《JSP程序设计》

实验作品

作品题目： 课表查询系统

院 系： 计算机与信息科学学院

专 业： 软件工程

学 号： 021321752215

姓 名： 舒洪凡

二零二一年十二月

**实验作品评分参考标准表**

|  |  |
| --- | --- |
| **评分标准及分值** | **得分** |
| **选题符合课程内容范围要求（10分）** |  |
| **设计思路和内容具有独特性（10分）** |  |
| **作品设计结构合理、功能完整（40分）** |  |
| **具有一定的实用研究价值（15分）** |  |
| **内容上无明显抄袭现象（15分）** |  |
| **图表格式正确，书写格式规范（10分）** |  |
| **合 计（100分）** |  |

**课表查询系统**

**摘要：**本作品主要实现了一个课表查询系统，采用了SSM（Spring + SpringMVC + MyBatis）的基础架构。后端使用Maven 进行包管理，使用spring-boot框架、模板引擎使用thymeleaf、web模块使用springMVC、数据库连接池使用Druid、数据库访问使用mybatis-plus、工具包使用hutool。前端使用H-ui模板，结合Timetables展示课程表。系统能够实现用户的登录退出，教室、班级、院系、地点、教工、学生、用户、课程、课程表的分页展示、增加、删除（批量）、修改、根据条件查询，排课（课程表增删改查）、课程表通过表格展示（可以通过教师、班级、学年、学期进行筛选）。并且使用拦截器通过session判断用户是否登录。

**目 录**

[第一章 作品概述 1](#_Toc90625264)

[第二章 作品设计 2](#_Toc90625265)

[2.1作品的构思 2](#_Toc90625266)

[2.1.1 起源 2](#_Toc90625267)

[2.1.2 总体架构 2](#_Toc90625268)

[2.2作品使用的第三方库 3](#_Toc90625269)

[第三章 作品的实现 4](#_Toc90625270)

[3.1作品框架 4](#_Toc90625271)

[3.1.1 数据库设计 4](#_Toc90625272)

[3.2作品实现要点 12](#_Toc90625273)

[3.2.1 用户登录模块 12](#_Toc90625274)

[3.2.2 教务管理系统首页 17](#_Toc90625275)

[3.2.3 教室模块 23](#_Toc90625276)

[3.2.4 班级模块 30](#_Toc90625277)

[3.2.5 院系模块 37](#_Toc90625278)

[3.2.6 地点模块 43](#_Toc90625279)

[3.2.7 教工模块 49](#_Toc90625280)

[3.2.8 用户模块 58](#_Toc90625281)

[3.2.9 课程模块 72](#_Toc90625282)

[3.2.10 课程表模块 78](#_Toc90625283)

[3.2.11 拦截器模块 99](#_Toc90625284)

[3.2.12 学生模块 102](#_Toc90625285)

[第四章 总结 111](#_Toc90625286)

# 第一章 作品概述

本作品是一个前后端不分离的课程表查询系统，基于SSM（Spring + SpringMVC + MyBatis），拥有信息管理和课表展示功能。

后端使用spring-boot作为基础，使用thymeleaf（基于Servlet）作为服务器端Java模板引擎。使用SpringMVC（spring内置的MVC框架，底层就是Servlet，它解决WEB开发中常见的问题(参数接收、文件上传、表单验证、国际化等等)，而且使用简单，与Spring无缝集成。支持 RESTful风格的URL请求）解决页面代码和后台代码的分离，降低代码的耦合性。使用mybatis-plus（一个 Mybatis 的增强工具，在 Mybatis 的基础上只做增强不做改变，为简化开发、提高效率而生），它已经封装好了一些crud方法，我们不需要再写xml了，直接调用这些方法就行，就类似于JPA。数据库使用MySQL数据库，数据库数据库连接池使用druid(Druid为监控而生的数据库连接池, 是Java语言中最好的数据库连接池)。使用Lombok在JavaBean中通过添加注解的方式，不需要为类编写getter/setter方法,也不需要编写构造器、equals等方法，同时可以自动化日志变量。同时引用hutool工具包，它能为我们自动生成验证码、格式化日期以及简化JDBC的操作。

前台使用H-ui 前端框架模板，它包含后台基本的增删改查页面，我们只要使用thymeleaf模板引擎循环替换H-UI模板的内容，并且更改静态文件路径，在springMVC的控制器中注册路由并且返回页面就能使用浏览器正常访问。课表的展示使用Timetables.js插件，它通过传入一个二维数组来展示课表，发送ajax请求从服务器获取课程表调用change方法进行展示。

系统能够实现用户的登录退出，教室、班级、院系、地点、教工、学生、用户、课程、课程表的分页展示、增加、删除（批量）、修改、根据条件查询，排课（课程表增删改查）、课程表通过表格展示（可以通过教师、班级、学年、学期进行筛选）。并且使用拦截器通过session判断用户是否登录，未登录用户跳转到登录页面。

# 第二章 作品设计

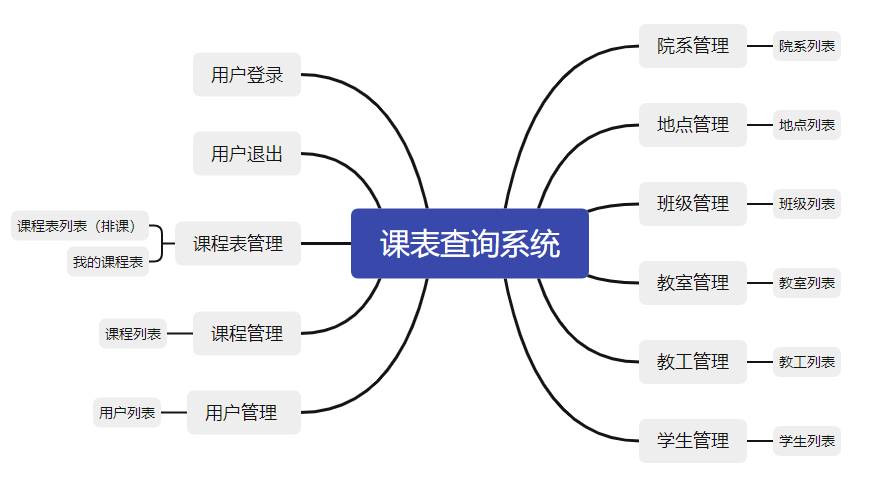
## 2.1作品的构思

### 2.1.1 起源

本作品为了方便学生和老师查询课程表信息，并管理和修改自己的信息。方便对全校的教室、院系、地点、学生、班级、老师、课程、课程表信息进行管理，并通过图表的方式进行查看。

### 2.1.2 总体架构

本作品整体结构如图2.1所示。实现用户的登录退出，教室、班级、院系、地点、教工、学生、用户、课程、课程表的分页展示、增加、删除（批量）、修改、根据条件查询，排课（课程表增删改查）、课程表通过表格展示（可以通过教师、班级、学年、学期进行筛选）。并且使用拦截器通过session判断用户是否登录，未登录用户跳转到登录页面。



**图2.1 总体架构图**

## 2.2作品使用的第三方库

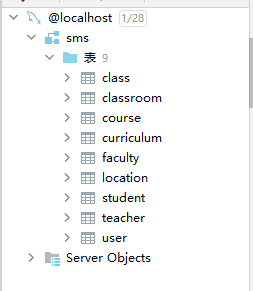
本作品使用的第三方库包括

1. spring-boot
2. spring-MVC
3. thymeleaf
4. mybatis-plus
5. druid
6. Lombok
7. Hutool
8. H-ui
9. Timetables.js

# 第三章 作品的实现

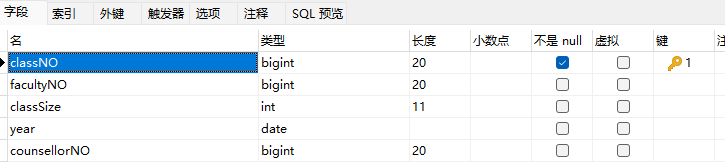
## 3.1作品框架

### 3.1.1 数据库设计

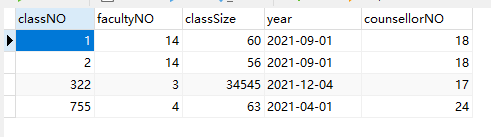


**图3.1 数据库总体架构图**

本课程表系统包含9张表名称如图3.1所示。数据表设计按照第一范式设计，要求字段是对属性的原子性约束，要求属性具有原子性，不可再分解；

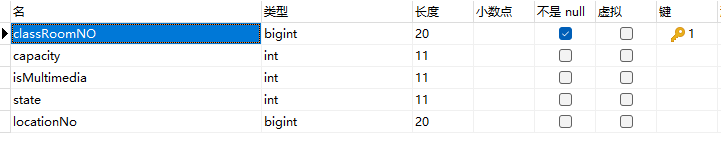


**图3.1.1 班级表字段**

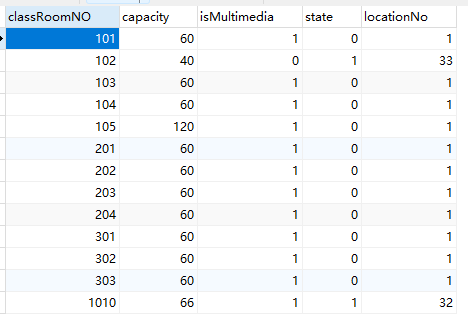


**图3.1.2 班级表数据**

班级表包括5个字段，字段如图3.1.1所示，部分数据如图3.12所示。分别为班级编号（classNO）、院系编号（facultyNO）、班级人数（classSize）、入学年份（year）、辅导员姓名（counsellorNO），院系编号和辅导员分别为院系表和教师表的外键。

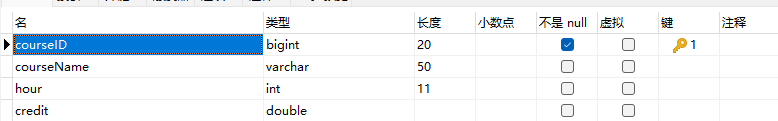


**图3.1.3 教室表字段**



**图3.1.4 教室表数据**

教室包括5个字段，字段如图3.1.3所示，部分数据如图3.1.4所示。分别为教室编号（classRoomNO）、教室容量（capacity）、是否多媒体教室（isMultimedia）、教室状态（state）、教室地点（locationNo），教室地点为为教室表的外键。

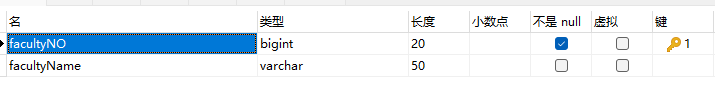


**图3.1.5 课程表字段**



**图3.1.6 课程表数据**

课表表包括4个字段，字段如图3.1.5所示，部分数据如图3.1.6所示。分别为课程编号（courseID）、课程名称（courseName）、学时（hour）、学分（credit）。

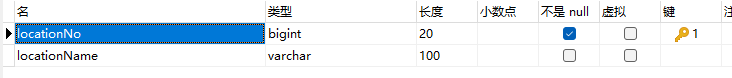


**图3.1.7 院系表字段**



**图3.1.8 院系表数据**

院系表包括2个字段，字段如图3.1.7所示，部分数据如图3.1.8所示。分别为院系编号（facultyNO）、院系名称（facultyName）。



**图3.1.9 地点表字段**

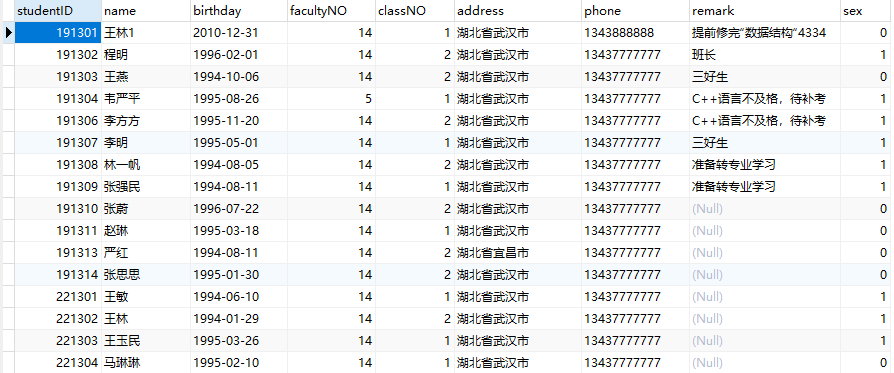


**图3.1.10 地点表数据**

地点表包括2个字段，字段如图3.1.9所示，部分数据如图3.1.10所示。分别为地点编号（locationNo）、地点名称（locationName）。

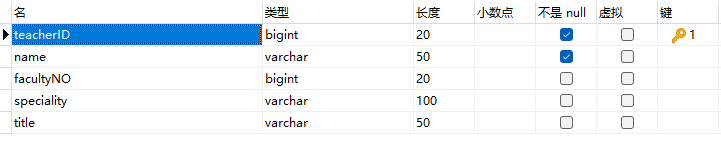


**图3.1.11 学生表字段**



**图3.1.12 学生表数据**

学生表包括9个字段，字段如图3.1.11所示，部分数据如图3.1.12所示。分别为学号（studentID）、姓名（name）、性别（sex）、出生日期（birthday）、院系（facultyNO）、班级（classNO）、家庭地址（address）、联系电话（phone）、备注（remark），其中院系、班级字段分别为院系表和班级表的外键。



**图3.1.13 教师表字段**

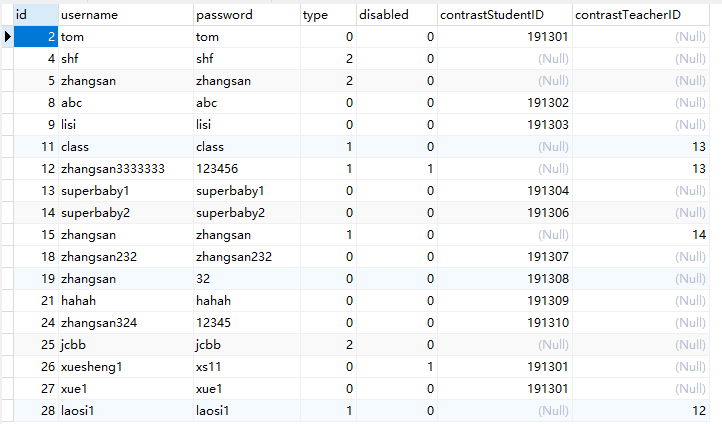


**图3.1.14 教师表数据**

教师表包括5个字段，字段如图3.1.13所示，部分数据如图3.1.14所示。分别为教工编号（teacherID）、姓名（name）、院系号（facultyNO）、主页方向（speciality）、职称（title），其中院系为院系表的外键。



**图3.1.15 用户表字段**

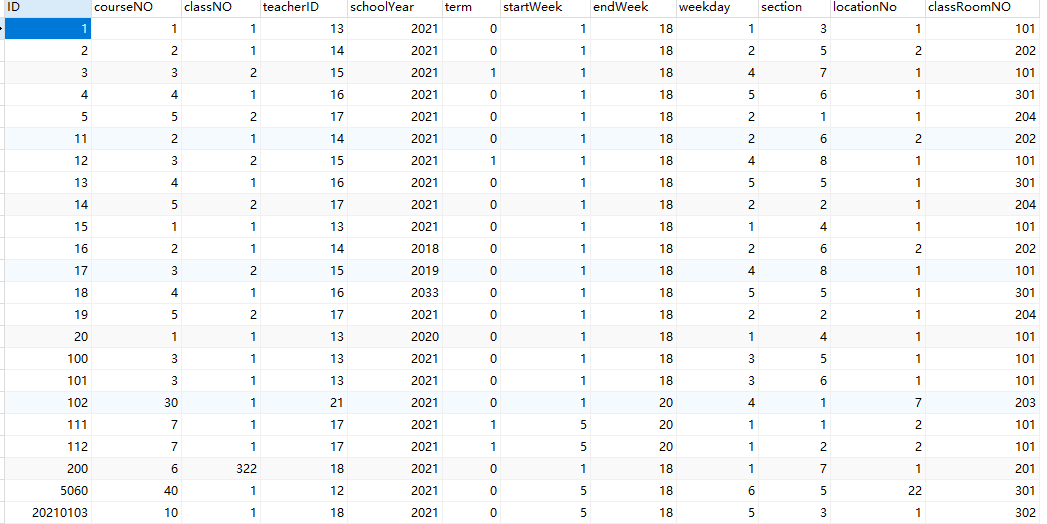


**图3.1.16 用户表数据**

用户表包括7个字段，字段如图3.1.15所示，部分数据如图3.1.16所示。分别为用户ID（id）、登录名（username）、密码（password）、用户类型【0学生 1老师 2管理员】（type）、是否启用【0启用1禁用】（disabled）、对应学生学号（contrastStudentID）、对应老师工号（contrastTeacherID），其中对应学生学号、对应老师工号字段分别为学生表和教师表的外键。



**图3.1.17 课程表 表字段**



**图3.1.18 课程表 表数据**

课程表表包括12个字段，字段如图3.1.17所示，部分数据如图3.1.18所示。分别为排课（ID）、课程ID（courseNO）、班级ID（classNO）、授课教师ID（teacherID）、开课学年（schoolYear）、学期【上学期0 下学期1】（term）、开课星期（startWeek）、结课星期（endWeek）、上课星期几（weekday）、第几节课上课（section）、地点ID（locationNo）、教室ID（classRoomNO），其中课程ID、班级ID、授课教师ID、教室ID、地点ID字段分别为课程表、班级表、教师表、教室表、地点表的外键。

## 3.2作品实现要点

### 3.2.1 用户登录模块



**图3.2.1.1 用户登录**



**图3.2.1.2 用户登录验证码错误**



**图3.2.1.3 用户登录用户名或密码错误**

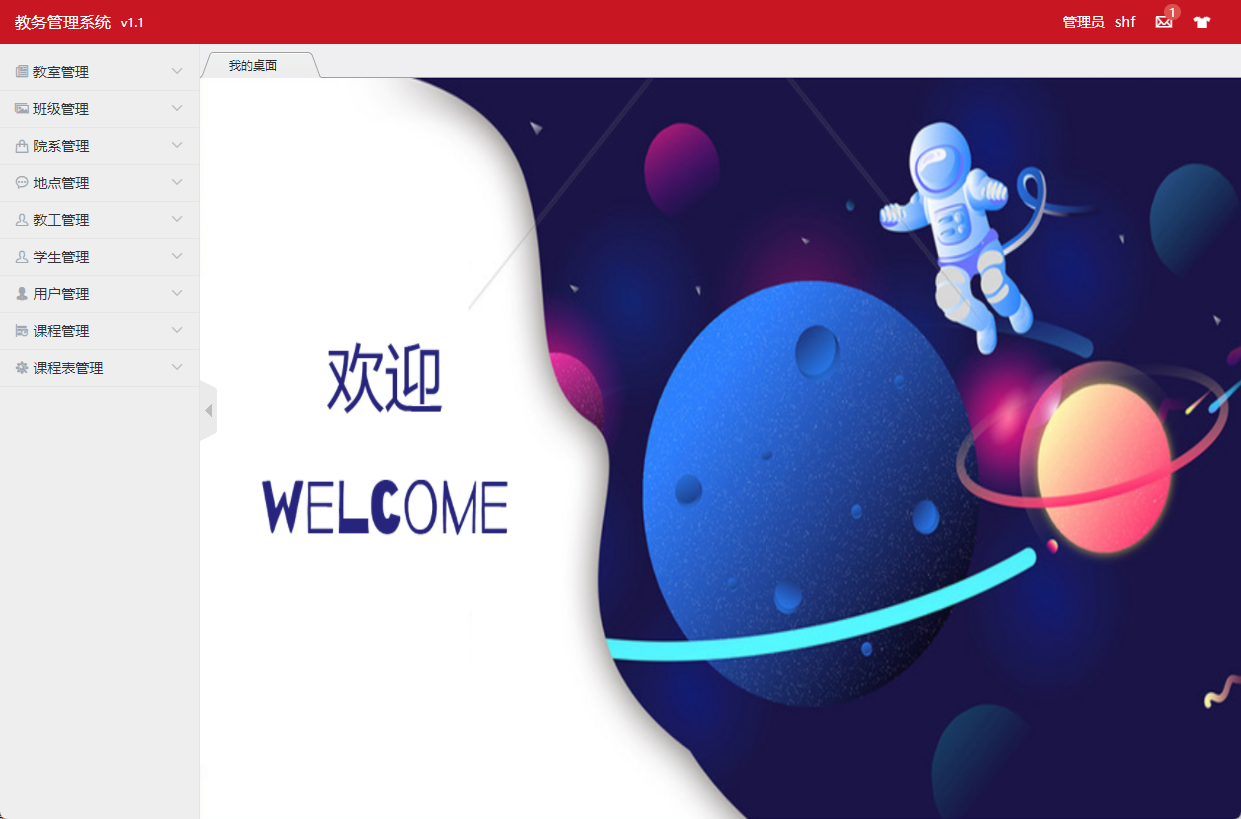
LoginController.java

@Slf4j  
@Controller  
public class LoginController *{* @Autowired  
 private UserMapper userMapper;  
  
 @Autowired  
 private StudentController studentController;  
  
 @Autowired  
 private TeacherController teacherController;  
  
 @GetMapping*(*"/login"*)* public String toLogin*(){* return "login";  
 *}* @PostMapping*(*"/login"*)* public String login*(*User user, HttpSession httpSession, Model model*){  
 log*.info*(*String.*valueOf(*user*))*;  
*// 验证码是否匹配* if *(*user.getCaptcha*()*.equalsIgnoreCase*((*String*)* httpSession.getAttribute*(*"SESSION\_KEY\_IMAGE\_CODE"*))) {  
// 用户是否存在* QueryWrapper*<*User*>* userQueryWrapper = new QueryWrapper*<>()*;  
 userQueryWrapper.eq*(*"username", user.getUsername*())*;  
 User u1 = userMapper.selectOne*(*userQueryWrapper*)*;  
 *log*.info*(*String.*valueOf(*u1*))*;  
 if *(*u1 != null*){  
 // 用户存在 判断密码是否一致* if *(*u1.getPassword*()*.equals*(*user.getPassword*())){* if *(*u1.getDisabled*()*==0*){* model.addAttribute*(*"info","登录成功！！！"*)*;  
 User u = userMapper.selectOne*(*new QueryWrapper*<*User*>()*.eq*(*"username",user.getUsername*()))*;  
 u.setPassword*(*"\*\*\*\*\*\*"*)*;  
 if *(*u.getContrastStudentID*()*!=null*){* Student student = studentController.getStudentById*(*u.getContrastStudentID*())*;  
 u.setClassNO*(*student.getClassNO*())*;  
 u.setContrastStudentName*(*student.getName*())*;  
 *}* if *(*u.getContrastTeacherID*()*!=null*){* Teacher teacher = teacherController.getTeacherById*(*u.getContrastTeacherID*())*;  
 u.setContrastTeacherName*(*teacher.getName*())*;  
 *}* httpSession.setAttribute*(*"USER",u*)*;  
 return "redirect:index";  
 *}* else *{* model.addAttribute*(*"info","用户已被禁用，请联系管理员！！！"*)*;  
 return "login";  
 *}  
 }* else *{* model.addAttribute*(*"info","用户名或密码错误，请重新输入！！！"*)*;  
 return "login";  
 *}  
 }* else *{  
// 用户不存在* model.addAttribute*(*"info","用户名或密码错误，请重新输入！！！"*)*;  
 return "login";  
 *}  
 }* else *{* model.addAttribute*(*"info","验证码错误，请重新输入！！！"*)*;  
 return "login";  
 *}  
 }  
}*

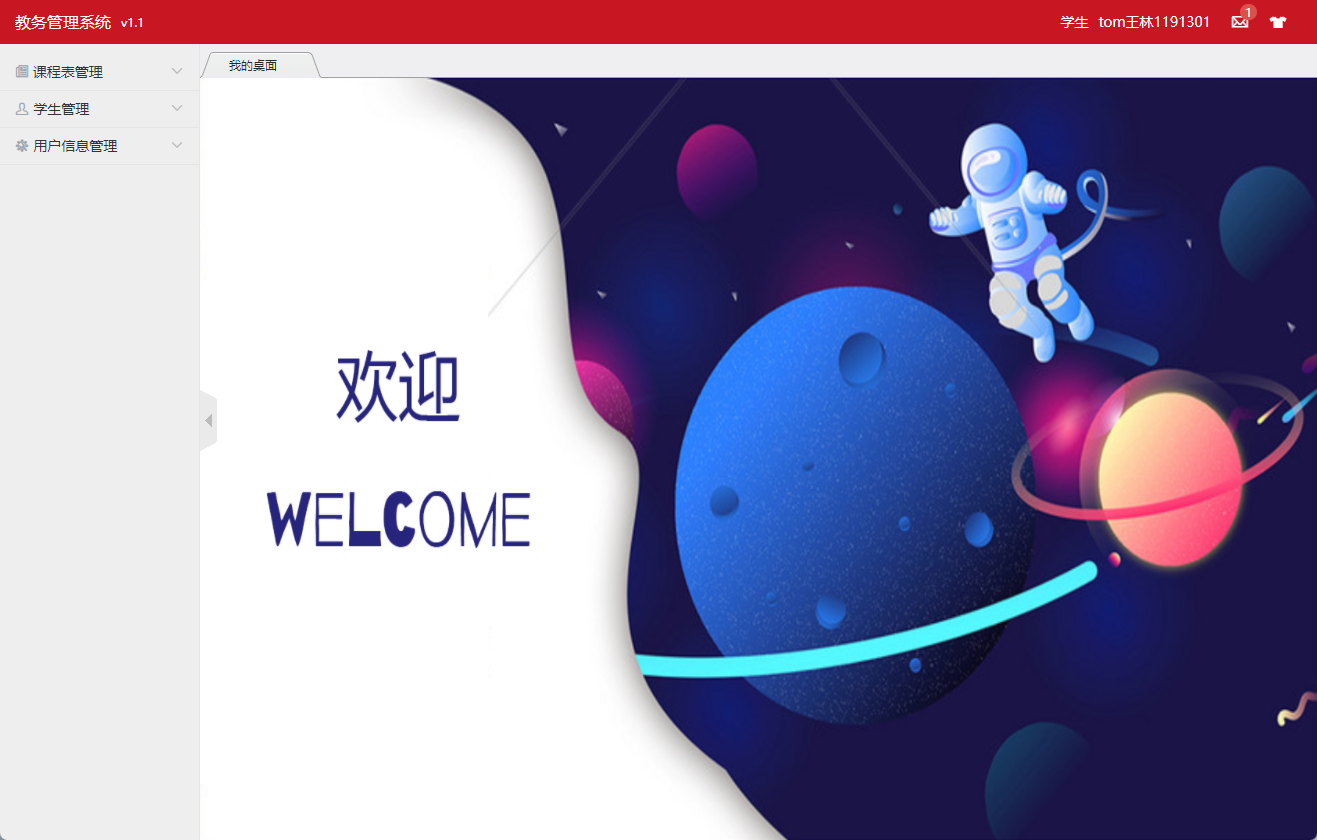
CaptchaController.java 验证码控制器

@Controller  
@Slf4j  
public class CaptchaController *{* public final static String *SESSION\_KEY\_IMAGE\_CODE* = "SESSION\_KEY\_IMAGE\_CODE";  
  
 @GetMapping*(*"/captcha"*)* public void createCode*(*HttpServletRequest request, HttpServletResponse response*)* throws IOException *{  
 //设置response响应* response.setCharacterEncoding*(*"UTF-8"*)*;  
 response.setHeader*(*"Pragma", "No-cache"*)*;  
 response.setHeader*(*"Cache-Control", "no-cache"*)*;  
 response.setDateHeader*(*"Expires", 0*)*;  
 response.setContentType*(*"image/jpeg"*)*;  
  
 *//定义图形验证码的长、宽、验证码字符数、干扰元素个数* CircleCaptcha captcha = CaptchaUtil.*createCircleCaptcha(*100, 38, 4, 20*)*;  
 System.*out*.println*(*captcha.getCode*())*;  
 *//将验证码放到HttpSession里面* request.getSession*()*.setAttribute*(SESSION\_KEY\_IMAGE\_CODE*, captcha.getCode*())*;  
 *log*.info*(*"本次生成的验证码为：" + captcha.getCode*()* + ",已存放到HttpSession中"*)*;  
  
 *//图形验证码写出，可以写出到文件，也可以写出到流  
 //输出浏览器* OutputStream out = response.getOutputStream*()*;  
 captcha.write*(*out*)*;  
 out.flush*()*;  
 out.close*()*;  
 *}  
}*

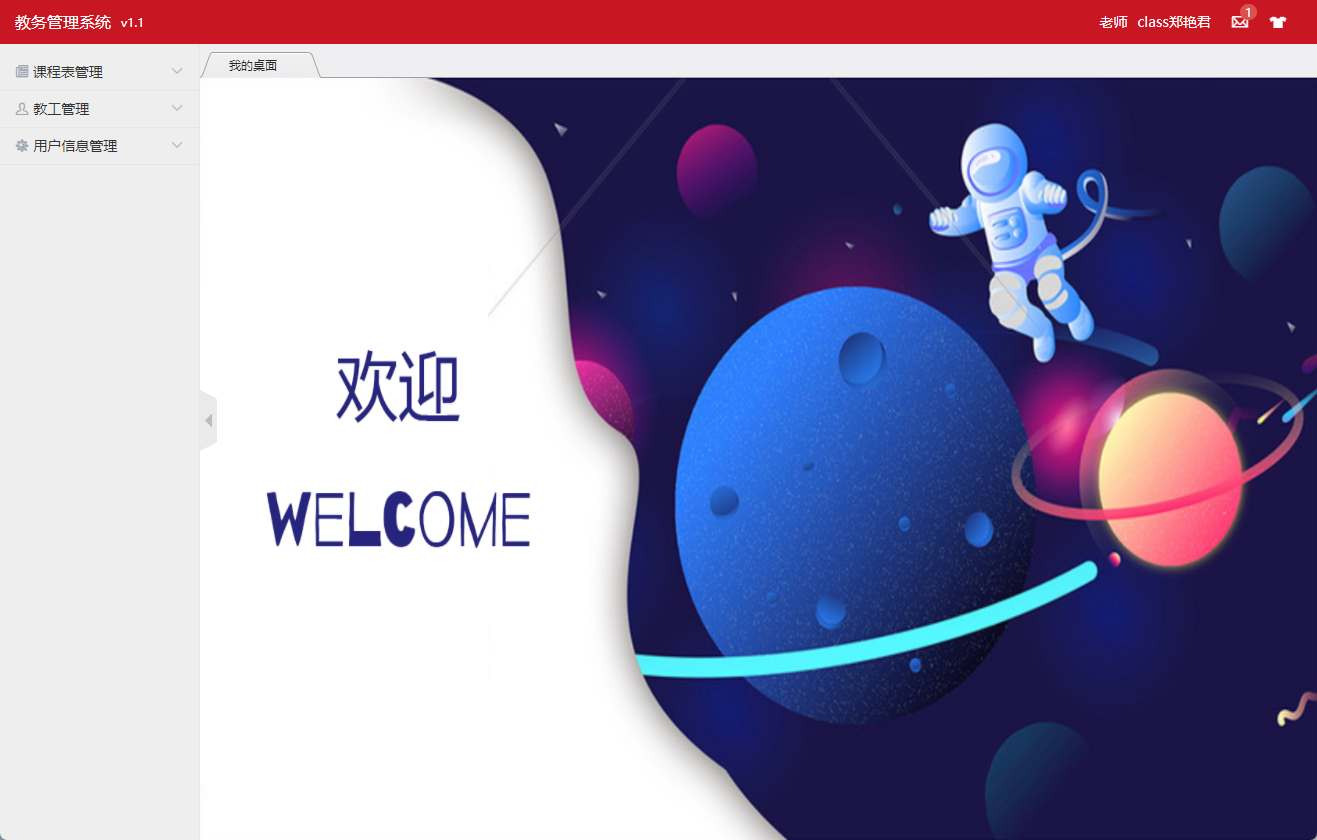
### 3.2.2 教务管理系统首页



**图3.2.2 管理员用户首页**



**图3.2.3 学生用户首页**



**图3.2.3 教师用户首页**

Indext.html 部分代码

*<*aside class="Hui-aside"*>  
 <*div th:if="*${*session.USER.type==2*}*" class="menu\_dropdown bk\_2"*>  
 <*dl id="menu-article"*>  
 <*dt*><*i class="Hui-iconfont"*>*&#xe616;*</*i*>* 教室管理*<*i class="Hui-iconfont menu\_dropdown-arrow"*>*&#xe6d5;*</*i*></*dt*>  
 <*dd*>  
 <*ul*>  
 <*li*><*a th:data-href="@{/classRoom/classRoomListPage}" data-title="教室列表" href="javascript:void(0)"*>*教室列表*</*a*></*li*>  
 </*ul*>  
 </*dd*>  
 </*dl*>  
 <*dl id="menu-picture"*>  
 <*dt*><*i class="Hui-iconfont"*>*&#xe613;*</*i*>* 班级管理*<*i class="Hui-iconfont menu\_dropdown-arrow"*>*&#xe6d5;*</*i*></*dt*>  
 <*dd*>  
 <*ul*>  
 <*li*><*a th:data-href="@{/class/classListPage}" data-title="班级列表" href="javascript:void(0)"*>*班级列表*</*a*></*li*>  
 </*ul*>  
 </*dd*>  
 </*dl*>  
 <*dl id="menu-product"*>  
 <*dt*><*i class="Hui-iconfont"*>*&#xe620;*</*i*>* 院系管理*<*i class="Hui-iconfont menu\_dropdown-arrow"*>*&#xe6d5;*</*i*></*dt*>  
 <*dd*>  
 <*ul*>  
 <*li*><*a th:data-href="@{/faculty/facultyListPage}" data-title="院系列表" href="javascript:void(0)"*>*院系列表*</*a*></*li*>  
 </*ul*>  
 </*dd*>  
 </*dl*>  
 <*dl id="menu-comments"*>  
 <*dt*><*i class="Hui-iconfont"*>*&#xe622;*</*i*>* 地点管理*<*i class="Hui-iconfont menu\_dropdown-arrow"*>*&#xe6d5;*</*i*></*dt*>  
 <*dd*>  
 <*ul*>  
 <*li*><*a th:data-href="@{/location/locationListPage}" data-title="地点列表" href="javascript:;"*>*地点列表*</*a*></*li*>  
 </*ul*>  
 </*dd*>  
 </*dl*>  
 <*dl id="menu-teacher"*>  
 <*dt*><*i class="Hui-iconfont"*>*&#xe60d;*</*i*>* 教工管理*<*i class="Hui-iconfont menu\_dropdown-arrow"*>*&#xe6d5;*</*i*></*dt*>  
 <*dd*>  
 <*ul*>  
 <*li*><*a th:data-href="@{/teacher/teacherListPage}" data-title="教工列表" href="javascript:;"*>*教工列表*</*a*></*li*>  
 </*ul*>  
 </*dd*>  
 </*dl*>  
 <*dl id="menu-student"*>  
 <*dt*><*i class="Hui-iconfont"*>*&#xe60d;*</*i*>* 学生管理*<*i class="Hui-iconfont menu\_dropdown-arrow"*>*&#xe6d5;*</*i*></*dt*>  
 <*dd*>  
 <*ul*>  
 <*li*><*a th:data-href="@{/student/studentListPage}" data-title="学生列表" href="javascript:;"*>*学生列表*</*a*></*li*>  
 </*ul*>  
 </*dd*>  
 </*dl*>  
 <*dl id="menu-admin"*>  
 <*dt*><*i class="Hui-iconfont"*>*&#xe62d;*</*i*>* 用户管理*<*i class="Hui-iconfont menu\_dropdown-arrow"*>*&#xe6d5;*</*i*></*dt*>  
 <*dd*>  
 <*ul*>  
 <*li*><*a th:data-href="@{/user/userListPage}" data-title="用户列表" href="javascript:void(0)"*>*用户列表*</*a*></*li*>  
 </*ul*>  
 </*dd*>  
 </*dl*>  
 <*dl id="menu-tongji"*>  
 <*dt*><*i class="Hui-iconfont"*>*&#xe61a;*</*i*>* 课程管理*<*i class="Hui-iconfont menu\_dropdown-arrow"*>*&#xe6d5;*</*i*></*dt*>  
 <*dd*>  
 <*ul*>  
 <*li*><*a th:data-href="@{/course/courseListPage}" data-title="课程列表" href="javascript:void(0)"*>*课程列表*</*a*></*li*>  
 </*ul*>  
 </*dd*>  
 </*dl*>  
 <*dl id="menu-system"*>  
 <*dt*><*i class="Hui-iconfont"*>*&#xe62e;*</*i*>* 课程表管理*<*i class="Hui-iconfont menu\_dropdown-arrow"*>*&#xe6d5;*</*i*></*dt*>  
 <*dd*>  
 <*ul*>  
 <*li*><*a th:data-href="@{/curriculum/curriculumListPage}" data-title="课程表列表" href="javascript:void(0)"*>*课程表列表*</*a*></*li*>  
 <*li*><*a th:data-href="@{/curriculum/myCurriculumPage}" data-title="我的课程表" href="javascript:void(0)"*>*我的课程表*</*a*></*li*>  
 </*ul*>  
 </*dd*>  
 </*dl*>  
 </*div*>  
 <*div th:if="*${*session.USER.type!=2*}*" class="menu\_dropdown bk\_2"*>  
 <*dl id="menu-article2"*>  
 <*dt*><*i class="Hui-iconfont"*>*&#xe616;*</*i*>* 课程表管理*<*i class="Hui-iconfont menu\_dropdown-arrow"*>*&#xe6d5;*</*i*></*dt*>  
 <*dd*>  
 <*ul*>  
 <*li*><*a th:data-href="@{/curriculum/myCurriculumPage}" data-title="我的课程表" href="javascript:void(0)"*>*我的课程表*</*a*></*li*>  
 </*ul*>  
 </*dd*>  
 </*dl*>  
 <*dl th:if="*${*session.USER.type==1*}*" id="menu-teacher2"*>  
 <*dt*><*i class="Hui-iconfont"*>*&#xe60d;*</*i*>* 教工管理*<*i class="Hui-iconfont menu\_dropdown-arrow"*>*&#xe6d5;*</*i*></*dt*>  
 <*dd*>  
 <*ul*>  
 <*li*><*a th:data-href="@{/teacher/teacherPage?teacherID=}+${session.USER.contrastTeacherID}" data-title="教工列表" href="javascript:;"*>*教工列表*</*a*></*li*>  
 </*ul*>  
 </*dd*>  
 </*dl*>  
 <*dl th:if="*${*session.USER.type==0*}*" id="menu-student2"*>  
 <*dt*><*i class="Hui-iconfont"*>*&#xe60d;*</*i*>* 学生管理*<*i class="Hui-iconfont menu\_dropdown-arrow"*>*&#xe6d5;*</*i*></*dt*>  
 <*dd*>  
 <*ul*>  
 <*li*><*a th:data-href="@{/student/studentPage?studentID=}+${session.USER.contrastStudentID}" data-title="学生列表" href="javascript:;"*>*学生列表*</*a*></*li*>  
 </*ul*>  
 </*dd*>  
 </*dl*>  
 <*dl id="menu-system2"*>  
 <*dt*><*i class="Hui-iconfont"*>*&#xe62e;*</*i*>* 用户信息管理*<*i class="Hui-iconfont menu\_dropdown-arrow"*>*&#xe6d5;*</*i*></*dt*>  
 <*dd*>  
 <*ul*>  
 <*li*><*a th:data-href="@{/user/userPage?username=}+${session.USER.username}" data-title="用户信息" href="javascript:void(0)"*>*用户信息*</*a*></*li*>  
 </*ul*>  
 </*dd*>  
 </*dl*>  
 </*div*>  
</*aside*>*

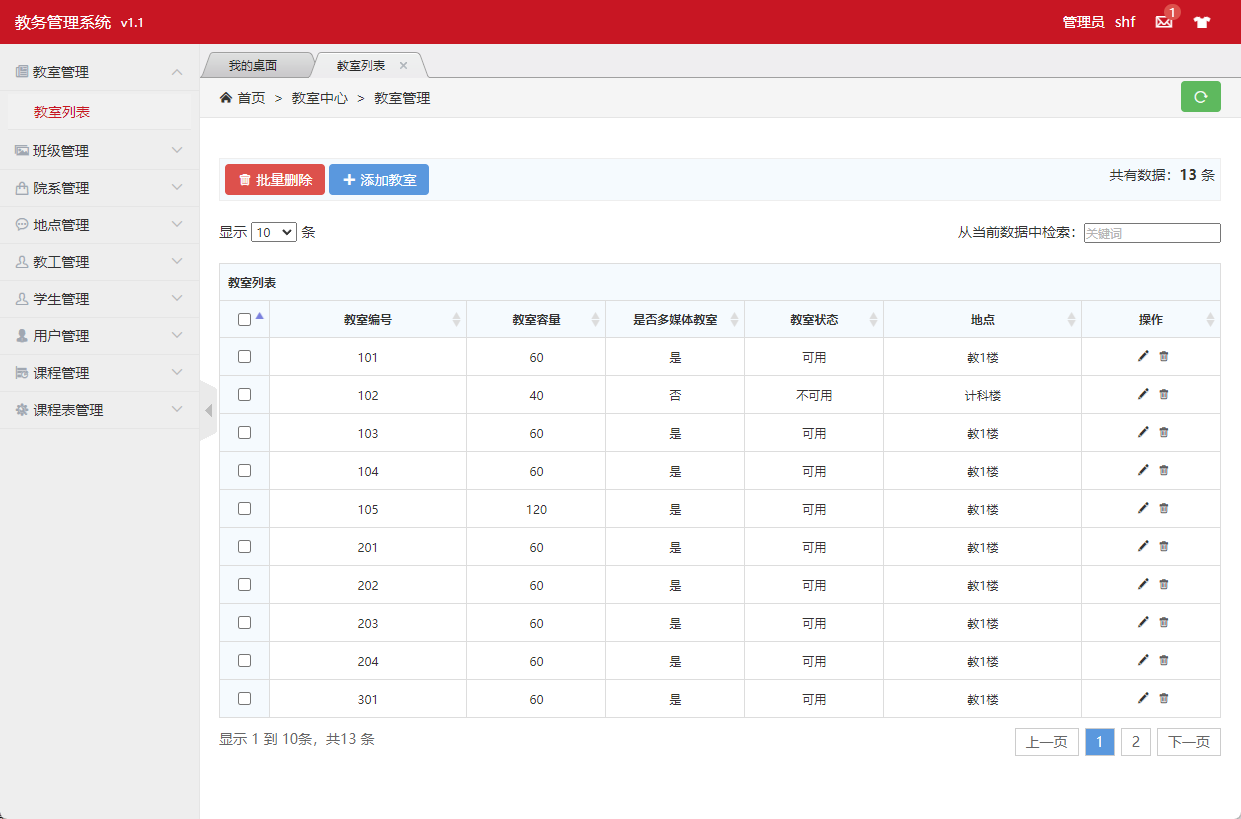
IndexController.java 首页控制器

@Controller  
public class IndexController *{* @GetMapping*(*value = *{*"/","/index"*})* public String index*(){* return "index";  
 *}* @GetMapping*(*"/welcome"*)* public String welcome*(){* return "welcome";  
 *}  
  
}*

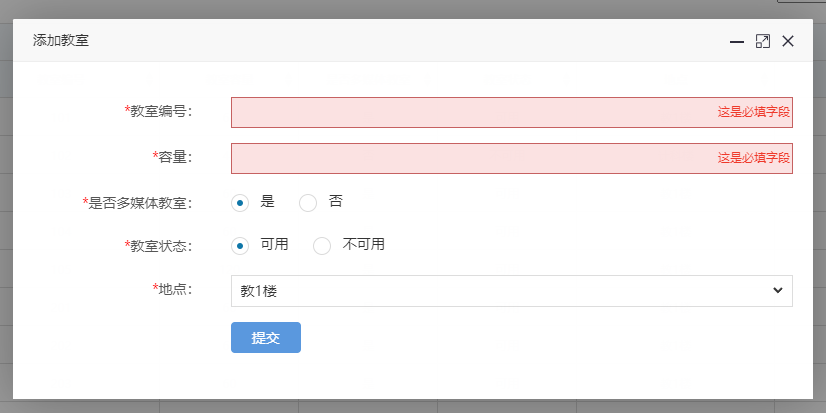
LogoutController.java 用户退出登录

@Controller  
public class LogoutController *{* @GetMapping*(*"/logout"*)* public String logout*(*HttpSession session*){* session.removeAttribute*(*"USER"*)*;  
 return "login";  
 *}  
}*

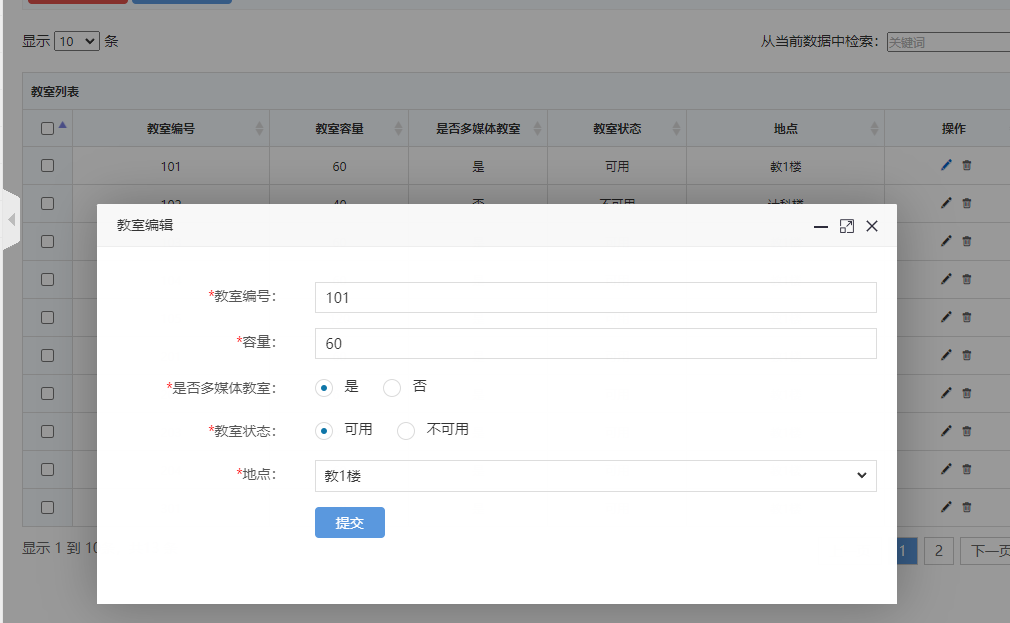
### 3.2.3 教室模块



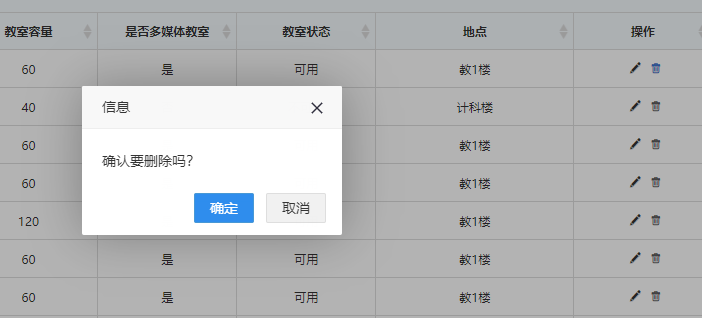
**图3.2.3.1 教室分页列表页面**



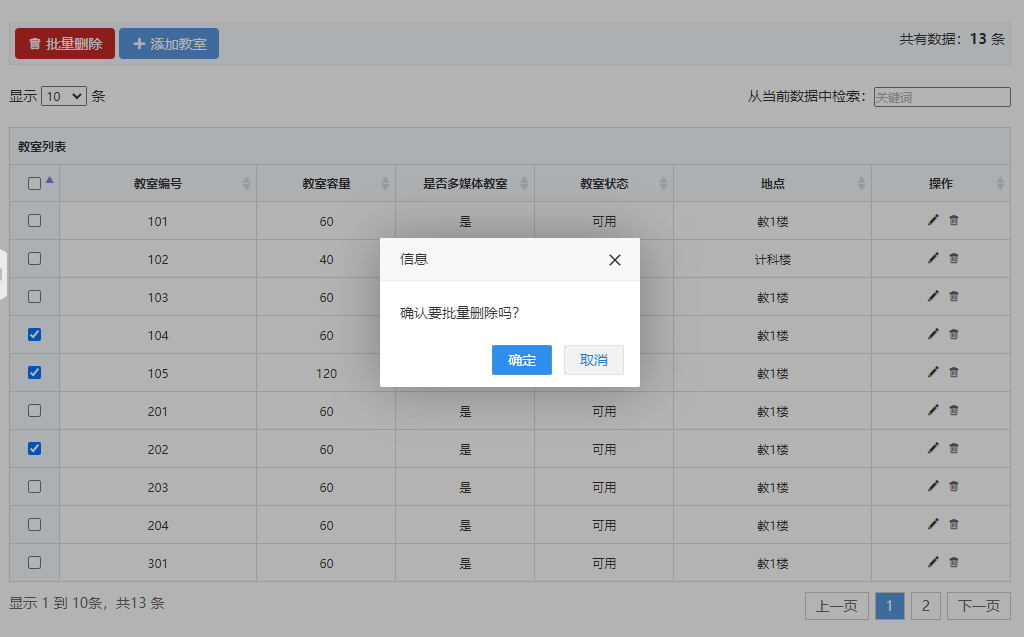
**图3.2.3.2 添加教室页面**



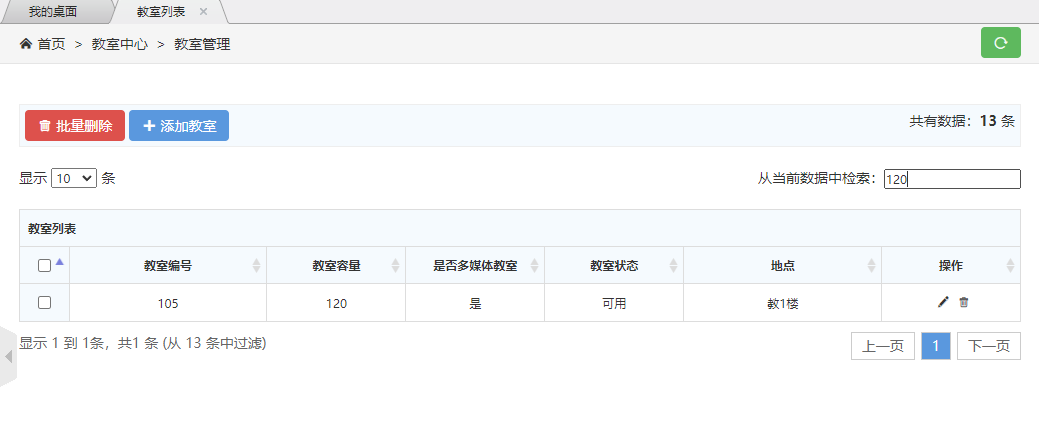
**图3.2.3.3 编辑教室页面**



**图3.2.3.3 删除教室页面**



**图3.2.3.3 批量删除教室页面**



**图3.2.3.4 通过条件查询教室页面**

ClassRoom.java 教室实体类

@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@TableName*(*"classroom"*)*public class ClassRoom implements Serializable *{* @TableId  
 private Integer classRoomNO; *// 教室编号* private Integer capacity; *// 容量* private Integer isMultimedia; *// 是否多媒体教室* private Integer state; *// 教室状态* private Integer locationNo; *// 地点* @TableField*(*exist = false*)* private String locationName; *//验证码  
}*

ClassRoomMapper.java 教室Mapper和Dao

@Repository  
public interface ClassRoomMapper extends BaseMapper*<*ClassRoom*> {  
}*

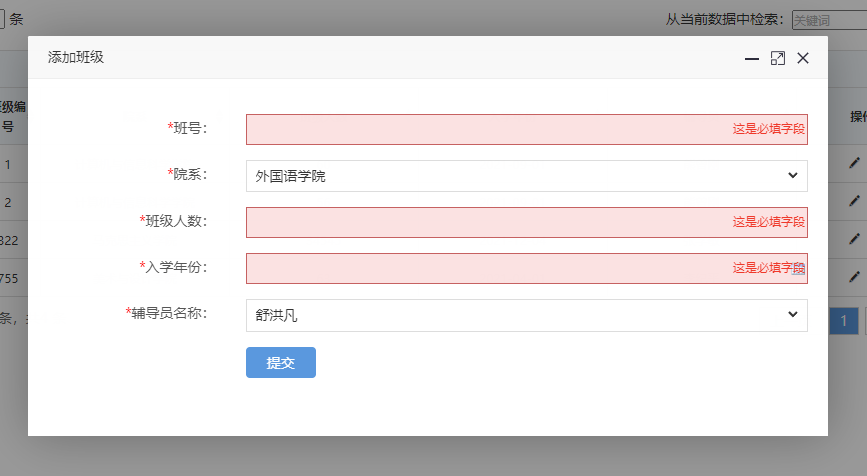
ClassRoomController.java 教室控制器

@Slf4j  
@Controller  
public class ClassRoomController *{* @Autowired  
 private ClassRoomMapper classRoomMapper;  
  
 @Autowired  
 private LocationController locationController;  
  
 @ResponseBody  
 @GetMapping*(*"/classRoom/isAvailableClassRoomNO"*)* public HashMap*<*String, Object*>* isAvailableClassRoomNO*(*ClassRoom classRoom*){* ClassRoom room = classRoomMapper.selectById*(*classRoom.getClassRoomNO*())*;  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*room == null*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",classRoom.getClassRoomNO*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",classRoom.getClassRoomNO*())*;  
 *}* return map;  
 *}* @ResponseBody  
 @GetMapping*(*"/classRoom/classRoomList"*)* public List*<*ClassRoom*>* toClassRoomList*(){* return classRoomMapper.selectList*(*null*)*;  
 *}* public ClassRoom getClassRoomById*(*Integer id*){* return classRoomMapper.selectById*(*id*)*;  
 *}* @GetMapping*(*"/classRoom/classRoomListPage"*)* public String toClassRoomList*(*Model model*){* List*<*ClassRoom*>* classRoomList = classRoomMapper.selectList*(*null*)*;  
 for *(*ClassRoom classRoom : classRoomList*) {* classRoom.setLocationName*(*locationController.getLocationById*(*classRoom.getLocationNo*())*.getLocationName*())*;  
 *}* model.addAttribute*(*"classRoomList",classRoomList*)*;  
 return "classRoom/admin-list";  
 *}* @ResponseBody  
 @PostMapping*(*"/classRoom/deleteClassRoomById"*)* public HashMap*<*String, Object*>* deleteClassRoomById*(*ClassRoom classRoom*){  
 log*.info*(*classRoom.toString*())*;  
  
 int count = classRoomMapper.deleteById*(*classRoom.getClassRoomNO*())*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",classRoom.getClassRoomNO*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",classRoom.getClassRoomNO*())*;  
 *}* return map;  
 *}* @ResponseBody  
 @PostMapping*(*"/classRoom/deleteBatchIds"*)* public HashMap*<*String, Object*>* deleteBatchIds*(*String ids*){* System.*out*.println*(*ids*)*;  
  
 JSON parse = JSONUtil.*parse(*ids*)*;  
 int count = classRoomMapper.deleteBatchIds*((*Collection*<*? extends Serializable*>)* parse*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count>=1*){* map.put*(*"code","100"*)*;  
 map.put*(*"ids",ids*)*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"ids",ids*)*;  
 *}* return map;  
 *}* @GetMapping*(*"/classRoom/classRoomAdd"*)* public String toUserAdd*(*Model model*){* List*<*Location*>* locations = locationController.locationList*()*;  
 model.addAttribute*(*"locationList",locations*)*;  
 return "classRoom/admin-add";  
 *}* @ResponseBody  
 @PostMapping*(*"/classRoom/classRoomAdd"*)* public HashMap*<*String, Object*>* classRoomAdd*(*ClassRoom classRoom*){  
 log*.info*(*classRoom.toString*())*;  
  
 int count = classRoomMapper.insert*(*classRoom*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",classRoom.getClassRoomNO*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",classRoom.getClassRoomNO*())*;  
 *}* return map;  
 *}* @GetMapping*(*"/classRoom/classRoomUpdate/{id}"*)* public String toClassRoomUpdate*(*@PathVariable*(*"id"*)* Integer id, Model model*){* List*<*Location*>* locations = locationController.locationList*()*;  
 model.addAttribute*(*"locationList",locations*)*;  
 ClassRoom t = classRoomMapper.selectById*(*id*)*;  
 model.addAttribute*(*"classRoom",t*)*;  
 return "classRoom/admin-update";  
 *}* @ResponseBody  
 @PostMapping*(*"/classRoom/classRoomUpdate"*)* public HashMap*<*String, Object*>* classRoomUpdate*(*ClassRoom classRoom*){  
 log*.info*(*classRoom.toString*())*;  
  
 int count = classRoomMapper.updateById*(*classRoom*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",classRoom.getClassRoomNO*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",classRoom.getClassRoomNO*())*;  
 *}* return map;  
 *}  
}*

### 3.2.4 班级模块



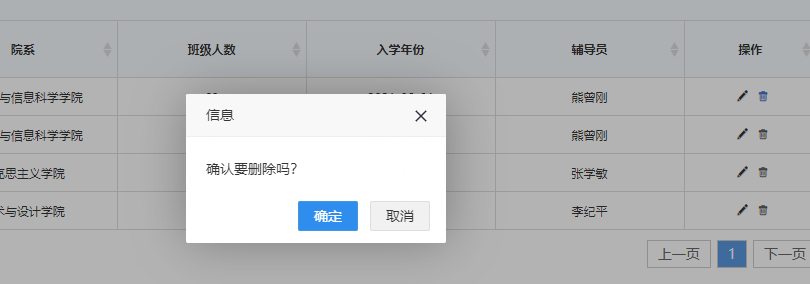
**图3.2.4.1 班级列表页面**



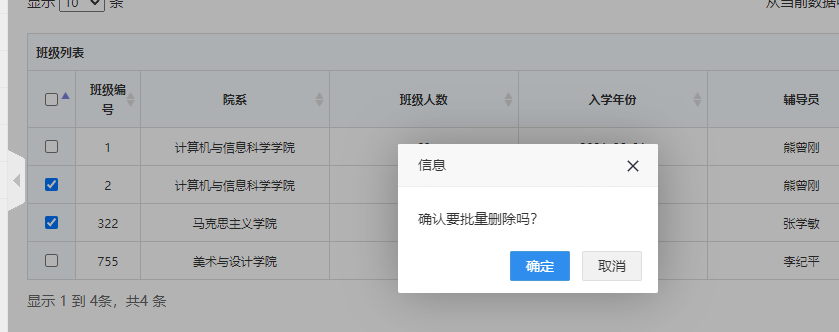
**图3.2.4.2 添加班级页面**



**图3.2.4.3 编辑班级页面**



**图3.2.4.4 删除班级页面**



**图3.2.4.5 批量删除班级页面**



**图3.2.4.5 筛选班级页面**

Class.java 班级实体类

@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@TableName*(*"class"*)*public class Class implements Serializable *{* @TableId  
 private Integer classNO; *//班号* private Integer facultyNO; *//院系* @TableField*(*exist = false*)* private String facultyName; *//院系名称* private Integer classSize; *// 班级人数* @DateTimeFormat*(*pattern = "yyyy-MM-dd"*)* private Date year; *// 入学年份* private Integer counsellorNO; *// 辅导员* @TableField*(*exist = false*)* private String counsellorName; *//辅导员名称  
}*

ClassMapper.java 班级mapper和Dao

@Repository  
public interface ClassMapper extends BaseMapper*<*Class*> {  
}*

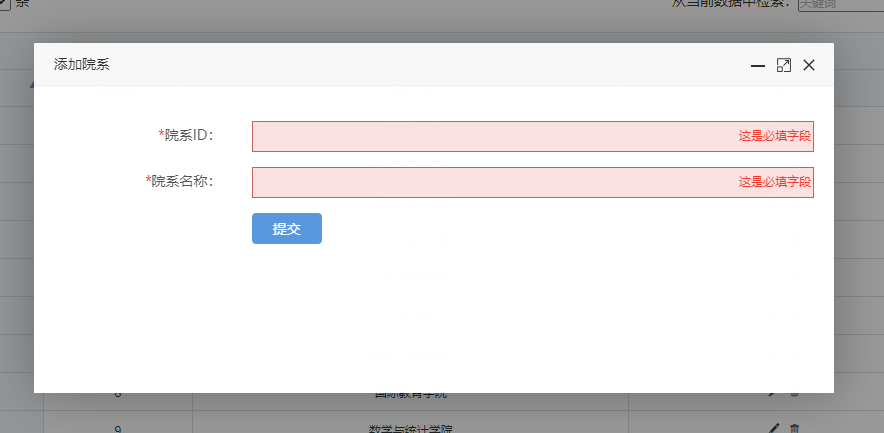
ClassController.java 班级控制器

@Slf4j  
@Controller  
public class ClassController *{* @Autowired  
 private ClassMapper classMapper;  
  
 @Autowired  
 private FacultyController facultyController;  
  
 @Autowired  
 private TeacherController teacherController;  
  
 @ResponseBody  
 @GetMapping*(*"/class/isAvailableClassNO"*)* public HashMap*<*String, Object*>* isAvailableID*(*Class classes*){* Class aClass = classMapper.selectById*(*classes.getClassNO*())*;  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*aClass == null*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",classes.getClassNO*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",classes.getClassNO*())*;  
 *}* return map;  
 *}* @ResponseBody  
 @GetMapping*(*"/class/classList"*)* public List*<*Class*>* toClassList*(){* return classMapper.selectList*(*null*)*;  
 *}* public Class getClassById*(*Integer id*){* return classMapper.selectById*(*id*)*;  
 *}* @GetMapping*(*"/class/classListPage"*)* public String toClassList*(*Model model*){* List*<*Class*>* classList = classMapper.selectList*(*null*)*;  
 for *(*Class classes : classList*) {* classes.setFacultyName*(*facultyController.getFacultyById*(*classes.getFacultyNO*())*.getFacultyName*())*;  
 classes.setCounsellorName*(*teacherController.getTeacherById*(*classes.getCounsellorNO*())*.getName*())*;  
 *}* model.addAttribute*(*"classList",classList*)*;  
 return "class/admin-list";  
 *}* @ResponseBody  
 @PostMapping*(*"/class/deleteClassById"*)* public HashMap*<*String, Object*>* deleteClassById*(*Class classes*){  
 log*.info*(*classes.toString*())*;  
  
 int count = classMapper.deleteById*(*classes.getClassNO*())*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",classes.getClassNO*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",classes.getClassNO*())*;  
 *}* return map;  
 *}* @ResponseBody  
 @PostMapping*(*"/class/deleteBatchIds"*)* public HashMap*<*String, Object*>* deleteBatchIds*(*String ids*){* System.*out*.println*(*ids*)*;  
  
 JSON parse = JSONUtil.*parse(*ids*)*;  
 int count = classMapper.deleteBatchIds*((*Collection*<*? extends Serializable*>)* parse*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count>=1*){* map.put*(*"code","100"*)*;  
 map.put*(*"ids",ids*)*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"ids",ids*)*;  
 *}* return map;  
 *}* @GetMapping*(*"/class/classAdd"*)* public String toUserAdd*(*Model model*){* List*<*Teacher*>* teachers = teacherController.teacherList*()*;  
 model.addAttribute*(*"teacherList",teachers*)*;  
 List*<*Faculty*>* facultyList = facultyController.toFacultyList*()*;  
 model.addAttribute*(*"facultyList",facultyList*)*;  
 return "class/admin-add";  
 *}* @ResponseBody  
 @PostMapping*(*"/class/classAdd"*)* public HashMap*<*String, Object*>* classAdd*(*Class classes*){  
 log*.info*(*classes.toString*())*;  
  
 int count = classMapper.insert*(*classes*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",classes.getClassNO*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",classes.getClassNO*())*;  
 *}* return map;  
 *}* @GetMapping*(*"/class/classUpdate/{id}"*)* public String toClassUpdate*(*@PathVariable*(*"id"*)* Integer id, Model model*){* List*<*Teacher*>* teachers = teacherController.teacherList*()*;  
 model.addAttribute*(*"teacherList",teachers*)*;  
 List*<*Faculty*>* facultyList = facultyController.toFacultyList*()*;  
 model.addAttribute*(*"facultyList",facultyList*)*;  
 Class t = classMapper.selectById*(*id*)*;  
 model.addAttribute*(*"class",t*)*;  
 return "class/admin-update";  
 *}* @ResponseBody  
 @PostMapping*(*"/class/classUpdate"*)* public HashMap*<*String, Object*>* classUpdate*(*Class classes*){  
 log*.info*(*classes.toString*())*;  
  
 int count = classMapper.updateById*(*classes*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",classes.getClassNO*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",classes.getClassNO*())*;  
 *}* return map;  
 *}  
}*

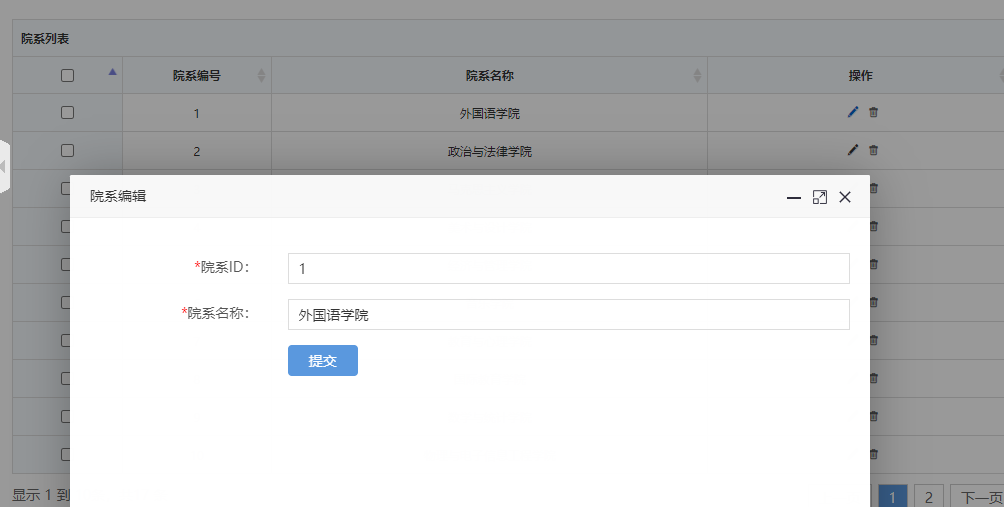
### 3.2.5 院系模块



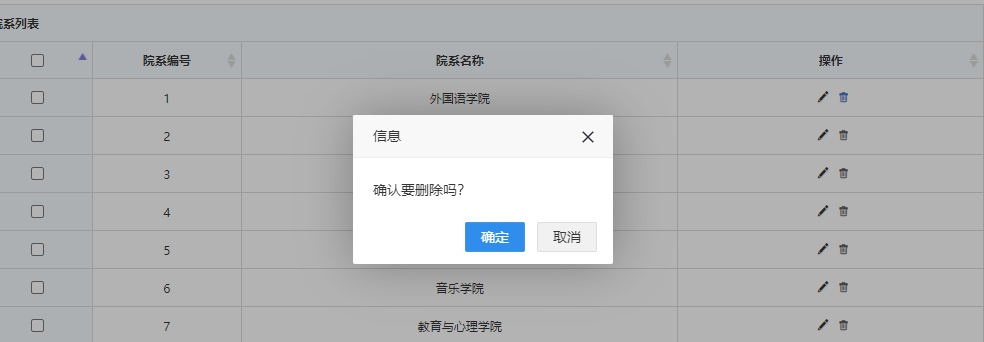
**图3.2.5.1 院系页面**



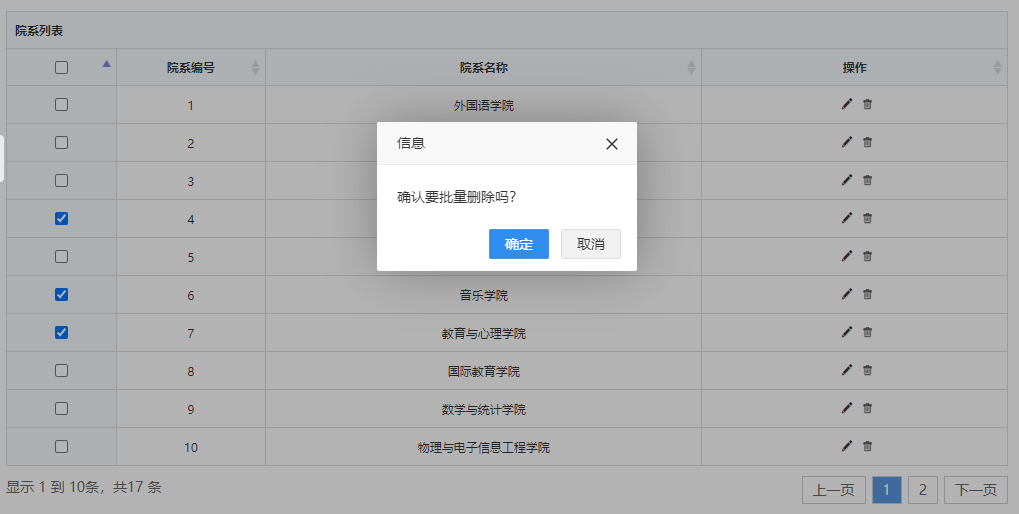
**图3.2.5.2 添加院系页面**



**图3.2.5.3 编辑院系页面**



**图3.2.5.4 删除院系页面**



**图3.2.5.5 批量删除院系页面**



**图3.2.5.6 根据条件查询院系页面**

Faculty.java 院系实体类

@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@TableName*(*"faculty"*)*public class Faculty implements Serializable *{* @TableId  
 private Integer facultyNO; *//#院系ID* private String facultyName; *//#院系名称  
}*

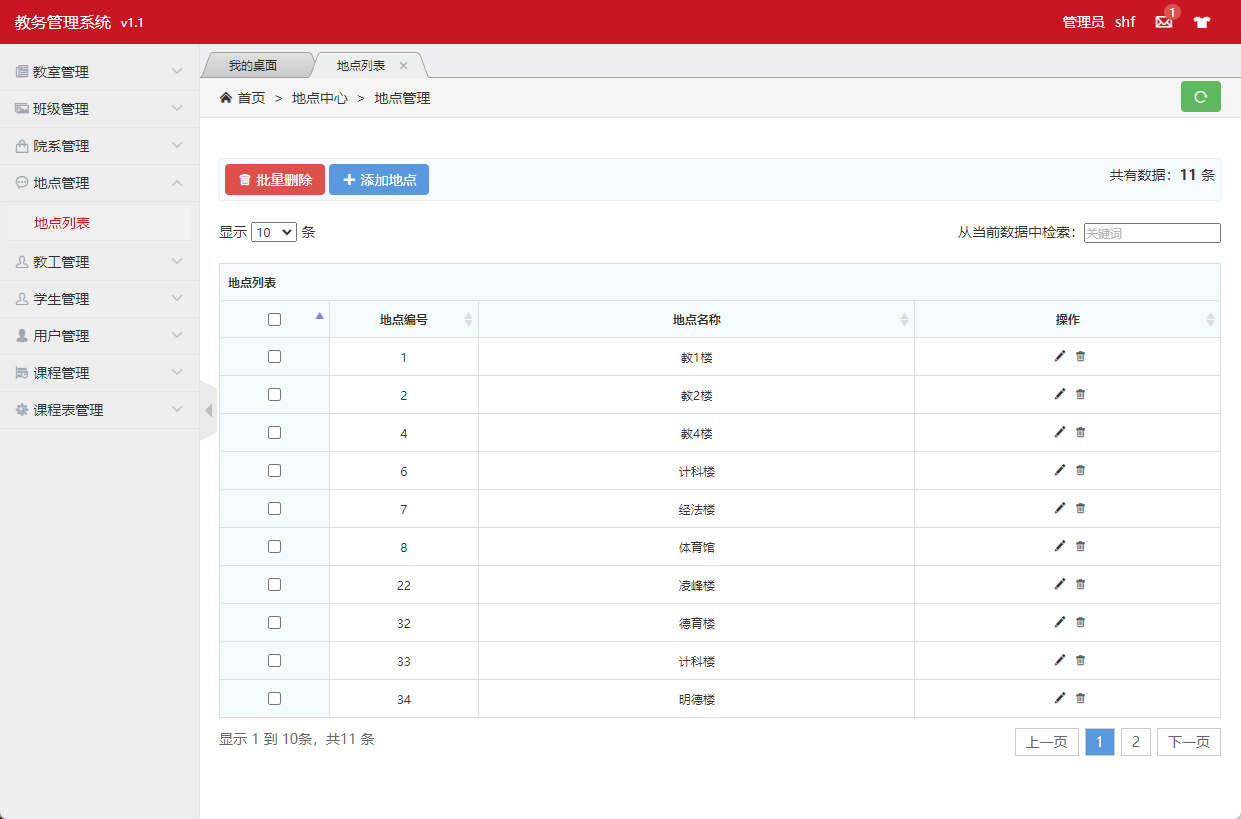
FacultyMapper.java 院系mapper和Dao

@Repository  
public interface FacultyMapper extends BaseMapper*<*Faculty*> {  
}*

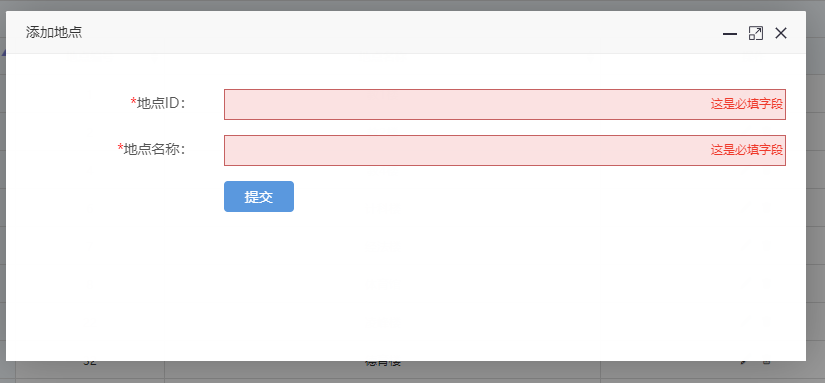
FacultyController.java 院系控制器

@Slf4j  
@Controller  
public class FacultyController *{* @Autowired  
 private FacultyMapper facultyMapper;  
  
 @ResponseBody  
 @GetMapping*(*"/faculty/isAvailableFacultyNO"*)* public HashMap*<*String, Object*>* isAvailableFacultyNO*(*Faculty faculty*){* Faculty room = facultyMapper.selectById*(*faculty.getFacultyNO*())*;  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*room == null*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",faculty.getFacultyNO*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",faculty.getFacultyNO*())*;  
 *}* return map;  
 *}* @ResponseBody  
 @GetMapping*(*"/faculty/facultyList"*)* public List*<*Faculty*>* toFacultyList*(){* return facultyMapper.selectList*(*null*)*;  
 *}* public Faculty getFacultyById*(*Integer id*){* return facultyMapper.selectById*(*id*)*;  
 *}* @GetMapping*(*"/faculty/facultyListPage"*)* public String toFacultyList*(*Model model*){* List*<*Faculty*>* facultyList = facultyMapper.selectList*(*null*)*;  
 model.addAttribute*(*"facultyList",facultyList*)*;  
 return "faculty/admin-list";  
 *}* @ResponseBody  
 @PostMapping*(*"/faculty/deleteFacultyById"*)* public HashMap*<*String, Object*>* deleteFacultyById*(*Faculty faculty*){  
 log*.info*(*faculty.toString*())*;  
  
 int count = facultyMapper.deleteById*(*faculty.getFacultyNO*())*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",faculty.getFacultyNO*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",faculty.getFacultyNO*())*;  
 *}* return map;  
 *}* @ResponseBody  
 @PostMapping*(*"/faculty/deleteBatchIds"*)* public HashMap*<*String, Object*>* deleteBatchIds*(*String ids*){* System.*out*.println*(*ids*)*;  
  
 JSON parse = JSONUtil.*parse(*ids*)*;  
 int count = facultyMapper.deleteBatchIds*((*Collection*<*? extends Serializable*>)* parse*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count>=1*){* map.put*(*"code","100"*)*;  
 map.put*(*"ids",ids*)*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"ids",ids*)*;  
 *}* return map;  
 *}* @GetMapping*(*"/faculty/facultyAdd"*)* public String toUserAdd*(){* return "faculty/admin-add";  
 *}* @ResponseBody  
 @PostMapping*(*"/faculty/facultyAdd"*)* public HashMap*<*String, Object*>* facultyAdd*(*Faculty faculty*){  
 log*.info*(*faculty.toString*())*;  
  
 int count = facultyMapper.insert*(*faculty*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",faculty.getFacultyNO*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",faculty.getFacultyNO*())*;  
 *}* return map;  
 *}* @GetMapping*(*"/faculty/facultyUpdate/{id}"*)* public String toFacultyUpdate*(*@PathVariable*(*"id"*)* Integer id, Model model*){* Faculty t = facultyMapper.selectById*(*id*)*;  
 model.addAttribute*(*"faculty",t*)*;  
 return "faculty/admin-update";  
 *}* @ResponseBody  
 @PostMapping*(*"/faculty/facultyUpdate"*)* public HashMap*<*String, Object*>* facultyUpdate*(*Faculty faculty*){  
 log*.info*(*faculty.toString*())*;  
  
 int count = facultyMapper.updateById*(*faculty*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",faculty.getFacultyNO*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",faculty.getFacultyNO*())*;  
 *}* return map;  
 *}  
}*

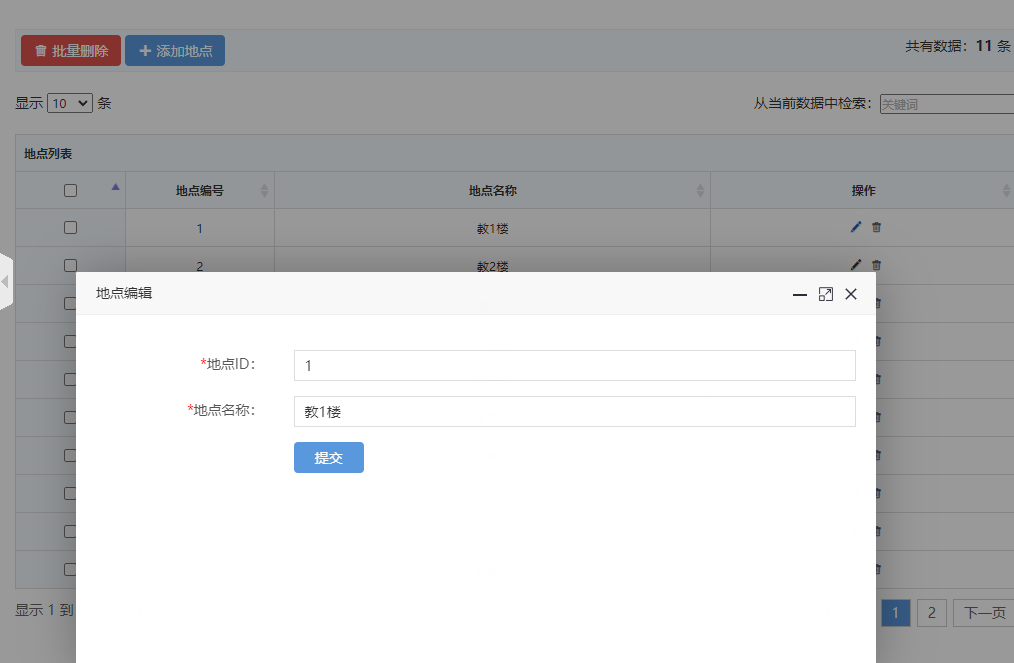
### 3.2.6 地点模块



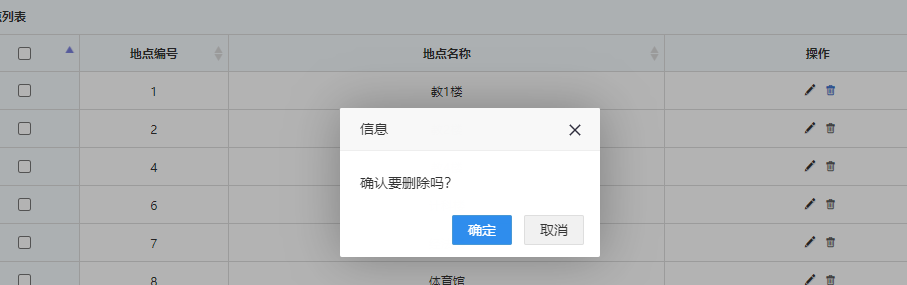
**图3.2.6.1 地点列表页面**



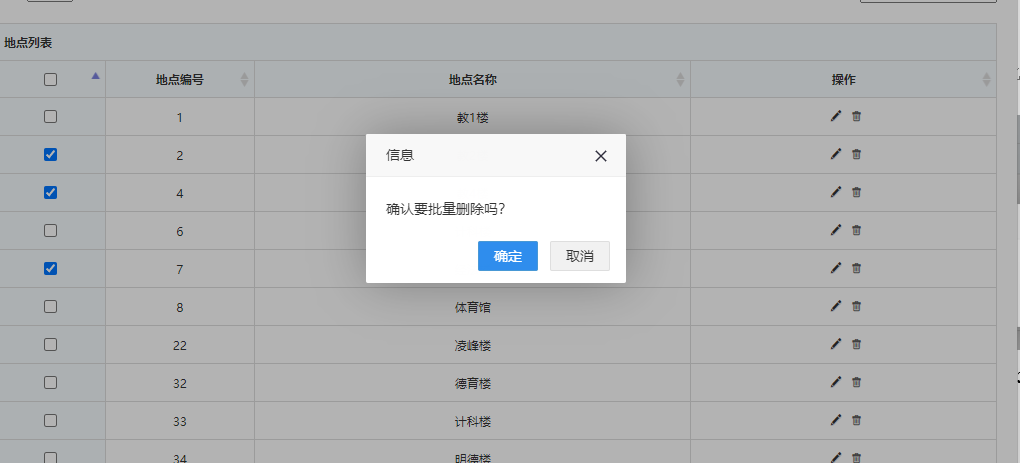
**图3.2.6.2 添加地点页面**



**图3.2.6.3 修改地点页面**



**图3.2.6.4 删除地点页面**



**图3.2.6.4 批量删除地点页面**



**图3.2.6.5 筛选地点页面**

Location.java 地点实体类

@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@TableName*(*"location"*)*public class Location implements Serializable *{* @TableId  
 private Integer locationNo; *// 地点ID* private String locationName; *// 地点名称  
}*

LocationMapper.java 地点mapper和Dao

@Repository  
public interface LocationMapper extends BaseMapper*<*Location*> {  
}*

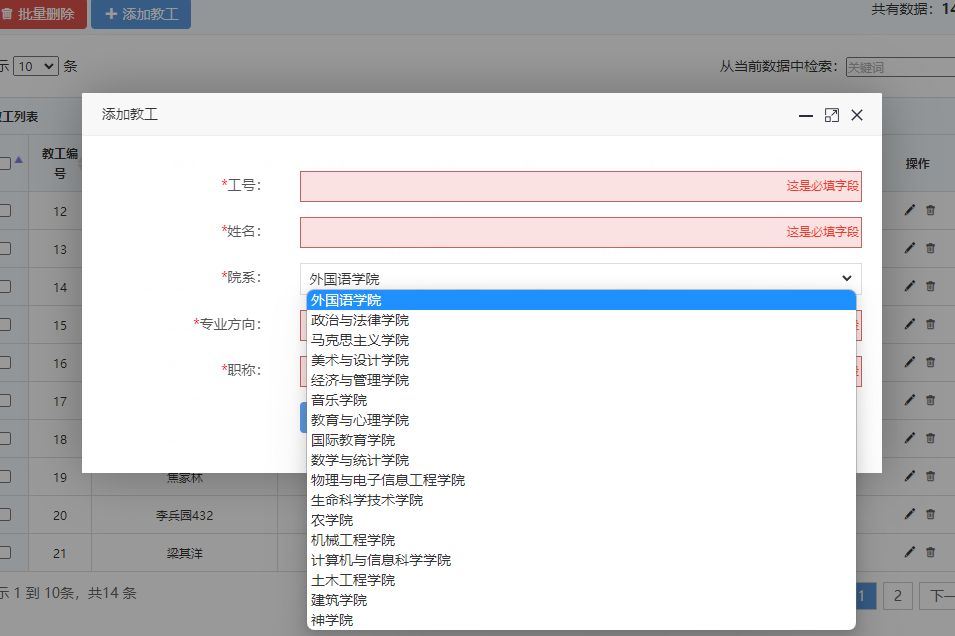
LocationController.java 地点控制器

@Slf4j  
@Controller  
public class LocationController *{* @Autowired  
 private LocationMapper locationMapper;  
  
 @ResponseBody  
 @GetMapping*(*"/location/isAvailableLocationNO"*)* public HashMap*<*String, Object*>* isAvailableLocationNO*(*Location location*){* Location room = locationMapper.selectById*(*location.getLocationNo*())*;  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*room == null*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",location.getLocationNo*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",location.getLocationNo*())*;  
 *}* return map;  
 *}* public Location getLocationById*(*Integer id*){* return locationMapper.selectById*(*id*)*;  
 *}* public List*<*Location*>* locationList*(){* return locationMapper.selectList*(*null*)*;  
 *}* @GetMapping*(*"/location/locationListPage"*)* public String toLocationList*(*Model model*){* List*<*Location*>* locationList = locationMapper.selectList*(*null*)*;  
 model.addAttribute*(*"locationList",locationList*)*;  
 return "location/admin-list";  
 *}* @ResponseBody  
 @PostMapping*(*"/location/deleteLocationById"*)* public HashMap*<*String, Object*>* deleteLocationById*(*Location location*){  
 log*.info*(*location.toString*())*;  
  
 int count = locationMapper.deleteById*(*location.getLocationNo*())*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",location.getLocationNo*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",location.getLocationNo*())*;  
 *}* return map;  
 *}* @ResponseBody  
 @PostMapping*(*"/location/deleteBatchIds"*)* public HashMap*<*String, Object*>* deleteBatchIds*(*String ids*){* System.*out*.println*(*ids*)*;  
  
 JSON parse = JSONUtil.*parse(*ids*)*;  
 int count = locationMapper.deleteBatchIds*((*Collection*<*? extends Serializable*>)* parse*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count>=1*){* map.put*(*"code","100"*)*;  
 map.put*(*"ids",ids*)*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"ids",ids*)*;  
 *}* return map;  
 *}* @GetMapping*(*"/location/locationAdd"*)* public String toUserAdd*(){* return "location/admin-add";  
 *}* @ResponseBody  
 @PostMapping*(*"/location/locationAdd"*)* public HashMap*<*String, Object*>* locationAdd*(*Location location*){  
 log*.info*(*location.toString*())*;  
  
 int count = locationMapper.insert*(*location*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",location.getLocationNo*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",location.getLocationNo*())*;  
 *}* return map;  
 *}* @GetMapping*(*"/location/locationUpdate/{id}"*)* public String toLocationUpdate*(*@PathVariable*(*"id"*)* Integer id,Model model*){* Location t = locationMapper.selectById*(*id*)*;  
 model.addAttribute*(*"location",t*)*;  
 return "location/admin-update";  
 *}* @ResponseBody  
 @PostMapping*(*"/location/locationUpdate"*)* public HashMap*<*String, Object*>* locationUpdate*(*Location location*){  
 log*.info*(*location.toString*())*;  
  
 int count = locationMapper.updateById*(*location*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",location.getLocationNo*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",location.getLocationNo*())*;  
 *}* return map;  
 *}  
  
}*

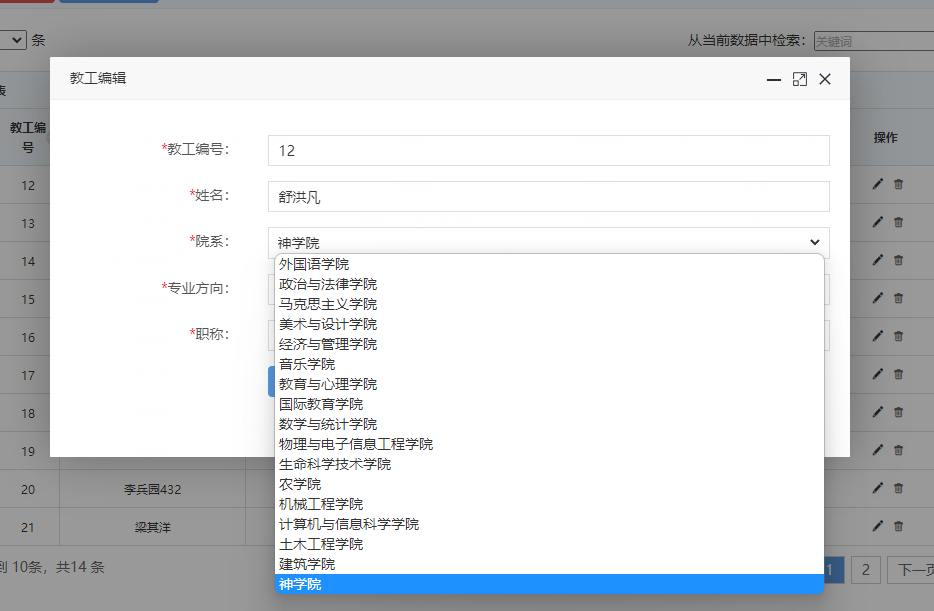
### 3.2.7 教工模块



**图3.2.7.1 教工页面**



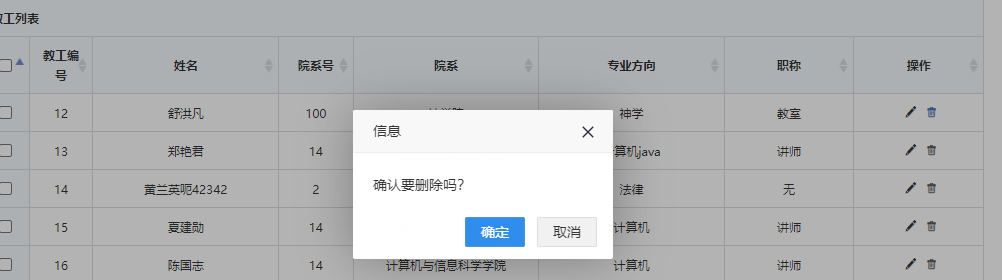
**图3.2.7.2 添加教工页面**



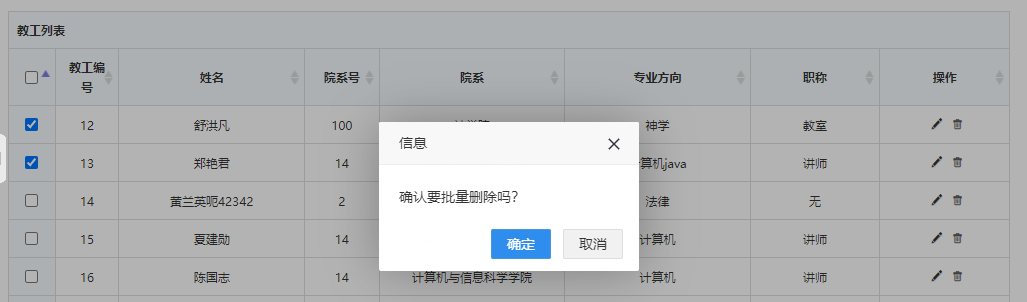
**图3.2.7.3 修改教工页面**



**图3.2.7.4 修改教工成功页面**



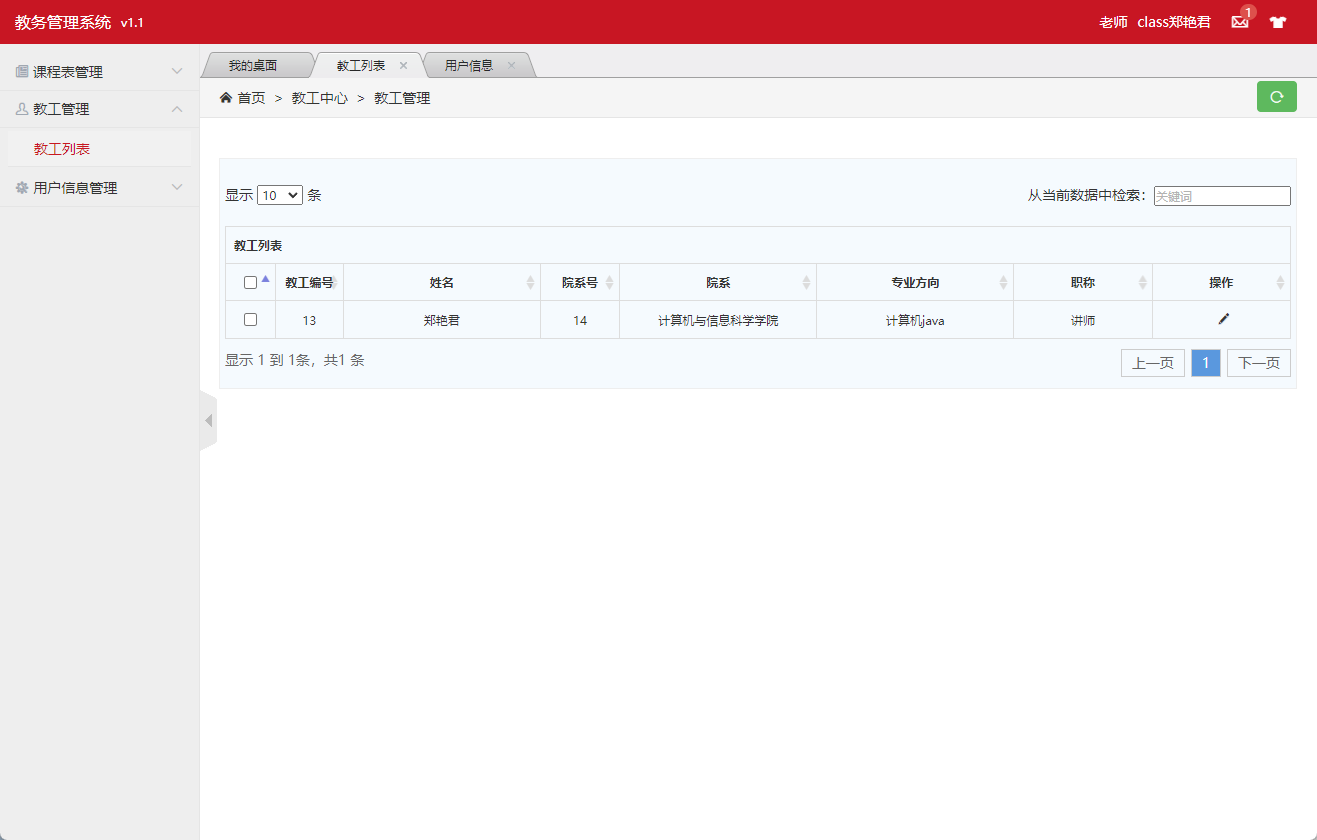
**图3.2.7.5 删除教工页面**



**图3.2.7.6 批量删除教工页面**



**图3.2.7.7 查询教工页面**



**图3.2.7.8 教工用户教工列表页面**



**图3.2.7.9 教工用户修改教工列表页面**

Teacher.java 教工实体类

@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@TableName*(*"teacher"*)*public class Teacher implements Serializable *{* @TableId  
 private Integer teacherID; *// 教工编号* private String name; *// 姓名* private Integer facultyNO; *// 院系* @TableField*(*exist = false*)* private String facultyName; *//#院系名称* private String speciality; *// 专业方向* private String title; *// 职称  
}*

TeacherMapper.java 教工mapper和Dao

@Repository  
public interface TeacherMapper extends BaseMapper*<*Teacher*> {  
}*

TeacherController.java 教工控制器

@Slf4j  
@Controller  
public class TeacherController *{* @Autowired  
 private TeacherMapper teacherMapper;  
  
 @Autowired  
 private FacultyController facultyController;  
  
 @ResponseBody  
 @GetMapping*(*"/teacher/isAvailableTeacherNO"*)* public HashMap*<*String, Object*>* isAvailableTeacherNO*(*Teacher teacher*){* Teacher room = teacherMapper.selectById*(*teacher.getTeacherID*())*;  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*room == null*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",teacher.getTeacherID*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",teacher.getTeacherID*())*;  
 *}* return map;  
 *}* public Teacher getTeacherById*(*Integer id*){* return teacherMapper.selectById*(*id*)*;  
 *}* public List*<*Teacher*>* teacherList*(){* return teacherMapper.selectList*(*null*)*;  
 *}* @GetMapping*(*"/teacher/teacherPage"*)* public String teacherPage*(*Teacher teacher,Model model*){* ArrayList*<*Teacher*>* teachers = new ArrayList*<>()*;  
 Teacher s = teacherMapper.selectById*(*teacher.getTeacherID*())*;  
 s.setFacultyName*(*facultyController.getFacultyById*(*s.getFacultyNO*())*.getFacultyName*())*;  
 teachers.add*(*s*)*;  
 model.addAttribute*(*"teacherList",teachers*)*;  
 return "teacher/admin-list-public";  
 *}* @GetMapping*(*"/teacher/teacherListPage"*)* public String toTeacherList*(*Model model*){* List*<*Teacher*>* teacherList = teacherMapper.selectList*(*null*)*;  
 for *(*Teacher teacher : teacherList*) {* teacher.setFacultyName*(*facultyController.getFacultyById*(*teacher.getFacultyNO*())*.getFacultyName*())*;  
 *}* model.addAttribute*(*"teacherList",teacherList*)*;  
 return "teacher/admin-list";  
 *}* @ResponseBody  
 @PostMapping*(*"/teacher/deleteTeacherById"*)* public HashMap*<*String, Object*>* deleteTeacherById*(*Teacher teacher*){  
 log*.info*(*teacher.toString*())*;  
  
 int count = teacherMapper.deleteById*(*teacher.getTeacherID*())*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",teacher.getTeacherID*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",teacher.getTeacherID*())*;  
 *}* return map;  
 *}* @ResponseBody  
 @PostMapping*(*"/teacher/deleteBatchIds"*)* public HashMap*<*String, Object*>* deleteBatchIds*(*String ids*){* System.*out*.println*(*ids*)*;  
  
 JSON parse = JSONUtil.*parse(*ids*)*;  
 int count = teacherMapper.deleteBatchIds*((*Collection*<*? extends Serializable*>)* parse*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count>=1*){* map.put*(*"code","100"*)*;  
 map.put*(*"ids",ids*)*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"ids",ids*)*;  
 *}* return map;  
 *}* @GetMapping*(*"/teacher/teacherAdd"*)* public String totTeacherAdd*(*Model model*){* List*<*Faculty*>* facultyList = facultyController.toFacultyList*()*;  
 model.addAttribute*(*"facultyList",facultyList*)*;  
 return "teacher/admin-add";  
 *}* @ResponseBody  
 @PostMapping*(*"/teacher/teacherAdd"*)* public HashMap*<*String, Object*>* teacherAdd*(*Teacher teacher*){  
 log*.info*(*teacher.toString*())*;  
  
 int count = teacherMapper.insert*(*teacher*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",teacher.getTeacherID*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",teacher.getTeacherID*())*;  
 *}* return map;  
 *}* @GetMapping*(*"/teacher/teacherUpdate/{id}"*)* public String toTeacherUpdate*(*@PathVariable*(*"id"*)* Integer id,Model model*){* List*<*Faculty*>* facultyList = facultyController.toFacultyList*()*;  
 model.addAttribute*(*"facultyList",facultyList*)*;  
 Teacher t = teacherMapper.selectById*(*id*)*;  
 model.addAttribute*(*"teacher",t*)*;  
 return "teacher/admin-update";  
 *}* @ResponseBody  
 @PostMapping*(*"/teacher/teacherUpdate"*)* public HashMap*<*String, Object*>* teacherUpdate*(*Teacher teacher*){  
 log*.info*(*teacher.toString*())*;  
  
 int count = teacherMapper.updateById*(*teacher*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",teacher.getTeacherID*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",teacher.getTeacherID*())*;  
 *}* return map;  
 *}  
  
}*

### 3.2.8 用户模块



**图3.2.8.1 用户页面**



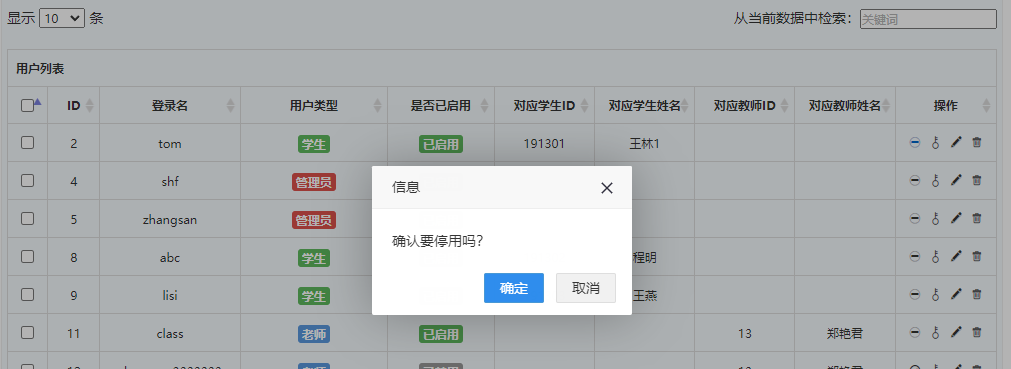
**图3.2.8.2 添加管理员用户页面**



**图3.2.8.3 添加学生用户页面**



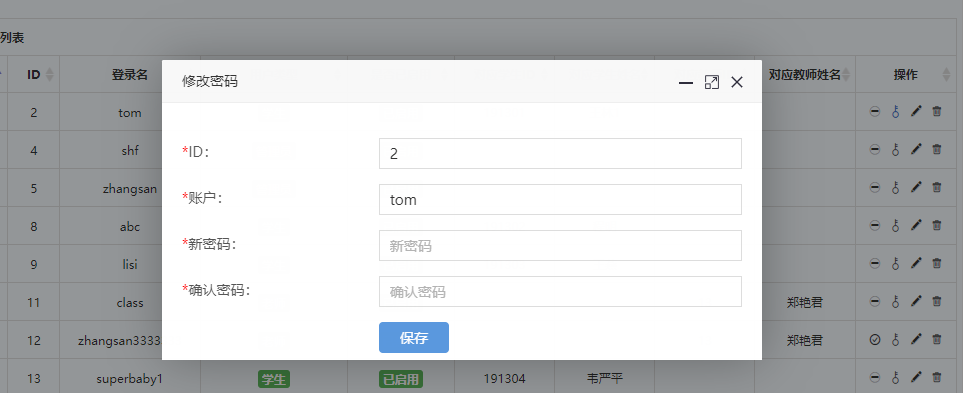
**图3.2.8.4 添加老师用户页面**



**图3.2.8.5 停用用户页面**



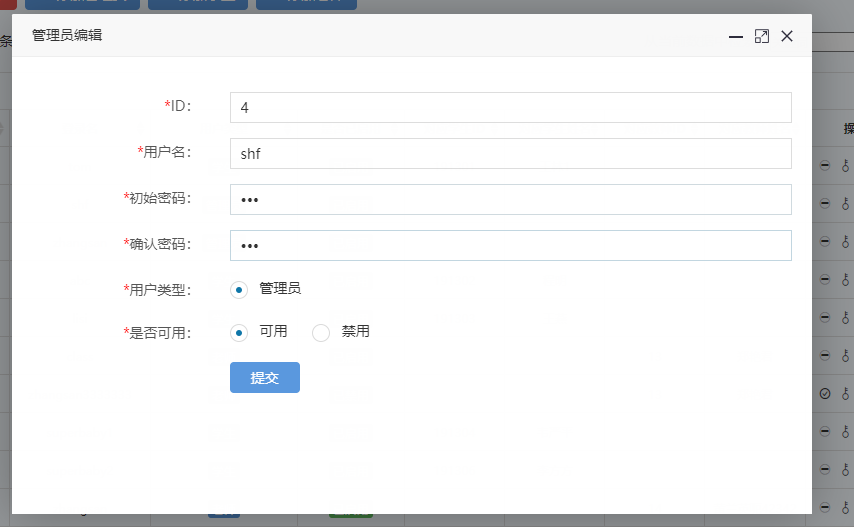
**图3.2.8.6 启用用户页面**



**图3.2.8.6 修改用户密码页面**



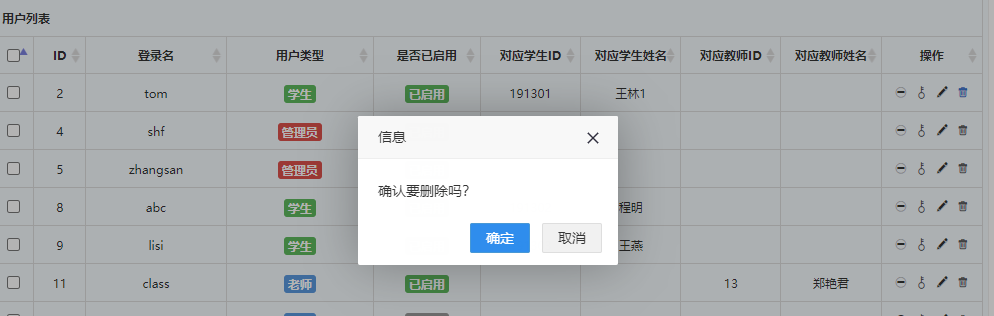
**图3.2.8.7 修改学生用户页面**



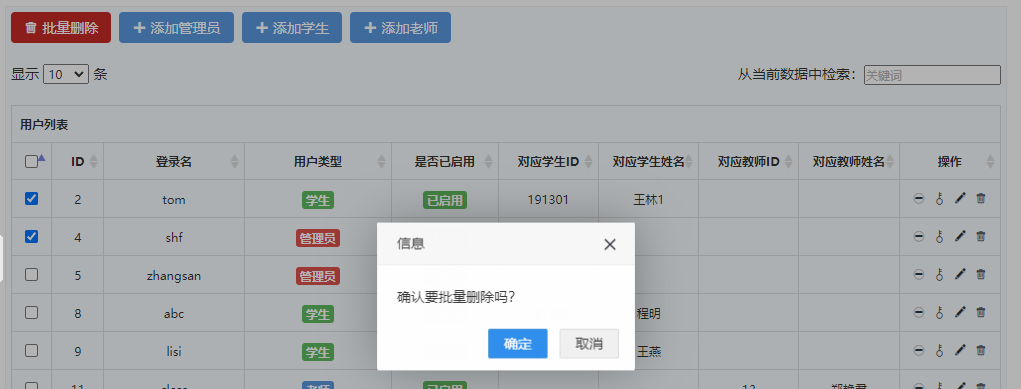
**图3.2.8.7 修改管理员用户页面**



**图3.2.8.7 修改老师用户页面**



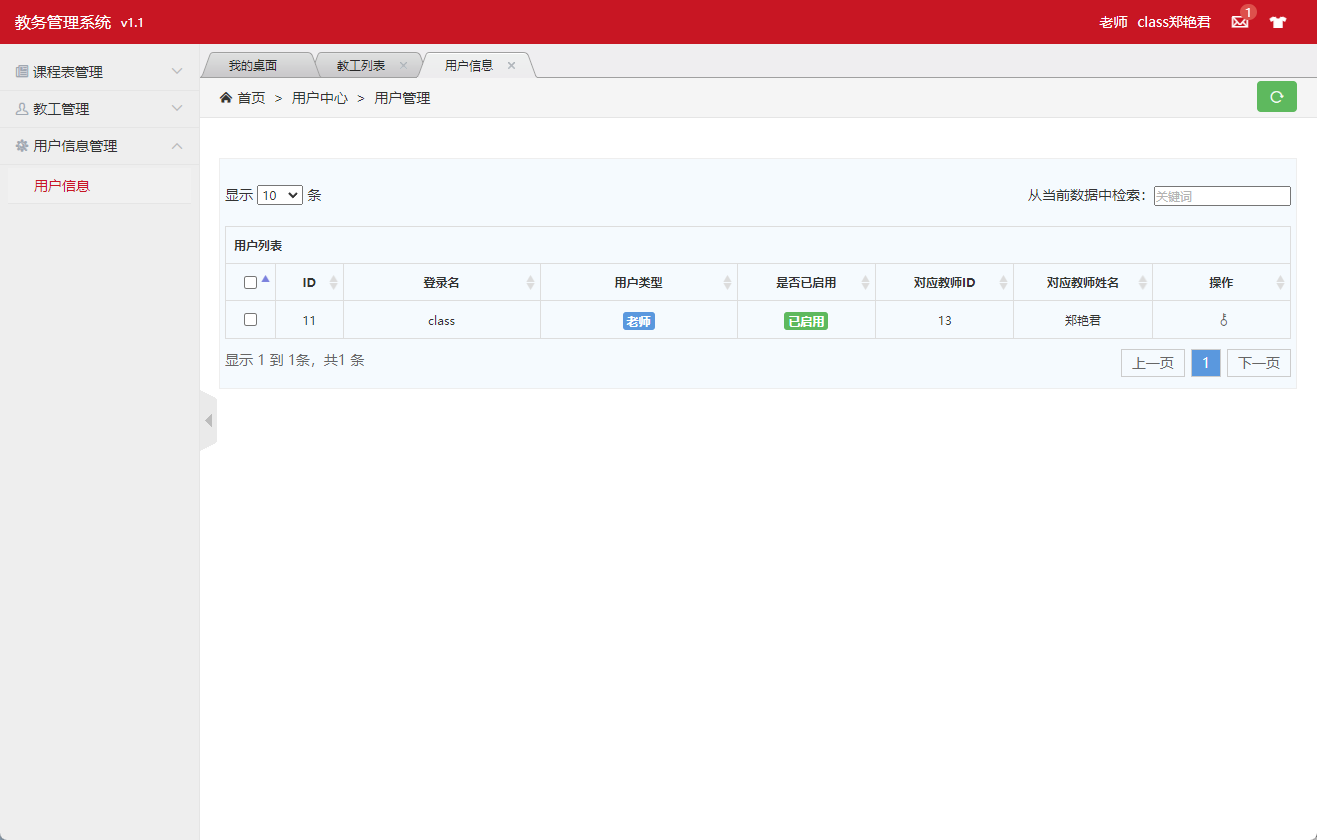
**图3.2.8.7 删除用户页面**



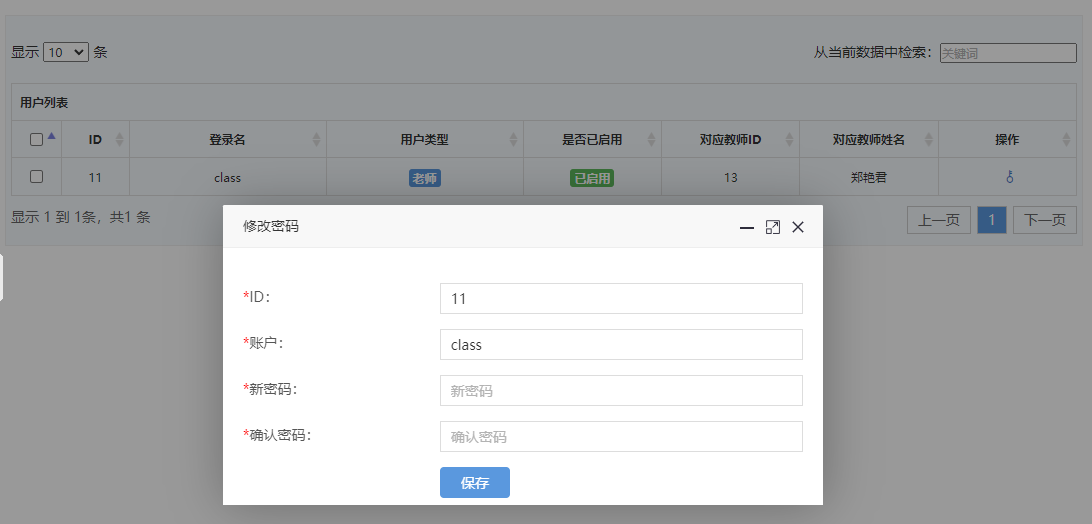
**图3.2.8.8 批量删除用户页面**



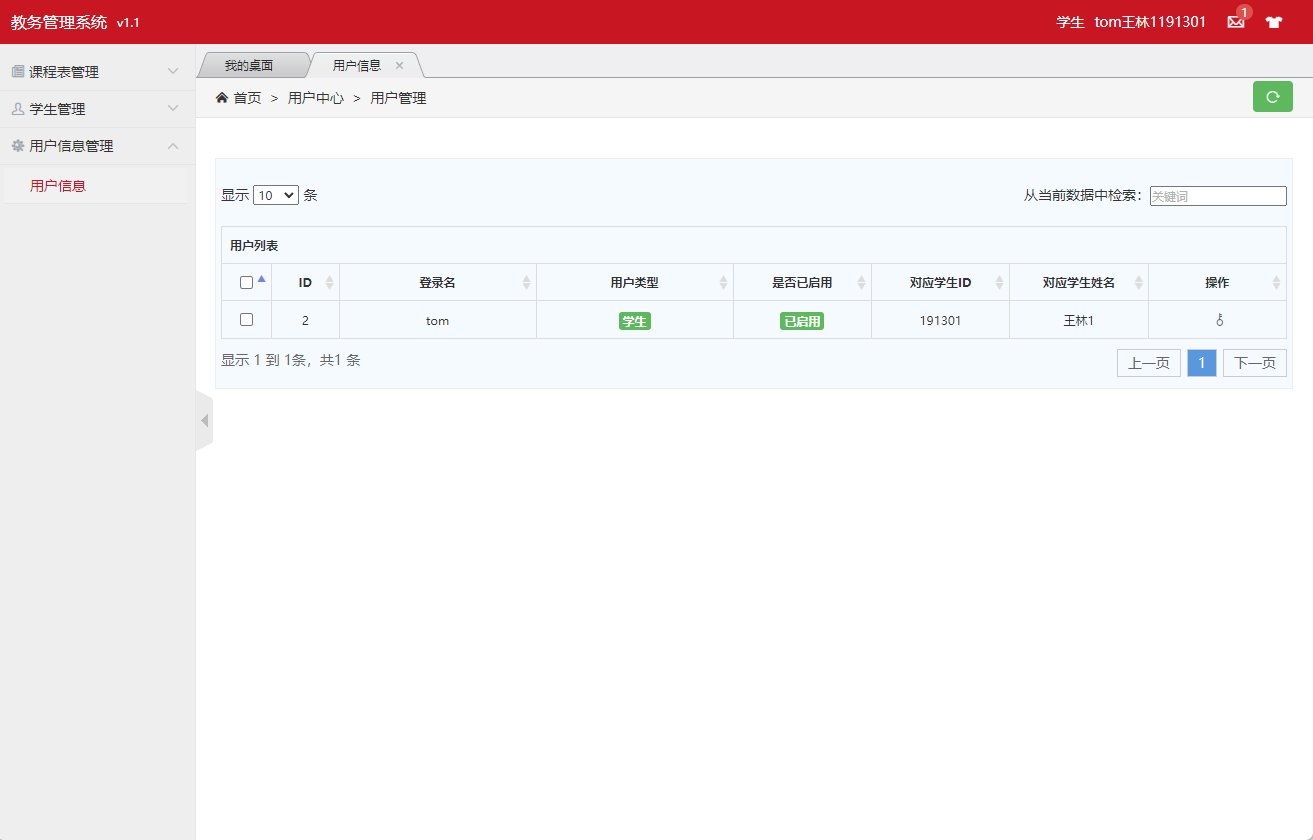
**图3.2.8.8 检索用户页面**



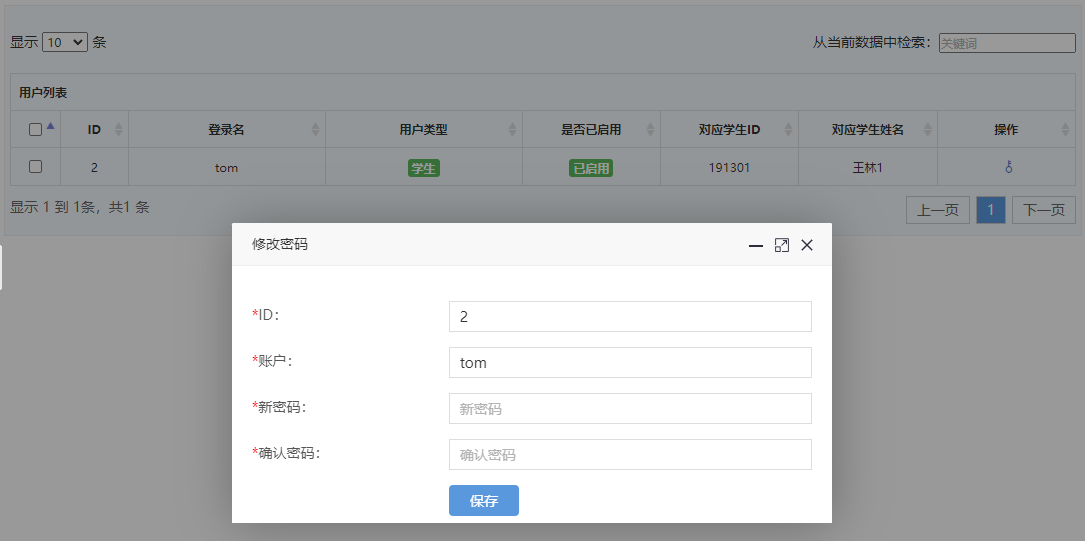
**图3.2.8.9 教师用户页面**



**图3.2.8.10 教师用户修改密码页面**



**图3.2.8.11 学生用户页面**



**图3.2.8.12 学生用户修改密码页面**

User.java 用户实体类

@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@TableName*(*"user"*)*public class User implements Serializable *{* @TableId*(*type = IdType.*AUTO)* private Integer id; *//#ID* private String username; *//#用户名* private String password; *//#密码* private Integer type; *//#用户类型 0学生 1教师 2管理员* private Integer disabled; *//#是否可用* private Integer contrastStudentID; *// 对应学生ID* private Integer contrastTeacherID; *// 对应老师ID* @TableField*(*exist = false*)* private String contrastStudentName; *//对应学生姓名* @TableField*(*exist = false*)* private String contrastTeacherName; *//对应老师姓名* @TableField*(*exist = false*)* private Integer classNO; *//对应学生班级* @TableField*(*exist = false*)* private String captcha; *//验证码  
}*

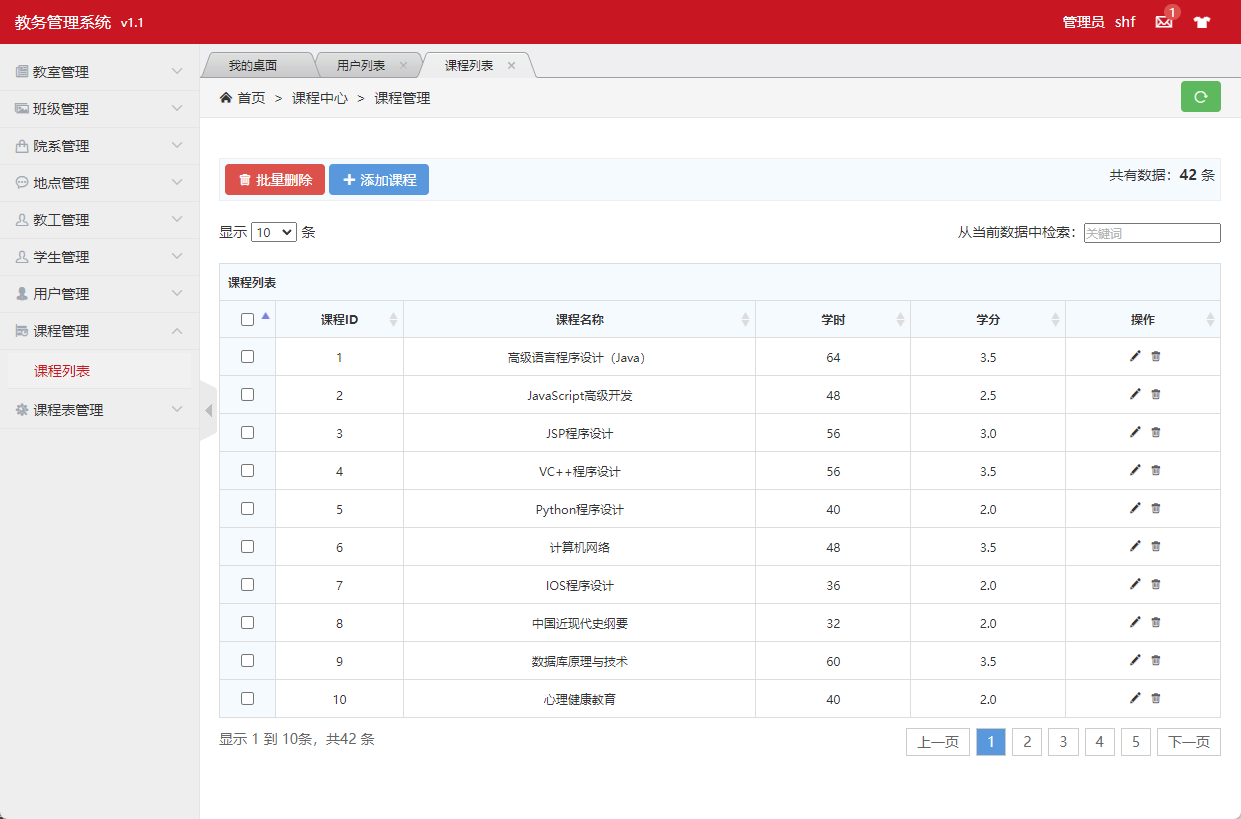
UserMapper.java 用户mapper和Dao

@Repository  
public interface UserMapper extends BaseMapper*<*User*> {  
}*

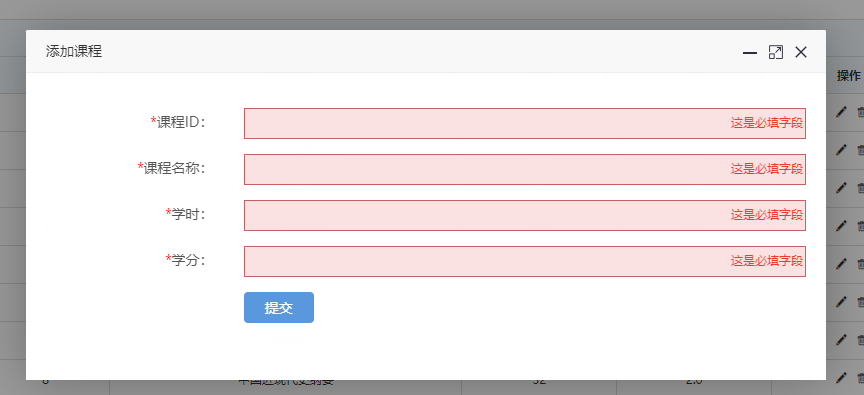
UserController.java 用户控制类类

@Slf4j  
@Controller  
public class UserController *{* @Autowired  
 private UserMapper userMapper;  
  
 @Autowired  
 private StudentController studentController;  
  
 @Autowired  
 private TeacherController teacherController;  
  
 @ResponseBody  
 @GetMapping*(*"/user/isAvailableUserName"*)* public HashMap*<*String, Object*>* isAvailableUserName*(*User useres*){* User aUser = userMapper.selectOne*(*new QueryWrapper*<*User*>()*.eq*(*"username",useres.getUsername*()))*;  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*aUser == null*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",useres.getUsername*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",useres.getUsername*())*;  
 *}* return map;  
 *}* @GetMapping*(*"/user/userListPage"*)* public String toUserList*(*Model model*){* List*<*User*>* userList = userMapper.selectList*(*null*)*;  
 for *(*User user : userList*) {* if *(*user.getContrastStudentID*()*!=null*){* user.setContrastStudentName*(*studentController.getStudentById*(*user.getContrastStudentID*())*.getName*())*;  
 *}* if *(*user.getContrastTeacherID*()*!=null*){* user.setContrastTeacherName*(*teacherController.getTeacherById*(*user.getContrastTeacherID*())*.getName*())*;  
 *}  
 }* model.addAttribute*(*"userList",userList*)*;  
 return "user/admin-list";  
 *}* @GetMapping*(*"/user/userPage"*)* public String userPage*(*User user,Model model*){* User u = userMapper.selectOne*(*new QueryWrapper*<*User*>()*.eq*(*"username",user.getUsername*()))*;  
 if *(*u.getContrastTeacherID*()*!=null*){* Teacher teacherById = teacherController.getTeacherById*(*u.getContrastTeacherID*())*;  
 u.setContrastTeacherName*(*teacherById.getName*())*;  
 *}* if *(*u.getContrastStudentID*()*!=null*){* Student studentById = studentController.getStudentById*(*u.getContrastStudentID*())*;  
 u.setContrastStudentName*(*studentById.getName*())*;  
 *}* ArrayList*<*User*>* users = new ArrayList*<>()*;  
 users.add*(*u*)*;  
 model.addAttribute*(*"userList",users*)*;  
 return "user/admin-list-public";  
 *}* @GetMapping*(*"/user/userListPaging/{num}/{size}"*)* @ResponseBody  
 public Page*<*User*>* pagelist*(*@PathVariable*(*"num"*)* int num, @PathVariable*(*"size"*)* int size*){* QueryWrapper*<*User*>* wrapper = new QueryWrapper*<>()*;  
 Page*<*User*>* page = new Page*<>(*num,size*)*;  
 System.*out*.println*(*page*)*;  
 return userMapper.selectPage*(*page,wrapper*)*;  
 *}* @ResponseBody  
 @PostMapping*(*"/user/stopUserById"*)* public HashMap*<*String, Object*>* stopUserById*(*User user*){  
 log*.info*(*user.toString*())*;  
  
 User u = new User*()*;  
 u.setId*(*user.getId*())*;  
 u.setDisabled*(*1*)*;  
 int count = userMapper.updateById*(*u*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",user.getId*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",user.getId*())*;  
 *}* return map;  
 *}* @ResponseBody  
 @PostMapping*(*"/user/startUserById"*)* public HashMap*<*String, Object*>* startUserById*(*User user*){  
 log*.info*(*user.toString*())*;  
  
 User u = new User*()*;  
 u.setId*(*user.getId*())*;  
 u.setDisabled*(*0*)*;  
 int count = userMapper.updateById*(*u*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",user.getId*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",user.getId*())*;  
 *}* return map;  
 *}* @ResponseBody  
 @PostMapping*(*"/user/deleteUserById"*)* public HashMap*<*String, Object*>* deleteUserById*(*User user*){  
 log*.info*(*user.toString*())*;  
  
 int count = userMapper.deleteById*(*user.getId*())*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",user.getId*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",user.getId*())*;  
 *}* return map;  
 *}* @ResponseBody  
 @PostMapping*(*"/user/deleteBatchIds"*)* public HashMap*<*String, Object*>* deleteBatchIds*(*String ids*){* System.*out*.println*(*ids*)*;  
  
 JSON parse = JSONUtil.*parse(*ids*)*;  
 int count = userMapper.deleteBatchIds*((*Collection*<*? extends Serializable*>)* parse*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count>=1*){* map.put*(*"code","100"*)*;  
 map.put*(*"ids",ids*)*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"ids",ids*)*;  
 *}* return map;  
 *}* @GetMapping*(*"/user/userAdd"*)* public String toUserAdd*(*Model model*){* List*<*Teacher*>* teachers = teacherController.teacherList*()*;  
 model.addAttribute*(*"teacherList",teachers*)*;  
 List*<*Student*>* studentList = studentController.getStudentList*()*;  
 model.addAttribute*(*"studentList",studentList*)*;  
 return "user/admin-add";  
 *}* @GetMapping*(*"/user/userAddStudent"*)* public String toUserAddStudent*(*Model model*){* List*<*Student*>* studentList = studentController.getStudentList*()*;  
 model.addAttribute*(*"studentList",studentList*)*;  
 return "user/admin-add-student";  
 *}* @GetMapping*(*"/user/userAddTeacher"*)* public String toUserAddTeacher*(*Model model*){* List*<*Teacher*>* teachers = teacherController.teacherList*()*;  
 model.addAttribute*(*"teacherList",teachers*)*;  
 return "user/admin-add-teacher";  
 *}* @ResponseBody  
 @PostMapping*(*"/user/userAdd"*)* public HashMap*<*String, Object*>* userAdd*(*User user*){  
 log*.info*(*user.toString*())*;  
  
 int count = userMapper.insert*(*user*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",user.getId*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",user.getId*())*;  
 *}* return map;  
 *}* @GetMapping*(*"/user/userUpdateStudent/{id}"*)* public String userUpdateStudent*(*@PathVariable*(*"id"*)* Integer id,Model model*){* User u = userMapper.selectById*(*id*)*;  
 model.addAttribute*(*"user",u*)*;  
 List*<*Student*>* studentList = studentController.getStudentList*()*;  
 model.addAttribute*(*"studentList",studentList*)*;  
 return "user/admin-update-student";  
 *}* @GetMapping*(*"/user/userUpdateTeacher/{id}"*)* public String userUpdateTeacher*(*@PathVariable*(*"id"*)* Integer id,Model model*){* User u = userMapper.selectById*(*id*)*;  
 model.addAttribute*(*"user",u*)*;  
 List*<*Teacher*>* teacherList = teacherController.teacherList*()*;  
 model.addAttribute*(*"teacherList",teacherList*)*;  
 return "user/admin-update-teacher";  
 *}* @GetMapping*(*"/user/userUpdate/{id}"*)* public String toUserUpdate*(*@PathVariable*(*"id"*)* Integer id,Model model*){* User u = userMapper.selectById*(*id*)*;  
 model.addAttribute*(*"user",u*)*;  
 return "user/admin-update";  
 *}* @GetMapping*(*"/user/userUpdatePassword/{id}"*)* public String toUserUpdatePassword*(*@PathVariable*(*"id"*)* Integer id,Model model*){* User u = userMapper.selectById*(*id*)*;  
 model.addAttribute*(*"user",u*)*;  
 return "user/change-password";  
 *}* @ResponseBody  
 @PostMapping*(*"/user/userUpdate"*)* public HashMap*<*String, Object*>* userUpdate*(*User user*){  
 log*.info*(*user.toString*())*;  
  
 int count = userMapper.updateById*(*user*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",user.getId*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",user.getId*())*;  
 *}* return map;  
 *}* @ResponseBody  
 @PostMapping*(*"/user/userUpdatePassword"*)* public HashMap*<*String, Object*>* userUpdatePassword*(*User user*){  
 log*.info*(*user.toString*())*;  
  
 int count = userMapper.update*(*user,new UpdateWrapper*<*User*>()* .set*(*"password",user.getPassword*())* .eq*(*"id",user.getId*()))*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",user.getId*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",user.getId*())*;  
 *}* return map;  
 *}  
  
}*

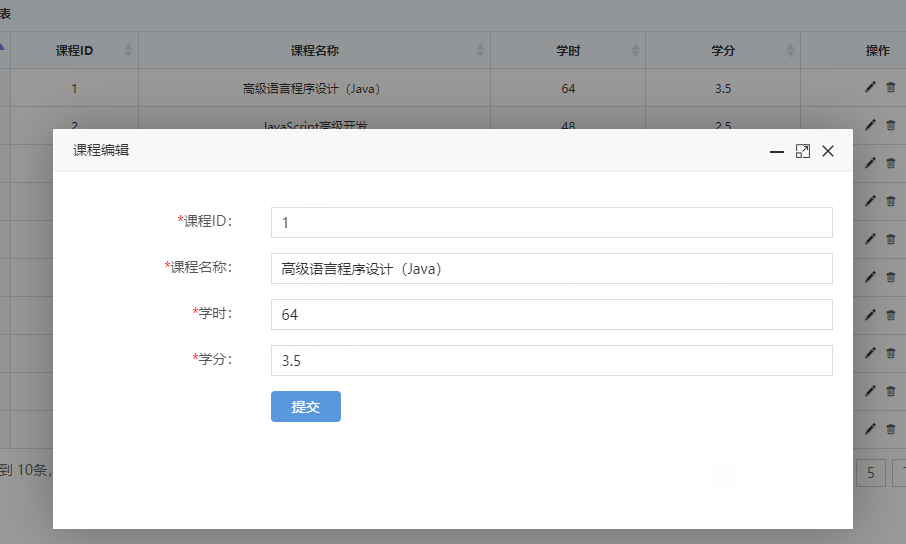
### 3.2.9 课程模块



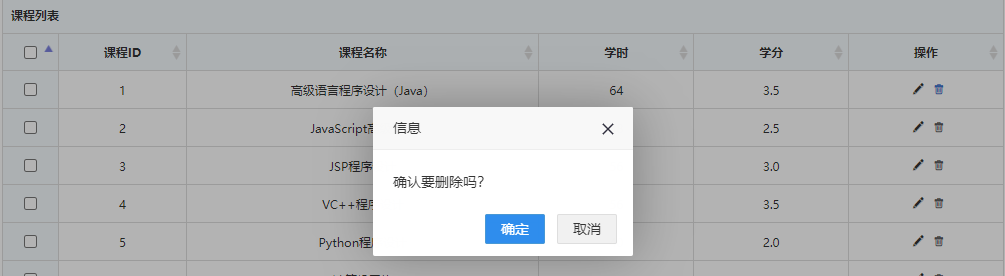
**图3.2.9.1 课程页面**



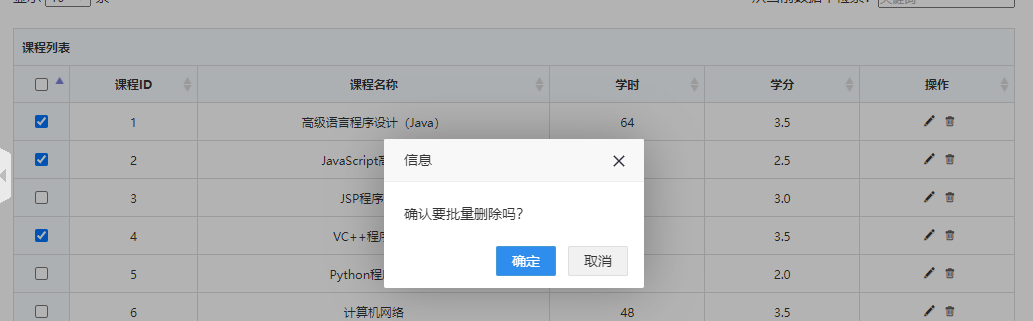
**图3.2.9.2 添加课程页面**



**图3.2.9.3 编辑课程页面**



**图3.2.9.4 删除课程页面**



**图3.2.9.5 批量删除课程页面**



**图3.2.9.6 筛选课程页面**

Course.java 课程实体类

@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@TableName*(*"course"*)*public class Course implements Serializable *{* @TableId  
 private Integer courseID; *//#课程ID* private String courseName; *//#课程名* private Integer hour; *//# 学时* private Double credit;*//# 学分  
}*

CourseMapper.java 课程mapper和DAO

@Repository  
public interface CourseMapper extends BaseMapper*<*Course*> {  
}*

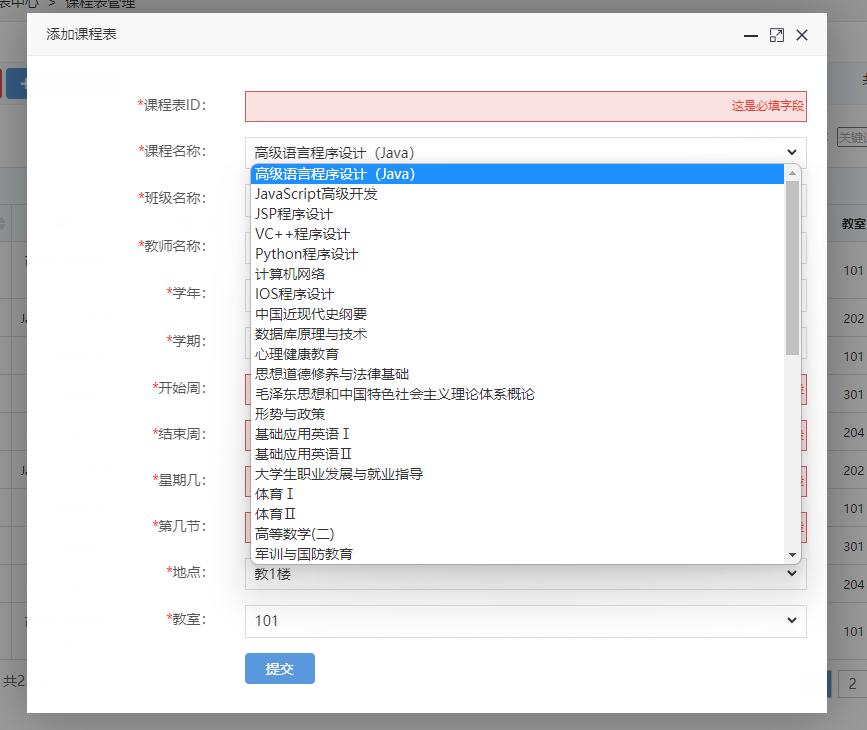
CourseController.java 课程控制器

@Slf4j  
@Controller  
public class CourseController *{* @Autowired  
 private CourseMapper courseMapper;  
  
 @ResponseBody  
 @GetMapping*(*"/course/isAvailableCourseNO"*)* public HashMap*<*String, Object*>* isAvailableCourseNO*(*Course course*){* Course room = courseMapper.selectById*(*course.getCourseID*())*;  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*room == null*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",course.getCourseID*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",course.getCourseID*())*;  
 *}* return map;  
 *}* @ResponseBody  
 @GetMapping*(*"/course/courseList"*)* public List*<*Course*>* toCourseList*(){* return courseMapper.selectList*(*null*)*;  
 *}* public Course getCourseById*(*Integer id*){* return courseMapper.selectById*(*id*)*;  
 *}* @GetMapping*(*"/course/courseListPage"*)* public String toCourseList*(*Model model*){* List*<*Course*>* courseList = courseMapper.selectList*(*null*)*;  
 model.addAttribute*(*"courseList",courseList*)*;  
 return "course/admin-list";  
 *}* @ResponseBody  
 @PostMapping*(*"/course/deleteCourseById"*)* public HashMap*<*String, Object*>* deleteCourseById*(*Course course*){  
 log*.info*(*course.toString*())*;  
  
 int count = courseMapper.deleteById*(*course.getCourseID*())*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",course.getCourseID*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",course.getCourseID*())*;  
 *}* return map;  
 *}* @ResponseBody  
 @PostMapping*(*"/course/deleteBatchIds"*)* public HashMap*<*String, Object*>* deleteBatchIds*(*String ids*){* System.*out*.println*(*ids*)*;  
  
 JSON parse = JSONUtil.*parse(*ids*)*;  
 int count = courseMapper.deleteBatchIds*((*Collection*<*? extends Serializable*>)* parse*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count>=1*){* map.put*(*"code","100"*)*;  
 map.put*(*"ids",ids*)*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"ids",ids*)*;  
 *}* return map;  
 *}* @GetMapping*(*"/course/courseAdd"*)* public String toUserAdd*(){* return "course/admin-add";  
 *}* @ResponseBody  
 @PostMapping*(*"/course/courseAdd"*)* public HashMap*<*String, Object*>* courseAdd*(*Course course*){  
 log*.info*(*course.toString*())*;  
  
 int count = courseMapper.insert*(*course*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",course.getCourseID*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",course.getCourseID*())*;  
 *}* return map;  
 *}* @GetMapping*(*"/course/courseUpdate/{id}"*)* public String toCourseUpdate*(*@PathVariable*(*"id"*)* Integer id, Model model*){* Course c = courseMapper.selectById*(*id*)*;  
 model.addAttribute*(*"course",c*)*;  
 return "course/admin-update";  
 *}* @ResponseBody  
 @PostMapping*(*"/course/courseUpdate"*)* public HashMap*<*String, Object*>* courseUpdate*(*Course course*){  
 log*.info*(*course.toString*())*;  
  
 int count = courseMapper.updateById*(*course*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",course.getCourseID*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",course.getCourseID*())*;  
 *}* return map;  
 *}  
}*

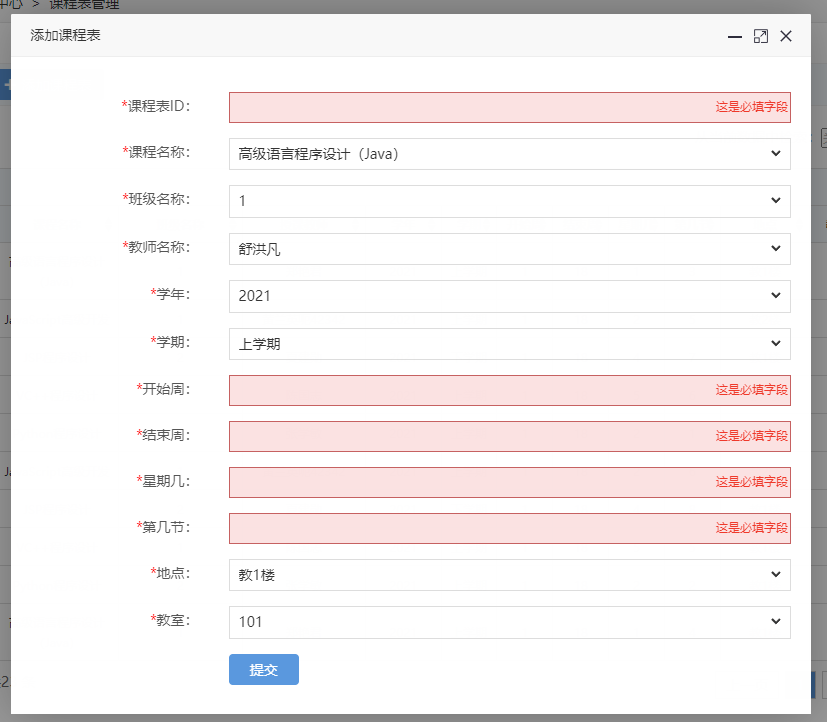
### 3.2.10 课程表模块



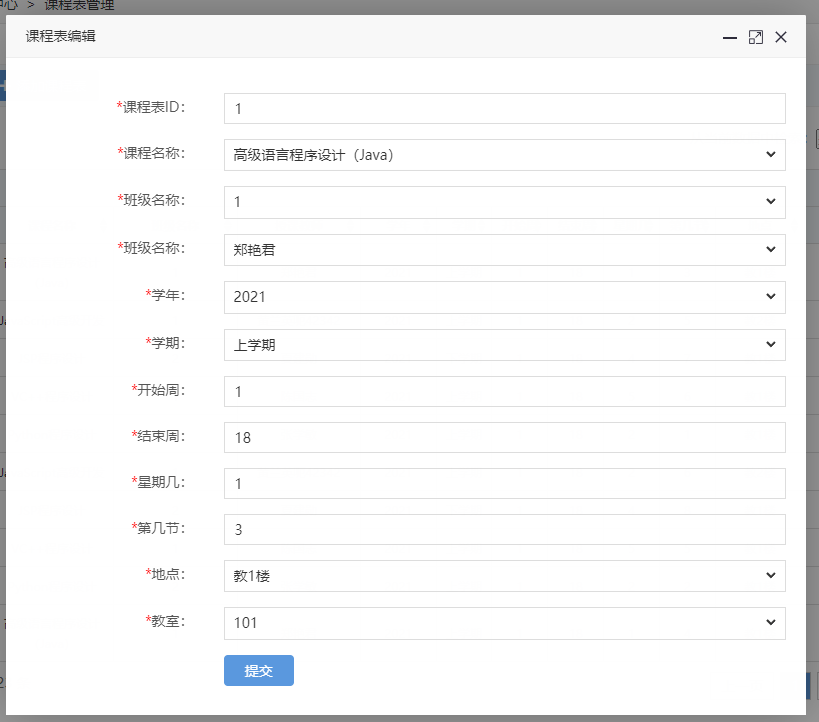
**图3.2.10.1课程表列表页面**



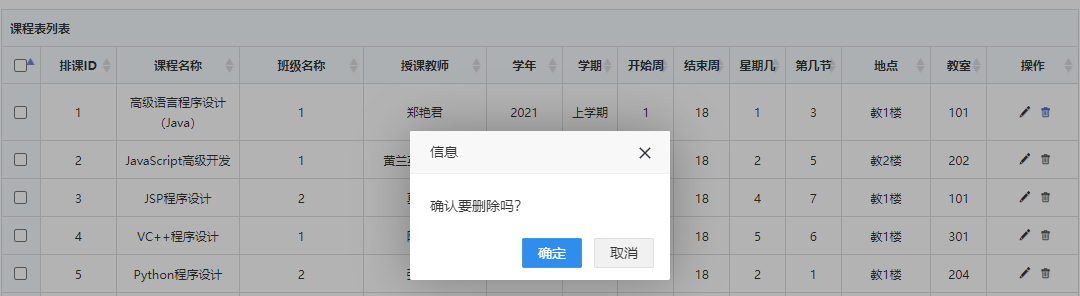
**图3.2.10.2添加课程表页面1**



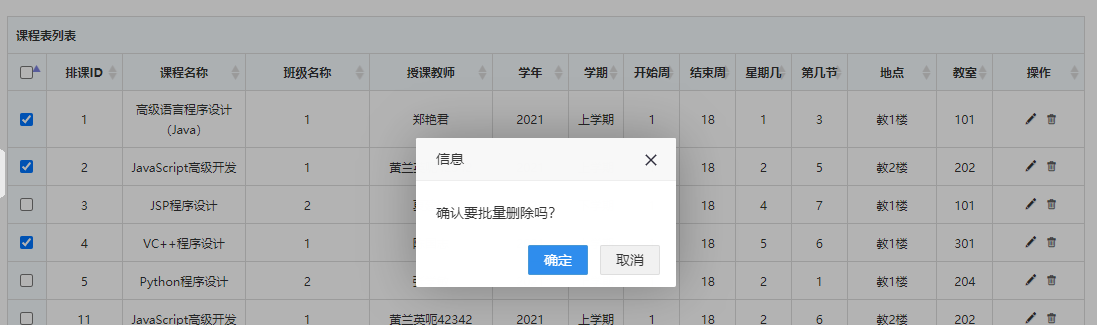
**图3.2.10.3添加课程表页面2**



**图3.2.10.4修改课程表页面**



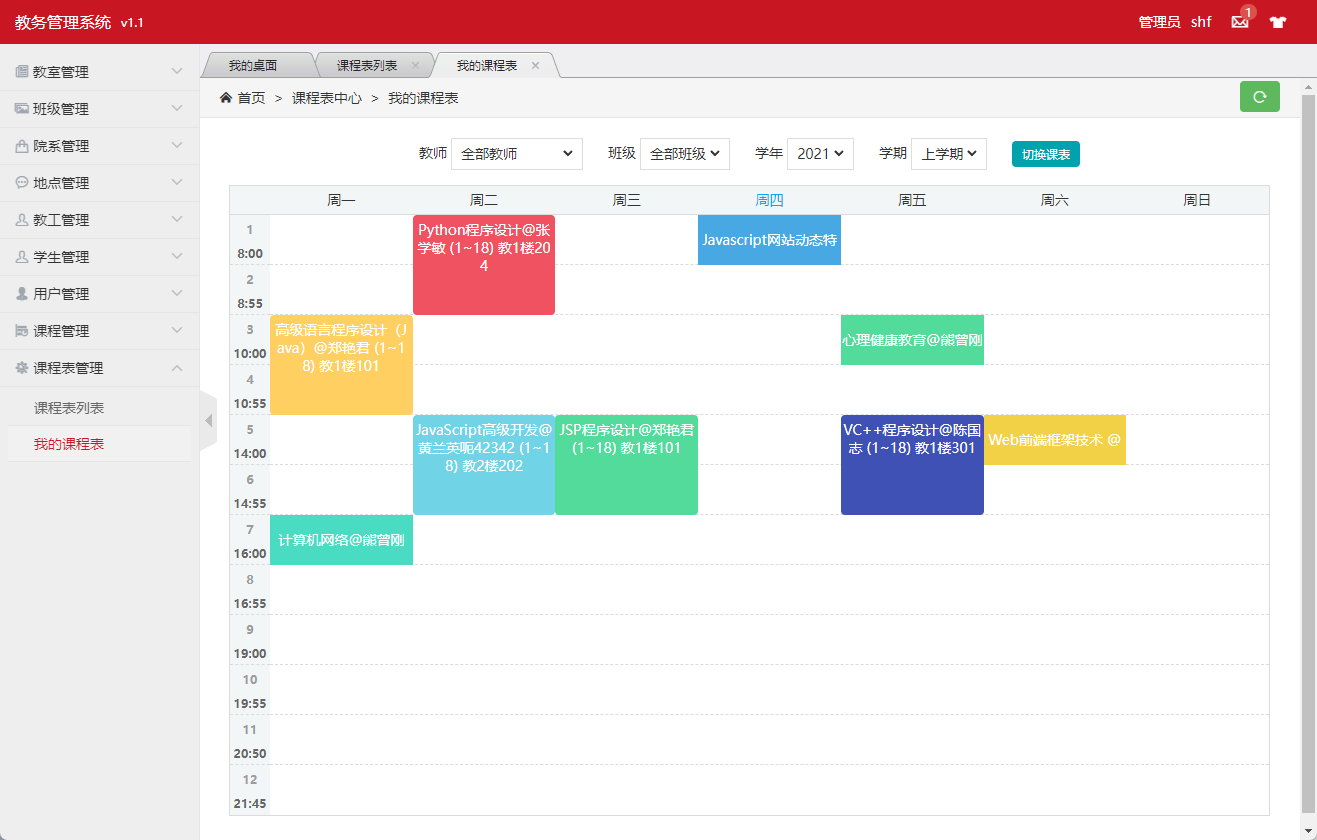
**图3.2.10.5删除课程表页面**



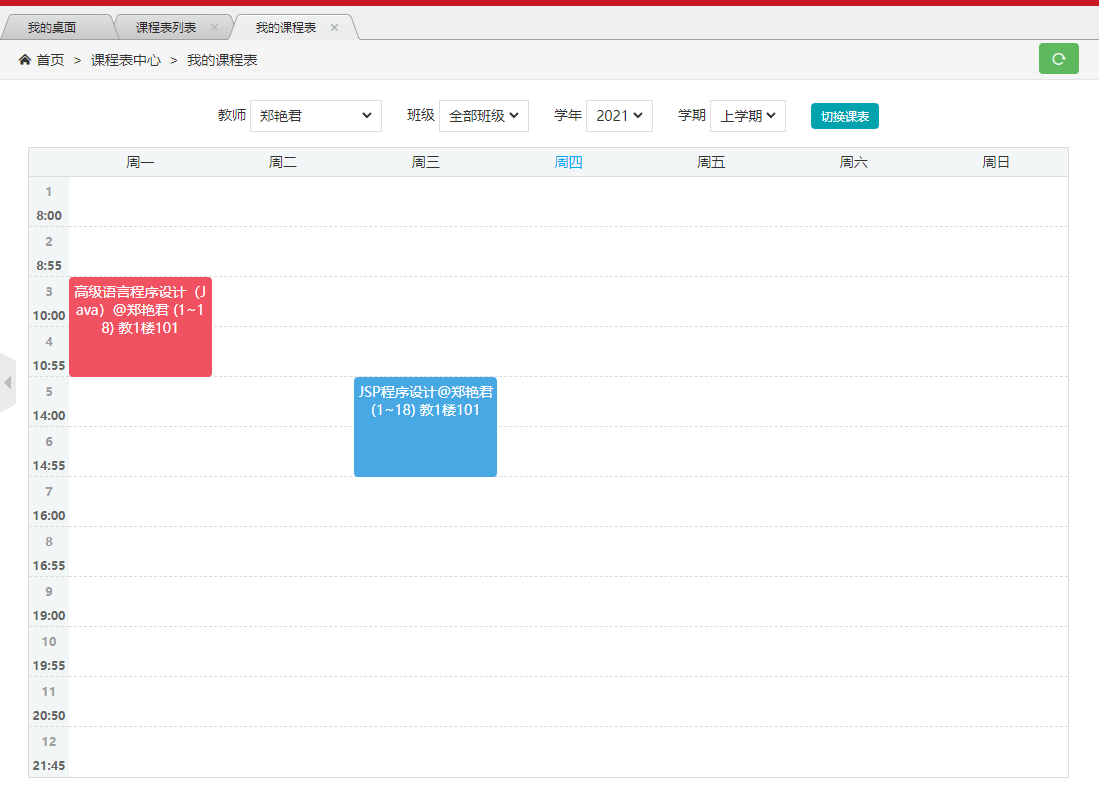
**图3.2.10.5批量删除课程表页面**



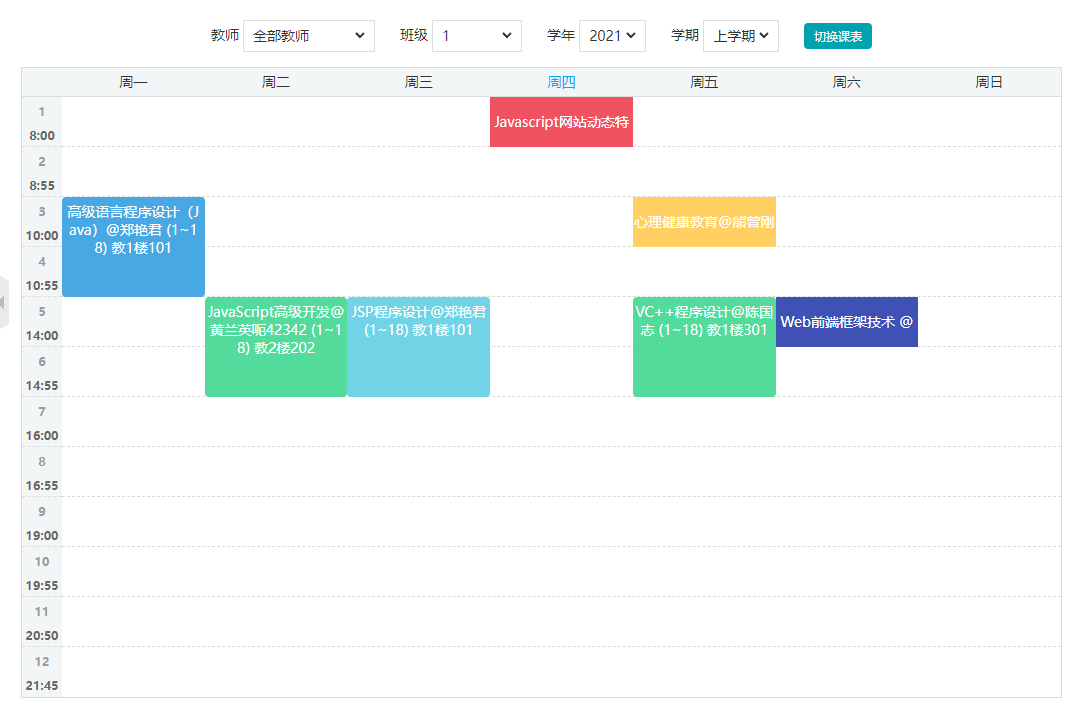
**图3.2.10.6筛选课程表页面**



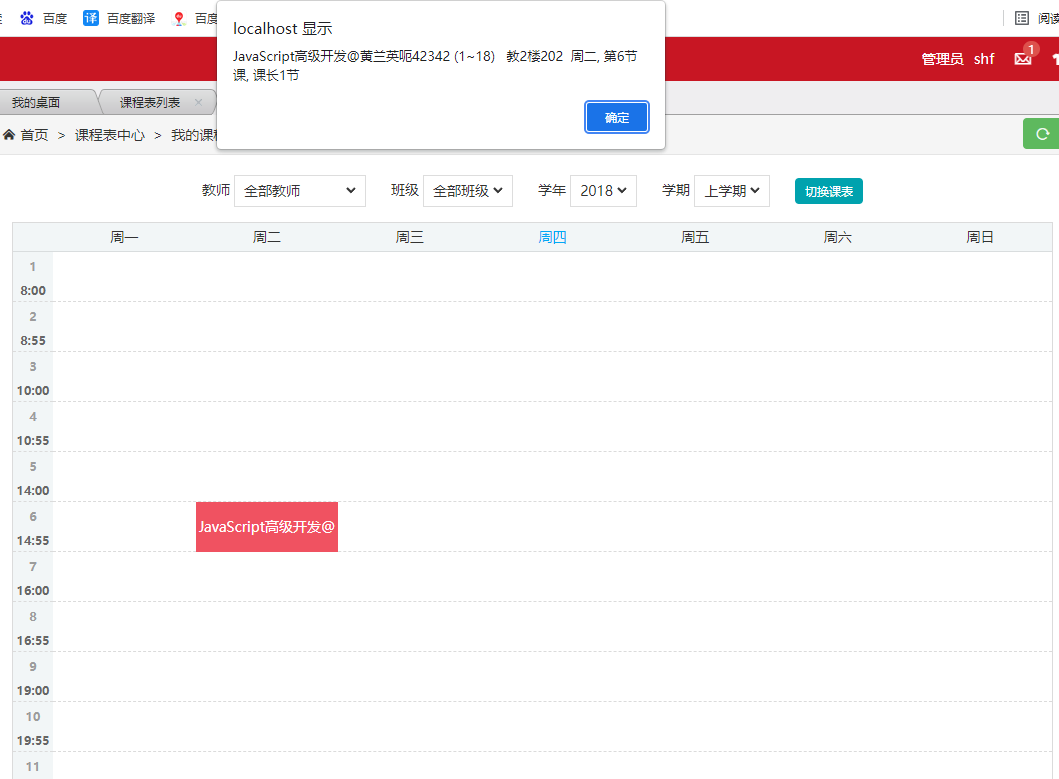
**图3.2.10.7 我的课程表页面**



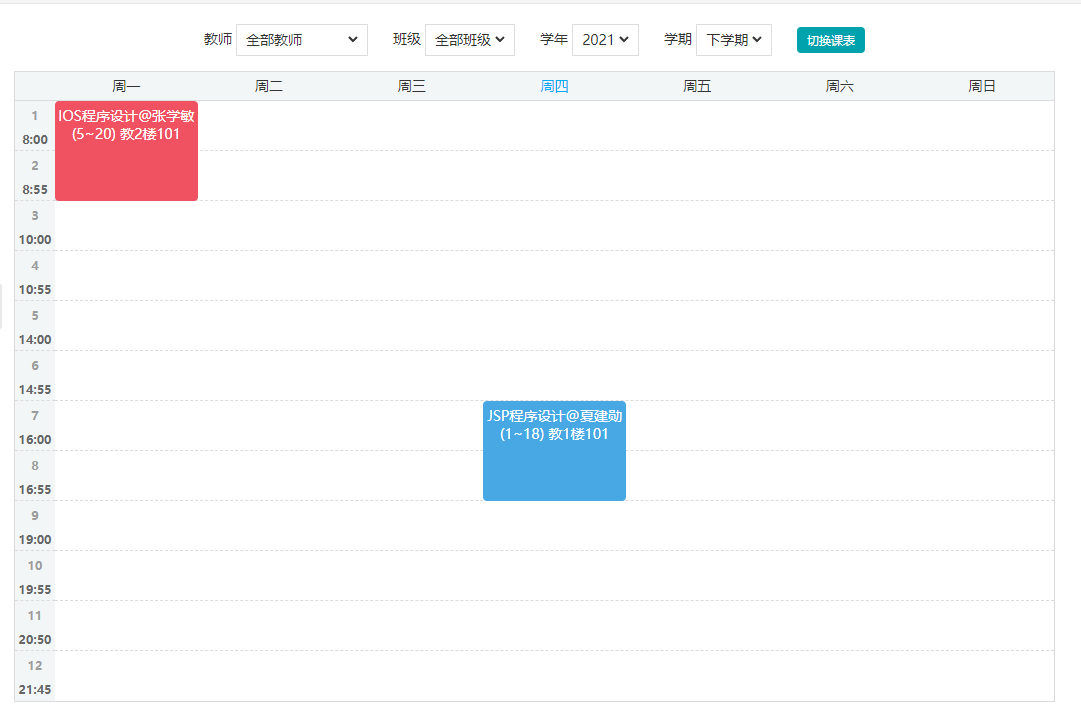
**图3.2.10.8 筛选课程表页面1**



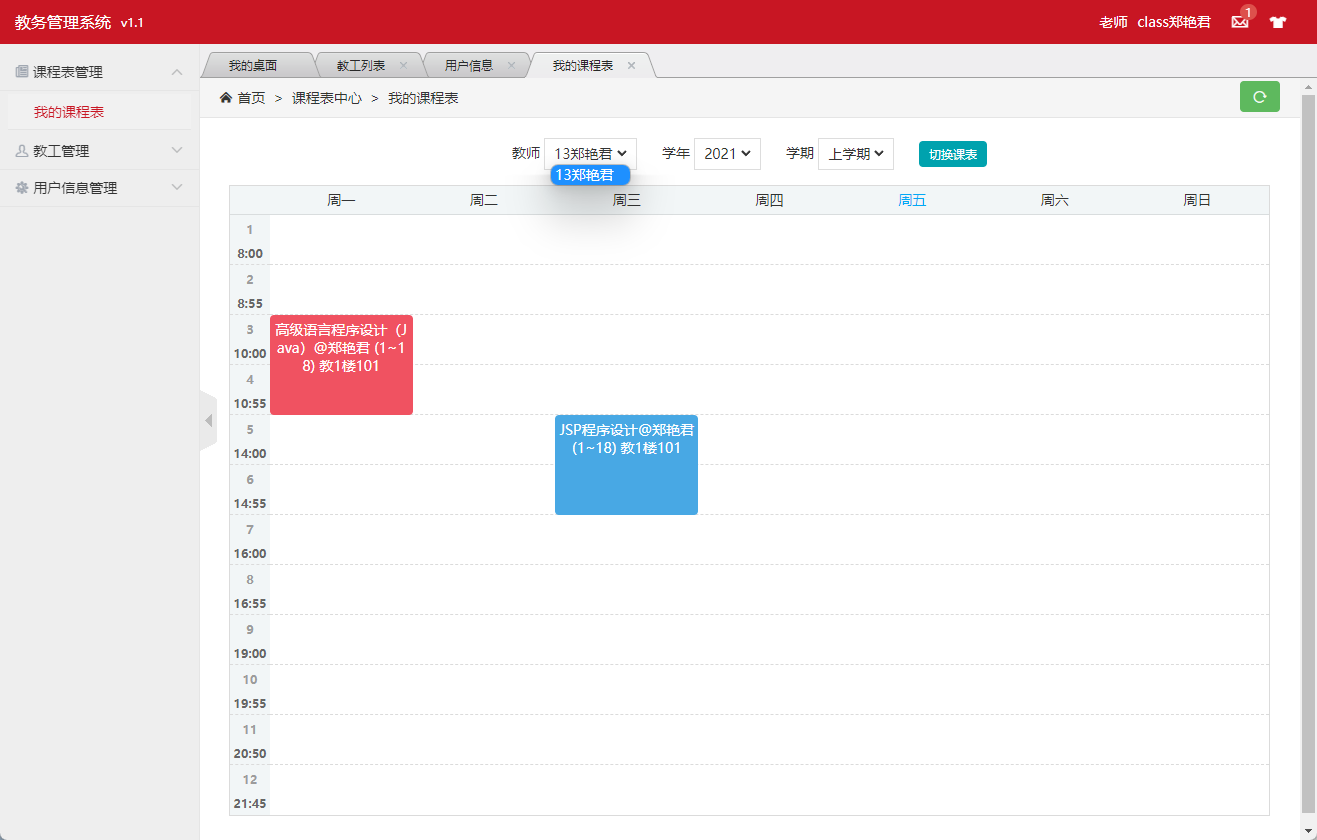
**图3.2.10.9 筛选课程表页面2**



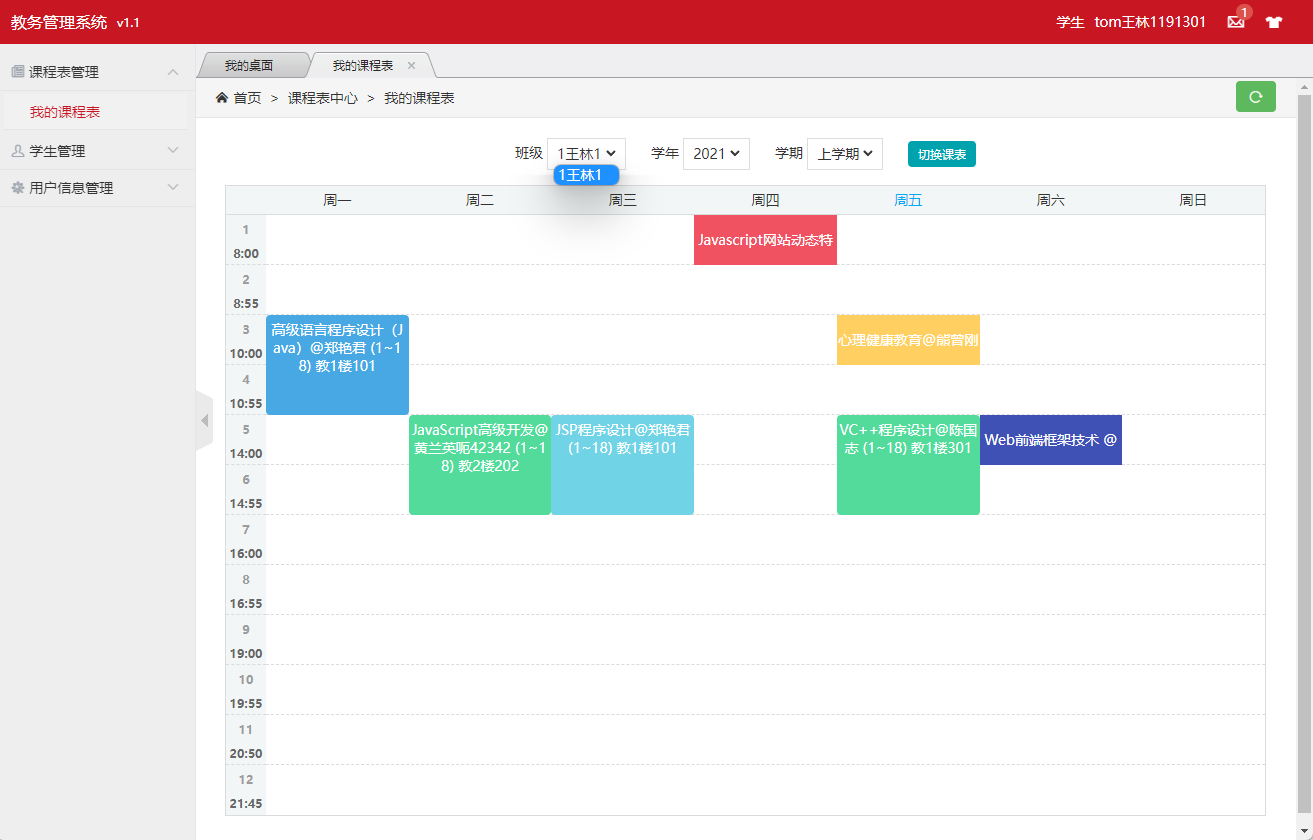
**图3.2.10.10 筛选课程表页面3**



**图3.2.10.11 筛选课程表页面4**



**图3.2.10.12 教工用户课程表页面**



**图3.2.10.12 学生用户课程表页面**

Curriculum.java 课程表实体类

@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@TableName*(*"curriculum"*)*public class Curriculum implements Serializable *{ // 课程* @TableId  
 private Integer ID; *// 排课ID* private Integer courseNO; *// 课程号* @TableField*(*exist = false*)* private String courseName; *//#课程名* private Integer classNO; *// 班级号* private Integer teacherID; *// 授课教师编号* @TableField*(*exist = false*)* private String name; *// 姓名* private Integer schoolYear; *// 学年* private Integer term; *// 学期* private Integer startWeek; *// 第几周开始* private Integer endWeek; *// 第几周结束* private Integer weekday; *// 星期几* private Integer section; *// 第几节* private Integer locationNo; *// 地点* @TableField*(*exist = false*)* private String locationName; *// 地点名称* private Integer classRoomNO; *// 教室  
}*

CurriculumMapper.java 课表Mapper和DAO

@Repository  
public interface CurriculumMapper extends BaseMapper*<*Curriculum*> {  
}*

CurriculumController.java 课程表控制器

@Slf4j  
@Controller  
public class CurriculumController *{* @Autowired  
 private CurriculumMapper curriculumMapper;  
  
 @Autowired  
 private CourseController courseController;  
  
 @Autowired  
 private ClassController classController;  
  
 @Autowired  
 private TeacherController teacherController;  
  
 @Autowired  
 private LocationController locationController;  
  
 @Autowired  
 private ClassRoomController classRoomController;  
  
 @ResponseBody  
 @GetMapping*(*"/curriculum/curriculumList"*)* public List*<*Curriculum*>* toCurriculumList*(){* return curriculumMapper.selectList*(*null*)*;  
 *}* public Curriculum getCurriculumById*(*Integer id*){* return curriculumMapper.selectById*(*id*)*;  
 *}* @GetMapping*(*"/curriculum/curriculumListPage"*)* public String toCurriculumList*(*Model model*){* List*<*Curriculum*>* curriculumList = curriculumMapper.selectList*(*null*)*;  
 for *(*Curriculum curriculum : curriculumList*) {* curriculum.setCourseName*(*courseController.getCourseById*(*curriculum.getCourseNO*())*.getCourseName*())*;  
 curriculum.setName*(*teacherController.getTeacherById*(*curriculum.getTeacherID*())*.getName*())*;  
 curriculum.setLocationName*(*locationController.getLocationById*(*curriculum.getLocationNo*())*.getLocationName*())*;  
 *}* model.addAttribute*(*"curriculumList",curriculumList*)*;  
 return "curriculum/admin-list";  
 *}* @ResponseBody  
 @PostMapping*(*"/curriculum/deleteCurriculumById"*)* public HashMap*<*String, Object*>* deleteCurriculumById*(*Curriculum curriculum*){  
 log*.info*(*curriculum.toString*())*;  
  
 int count = curriculumMapper.deleteById*(*curriculum.getID*())*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",curriculum.getID*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",curriculum.getID*())*;  
 *}* return map;  
 *}* @ResponseBody  
 @PostMapping*(*"/curriculum/deleteBatchIds"*)* public HashMap*<*String, Object*>* deleteBatchIds*(*String ids*){* System.*out*.println*(*ids*)*;  
  
 JSON parse = JSONUtil.*parse(*ids*)*;  
 int count = curriculumMapper.deleteBatchIds*((*Collection*<*? extends Serializable*>)* parse*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count>=1*){* map.put*(*"code","100"*)*;  
 map.put*(*"ids",ids*)*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"ids",ids*)*;  
 *}* return map;  
 *}* @GetMapping*(*"/curriculum/curriculumAdd"*)* public String toUserAdd*(*Model model*){* List*<*Course*>* courses = courseController.toCourseList*()*;  
 model.addAttribute*(*"courseList",courses*)*;  
 List*<*Class*>* classes = classController.toClassList*()*;  
 model.addAttribute*(*"classList",classes*)*;  
 List*<*Teacher*>* teachers = teacherController.teacherList*()*;  
 model.addAttribute*(*"teacherList",teachers*)*;  
 List*<*Location*>* locations = locationController.locationList*()*;  
 model.addAttribute*(*"locationList",locations*)*;  
 List*<*ClassRoom*>* classRooms = classRoomController.toClassRoomList*()*;  
 model.addAttribute*(*"classRoomList",classRooms*)*;  
 return "curriculum/admin-add";  
 *}* @ResponseBody  
 @PostMapping*(*"/curriculum/curriculumAdd"*)* public HashMap*<*String, Object*>* curriculumAdd*(*Curriculum curriculum*){  
 log*.info*(*curriculum.toString*())*;  
  
 int count = curriculumMapper.insert*(*curriculum*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",curriculum.getID*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",curriculum.getID*())*;  
 *}* return map;  
 *}* @GetMapping*(*"/curriculum/curriculumUpdate/{id}"*)* public String toCurriculumUpdate*(*@PathVariable*(*"id"*)* Integer id, Model model*){* List*<*Course*>* courses = courseController.toCourseList*()*;  
 model.addAttribute*(*"courseList",courses*)*;  
 List*<*Class*>* classes = classController.toClassList*()*;  
 model.addAttribute*(*"classList",classes*)*;  
 List*<*Teacher*>* teachers = teacherController.teacherList*()*;  
 model.addAttribute*(*"teacherList",teachers*)*;  
 List*<*Location*>* locations = locationController.locationList*()*;  
 model.addAttribute*(*"locationList",locations*)*;  
 List*<*ClassRoom*>* classRooms = classRoomController.toClassRoomList*()*;  
 model.addAttribute*(*"classRoomList",classRooms*)*;  
 Curriculum t = curriculumMapper.selectById*(*id*)*;  
 model.addAttribute*(*"curriculum",t*)*;  
 return "curriculum/admin-update";  
 *}* @ResponseBody  
 @PostMapping*(*"/curriculum/curriculumUpdate"*)* public HashMap*<*String, Object*>* curriculumUpdate*(*Curriculum curriculum*){  
 log*.info*(*curriculum.toString*())*;  
  
 int count = curriculumMapper.updateById*(*curriculum*)*;  
 *log*.info*(*String.*valueOf(*count*))*;  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",curriculum.getID*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",curriculum.getID*())*;  
 *}* return map;  
 *}* @ResponseBody  
 @GetMapping*(*"/curriculum/isAvailableID"*)* public HashMap*<*String, Object*>* isAvailableID*(*Curriculum curriculum*){* Curriculum byId = curriculumMapper.selectById*(*curriculum.getID*())*;  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*byId == null*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",curriculum.getID*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",curriculum.getID*())*;  
 *}* return map;  
 *}* @GetMapping*(*"/curriculum/myCurriculumPage"*)* public String myCurriculumPage*(*Model model*){* List*<*Curriculum*>* distinct\_teacherID = curriculumMapper.selectList*(* new QueryWrapper*<*Curriculum*>()* .select*(*"distinct teacherID"*)* .orderByAsc*(*"teacherID"*)  
 )*;  
 for *(*Curriculum curriculum : distinct\_teacherID*) {* curriculum.setName*(*teacherController.getTeacherById*(*curriculum.getTeacherID*())*.getName*())*;  
 *}* model.addAttribute*(*"distinctTeacherID",distinct\_teacherID*)*;  
 List*<*Curriculum*>* distinct\_schoolYear = curriculumMapper.selectList*(* new QueryWrapper*<*Curriculum*>()* .select*(*"distinct schoolYear"*)* .orderByAsc*(*"schoolYear"*)  
 )*;  
 model.addAttribute*(*"distinctSchoolYear",distinct\_schoolYear*)*;  
 List*<*Curriculum*>* distinct\_classNO = curriculumMapper.selectList*(* new QueryWrapper*<*Curriculum*>()* .select*(*"distinct classNO"*)* .orderByAsc*(*"classNO"*)  
 )*;  
 model.addAttribute*(*"distinctClassNO",distinct\_classNO*)*;  
 return "curriculum/curriculum";  
 *}* @ResponseBody  
 @GetMapping*(*"/curriculum/myCurriculum"*)* public HashMap*<*String, Object*>* myCurriculum*(*Curriculum curriculum*){  
 log*.info*(*String.*valueOf(*curriculum*))*;  
 List*<*Curriculum*>* curriculumList = null;  
 if *(*curriculum.getClassNO*()*==404 && curriculum.getTeacherID*()*==404*){* curriculumList = curriculumMapper.selectList*(* new QueryWrapper*<*Curriculum*>()* .eq*(*"schoolYear",curriculum.getSchoolYear*())* .eq*(*"term",curriculum.getTerm*())  
 )*;  
 *}* else if *(*curriculum.getClassNO*()*==404*){* curriculumList = curriculumMapper.selectList*(* new QueryWrapper*<*Curriculum*>()* .eq*(*"teacherID",curriculum.getTeacherID*())* .eq*(*"schoolYear",curriculum.getSchoolYear*())* .eq*(*"term",curriculum.getTerm*())  
 )*;  
 *}* else if*(*curriculum.getTeacherID*()*==404*){* curriculumList = curriculumMapper.selectList*(* new QueryWrapper*<*Curriculum*>()* .eq*(*"schoolYear",curriculum.getSchoolYear*())* .eq*(*"term",curriculum.getTerm*())* .eq*(*"classNO",curriculum.getClassNO*())  
 )*;  
 *}* else *{* curriculumList = curriculumMapper.selectList*(* new QueryWrapper*<*Curriculum*>()* .eq*(*"teacherID",curriculum.getTeacherID*())* .eq*(*"schoolYear",curriculum.getSchoolYear*())* .eq*(*"term",curriculum.getTerm*())* .eq*(*"classNO",curriculum.getClassNO*())  
 )*;  
 *}* HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*curriculumList.size*()* > 0*){* map.put*(*"code","100"*)*;  
 for *(*Curriculum c : curriculumList*) {* c.setCourseName*(*courseController.getCourseById*(*c.getCourseNO*())*.getCourseName*())*;  
 c.setName*(*teacherController.getTeacherById*(*c.getTeacherID*())*.getName*())*;  
 c.setLocationName*(*locationController.getLocationById*(*c.getLocationNo*())*.getLocationName*())*;  
 *}* map.put*(*"curriculumList",curriculumList*)*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",curriculum.getID*())*;  
 *}* return map;  
 *}  
}*

curriculum.html 课程表部分静态资源

*<*div class="page-container"*>  
 <*div class="text-c"*>  
 <*span th:if="*${*session.USER.type==2*}*"*>  
 <*label for="teacherID"*>*教师*</*label*>  
 <*span class="select-box inline"*>  
 <*select id="teacherID" class="select"*>  
 <*option value="404"*>*全部教师*</*option*>  
 <*option th:each="t: ${distinctTeacherID}" th:value="*${*t.teacherID*}*" th:text="*${*t.name*}*"*>*全部分类*</*option*>  
 </*select*>  
 </*span*>  
 <*span*>*&nbsp;&nbsp;&nbsp;&nbsp;*</*span*>  
  
 <*label for="classNO"*>*班级*</*label*>  
 <*span class="select-box inline"*>  
 <*select id="classNO" class="select"*>  
 <*option value="404"*>*全部班级*</*option*>  
 <*option th:each="c: ${distinctClassNO}" th:selected="${#dates.format(new java.util.Date().getTime(), 'yyyy')}==${c.classNO}" th:value="*${*c.classNO*}*" th:text="*${*c.classNO*}*"*>*全部分类*</*option*>  
 </*select*>  
 </*span*>  
 <*span*>*&nbsp;&nbsp;&nbsp;&nbsp;*</*span*>  
 </*span*>  
 <*span th:if="*${*session.USER.type==0*}*"*>  
 <*label for="classNO"*>*班级*</*label*>  
 <*span class="select-box inline"*>  
 <*select id="classNO" class="select"*>  
 <*option selected th:value="*${*session.USER.classNO*}*" th:text="*${*session.USER.classNO+session.USER.contrastStudentName*}*"*>*全部分类*</*option*>  
 </*select*>  
 </*span*>  
 <*span*>*&nbsp;&nbsp;&nbsp;&nbsp;*</*span*>  
 </*span*>  
  
 <*span th:if="*${*session.USER.type==1*}*"*>  
 <*label for="teacherID"*>*教师*</*label*>  
 <*span class="select-box inline"*>  
 <*select id="teacherID" class="select"*>  
 <*option checked th:value="*${*session.USER.contrastTeacherID*}*" th:text="*${*session.USER.contrastTeacherID+session.USER.contrastTeacherName*}*"*>*全部分类*</*option*>  
 </*select*>  
 </*span*>  
 <*span*>*&nbsp;&nbsp;&nbsp;&nbsp;*</*span*>  
 </*span*>  
  
 <*label for="schoolYear"*>*学年*</*label*>  
 <*span class="select-box inline"*>  
 <*select id="schoolYear" class="select"*>  
 <*option th:each="y: ${distinctSchoolYear}" th:selected="${#dates.format(new java.util.Date().getTime(), 'yyyy')}==${y.schoolYear}" th:value="*${*y.schoolYear*}*" th:text="*${*y.schoolYear*}*"*>*全部分类*</*option*>  
 </*select*>  
 </*span*>  
 <*span*>*&nbsp;&nbsp;&nbsp;&nbsp;*</*span*>  
 <*label for="term"*>*学期*</*label*>  
 <*span class="select-box inline"*>  
 <*select id="term" class="select"*>  
 <*option value="0"*>*上学期*</*option*>  
 <*option value="1"*>*下学期*</*option*>  
 </*select*>  
 </*span*>  
 <*span*>*&nbsp;&nbsp;&nbsp;&nbsp;*</*span*>  
 <*button onclick="*onChange()*"  
 style="background-color: #00a2ae; color: #fff; padding: 5px 10px; border-radius: 4px;border: none"*>* 切换课表  
 *</*button*>  
 </*div*>  
 <*div id="coursesTable"*></*div*>  
</*div*>  
  
<*script type="text/javascript" th:inline="javascript"*>  
 // 新建二维数组* let arr = new Array*(*7*)* for*(*let i = 0;i < arr.length;++i*){* arr*[*i*]* = new Array*(*12*)  
 }* for*(*let j = 0; j <arr.length;++j*){* for *(*let i = 0; i < arr*[*j*]*.length; i++*) {* arr*[*j*][*i*]*=""  
 *}  
 }* var week = window.innerWidth > 360 ? *[*'周一', '周二', '周三', '周四', '周五','周六','周日'*]* :  
 *[*'一', '二', '三', '四', '五','六','日'*]*;  
 var day = new Date*()*.getDay*()*;  
 var courseType = *[  
 [{*index: '1', name: '8:00'*}*, 1*]*,  
 *[{*index: '2', name: '8:55'*}*, 1*]*,  
 *[{*index: '3', name: '10:00'*}*, 1*]*,  
 *[{*index: '4', name: '10:55'*}*, 1*]*,  
 *[{*index: '5', name: '14:00'*}*, 1*]*,  
 *[{*index: '6', name: '14:55'*}*, 1*]*,  
 *[{*index: '7', name: '16:00'*}*, 1*]*,  
 *[{*index: '8', name: '16:55'*}*, 1*]*,  
 *[{*index: '9', name: '19:00'*}*, 1*]*,  
 *[{*index: '10', name: '19:55'*}*, 1*]*,  
 *[{*index: '11', name: '20:50'*}*, 1*]*,  
 *[{*index: '12', name: '21:45'*}*, 1*]  
 ]*;  
  
 *// 实例化(初始化课表)* var Timetable = new Timetables*({* el: '#coursesTable',  
 timetables: arr,  
 week: week,  
 timetableType: courseType,  
 highlightWeek: day,  
 gridOnClick: function *(*e*) {  
 alert(*e.name + ' ' + e.week + ', 第' + e.index + '节课, 课长' + e.length + '节'*)*;  
 console.log*(*e*)*;  
 *}*,  
 styles: *{* Gheight: 50  
 *}  
 })*;  
  
 *//切换课表* function *onChange() {* var ctxPath = [[@*{*/}]];  
 let schoolYear=$*(*"#schoolYear*")*.val*()* let term=$*(*"#term*")*.val*()* let classNO=$*(*"#classNO*")*.val*()*==null?"404":$*(*"#classNO*")*.val*()* let teacherID=$*(*"#teacherID*")*.val*()*==null?"404":$*(*"#teacherID*")*.val*()* $.get*(*ctxPath+"curriculum/myCurriculum?schoolYear="+schoolYear+"&term="+term+"&classNO="+classNO+"&teacherID="+teacherID,function*(*data,status*){  
 // 新建二维数组* var arr = new Array*(*7*)* for*(*let i = 0;i < arr.length;++i*){* arr*[*i*]* = new Array*(*12*)  
 }* for*(*let j = 0; j <arr.length;++j*){* for *(*let i = 0; i < arr*[*j*]*.length; i++*) {* arr*[*j*][*i*]*=""  
 *}  
 }* if *(*data.code ==100*) {* console.log*(*"数据: " + data + "\n状态: " + status*)*;  
 var curriculumList = data.curriculumList  
 *// 新建二维数组* arr = new Array*(*7*)* for*(*let i = 0;i < arr.length;++i*){* arr*[*i*]* = new Array*(*12*)  
 }* for*(*let j = 0; j <arr.length;++j*){* for *(*let i = 0; i < arr*[*j*]*.length; i++*) {* arr*[*j*][*i*]*=""  
 *}  
 }* for *(*let i = 0; i < curriculumList.length; i++*) {* let weekday=curriculumList*[*i*]*.weekday  
 let section=curriculumList*[*i*]*.section  
 arr*[*weekday-1*][*section-1*]*=curriculumList*[*i*]*.courseName+"@"+curriculumList*[*i*]*.name+" ("+  
 curriculumList*[*i*]*.startWeek+"~"+curriculumList*[*i*]*.endWeek+") "+  
 curriculumList*[*i*]*.locationName+curriculumList*[*i*]*.classRoomNO  
 *}* console.log*(*arr*)  
 }* Timetable.setOption*({* timetables: arr,  
 week: week,  
 timetableType: courseType,  
 highlightWeek: day,  
 gridOnClick: function *(*e*) {  
 alert(*e.name + ' ' + e.week + ', 第' + e.index + '节课, 课长' + e.length + '节'*)*;  
 console.log*(*e*)*;  
 *}*,  
 styles: *{* Gheight: 50  
 *}  
 })*;  
 *})*;  
 *}  
</*script*>  
</*body*>  
</*html*>*

### 3.2.11 拦截器模块

AdminInterceptor.java 未登录用户跳回登录页面

public class AdminInterceptor implements HandlerInterceptor *{  
 /\*\*  
 \* 在请求处理之前进行调用（Controller方法调用之前）  
 \*/* @Override  
 public boolean preHandle*(*HttpServletRequest request, HttpServletResponse response, Object handler*) {  
// System.out.println("执行了TestInterceptor的preHandle方法");* try *{  
 //统一拦截（查询当前session是否存在user）(这里user会在每次登陆成功后，写入session)* User user=*(*User*)*request.getSession*()*.getAttribute*(*"USER"*)*;  
 if*(*user!=null*){* return true;  
 *}* response.sendRedirect*(*request.getContextPath*()*+"/login"*)*;  
 *}* catch *(*IOException e*) {* e.printStackTrace*()*;  
 *}* return false;*//如果设置为false时，被请求时，拦截器执行到此处将不会继续操作  
 //如果设置为true时，请求将会继续执行后面的操作  
 }  
  
 /\*\*  
 \* 请求处理之后进行调用，但是在视图被渲染之前（Controller方法调用之后）  
 \*/* @Override  
 public void postHandle*(*HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView*) {  
// System.out.println("执行了TestInterceptor的postHandle方法");  
 }  
  
 /\*\*  
 \* 在整个请求结束之后被调用，也就是在DispatcherServlet 渲染了对应的视图之后执行（主要是用于进行资源清理工作）  
 \*/* @Override  
 public void afterCompletion*(*HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex*) {  
// System.out.println("执行了TestInterceptor的afterCompletion方法");  
 }  
}*

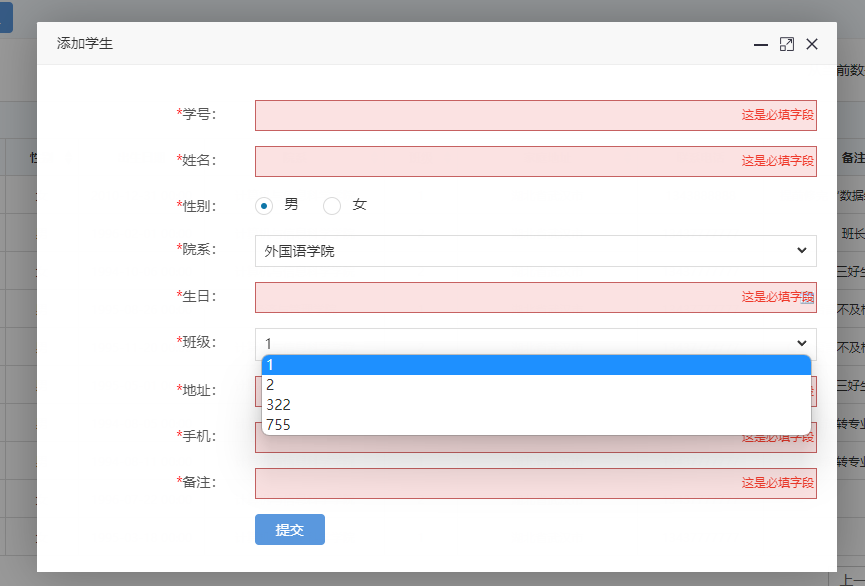
LoginConfig.java 让拦截器不拦截静态资源、登录页面和验证码

@Configuration  
public class LoginConfig implements WebMvcConfigurer *{* @Override  
 public void addInterceptors*(*InterceptorRegistry registry*) {  
 //注册TestInterceptor拦截器* InterceptorRegistration registration = registry.addInterceptor*(*new AdminInterceptor*())*;  
 registration.addPathPatterns*(*"/\*\*"*)*; *//所有路径都被拦截* registration.excludePathPatterns*( //添加不拦截路径* "/login", *//登录* "/captcha", *//验证码* "/\*\*/\*.html", *//html静态资源* "/\*\*/\*.js", *//js静态资源* "/\*\*/\*.css", *//css静态资源* "/\*\*/\*.ico",  
 "/\*\*/\*.jpg",  
 "/\*\*/\*.png",  
 "/\*\*/\*.woff",  
 "/\*\*/\*.ttf"  
 *)*;  
 *}  
}*

### 3.2.12 学生模块



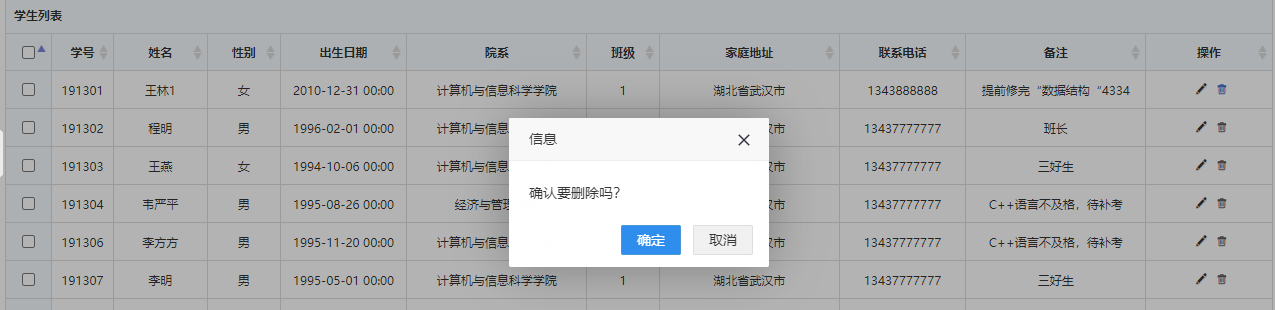
**图3.2.12.1 学生页面**



**图3.2.12.2 添加学生页面**



**图3.2.12.3 修改学生页面**



**图3.2.12.4 删除学生页面**



**图3.2.12.5 批量删除学生页面**



**图3.2.12.6 条件查询学生页面**



**图3.2.12.7 学生用户学生页面**



**图3.2.12.8 学生用户修改学生信息页面**

Student.java 学生实体类

@Data  
@NoArgsConstructor  
@AllArgsConstructor  
@TableName*(*"student"*)*public class Student implements Serializable *{* @TableId  
 private Integer studentID; *// 学号* private String name; *// 姓名* private Integer sex; *// 性别* @DateTimeFormat*(*pattern = "yyyy-MM-dd HH:mm"*)* private Date birthday; *// 出生日期* private Integer facultyNO; *// 院系* @TableField*(*exist = false*)* private String facultyName; *//#院系名称* private Integer classNO; *// 班级* private String address; *// 家庭地址* private String phone; *// 联系电话* private String remark; *// 备注  
}*

StudentMapper.java 学生mapper和DAO

@Repository  
public interface StudentMapper extends BaseMapper*<*Student*> {  
}*

StudentController.java 学生控制器

@Slf4j  
@Controller  
public class StudentController *{* @Autowired  
 private StudentMapper StudentMapper;  
  
 @Autowired  
 private FacultyController facultyController;  
  
 @Autowired  
 private ClassController classController;  
  
 public List*<*Student*>* getStudentList*(){* return StudentMapper.selectList*(*null*)*;  
 *}* public Student getStudentById*(*Integer id*){* return StudentMapper.selectById*(*id*)*;  
 *}* @ResponseBody  
 @GetMapping*(*"/student/isAvailableStudentNO"*)* public HashMap*<*String, Object*>* isAvailableStudentNO*(*Student student*){* Student room = StudentMapper.selectById*(*student.getStudentID*())*;  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*room == null*){* map.put*(*"code","100"*)*;  
 map.put*(*"id",student.getStudentID*())*;  
 *}* else *{* map.put*(*"code","500"*)*;  
 map.put*(*"id",student.getStudentID*())*;  
 *}* return map;  
 *}* @GetMapping("/student/studentPage")  
 public String studentPage*(*Student student,Model model*){* System.*out*.println*(*student*)*;  
 ArrayList*<*Student*>* students = new ArrayList*<>()*;  
 Student s = StudentMapper.selectById(student.getStudentID());  
 s.setFacultyName(facultyController.getFacultyById(s.getFacultyNO()).getFacultyName());  
 students.add(s);  
 model.addAttribute("studentList",students);  
 return "student/admin-list-public";  
 }  
  
 @GetMapping*(*"/student/studentListPage")  
 public String toTeacherList(Model model){  
 List<Student> studentList = StudentMapper.selectList(null);  
 for (Student student : studentList) {  
 student.setFacultyName(facultyController.getFacultyById(student.getFacultyNO()).getFacultyName());  
 }  
 model.addAttribute("studentList",studentList);  
 return "student/admin-list";  
 *}* @ResponseBody  
 @PostMapping*(*"/student/deleteStudentById"*)* public HashMap<String, Object> deleteStudentById(Student student){  
 log.info(student.toString());  
  
 int count = StudentMapper.deleteById(student.getStudentID());  
 log.info(String.valueOf(count));  
  
 HashMap*<*String, Object*>* map = new HashMap*<>()*;  
 if *(*count==1*){* map.put*(*"code","100");  
 map.put("id",student.getStudentID());  
 } else {  
 map.put("code","500");  
 map.put("id",student.getStudentID());  
 }  
 return map;  
 *}* @ResponseBody  
 @PostMapping*(*"/student/deleteBatchIds")  
 public HashMap<String, Object> deleteBatchIds(String ids){  
 System.out.println(ids);  
  
 JSON parse = JSONUtil.parse(ids);  
 int count = StudentMapper.deleteBatchIds*((*Collection*<*? extends Serializable*>)* parse);  
 log.info(String.valueOf(count));  
 HashMap<String, Object> map = new HashMap<>();  
 if (count>=1){  
 map.put("code","100");  
 map.put("ids",ids);  
 } else {  
 map.put("code","500");  
 map.put("ids",ids);  
 *}* return map;  
 *}* @GetMapping*(*"/student/studentAdd"*)* public String toStudentAdd*(*Model model*){* List*<*Faculty*>* facultyList = facultyController.toFacultyList();  
 model.addAttribute("facultyList",facultyList);  
 List<Class> classList = classController.toClassList();  
 model.addAttribute("classList",classList);  
 return "student/admin-add";  
 }  
  
 @ResponseBody  
 @PostMapping("/student/studentAdd")  
 public HashMap<String, Object> studentAdd(Student student){  
 log.info(student.toString());  
  
 int count = StudentMapper.insert(student);  
 *log*.info*(*String.valueOf(count));  
  
 HashMap<String, Object> map = new HashMap<>();  
 if (count==1){  
 map.put("code","100");  
 map.put("id",student.getStudentID());  
 } else {  
 map.put("code","500");  
 map.put("id",student.getStudentID());  
 *}* return map;  
 *}* @GetMapping*(*"/student/studentUpdate/{id}")  
 public String toStudentUpdate(@PathVariable("id") Integer id,Model model){  
 List<Faculty> facultyList = facultyController.toFacultyList();  
 model.addAttribute("facultyList",facultyList);  
 List<Class> classList = classController.toClassList();  
 model.addAttribute("classList",classList);  
 Student s = StudentMapper.selectById(id);  
 model.addAttribute("student",s);  
 return "student/admin-update";  
 }  
  
 @ResponseBody  
 @PostMapping("/student/studentUpdate")  
 public HashMap<String, Object> studentUpdate(Student student){  
 *log*.info*(*student.toString*())*;  
  
 int count = StudentMapper.updateById*(*student*)*;  
 *log*.info*(*String.valueOf(count));  
  
 HashMap<String, Object> map = new HashMap<>();  
 if (count==1){  
 map.put("code","100");  
 map.put("id",student.getStudentID());  
 } else {  
 map.put("code","500");  
 map.put("id",student.getStudentID());  
 }  
 return map;  
 }  
  
}

# 第四章 总结

通过实现本作品，我学会使用SSM框架搭建一个前后端不分离的WEB项目，并且能够把mybatis-plus、thymeleaf、springMVC、druid、lombok、hutool整合进入SSM框架中。在项目中能够使用基于注解的开发模式，减少代码冗余，提到代码复用性。在项目中我使用实体类接收前台传送过来的数据，springMVC自动为我们封装数据，简化开发步骤。在课程表模块中，我们使用SpringMVC的@ResponseBody注解返回给前台JOSN数据并产生课表。还能够使用拦截器对未登录的请求进行拦截，未登录就直接跳转登录页面。并且还对用户的权限进行判断，数据表user中的type字段为权限字段（0学生 1老师 2管理员），管理员用户的权限最大，能够对所有的表进行增删改查，并且能够给课表进行排课，能查询所有学年学期、所有老师、所有班级的课表，学生和老师的权限最小，只能够修改自己的密码和基础的个人信息，并查询自己的课程表，每个用户登录之后的界面不同。在课程表页面我们通过选择课程名称、班级名称、教师姓名、学年、学期、开始周、结束周、星期几、第几节、地点、教师进行排课，排课完成之后就能在我的课程表页面查询出来，学生用户只能查询本班级课表，教师用户只能查询自己的课表。