分Java应用都可以从Spring中受益。

Spring mvc首先是一个MVC框架。在web模型中，MVC是一种很流行的框架，通过把Model，View，Controller分离，把较为复杂的web应用分成逻辑清晰的几部分，是为了简化开发，减少出错。还是为了组内开发人员之间的配合。总之就是一种分层工作的办法。并且是spring的一个子框架，当然拥有spring的特性，如依赖注入。

**第三章 系统设计**

## 3.1功能需求

本系统的目的是实现基于RBAC模型的权限管理系统的基本功能。系统主要分为用户模块：主要实现对用户的增删改查，另外还可以为用户分配角色。角色模块：完成对角色的增删改查，另外还可以对角色拥有的权限进行分配。权限模块：主要对权限树进行增删改查。另外用户登录还可以对自己的个人信息进行修改，比如修改密码。

## 3.2可行性分析

### 3.2.1技术的可行性

主要用IntelliJ IDEA进行本系统的发布， 数据库也是开源免费的Mysql数据库。主要开发语言还是面向对象的开发语言Java。这个系统主要完成一些简单业务 前端界面采用html+css等技术进行页面设计，后台采用java来操作数据库进行增删改查 同时java也是开源的，为系统开发也提供了一些系统架包。由此可以看出，现有的技术是完全可以支撑起本系统的开发。

### 3.2.2经济的可行性

根据技术可行性可以看出，成绩管理系统主要采用免费开源的软件进行开发，通过这样做，我们的开发成本得到了有效的管理，节约了大量开发工具的费用，不需要大量的软件开发成本需求。采用免费开源的软件还可以根据自己的需求对其进行改造适应自己的需求和发展，因此可以看出，从经济的角度对权限管理系统分析可以得出权限管理系统在经济方面也是可行的。

### 3.2.3操作可行性

权限管理系统操作简单方便，大多都是通过鼠标进行操作和管理。而且由于现在电脑技术的迅速普及，基本每个人都具备基本的电脑操作技巧。

### 3.2.4法律的可行性

本权限管理系统是自己进行独立开发和测试完善的 。另外，开发所使用的软件大多数是开源免费的，因此不存在侵犯版权之类的问题，是在法律允许的范围之类进行开发和测试的。因此，在法律上是可行的。

## 3.3分析模型

为了确保正确的数据在系统中能够客观真实地反映世界，其中的一些完整性规则通过一些控制的标准作业应用程序来实现。数据库要具有安全性，这数据库系统重要的一个方面，它是指保护数据库以防止非法使用，包括破坏和未授权访问等行为。

但是由于本次设计开发的系统涉及到的数据都不具备保密性的数据，因此不需要采取特别复杂的安全措施，只须防止数据被人随意改动即可。根据系统要求，可以列出以下数据项和数据结构.

#### 3.3.1 系统流程图



#### 3.3.2 用例图



上图为用例图，用户需要登录成功才能使用相应的功能，包括查看首页内容，对个人信息的修改，比如对密码进行修改.

#### 3.3.3 E-R图



上图为E-R图，可以看出权限和角色，用户和角色都是多对多关系，在后续的数据库关系模式的选择上可以考虑把两个关系单独建表。

#### 3.3.4 包图



由于是采用MVC设计思想，所以落实到java中就是分层的思想，控制器接受所有的请求，找到对应的处理器，而处理器是通过调用service层的方法实现的，当然最终对数据库的访问都是在dao中实现的。另外，并不是所有的请求都是合法的，需要控制器去管理，所以我还分了一个拦截器的类，实现权限的拦截功能。最后包工作过程中，需要使用可重复使用组件bean,所以为其单独建立一个包。

#### 3.3.5 部分功能的顺序图

注册功能



登录功能：



**第四章 系统功能实现**

## 4.1 系统架构

本次设计开发的系统是B / S模式系统。后台数据处理层，中间业务逻辑层，用户界面层组成三层结构的体系。后台使用数据库，中间层使用数据的分析和处理程序，前台使用WEB连接的三种层次结构的发展模式。使用和MySQL数据库和JSP技术平台开发。

前端软件使用Web技术开发，通过在服务器端下载的方式使客户端运行，这样可以在最大程度上提高整个系统的安全性能;客户界面采用浏览器，操作起来非常简单，在具体实施的时候，我们可以充分的利用JSP技术，WEB开发出来大大提高了应用的扩展性、功能性及可靠性，最终的结果就不是简单的HTML页面了，而是可在它之上构筑一种应用完整的平台。系统主要是针对新闻发布工作的管理。根据分析系统确定的功能以及目标，采用逐步求精、自顶到底的设计结构化方法，系统进行了结构设计。系统开发设计分析图如图4-1所示

用户需求分析析

系统分析

功能分析

基于B/S模式开发

应用开发

数据库设计

图4-1系统开发设计基本设计图

在具体的技术选用方面，MVC框架采用spring mvc ,数据库采用mysql,mybatis封装JDBC，完成对数据库的访问，连接池方面，选用阿里巴巴的druid连接池，日志记录选择log4j,使用git完成版本的控制，使用maven整合项目，我主要用到的是maven对jar包的管理，spring 完成各个框架的整合和bean的管理。下图是架构图：



图4-2系统开发所用技术的架构图

## 4.2 系统设计

#### 4.2.1 maven的配置

系统采用maven+ssm实现，所以先创建maven项目，我采用的是IDEA编辑器。创建完毕，对maven的配置文件 pom.xml进行依赖引入，包括spring mvc ,spring

Mybatis的依赖，测试工具junit的依赖，spring 整合 junit的依赖 spring-test,jsp的依赖等，数据库MySQL的依赖。这里以jsp和junit为例，

<**dependency**>  
 <**groupId**>org.apache.tomcat</**groupId**>  
 <**artifactId**>tomcat-jsp-api</**artifactId**>  
 <**version**>8.0.28</**version**>  
</**dependency**>  
<**dependency**>  
 <**groupId**>junit</**groupId**>  
 <**artifactId**>junit</**artifactId**>  
 <**version**>RELEASE</**version**>  
 <**scope**>test</**scope**>  
</**dependency**>

其中 scope属性为test表示只在测试期间有效。

#### 4.2.2 ssm的整合

Ssm框架的整合，首先是spring mvc的配置文件，主要包括拦截器的配置，视图解析器的配置，静态资源的放行配置，这里是视图解析器为例：

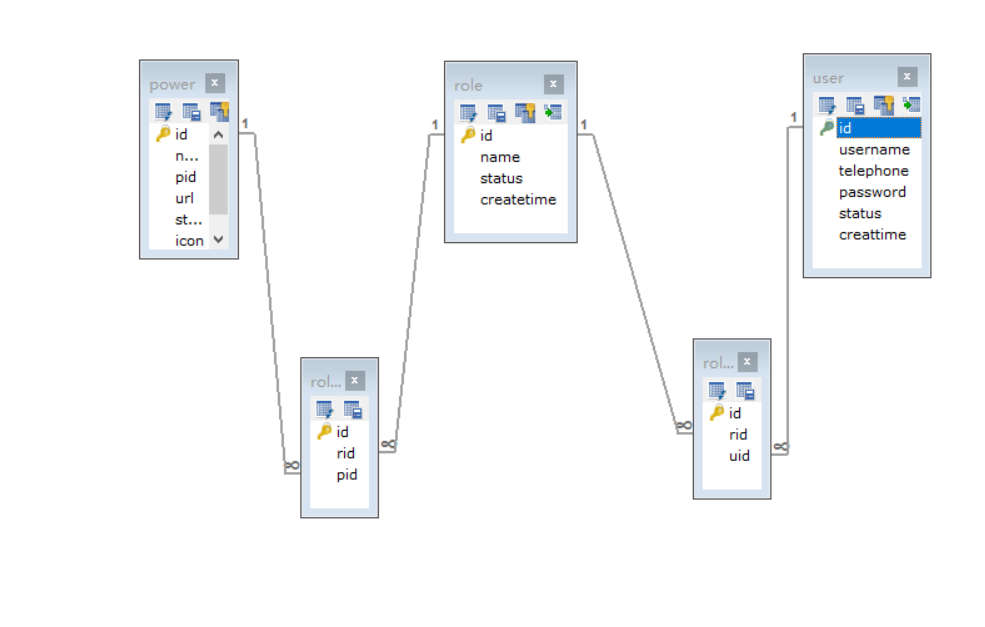
<**bean** *class***="org.springframework.web.servlet.view.BeanNameViewResolver"**/>  
<**bean** *id***="jsonView"** *class***="org.springframework.web.servlet.view.json.MappingJackson2JsonView"**/>  
<!--视图解析器配置-->  
<**bean** *class***="org.springframework.web.servlet.view.InternalResourceViewResolver"**>  
 <**property** *name***="prefix"** *value***="/WEB-INF/views/"** />  
 <**property** *name***="suffix"** *value***=".jsp"** />  
</**bean**>

其次是spring对mybatis的整合，主要是对各类bean的管理，本次开发主要采用注解开发，所以需要在配置文件中开启注解，此外对数据库连接池进行配置，对mybatis的会话进行管理。这里以数据库连接池为例：

<!--数据源的配置-->  
<**bean** *id***="dataSource"** *class***="com.alibaba.druid.pool.DruidDataSource"**>  
 <**property** *name***="driverClassName"** *value***="${db.driverClassName}"** />  
 <**property** *name***="url"** *value***="${db.url}"** />  
 <**property** *name***="username"** *value***="${db.username}"** />  
 <**property** *name***="password"** *value***="${db.password}"** />  
 <**property** *name***="initialSize"** *value***="3"**/>  
 <**property** *name***="minIdle"** *value***="3"**/>  
 <**property** *name***="maxActive"** *value***="20"**/>  
 <**property** *name***="filters"** *value***="stat,wall"**></**property**>  
</**bean**>

## 4.3 系统数据库设计

根据E-R图设计的关系模式如下：

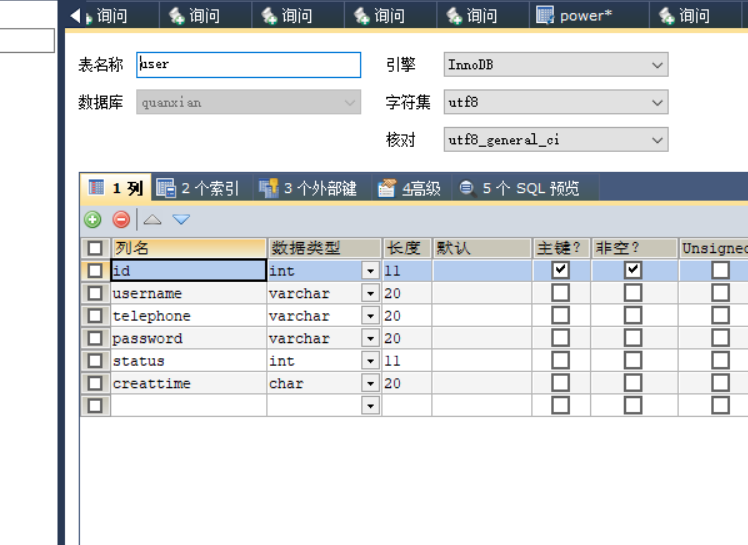


具体的表结构如下：

权限表：



用户表：



角色表：



## 4.4前后台交互

本次web设计主要采用MVC设计模式，前台接受用户的请求，后台对请求进行相关处理，其中后台语言采用的是java.具体过程为，前台发送一个请求，交给控制器（java中为servlet）,控制器对请求进行分发到业务层进行处理，业务层通过查询数据库得到数据，本系统采用的是JDBC的一个框架也就是mybatis,采集到数据之后，数据发给控制器，再有控制器把数据发给视图层进行显示，最终完成前后台的交互。下面以几个核心功能为例，介绍这个过程：

### 4.4.1登录功能的实现

**@RequestMapping**(**"/doLogin"**)  
**@ResponseBody  
public JsonData** doLogin(**User** *user*, **HttpSession** *session*){  
 **User** us =userService.doLogin(*user*);  
 **if**(us!=**null**) {  
 *session*.setAttribute(**"user"**, us);  
 **List**<**Power**> list=userService.selectPowerByUserId(us.getId());  
 **Map**<**Integer**,**Power**> map=**new** HashMap<>();  
 **Power** userPower=**new** Power();  
 **Set**<**String**>userUrl=**new** HashSet<>();  
 **for**(**Power** power:list){  
 map.put(power.getId(), power);  
 **if**(power.getUrl()!=**null**&& !power.getUrl().equals(**""**)) {  
 userUrl.add(power.getUrl());  
 }  
 }  
 **for**(**Power** power:list){  
 **if**(power.getPid()==**0**)  
 userPower=power;  
 **else**{  
 map.get(power.getPid()).getChildren().add(power);  
 }  
 *session*.setAttribute(**"userPower"**, userPower);  
 *session*.setAttribute(**"userUrl"**, userUrl);  
 }  
 **return JsonData**.**success**();  
 }  
 **else** {  
 **return JsonData**.**fail**(**"用户名与密码不匹配"**);  
 }  
}

登录分为两种情况：如果登录成功，则把用户的信息和用户拥有的权限放到session中去，

并通过ajax的回调函数跳转页面。如果登陆不成功就打印错误信息。

### 4.4.2用户增加实现

**@RequestMapping**(**"/insert"**)  
**@ResponseBody  
 public JsonData** insert(**User** *user*){  
 *user*.setPassword(**"123456"**);  
 *user*.setCreattime(**new** Date(**System**.**currentTimeMillis**()).toString());  
 userService.insertUser(*user*);  
 **return JsonData**.**success**();  
}

把从前台拿到的正确注册信息，并设置一些初始信息：密码和创建时间。然后调用数据访问层实现对数据库的更新操作。

### 4.4.3用户分配角色的实现

**@RequestMapping**(**"/doUnAssign"**)  
**@ResponseBody  
public JsonData** doUnAssign(**String** *userId*,**Integer**[] *assign*){  
 **Map**<**String** ,**Object**> map=**new** HashMap<>();  
 map.put(**"userId"**, *userId*);  
 map.put(**"assign"**, *assign*);  
 userService.deleteUserRole(map);  
 **return JsonData**.**success**();  
}

<**insert** *id=***"insertUserRole"** *parameterType=***"map"**>  
 insert into roleuser(uid,rid) values  
 <**foreach** *collection=***"unAssign"** *item=***"roleId"** *separator=***","**>  
 (*#{userId}*,*#{roleId}*)  
 </**foreach**>  
</**insert**>

从前台拿到数据后，由于分配的角色可能是多个，所以这里使用一个数组进行接收，不同的是此时数据访问层要对多组数据进行插入，此时，用到了mybatis里动态sql,用一个foreach标签实现对数组的遍历，以达到对多组数据进行插入的效果。

**4.4.4权限查看的实现**

**@RequestMapping**(**"/loadData"**)  
**@ResponseBody  
public Object** loadData(){  
 **List**<**Power**> list=**new** ArrayList<>();  
  
 **List**<**Power**> all =powerService.selectAll();  
 **Map**<**Integer**,**Power**> map=**new** HashMap<>();  
 **for**(**Power** power:all){  
 map.put(power.getId(), power);  
 }  
 **for**(**Power** power:all){  
 **if**(power.getPid()==**0**)  
 list.add(power);  
 **else** {  
 map.get((power.getPid())).getChildren().add(power);  
 }  
  
 }  
 **return** list;  
}

首先是前台方面，由于对前端语言不是特别熟悉，而权限的查找又不是传统的顺序结构，所以这里采用jquer的ztree组件。而后台的获取树结构的算法是使用hashmap和list实现的：首先查出所有的数据，然后以为id为key存储到hashmap中，方便后序建立父子关系。

**4.4.5为角色分配权限的实现**

**@RequestMapping("/doAssign")  
@ResponseBody  
public JsonData doAssign(Integer rid,Integer [] pid){  
 Map<String ,Object> map=new HashMap<>();  
 map.put("rid", rid);  
 map.put("pid", pid);  
 try{  
 roleService.deleteRolePower(rid);  
 roleService.updateRolePower(map);  
 }catch (Exception e){  
 e.printStackTrace();  
 return JsonData.fail("失败");  
 }  
 return JsonData.success();  
}**

由于会出现重新分配权限的方法，我采取的是把原理所有分配的权限都删除再进行新的权限的分配。

**4.4.6权限拦截的情况**

菜单页面可以借助数据库中的数据实现，但是如果未授权的用户直接访问url,也可以访问页面，这里采用spring的拦截器实现权限的拦截，主要有两个方面：一是未登录用户不能访问系统，二是登录用户不能超过自己的权限。

**public boolean preHandle(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse, Object o) throws Exception {  
 HttpSession session = httpServletRequest.getSession();  
 User user= (User) session.getAttribute("user");  
 if(user==null){  
 httpServletResponse.sendRedirect(session.getServletContext().getContextPath()+"/login");  
 return false;  
  
 }  
 else  
 return true;  
}**

**第五章 系统测试**

# 5.1用户模块

### 5.1.1登录模块的测试

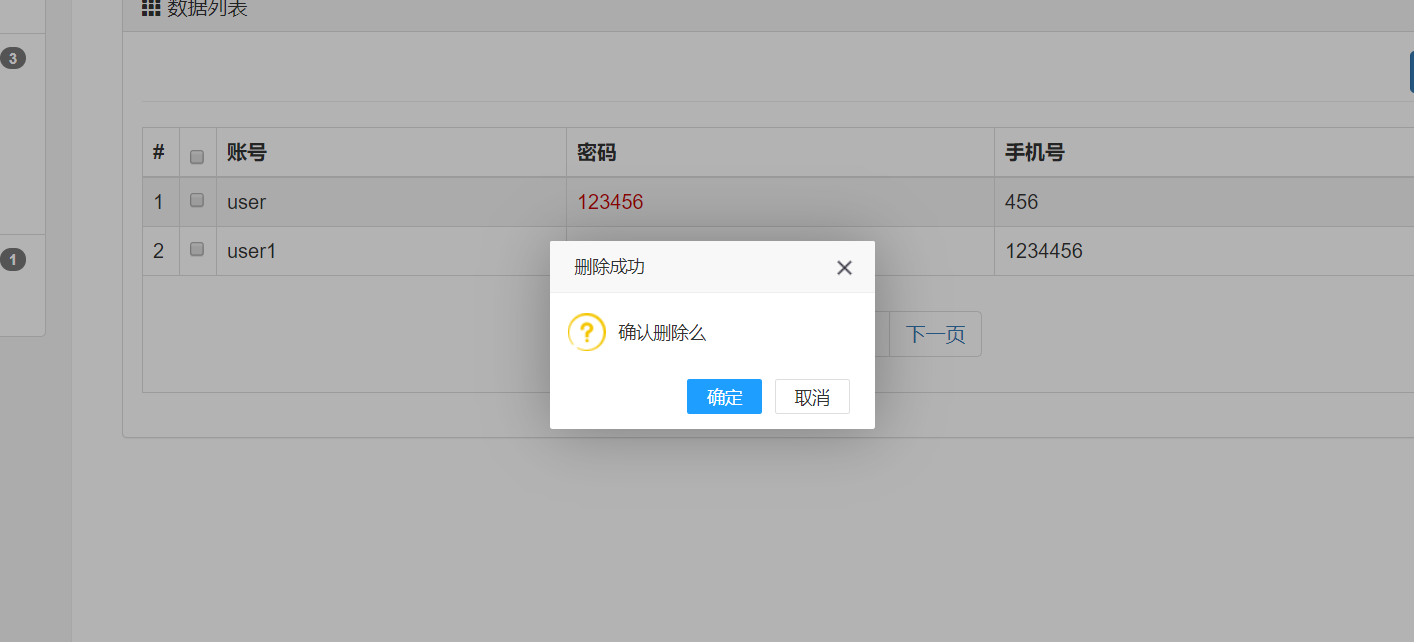
登录界面如图所示，在user和学生表中分别填入测试数据，如果用户名密码不匹配会给出相应的提示：



### 5.1.2个人信息模块的测试



### 5.1.3删除模块的实现



点取消什么事都不会发生，而选择确认会把选定的数据删除。

### 5.2权限模块

### 5.2.1更新功能的实现



### 5.2.2角色分配功能的实现





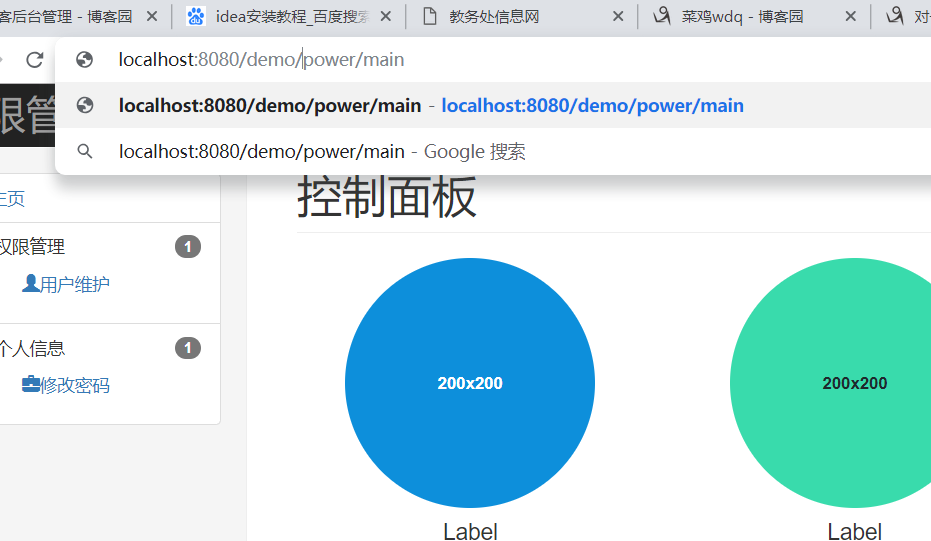
### 5.3权限模块

### 5.3.1查询功能的实现



并且可以通过点击图标完成增删改查。

### 5.3.2用户尝试访问未分配的权限



页面给出提示：



# 第六章 总结

该系统使用的是MVC架构模式，能对系统的功能进行分离，对系统分成三个层次其中Model是应用程序中用于处理应用程序数据逻辑的部分是系统的模型也是核心层，View应用程序中处理数据显示的部分是系统的视图层，Controller是应用程序中处理用户交互的部分是系统的控制层，用户通过系统对数据的访问通过控制层对数据的控制之后通过模型处理再返回数据给控制层之后反馈给用户就能获取信息。

实验中遇到较多的问题：

1. 为了搭建完整的项目环境，必要的测试必不可少，我采用的junit,但是由于spring 框架统一管理bean,但是测试模块启动时，spring的相关配置文件还没有加载，造成junit不能使用spring管理bean的情况。经过网上查阅资料，先是在依赖中导入依赖：

<**dependency**>  
 <**groupId**>org.springframework</**groupId**>  
 <**artifactId**>spring-test</**artifactId**>  
 <**version**>${spring-version}</**version**>  
 </**dependency**>

然后在写测试模块时，加上相关注解读入配置文件。

2.树型结构难以开发的问题，对于用户模块，角色模块的增删改查比较容易实现，但是权限的结构不是顺序结构，它逻辑上是一个树状结构。后台可以采用相关算法实现，但是由于我对前台模块掌握的不好，难以实现对权限树的渲染。最后选用了jQuery的ztree实现。

3.ajax请求失效的问题，考虑到用户的实用性，本项目多次采用了ajax交换信息的方式，但是在开发一个模块的时候，出现了ajax请求失效的问题。具体表现为：ajax请求可以发到后台，但是却接收不到后台返回的json数据，二是刷新当前页面。我先是查看后台的业务逻辑部分，发现没什么问题，于是检查前台代码，最后发现是所用的前端框架bootstrap中button标签默认绑定的事件是刷新当前页面

通过开发本次权限管理系统，我加深了对数据库的了解。我完成了通过java语言完成了对前台页面和数据库的相关数据联系起来。具体的收获如下：

1.完成了maven+ssm+junit开发框架的整合，以前虽然也写过一些项目，但是很多时候没采用框架，或者只使用mybatis封装JDBC，存在大量的重复代码，并且项目结构也不是很规范，通过本次课程的系统学习，老师的耐心教导，学会了javaweb项目在MVC模式下的一些规范。

2. 锻炼了前后台连接的能力，由于自己是偏向于后台的开发，对前台的一些语言，规范学习不是很深入。以前写项目的时候，多采用一套模板页面，直接发送请求的方式，结果就是整个项目静态页面部分，没有明显的分类，结构混乱。此次采用很多ajax请求的方式。页面也针对模块进行了分类，在权限拦截的时候十分方便.

3.使用日志文件加快开发速度。以前开发项目时没有写日志记录的情况，本次实验过程中 采用日志记录快速发现功能开发的问题。比如在相关控制器加上日志记录，记录参数，由此可以轻松的发现前后台的数据有没有出现问题。