目 录

[枣庄学院 1](#_Toc4513)

[人工智能学院课程设计 1](#_Toc9603)

[目 录 5](#_Toc17418)

[1.项目主题 7](#_Toc12577)

[1.1项目主题 7](#_Toc19018)

[1.2项目模块 7](#_Toc28217)

[1.3 技术栈 9](#_Toc16922)

[1.3.1 VUE 10](#_Toc7576)

[1.3.2Spring 10](#_Toc811)

[1.3.3SpringMVC 10](#_Toc12792)

[1.3.4mybatis-plus 10](#_Toc28657)

[1.3.5MVC项目架构 10](#_Toc23577)

[2.个人负责部分 11](#_Toc21979)

[2.1主要功能 11](#_Toc23697)

[2.2主要代码 11](#_Toc8298)

[2.3功能截图 13](#_Toc28959)

[3.智慧校园系统功能开发 14](#_Toc32182)

[3.1项目准备 14](#_Toc8576)

[3.1.1 准备项目结构 14](#_Toc18531)

[3.1.2准备分页插件的配置类 14](#_Toc4130)

[3.1.3准备Swagger2的配置类 16](#_Toc27170)

[3.1.4准备util下的工具类 18](#_Toc21791)

[3.2 登录功能后台实现 32](#_Toc17183)

[3.2.1 验证码功能实现 32](#_Toc24170)

[3.2.2 登录校验功能实现 34](#_Toc17943)

[3.2.3 跳转至首页功能实现 41](#_Toc26137)

[3.3 年级管理功能实现 43](#_Toc13801)

[3.3.1 查询年级信息【分页带条件】 43](#_Toc22564)

[3.3.2 添加和修改年级信息 46](#_Toc6366)

[3.3.3 删除和批量删除年级信息 48](#_Toc2848)

[3.4 班级管理功能实现 49](#_Toc22266)

[3.4.1 回显搜索条件中的年级选项 49](#_Toc2998)

[3.4.2 查询班级信息【分页带条件】 51](#_Toc25624)

[3.4.3 添加和修改学生信息 53](#_Toc30947)

[3.4.4 删除和批量删除班级信息 54](#_Toc7069)

[3.5 学生管理功能实现 57](#_Toc8016)

[3.5.1 回显搜索条件中的班级选项 57](#_Toc25347)

[3.5.2 查询学生信息【分页带条件】 58](#_Toc13370)

[3.5.3 异步图片上传处理头像 61](#_Toc4497)

[3.5.4 添加和修改学生信息 64](#_Toc21699)

[3.5.6 删除和批量删除学生信息 66](#_Toc20217)

[3.6 教师管理功能实现 68](#_Toc21668)

[3.6.1 查询教师信息【分页带条件】 68](#_Toc26543)

[3.6.2 添加和修改教师信息 70](#_Toc32663)

[3.6.3 删除和批量删除教师信息 71](#_Toc19769)

[3.7管理员管理功能实现 72](#_Toc19188)

[3.7.1 查询管理员信息【分页带条件】 73](#_Toc27084)

[3.7.2 添加和修改管理员信息 75](#_Toc8535)

[3.7.3 删除和批量删除管理员信息 76](#_Toc27114)

[3.7.4 管理员修改自己的密码 77](#_Toc71)

[4.课程设计总结 80](#_Toc29665)

# 1.项目主题

## 1.1项目主题

智慧校园管理系统：主要是以年级、班级为单位，进行老师和学生信息记录和统计功能。 项目采用前后端分离架构思想，前端采用HTML+CSS+VUE来实现页面效果展示，后端采用 SpringBoot+MybatisPlus框架实现数据存储等服务。存储层使用高性能的MySQL，服务器 使用SpringBoot内置的Tomcat9.x，项目构建工具使用Maven来管理jar包和项目构建。

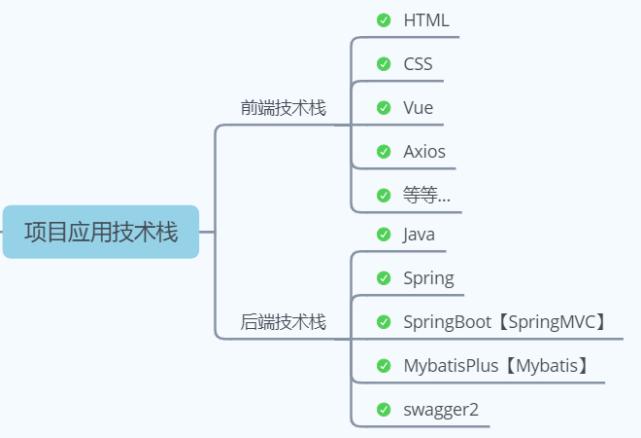
## 1.2项目模块







## 1.3 技术栈



### 1.3.1 VUE

是一套用于构建用户界面的渐进式框架。与其它大型框架不同的是， Vue 被设计为 可以自底向上逐层应用。 Vue 的核心库只关注视图层，不仅易于上手，还便于与第 三方库或既有项目整合。另一方面，当与现代化的工具链以及各种支持类库结合使 用时， Vue 也完全能够为复杂的单页应用提供驱动。 前后端分离是目前一种非常流 行的开发模式，它使项目的分工更加明确：后端：负责处理、存储数据. 前端：负责 显示数据 前端和后端开发人员通过 接口进行数据的交换。

### 1.3.2Spring

Spring就像是整个项目中装配bean的大工厂，在配置文件中可以指定使用特定的 参数去调用实体类的构造方法来实例化对象。也可以称之为项目中的粘合剂。

Spring的核心思想是IoC (控制反转)，即不再需要程序员去显式地 new一个对 象，而是让Spring框架帮你来完成这一切。

### 1.3.3SpringMVC

SpringMVC在项目中拦截用户请求，它的核心Servlet即DispatcherServlet承担中介 或是前台这样的职责，将用户请求通过HandlerMapping去匹配Controller， Controller就是具体对应请求所执行的操作。 SpringMVC相当于SSH框架中struts。

### 1.3.4mybatis-plus

mybatis是对jdbc的封装，它让数据库底层操作变的透明。 mybatis的操作都是围绕 一个sqlSessionFactory实例展开的。 mybatis通过配置文件关联到各实体类的Mapper 文件， Mapper文件中配置了每个类对数据库所需进行的sql语句映射。在每次与数据 库交互时，通过sqlSessionFactory拿到一个sqlSession，再执行sql命令。 MyBatis- plus就是在MyBatis的基础上,为Mapper接口,Service层提供一些比较全面的CURD的 业务逻辑功能,使程序员可以减少在Mapper和Service层的代码编写

### 1.3.5MVC项目架构

页面发送请求给控制器，控制器调用业务层处理逻辑，逻辑层向持久层发送请求， 持久层与数据库交互，后将结果返回给业务层，业务层将处理逻辑发送给控制器， 控制器再调用视图展现数据。

# 2.个人负责部分

## 2.1主要功能

·查询年级信息

·添加和修改年级信息

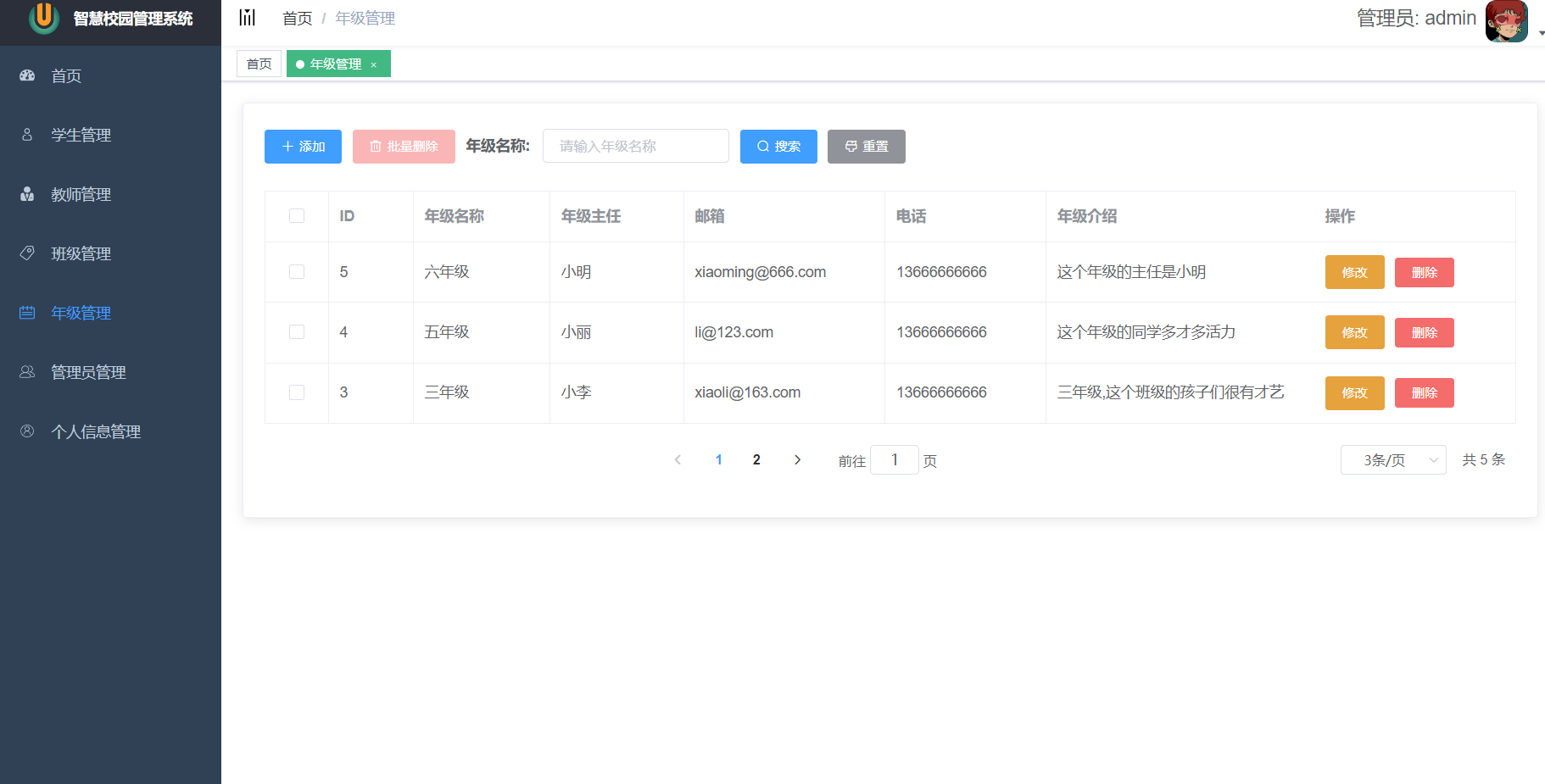
·删除和批量删除年级信息

**2.2主要代码**

**package** com.chen.zhxy.controller;  
**import** com.baomidou.mybatisplus.core.metadata.IPage;  
**import** com.baomidou.mybatisplus.extension.plugins.pagination.Page;  
**import** com.chen.zhxy.entity.Grade;  
**import** com.chen.zhxy.service.GradeService;  
**import** com.chen.zhxy.utils.Result;  
**import** io.swagger.annotations.Api;  
**import** io.swagger.annotations.ApiOperation;  
**import** io.swagger.annotations.ApiParam;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.web.bind.annotation.\*;  
**import** java.util.List;  
  
*/\*\*  
 \* <p>  
 \* 前端控制器  
 \* </p>  
 \*  
 \** ***@author*** *galaxy  
 \** ***@since*** *2022-04-18  
 \*/*@Api(tags = **"年级控制器"**)  
@RestController  
@RequestMapping(**"sms/gradeController"**)  
**public class** GradeController {  
 @Autowired  
 **private** GradeService **gradeService**;  
  
 @ApiOperation(**"查询年级信息,分页带条件"**)  
 @GetMapping(**"/getGrades/{pageNo}/{pageSize}"**)  
 **public** Result getGradeByOpr(@ApiParam(**"分页查询页码数"**) @PathVariable Integer pageNo,  
 @ApiParam(**"分页查询页大小"**) @PathVariable Integer pageSize,  
 @ApiParam(**"分页查询模糊匹配班级名"**) String gradeName) {  
 *//设置分页信息 封装一个Page对象* Page<Grade> page = **new** Page<>(pageNo, pageSize);  
 *//传入分页条件进行查询* IPage<Grade> iPage = **gradeService**.getGradeByOpr(page, gradeName);  
 *//快速封装Result对象并返回* **return** Result.*ok*(iPage);  
 }  
  
 @PostMapping(**"/saveOrUpdateGrade"**)  
 **public** Result saveOrUpdateGrade(@RequestBody Grade grade) {  
 *//接收对象 ：用@RequestBody 将json串转成Grade grade对象  
 //完成增加或者修改* **gradeService**.saveOrUpdate(grade);  
 **return** Result.*ok*();  
 }  
  
 @DeleteMapping(**"/deleteGrade"**)  
 *//@RequestBody将请求体中的json串转换成List类型* **public** Result deleteGrade(@RequestBody List<Integer> ids) {  
 **gradeService**.removeByIds(ids);  
 **return** Result.*ok*();  
 }  
  
 @GetMapping(**"/getGrades"**)  
 **public** Result getGrades() {  
 List<Grade> grades = **gradeService**.list();  
 **return** Result.*ok*(grades);  
 }  
}

**package** com.chen.zhxy.service.impl;  
  
**import** com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;  
**import** com.baomidou.mybatisplus.core.metadata.IPage;  
**import** com.baomidou.mybatisplus.extension.plugins.pagination.Page;  
**import** com.chen.zhxy.entity.Grade;  
**import** com.chen.zhxy.mapper.GradeMapper;  
**import** com.chen.zhxy.service.GradeService;  
**import** com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;  
**import** org.springframework.context.annotation.Bean;  
**import** org.springframework.stereotype.Service;  
**import** org.springframework.util.StringUtils;  
  
**import** java.util.List;  
  
*/\*\*  
 \* <p>  
 \* 服务实现类  
 \* </p>  
 \*分页带条件查询  
 \** ***@author*** *galaxy  
 \** ***@since*** *2022-04-18  
 \*/*@Service  
**public class** GradeServiceImpl **extends** ServiceImpl<GradeMapper, Grade> **implements** GradeService {  
  
 @Override  
 **public** IPage<Grade> getGradeByOpr(Page<Grade> page, String gradeName) {  
 QueryWrapper<Grade> wrapper = **new** QueryWrapper<>();  
 **if** (!StringUtils.*isEmpty*(gradeName)) {  
 wrapper.like(**"name"**, gradeName);  
 }  
 *//根据id属性降序查询* wrapper.orderByDesc(**"id"**);  
 **return baseMapper**.selectPage(page, wrapper);*//括号里是Page对象和查询条件* }  
}

**2.3功能截图**



# 

# 3.智慧校园系统功能开发

## 3.1项目准备

### 3.1.1 准备项目结构

java目录下

a. config : 项目的配置类

b. controller: 控制层

c. mapper : 持久层接口

d. pojo : 实体类

e. service: 服务层

f. util: 工具类包

g. ZhxyApplication : 启动类

resources目录下

a. mapper :持久层映射文件

b. public/upload:文件上传目录

c. static: 静态资源目录

d. application.yml :SpringBoot核心配置文件

### 3.1.2准备分页插件的配置类

import

com.baomidou.mybatisplus.extension.plugins.PaginationInterceptor; import org.mybatis.spring.annotation.MapperScan;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

@MapperScan("com.atguigu.zhxy.mapper")

public class MpConfig {

/\*\*

\* 分页插件

\*/

@Bean

public PaginationInterceptor paginationInterceptor() {

PaginationInterceptor paginationInterceptor = new PaginationInterceptor();

// paginationInterceptor.setLimit(你的最大单页限制数量，默认 500 条，小于 0 如 -1 不受限制);

return paginationInterceptor;

}

}

### 3.1.3准备Swagger2的配置类

package com.atguigu.zhxy.config;

import com.google.common.base.Predicates;

import io.swagger.annotations.ApiOperation;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration; import springfox.documentation.builders.ApiInfoBuilder;

import springfox.documentation.builders.ParameterBuilder;

import springfox.documentation.builders.PathSelectors;

import springfox.documentation.builders.RequestHandlerSelectors; import springfox.documentation.schema.ModelRef;

import springfox.documentation.service.ApiInfo;

import springfox.documentation.service.Contact;

import springfox.documentation.service.Parameter;

import springfox.documentation.spi.DocumentationType;

import springfox.documentation.spring.web.plugins.Docket;

import springfox.documentation.swagger2.annotations.EnableSwagger2;

import java.util.ArrayList;

import java.util.List;

/\*\*

\* Swagger2配置信息

\*/

@Configuration

@EnableSwagger2

public class Swagger2Config {

@Bean

public Docket webApiConfig(){

//添加head参数start

List<Parameter> pars = new ArrayList<>(); ParameterBuilder tokenPar = new ParameterBuilder(); tokenPar.name("userId")

.description("用户ID")

.defaultValue("1")

.modelRef(new ModelRef("string"))

.parameterType("header")

.required(false)

.build();

pars.add(tokenPar.build());

ParameterBuilder tmpPar = new ParameterBuilder();

tmpPar.name("userTempId")

.description("临时用户ID")

.defaultValue("1")

.modelRef(new ModelRef("string"))

.parameterType("header")

.required(false)

.build();

pars.add(tmpPar.build());

//添加head参数end

return new Docket(DocumentationType.SWAGGER\_2)

.groupName("webApi")

.apiInfo(webApiInfo())

.select()

//可以测试请求头中：输入token

.apis(RequestHandlerSelectors.withClassAnnotation(ApiOperation.clas s))

//过滤掉admin路径下的所有页面

//.paths(Predicates.and(PathSelectors.regex("/sms/.\*"))) //过滤掉所有error或error.\*页面

//.paths(Predicates.not(PathSelectors.regex("/error.\*")))

.build()

.globalOperationParameters(pars);

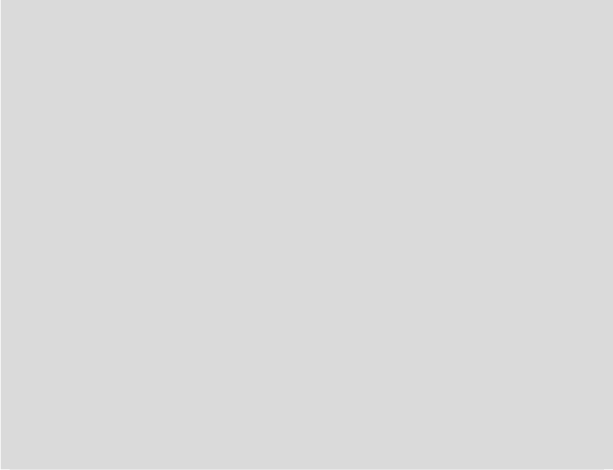
}

private ApiInfo webApiInfo(){

return new ApiInfoBuilder()

.title("网站-API文档")

.description("本文档描述了网站微服务接口定义") .version("1.0")



.contact(new Contact("atguigu",

"<http://atguigu.com>", "512111559@qq.com"))

.build();

}

private ApiInfo adminApiInfo(){

return new ApiInfoBuilder()

.title("后台管理系统-API文档")

.description("本文档描述了后台管理系统微服务接口定义") .version("1.0")

.contact(new Contact("atguigu",

"<http://atguigu.com>", "512111559@qq.com"))

.build();

}

}

### 3.1.4准备util下的工具类

 验证码图片工具类

package com.atguigu.zhxy.util;

import java.awt.\*;

import java.awt.image.BufferedImage;

import java.util.Random;

/\*\*

\* @project: ssm\_sms

\* @description: 绘制验证码图片 \*/

public class CreateVerifiCodeImage {

private static int WIDTH = 90;

private static int HEIGHT = 35;

private static int FONT\_SIZE = 20; //字符大小

private static char[] verifiCode; //验证码

private static BufferedImage verifiCodeImage; //验证码图片

/\*\*

\* @description: 获取验证码图片

\* @param: no

\* @return: java.awt.image.BufferedImage \*/

public static BufferedImage getVerifiCodeImage() {

verifiCodeImage = new BufferedImage(WIDTH, HEIGHT, BufferedImage.TYPE\_INT\_BGR);// create a image

Graphics graphics = verifiCodeImage.getGraphics();

verifiCode = generateCheckCode();

drawBackground(graphics);

drawRands(graphics, verifiCode);

graphics.dispose();

return verifiCodeImage;

}

/\*\*

\* @description: 获取验证码

\* @param: no

\* @return: char[] \*/

public static char[] getVerifiCode() {

return verifiCode;

}

/\*\*

\* @description: 随机生成验证码

\* @param: no

\* @return: char[] \*/

private static char[] generateCheckCode() {

String chars = "0123456789abcdefghijklmnopqrstuvwxyz" + "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

char[] rands = new char[4];

for (int i = 0; i < 4; i++) {

int rand = (int) (Math.random() \* (10 + 26 \* 2));

rands[i] = chars.charAt(rand);

}

return rands;

}

/\*\*

\* @description: 绘制验证码

\* @param: g

\* @param: rands

\* @return: void \*/

private static void drawRands(Graphics g, char[] rands) { g.setFont(new Font("Console", Font.BOLD, FONT\_SIZE));

for (int i = 0; i < rands.length; i++) {

g.setColor(getRandomColor());

g.drawString("" + rands[i], i \* FONT\_SIZE + 10, 25);

}

}

/\*\*

\* @description: 绘制验证码图片背景

\* @param: g

\* @return: void

\*/

private static void drawBackground(Graphics g) {

g.setColor(Color.white);

g.fillRect(0, 0, WIDTH, HEIGHT);

// 绘制验证码干扰点

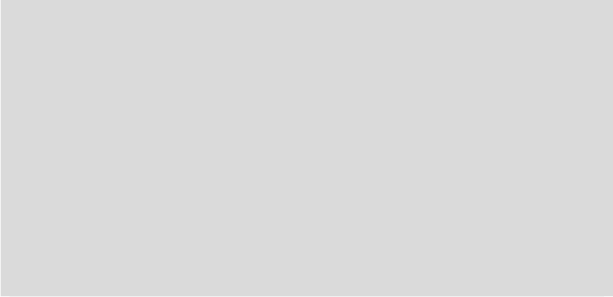
for (int i = 0; i < 200; i++) {

int x = (int) (Math.random() \* WIDTH);

int y = (int) (Math.random() \* HEIGHT);

g.setColor(getRandomColor());

g.drawOval(x, y, 1, 1);



/\*\*

\* @description: 获取随机颜色

\* @param: no

\* @return: java.awt.Color \*/

private static Color getRandomColor() {

Random ran = new Random();

return new Color(ran.nextInt(220), ran.nextInt(220), ran.nextInt(220));

}

}

*token*口令生成工具 *JwtHelper*

package com.atguigu.zhxy.util;

import io.jsonwebtoken.\*;

import org.springframework.util.StringUtils;

import java.util.Date;

public class JwtHelper {

private static long tokenExpiration = 24\*60\*60\*1000; private static String tokenSignKey = "123456";

//生成token字符串

public static String createToken(Long userId, Integer userType) {

String token = Jwts.builder()

.setSubject("YYGH-USER")

.setExpiration(new Date(System.currentTimeMillis() + tokenExpiration))

.claim("userId", userId)

// .claim("userName", userName)

.claim("userType", userType)

.signWith(SignatureAlgorithm.HS512, tokenSignKey) .compressWith(CompressionCodecs.GZIP)

.compact();

return token;

}

//从token字符串获取userid

public static Long getUserId(String token) {

if(StringUtils.isEmpty(token)) return null;

Jws<Claims> claimsJws =

Jwts.parser().setSigningKey(tokenSignKey).parseClaimsJws(token); Claims claims = claimsJws.getBody();

Integer userId = (Integer)claims.get("userId"); return userId.longValue();

}

//从token字符串获取userType

public static Integer getUserType(String token) {

if(StringUtils.isEmpty(token)) return null;

Jws<Claims> claimsJws

=

Jwts.parser().setSigningKey(tokenSignKey).parseClaimsJws(token); Claims claims = claimsJws.getBody();

return (Integer)(claims.get("userType"));

}

//从token字符串获取userName

public static String getUserName(String token) {

if(StringUtils.isEmpty(token)) return "";

Jws<Claims> claimsJws

=

Jwts.parser().setSigningKey(tokenSignKey).parseClaimsJws(token); Claims claims = claimsJws.getBody();

return (String)claims.get("userName");

}

//判断token是否有效

public static boolean isExpiration(String token){

try {

boolean isExpire = Jwts.parser()

.setSigningKey(tokenSignKey)

.parseClaimsJws(token)

.getBody()

.getExpiration().before(new Date());

//没有过期，有效，返回false

return isExpire;

}catch(Exception e) {

//过期出现异常，返回true

return true;

}

}

/\*\*

\* 刷新Token

\* @param token

\* @return \*/

public String refreshToken(String token) {

String refreshedToken;

try {

final Claims claims = Jwts.parser()

.setSigningKey(tokenSignKey)

.parseClaimsJws(token)

.getBody();

refreshedToken =

JwtHelper.createToken(getUserId(token), getUserType(token)); } catch (Exception e) {

refreshedToken = null;

}

return refreshedToken;

}

//

//

//

//

}

}

public static void main(String[] args) {

String token = JwtHelper.createToken(1L, "lucy"); System.out.println(token);

System.out.println(JwtHelper.getUserId(token)); System.out.println(JwtHelper.getUserName(token));

解析*request*请求中的 *token*口令的工具*AuthContextHolder*

package com.atguigu.zhxy.util;

import javax.servlet.http.HttpServletRequest;

public class AuthContextHolder {

//从请求头token获取userid

public static Long getUserIdToken(HttpServletRequest request) {

//从请求头token

String token = request.getHeader("token");

//调用工具类

Long userId = JwtHelper.getUserId(token);

return userId;

}

//从请求头token获取name

public static String getUserName(HttpServletRequest request) {

//从header获取token

String token = request.getHeader("token");

//jwt从token获取username

String userName = JwtHelper.getUserName(token); return userName;

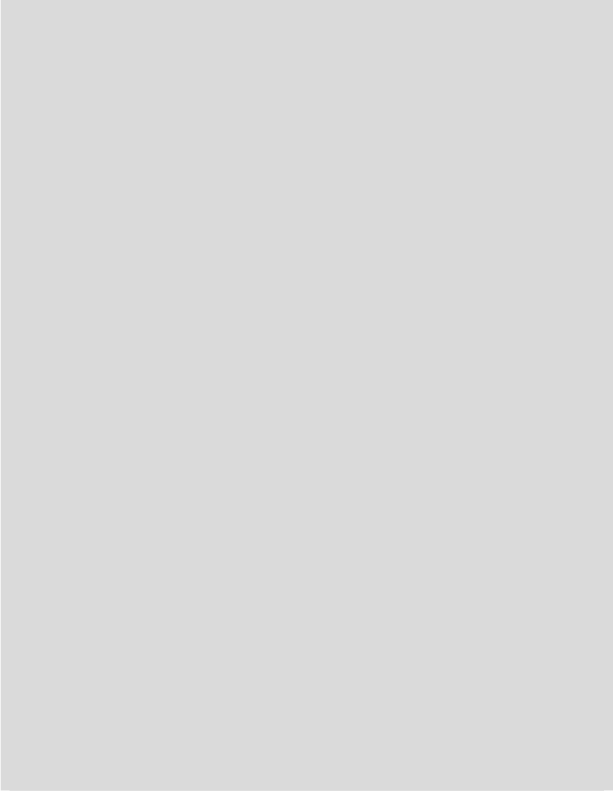
}

}

*MD5*加密工具类

package com.atguigu.zhxy.util;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

public final class MD5 {

public static String encrypt(String strSrc) {

try {

char hexChars[] = { '0', '1', '2', '3', '4', '5', '6',

'7', '8',

'9', 'a', 'b', 'c', 'd', 'e', 'f' }; byte[] bytes = strSrc.getBytes();

MessageDigest md = MessageDigest.getInstance("MD5"); md.update(bytes);

bytes = md.digest();

int j = bytes.length;

char[] chars = new char[j \* 2];

int k = 0;

for (int i = 0; i < bytes.length; i++) {

byte b = bytes[i];

chars[k++] = hexChars[b >>> 4 & 0xf];

chars[k++] = hexChars[b & 0xf];

}

return new String(chars);

} catch (NoSuchAlgorithmException e) {

e.printStackTrace();

throw new RuntimeException("MD5加密出错！！+" + e);

}

}

}

*JSON*响应结果格式封装类

package com.atguigu.zhxy.util;

import io.swagger.annotations.ApiModel;

import io.swagger.annotations.ApiModelProperty;

import lombok.Data;

/\*\*

\* 全局统一返回结果类

\*

\*/

@Data

@ApiModel(value = "全局统一返回结果")

public class Result<T> {

@ApiModelProperty(value = "返回码")

private Integer code;

@ApiModelProperty(value = "返回消息")

private String message;

@ApiModelProperty(value = "返回数据")

private T data;

public Result(){}

// 返回数据

protected static <T> Result<T> build(T data) {

Result<T> result = new Result<T>();

if (data != null)

result.setData(data);

return result;

}

public static <T> Result<T> build(T body, ResultCodeEnum resultCodeEnum) {

Result<T> result = build(body);

result.setCode(resultCodeEnum.getCode()); result.setMessage(resultCodeEnum.getMessage()); return result;

}

public static<T> Result<T> ok(){

return Result.ok(null);

}

/\*\*

\* 操作成功

\* @param data

\* @param <T>

\* @return

\*/

public static<T> Result<T> ok(T data){

Result<T> result = build(data);

return build(data, ResultCodeEnum.SUCCESS);

}

public static<T> Result<T> fail(){

return Result.fail(null);

}

/\*\*

\* 操作失败

\* @param data

\* @param <T>

\* @return

\*/

public static<T> Result<T> fail(T data){

Result<T> result = build(data);

return build(data, ResultCodeEnum.FAIL);

}

public Result<T> message(String msg){

this.setMessage(msg);

return this;

}

public Result<T> code(Integer code){

this.setCode(code);

return this;

}

public boolean isOk() {

if(this.getCode().intValue() ==

ResultCodeEnum.SUCCESS.getCode().intValue()) {

return true;

}

return false;

}

}

响应结果类型码枚举

package com.atguigu.zhxy.util;

import lombok.Getter;

/\*\*

\* 统一返回结果状态信息类

\*

\*/

@Getter

public enum ResultCodeEnum {

SUCCESS(200,"成功"),

FAIL(201, "失败"),

SERVICE\_ERROR(2012, "服务异常"),

ILLEGAL\_REQUEST( 204, "非法请求"),

PAY\_RUN(205, "支付中"),

ARGUMENT\_VALID\_ERROR(206, "参数校验错误"),

LOGIN\_ERROR(207, "用户名或密码错误"),

LOGIN\_AUTH(208, "未登陆"),

PERMISSION(209, "没有权限"),

SECKILL\_NO\_START(210, "秒杀还没开始"),

SECKILL\_RUN(211, "正在排队中"),

SECKILL\_NO\_PAY\_ORDER(212, "您有未支付的订单"),

SECKILL\_FINISH(213, "已售罄"),

SECKILL\_END(214, "秒杀已结束"),

SECKILL\_SUCCESS(215, "抢单成功"),

SECKILL\_FAIL(216, "抢单失败"),

SECKILL\_ILLEGAL(217, "请求不合法"),

SECKILL\_ORDER\_SUCCESS(218, "下单成功"),

COUPON\_GET(220, "优惠券已经领取"),

COUPON\_LIMIT\_GET(221, "优惠券已发放完毕"),

//2022-02-22

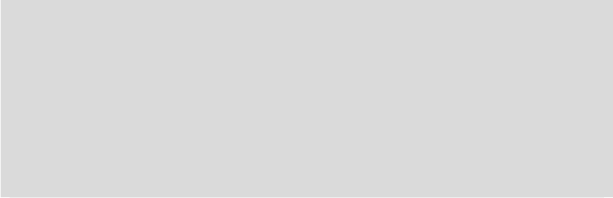
LOGIN\_CODE(222,"长时间未操作 ,会话已失效 ,请刷新页面后重试!"),

CODE\_ERROR(223,"验证码错误!"),

TOKEN\_ERROR(224,"Token无效!")

;

private Integer code;



private String message;

private ResultCodeEnum(Integer code, String message) {

this.code = code;

this.message = message;

}

}

文件上传工具类

package com.atguigu.zhxy.util;

import org.apache.commons.io.filefilter.SuffixFileFilter; import org.springframework.web.multipart.MultipartFile;

import java.io.File;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

import java.util.UUID;

/\*\*

\* @project: zhxy

\* @description: 上传文件的工具类 \*/

public class UploadFile {

//存储文件上传失败的错误信息

private static Map<String, Object> error\_result = new HashMap<> ();

//存储头像的上传结果信息

private static Map<String, Object> upload\_result = new HashMap<>();

/\*\*

\* @description: 效验所上传图片的大小及格式等信息 ...

\* @param: photo

\* @param: path

\* @return: java.util.Map<java.lang.String, java.lang.Object> \*/

private static Map<String, Object> uploadPhoto(MultipartFile photo, String path) {

//限制头像大小(20M)

int MAX\_SIZE = 20971520;

//获取图片的原始名称

String orginalName = photo.getOriginalFilename();

//如果保存文件的路径不存在 ,则创建该目录

File filePath = new File(path);

if (!filePath.exists()) {

filePath.mkdirs();

}

//限制上传文件的大小

if (photo.getSize() > MAX\_SIZE) {

error\_result.put("success", false);

error\_result.put("msg", "上传的图片大小不能超过20M哟!"); return error\_result;

}

// 限制上传的文件类型

String[] suffixs = new String[]{".png", ".PNG", ".jpg",

".JPG", ".jpeg", ".JPEG", ".gif", ".GIF", ".bmp", ".BMP"};

SuffixFileFilter suffixFileFilter = new

SuffixFileFilter(suffixs);

if (!suffixFileFilter.accept(new File(path + orginalName))) {

error\_result.put("success", false);

error\_result.put("msg", "禁止上传此类型文件! 请上传图片哟!"); return error\_result;

}

return null;

}

/\*\*

\* @description: (提取公共代码 : 提高代码的可重用性)获取头像的上传结果信 息

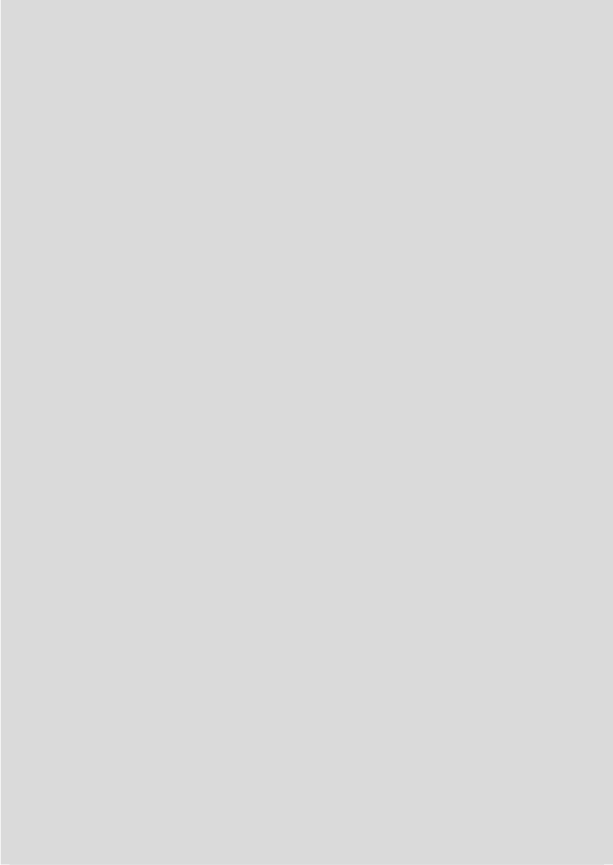
\* @param: photo

\* @param: dirPaht

\* @param: portraitPath

\* @return: java.util.Map<java.lang.String, java.lang.Object> \*/

public static Map<String, Object> getUploadResult(MultipartFile photo, String dirPaht, String portraitPath) {

if (!photo.isEmpty() && photo.getSize() > 0) {

//获取图片的原始名称

String orginalName = photo.getOriginalFilename();

//上传图片 ,error\_result:存储头像上传失败的错误信息

Map<String, Object> error\_result =

UploadFile.uploadPhoto(photo, dirPaht);

if (error\_result != null) {

return error\_result;

}

//使用UUID重命名图片名称(uuid\_\_原始图片名称)

String newPhotoName = UUID.randomUUID() + "\_\_" + orginalName;

//将上传的文件保存到目标目录下

try {

photo.transferTo(new File(dirPaht + newPhotoName)); upload\_result.put("success", true);

upload\_result.put("portrait\_path", portraitPath + newPhotoName);//将存储头像的项目路径返回给页面

} catch (IOException e) {

e.printStackTrace();

upload\_result.put("success", false); upload\_result.put("msg", "上传文件失败! 服务器端发生异

常!");

return upload\_result;

}

} else {

upload\_result.put("success", false);

upload\_result.put("msg", "头像上传失败! 未找到指定图片!"); }

return upload\_result;

}

}

## 3.2 登录功能后台实现

### 3.2.1 验证码功能实现

获取验证码图片及向*session*中存储验证码功能实现



*SystemController*层代码实现

package com.atguigu.zhxy.controller;

import com.atguigu.zhxy.pojo.Admin;

import com.atguigu.zhxy.pojo.LoginForm;

import com.atguigu.zhxy.service.AdminService;

import com.atguigu.zhxy.service.StudentService;

import com.atguigu.zhxy.service.TeacherService;

import com.atguigu.zhxy.util.CreateVerifiCodeImage;

import com.atguigu.zhxy.util.JwtHelper;

import com.atguigu.zhxy.util.Result;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.\*;

import javax.imageio.ImageIO;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import java.awt.image.BufferedImage;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

@Api(tags = "系统控制器")

@RestController

@RequestMapping("/sms/system")

public class SystemController {

@ApiOperation("获取验证码图片")

@GetMapping("/getVerifiCodeImage")

public void getVerifiCodeImage(HttpServletRequest request, HttpServletResponse response){

//获取验证码图片

BufferedImage verifiCodeImage =

CreateVerifiCodeImage.getVerifiCodeImage();

//获取验证码字符串

String verifiCode =

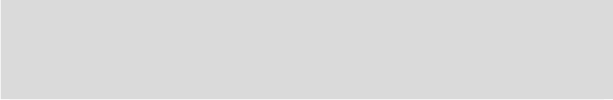
String.valueOf(CreateVerifiCodeImage.getVerifiCode());

/\*将验证码放入当前请求域\*/

request.getSession().setAttribute("verifiCode",verifiCode); try {

//将验证码图片通过输出流做出响应

ImageIO.write(verifiCodeImage,"JPEG",response.getOutputStream()); } catch (IOException e) {

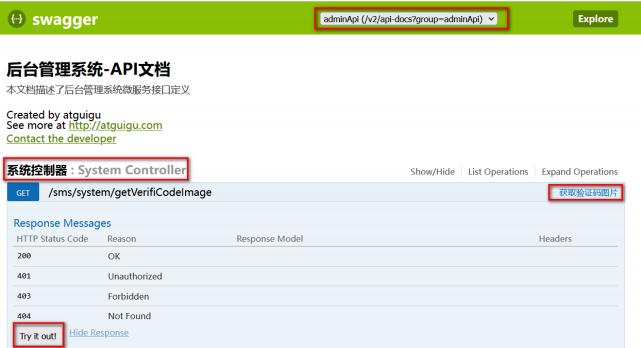


e.printStackTrace(); }

}

}

*swagger2*接口文档查阅



### 3.2.2 登录校验功能实现

*SystemController*层代码实现

package com.atguigu.zhxy.controller;

import com.atguigu.zhxy.pojo.Admin;

import com.atguigu.zhxy.pojo.LoginForm;

import com.atguigu.zhxy.pojo.Student;

import com.atguigu.zhxy.pojo.Teacher;

import com.atguigu.zhxy.service.AdminService;

import com.atguigu.zhxy.service.StudentService;

import com.atguigu.zhxy.service.TeacherService;

import com.atguigu.zhxy.util.CreateVerifiCodeImage;

import com.atguigu.zhxy.util.JwtHelper;

import com.atguigu.zhxy.util.Result;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.\*;

import javax.imageio.ImageIO;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import java.awt.image.BufferedImage;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

@Api(tags = "系统控制器")

@RestController

@RequestMapping("/sms/system")

public class SystemController {

@Autowired

private AdminService adService;

@Autowired

private StudentService studentService;

@Autowired

private TeacherService teacherService;

@ApiOperation("登录请求验证")

@PostMapping("/login")

public Result login(@RequestBody LoginForm loginForm, HttpServletRequest request){

// 获取用户提交的验证码和session域中的验证码

HttpSession session = request.getSession();

String systemVerifiCode =(String)

session.getAttribute("verifiCode");

String loginVerifiCode = loginForm.getVerifiCode(); if("".equals(systemVerifiCode)){

// session过期 ,验证码超时 ,

return Result.fail().message("验证码失效 ,请刷新后重试"); }

if (!loginVerifiCode.equalsIgnoreCase(systemVerifiCode)){

// 验证码有误

return Result.fail().message("验证码有误 ,请刷新后重新输入"); }

// 验证码使用完毕 ,移除当前请求域中的验证码

session.removeAttribute("verifiCode");

// 准备一个Map集合 ,用户存放响应的信息

Map<String,Object> map=new HashMap<>();

// 根据用户身份 ,验证登录的用户信息

switch (loginForm.getUserType()){

case 1:// 管理员身份

try {

// 调用服务层登录方法 ,根据用户提交的LoginInfo信息 ,查询 对应的Admin对象 ,找不到返回Null

Admin login = adService.login(loginForm); if (null != login) {

// 登录成功 ,将用户id和用户类型转换为token口令 ,作为

信息响应给前端

map.put("token",JwtHelper.createToken(login.getId().longValue(), 1));

}else{

throw new RuntimeException("用户名或者密码有

误!");

}

return Result.ok(map);

} catch (RuntimeException e) {

e.printStackTrace();

// 捕获异常 ,向用户响应错误信息

return Result.fail().message(e.getMessage()); }

case 2:// 学生身份

try {

// 调用服务层登录方法 ,根据用户提交的LoginInfo信息 ,查询 对应的Student对象 ,找不到返回Null

Student login =

studentService.login(loginForm);

if (null != login) {

// 登录成功 ,将用户id和用户类型转换为token口令 ,作为

信息响应给前端

map.put("token",JwtHelper.createToken(login.getId().longValue(), 2));

}else{

throw new RuntimeException("用户名或者密码有

误!");

}

return Result.ok(map);

} catch (RuntimeException e) {

e.printStackTrace();

// 捕获异常 ,向用户响应错误信息

return Result.fail().message(e.getMessage()); }

case 3:// 教师身份

// 调用服务层登录方法 ,根据用户提交的LoginInfo信息 ,查询对应的 Teacher对象 ,找不到返回Null

try {

// 调用服务层登录方法 ,根据用户提交的LoginInfo信息 ,查询 对应的Student对象 ,找不到返回Null

Teacher login =

teacherService.login(loginForm);

if (null != login) {

// 登录成功 ,将用户id和用户类型转换为token口令 ,作为

信息响应给前端

map.put("token",JwtHelper.createToken(login.getId().longValue(), 3));

}else{

throw new RuntimeException("用户名或者密码有

误!");

}

return Result.ok(map);

} catch (RuntimeException e) {

e.printStackTrace();

// 捕获异常 ,向用户响应错误信息

return Result.fail().message(e.getMessage()); }

}

// 查无此用户 ,响应失败

return Result.fail().message("查无此用户");

}

}

*AdminService*中的*login*方法

public interface AdminService extends IService<Admin> {

/\*\*

\* 登录

\* @param loginForm

\* @return \*/

Admin login(LoginForm loginForm);

}

@Service("adService")

@Transactional

public class AdminServiceImpl extends ServiceImpl<AdminMapper, Admin> implements AdminService {

/\*\*

\* 超级管理员登录

\* @param loginForm

\* @return \*/

@Override

public Admin login(LoginForm loginForm) {

//创建QueryWrapper对象

QueryWrapper<Admin> queryWrapper = new QueryWrapper<>();

//拼接查询条件

queryWrapper.eq("name",loginForm.getUsername());

// 转换成密文进行查询

queryWrapper.eq("password",

MD5.encrypt(loginForm.getPassword()));

Admin admin = baseMapper.selectOne(queryWrapper);

return admin;

}

}

*StudentService*中的*login*方法

public interface StudentService extends IService<Student> {

/\*\*

\* 学生登录方法

\* @return

\*/

public Student login(LoginForm loginForm);

}

*TeacherService*中的*login*方法

public interface TeacherService extends IService<Teacher> {

/\*\*

\* 登录方法

\*/

public Teacher login(LoginForm loginForm);

}

@Service("teaService")

@Transactional

public class TeacherServiceImpl extends ServiceImpl<TeacherMapper, Teacher> implements TeacherService {

/\*\*

\* Teacher登录方法

\* @param loginForm

\* @return \*/

@Override

public Teacher login(LoginForm loginForm) {

QueryWrapper queryWrapper = new QueryWrapper(); queryWrapper.eq("name",loginForm.getUsername());

queryWrapper.eq("password",

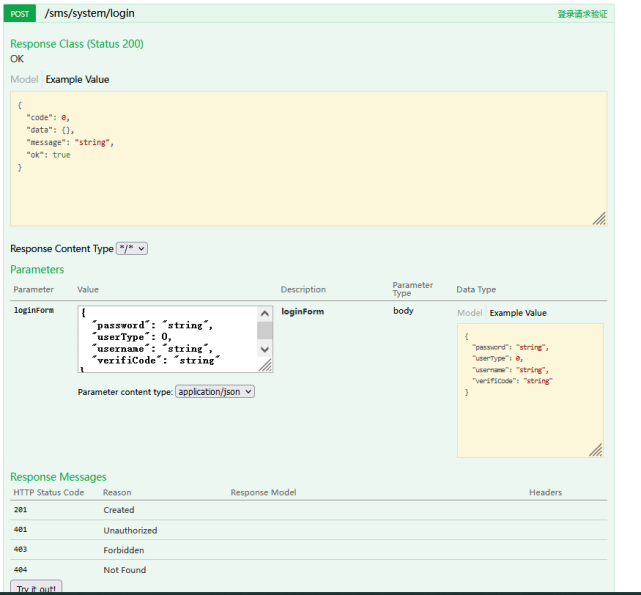
MD5.encrypt(loginForm.getPassword()));

Teacher teacher = baseMapper.selectOne(queryWrapper); return teacher;

}

}

通过*SwaggerUI* 查看和测试接口



### 3.2.3 跳转至首页功能实现

当验证通过后*,*前端会产生第二个请求*,*用以获取用户的类型*,*然后根据用户的类型来 展现不同的页面*,*所以后端要有一个根据*token*解析用户类型并做出结果响应的控制层

@Api(tags = "系统控制器")

@RestController

@RequestMapping("/sms/system")

public class SystemController {

@Autowired

private AdminService adService;

@Autowired

private StudentService studentService;

@Autowired

private TeacherService teacherService;

@ApiOperation("通过token获取用户信息")

@GetMapping("/getInfo")

public Result getUserInfoByToken(HttpServletRequest request, @RequestHeader("token")String token){

// 获取用户中请求的token

// 检查token 是否过期 20H

boolean isEx = JwtHelper.isExpiration(token);

if (isEx) {

return Result.build(null, ResultCodeEnum.TOKEN\_ERROR); }

// 解析token,获取用户id和用户类型

Long userId =JwtHelper.getUserId(token); Integer userType =JwtHelper.getUserType(token);

// 准备一个Map集合用于存响应的数据

Map<String,Object> map=new HashMap<>();

switch (userType){

case 1:

Admin admin =

adService.getAdminById(userId.intValue());

map.put("user",admin);

map.put("userType",1);

break;

case 2:

Student student =

studentService.getStudentById(userId.intValue());

map.put("user",student);

map.put("userType",1);

break;

case 3:

Teacher teacher =

teacherService.getTeacherById(userId.intValue());

map.put("user",teacher);

map.put("userType",1);

break;

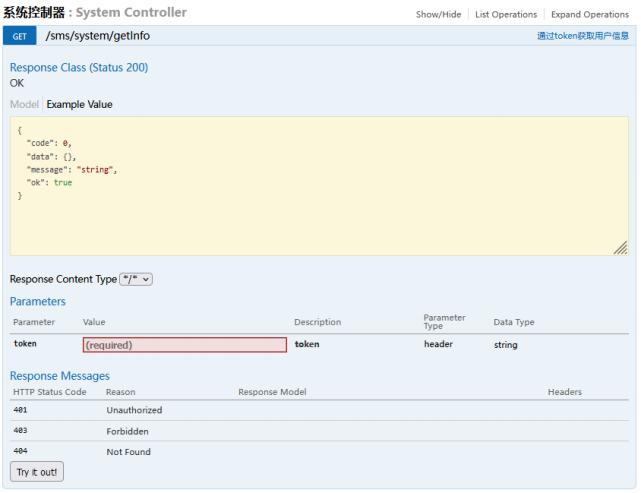
}

return Result.ok(map);



}

}



## 3.3 年级管理功能实现

### 3.3.1 查询年级信息【分页带条件】

*Controller*层代码

@Api(tags = "年级控制器")

@RestController

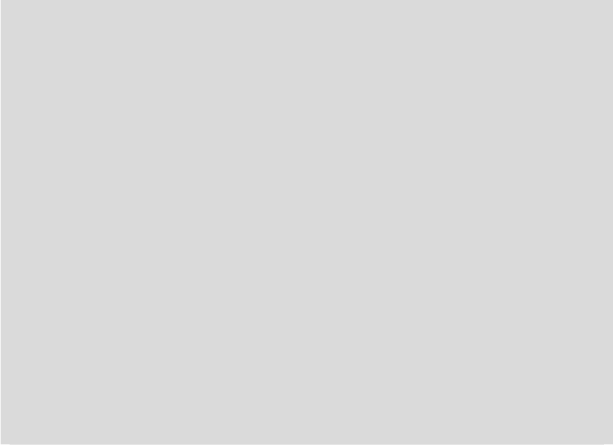
@RequestMapping("/sms/gradeController")

public class GradeController {

@Autowired

private GradeService gradeService;

@ApiOperation("查询年级信息 ,分页带条件")



@GetMapping("/getGrades/{pageNo}/{pageSize}")

public Result getGradeByOpr(

@ApiParam("分页查询页码数") @PathVariable(value = "pageNo") Integer pageNo, // 页码数

@ApiParam("分页查询页大小") @PathVariable(value = "pageSize")Integer pageSize, // 页大小

@ApiParam("分页查询模糊匹配班级名") String gradeName)// 模糊 查询条件

{

// 设置分页信息

Page<Grade> page =new Page<>(pageNo,pageSize);

// 调用服务层方法 ,传入分页信息 ,和查询的条件

IPage<Grade> pageRs = gradeService.getGradeByOpr(page, gradeName);

return Result.ok(pageRs);

}

}

*Service*层代码

@Service

@Transactional

public class GradeServiceImpl extends ServiceImpl<GradeMapper, Grade> implements GradeService {

@Override

public IPage<Grade> getGradeByOpr(Page<Grade> pageParam, String gradeName) {

// 设置查询条件

QueryWrapper queryWrapper = new QueryWrapper(); if(!StringUtils.isEmpty(gradeName)){

queryWrapper.like("name",gradeName);

}

// 设置排序规则

queryWrapper.orderByDesc("id");

queryWrapper.orderByAsc("name");

// 分页查询数据

Page page = baseMapper.selectPage(pageParam, queryWrapper);

return page;

}



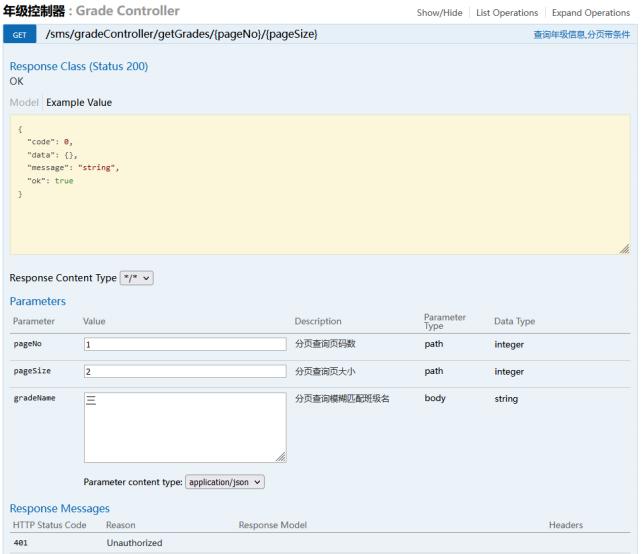
}

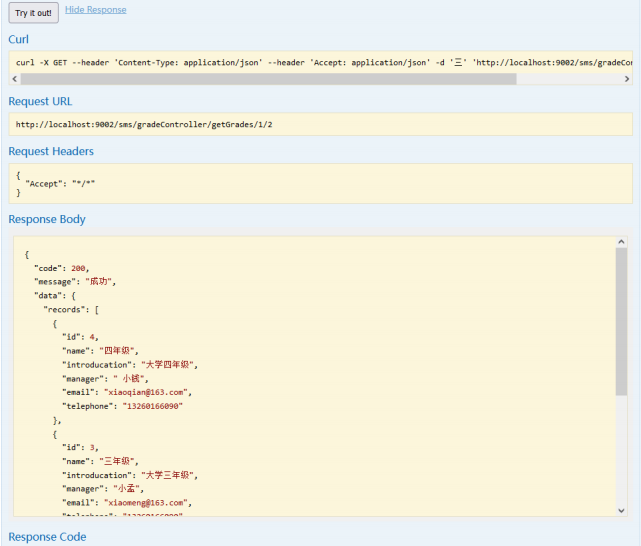
持久层代码

// MyBatisPlus 的BaseMapper已经封装好分页带条件查询方法

<E extends IPage<T>> E selectPage(E page, @Param("ew") Wrapper<T> queryWrapper);

*swagger*测试接口功能





### 3.3.2 添加和修改年级信息

*Controller*层代码

@Api(tags = "年级控制器")

@RestController

@RequestMapping("/sms/gradeController")

public class GradeController {

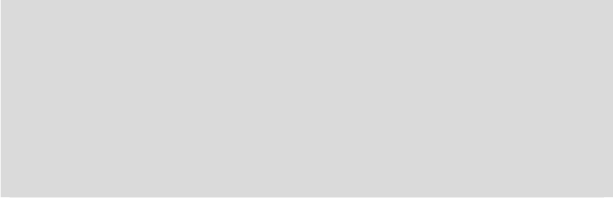
@Autowired

private GradeService gradeService;

@ApiOperation("添加或者修改年级信息")

@PostMapping("/saveOrUpdateGrade")

public Result saveOrUpdateGrade(



@ApiParam("JSON的grade对象转换后台数据模型")@RequestBody

Grade grade

){

// 调用服务层方法 ,实现添加或者修改年级信息

gradeService.saveOrUpdate(grade);

return Result.ok();

}

}

*service*层代码

// MyBatisPlus源码可见

public boolean saveOrUpdate(T entity) {

if (null == entity) {

return false;

} else {

Class<?> cls = entity.getClass();

TableInfo tableInfo =

TableInfoHelper.getTableInfo(cls);

Assert.notNull(tableInfo, "error: can not execute.

because can not find cache of TableInfo for entity!", new Object[0]);

String keyProperty = tableInfo.getKeyProperty(); Assert.notEmpty(keyProperty, "error: can not execute.

because can not find column for id from entity!", new Object[0]);

Object idVal = ReflectionKit.getMethodValue(cls, entity, tableInfo.getKeyProperty());

return !StringUtils.checkValNull(idVal) &&

!Objects.isNull(this.getById((Serializable)idVal)) ?

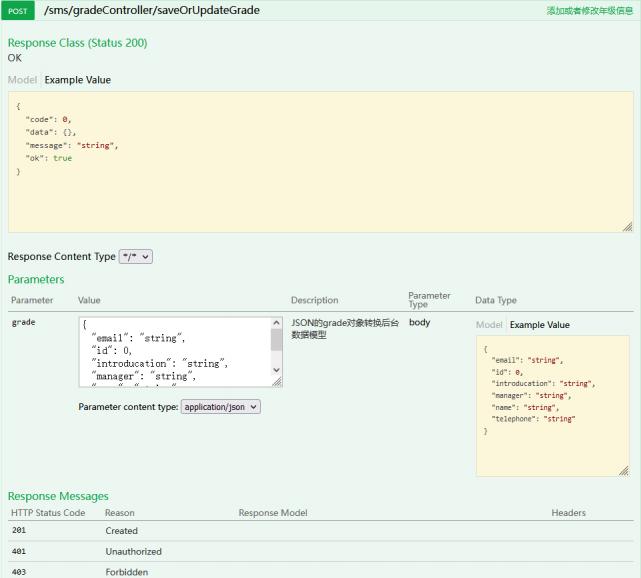
this.updateById(entity) : this.save(entity);

}

}

*mapper*层代码

略



### 3.3.3 删除和批量删除年级信息

 *Controller*层代码

@Api(tags = "年级控制器")

@RestController

@RequestMapping("/sms/gradeController")

public class GradeController {

@Autowired

private GradeService gradeService;

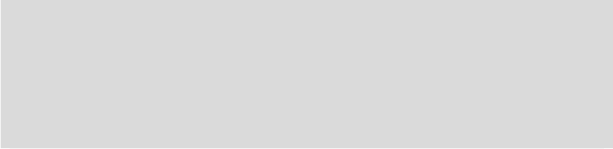
@ApiOperation("删除一个或者多个grade信息")

@DeleteMapping("/deleteGrade")

public Result deleteGradeById(

@ApiParam("JSON的年级id集合 ,映射为后台

List<Integer>")@RequestBody List<Integer> ids



)

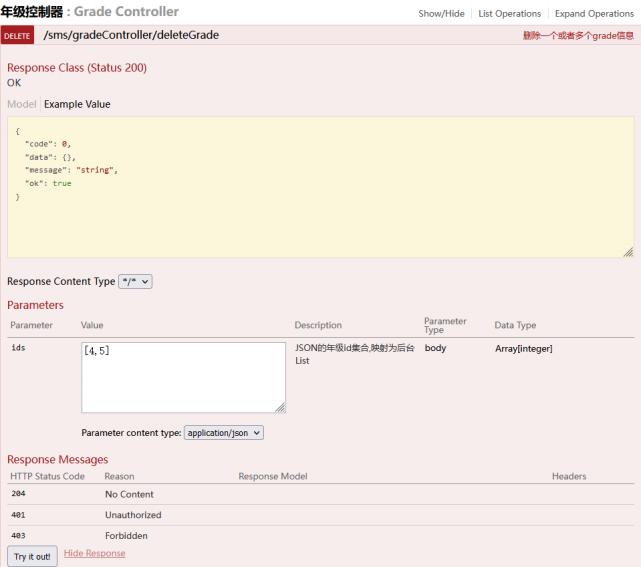
{

gradeService.removeByIds(ids); return Result.ok();

}

}

*swagger*测试接口



## 3.4 班级管理功能实现

### 3.4.1 回显搜索条件中的年级选项

*Controller*层代码

@Api(tags = "年级控制器")

@RestController

@RequestMapping("/sms/gradeController")

public class GradeController {

@Autowired

private GradeService gradeService;

@ApiOperation("获取所有Grade信息")

@GetMapping("/getGrades")

public Result getGrades(){

List<Grade> grades = gradeService.getGrades();

return Result.ok(grades);

}

}

*service*层代码

@Service

@Transactional

public class GradeServiceImpl extends ServiceImpl<GradeMapper, Grade> implements GradeService {

public List<Grade> getGrades(){

List<Grade> grades = baseMapper.selectList(null); return grades;

}

}

*mapper*层代码

public interface BaseMapper<T> extends Mapper<T>{

List<T> selectList(@Param("ew") Wrapper<T> queryWrapper); }

[](af://n472/)***swagger***测试接口



### 3.4.2 查询班级信息【分页带条件】

*Controller*层代码

@Api(tags = "班级控制器")

@RestController

@RequestMapping("/sms/clazzController")

public class ClazzController {

@Autowired

private ClazzService clazzService;

@ApiOperation("查询班级信息 ,分页带条件")

@GetMapping("/getClazzsByOpr/{pageNo}/{pageSize}")

public Result getClazzsByOpr(

@ApiParam("页码数") @PathVariable("pageNo") Integer

pageNo,

@ApiParam("页大小") @PathVariable("pageSize") Integer

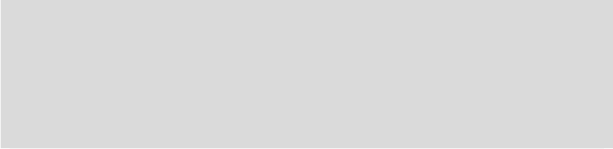
pageSize,

@ApiParam("查询条件") Clazz clazz

){

//设置分页信息

Page<Clazz> page =new Page<>(pageNo,pageSize);



IPage<Clazz> iPage = clazzService.getClazzsByOpr(page, clazz);

return Result.ok(iPage);

}

}

*service*层代码

@Service

@Transactional

public class ClazzServiceImpl extends ServiceImpl<ClazzMapper, Clazz> implements ClazzService {

/\*\*

\* 分页查询所有班级信息【带条件】

\* @param clazz

\* @return \*/

@Override

public IPage<Clazz> getClazzsByOpr(Page<Clazz> pageParam,Clazz clazz) {

QueryWrapper queryWrapper = new QueryWrapper(); if(clazz != null){

//年级名称条件

String gradeName = clazz.getGradeName();

if(!StringUtils.isEmpty(gradeName)){

queryWrapper.eq("grade\_name",gradeName); }

//班级名称条件

String clazzName = clazz.getName();

if(!StringUtils.isEmpty(clazzName)){

queryWrapper.like("name",clazzName);

}

queryWrapper.orderByDesc("id");

queryWrapper.orderByAsc("name");

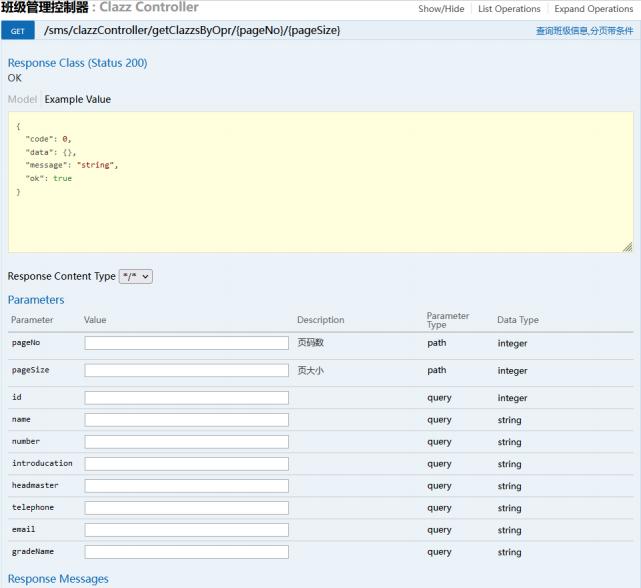
}

Page page = baseMapper.selectPage(pageParam, queryWrapper); return page;

}

}

*swagger*测试接口



3.4.3 添加和修改学生信息

 *Controller*层代码

@Api(tags = "班级控制器")

@RestController

@RequestMapping("/sms/clazzController")

public class ClazzController {

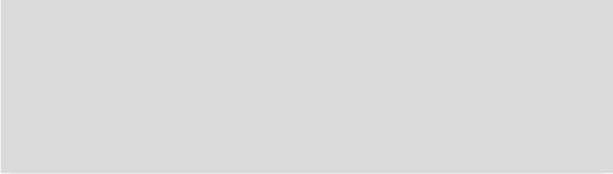
@Autowired

private ClazzService clazzService;

@ApiOperation("保存或者修改班级信息")

@PostMapping("/saveOrUpdateClazz")

public Result saveOrUpdateClazz(



@ApiParam("JSON转换后端Clazz数据模型") @RequestBody Clazz

clazz

){

clazzService.saveOrUpdate(clazz);

return Result.ok();

}

}

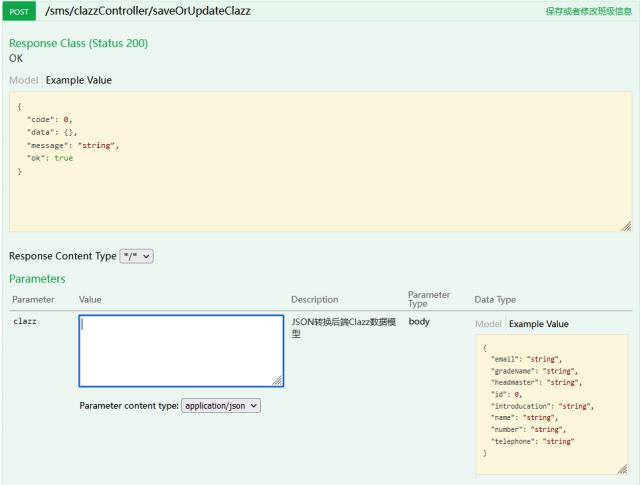
*service*层代码

见IService

*mapper*层代码

略

*swagger*测试接口



### 3.4.4 删除和批量删除班级信息

 *Controller*层代码

@Api(tags = "班级控制器")

@RestController

@RequestMapping("/sms/clazzController")

public class ClazzController {

@Autowired

private ClazzService clazzService;

@ApiOperation("删除一个或者多个班级信息")

@DeleteMapping("/deleteClazz")

public Result deleteClazzByIds(

@ApiParam("多个班级id的JSON") @RequestBody List<Integer>

ids

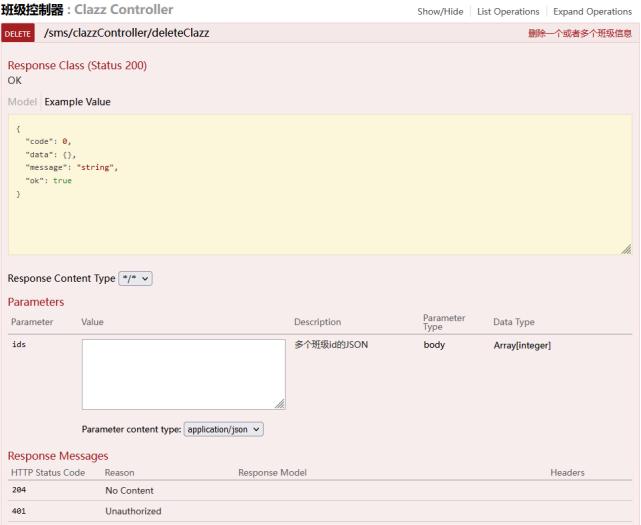
){

clazzService.removeByIds(ids);

return Result.ok();

}

}



## 3.5 学生管理功能实现

### 3.5.1 回显搜索条件中的班级选项

*Controller*层代码

@Api(tags = "班级控制器")

@RestController

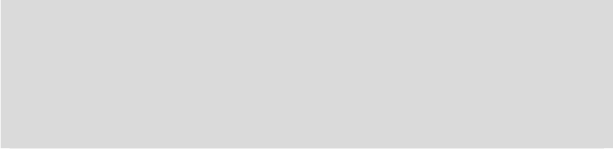
@RequestMapping("/sms/clazzController")

public class ClazzController {

@Autowired

private ClazzService clazzService;

@ApiOperation("获取所有班级的JSON")



@GetMapping("/getClazzs")

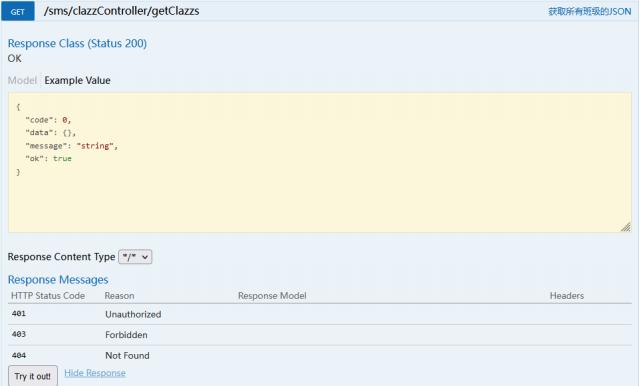
public Result getClazzs(){

List<Clazz> clazzList = clazzService.getClazzs(); return Result.ok(clazzList);

}

}

*swagger*测试接口



### 3.5.2 查询学生信息【分页带条件】

*Controller*层代码

@Api(tags = "学生控制器")

@RestController

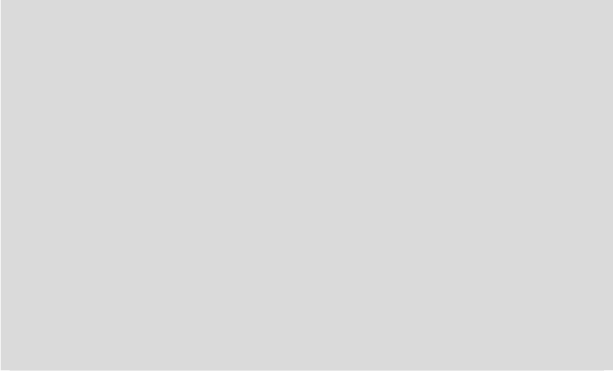
@RequestMapping("/sms/studentController")

public class StudentController {

@Autowired

private StudentService studentService;

@ApiOperation("查询学生信息 ,分页带条件") @GetMapping("/getStudentByOpr/{pageNo}/{pageSize}") public Result getStudentsByOpr(



@ApiParam("页码数") @PathVariable("pageNo")Integer

pageNo,

@ApiParam("页大小") @PathVariable("pageSize")Integer

pageSize,

@ApiParam("查询条件转换后端数据模型") Student student ){

// 准备分页信息封装的page对象

Page<Student> page =new Page<>(pageNo,pageSize);

// 获取分页的学生信息

IPage<Student> iPage = studentService.getStudentByOpr(page, student);

// 返回学生信息

return Result.ok(iPage);

}

}

*service*层代码

@Service("stuService")

@Transactional

public class StudentServiceImpl extends ServiceImpl<StudentMapper, Student> implements StudentService {

/\*\*

\* 按条件查询学生信息【带分页】

\*/

public IPage<Student> getStudentByOpr(Page<Student> pageParam,Student student){

QueryWrapper<Student> queryWrapper = null;

if(student != null) {

queryWrapper = new QueryWrapper<>();

if (student.getClazzName() != null) {

queryWrapper.eq("clazz\_name",

student.getClazzName());

}

if (student.getName() != null) {

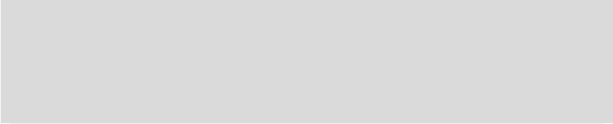
queryWrapper.like("name", student.getName()); }

queryWrapper.orderByDesc("id");

queryWrapper.orderByAsc("name");

}

//创建分页对象

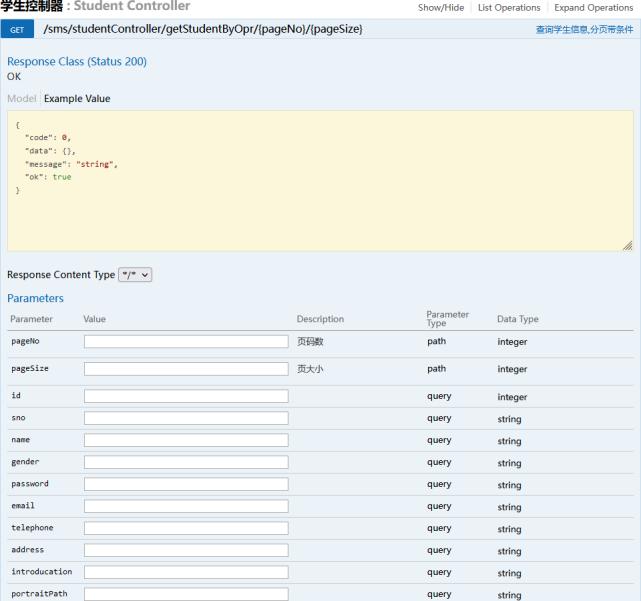


IPage<Student> pages = baseMapper.selectPage(pageParam, queryWrapper);

return pages;

}

*swagger*测试接口



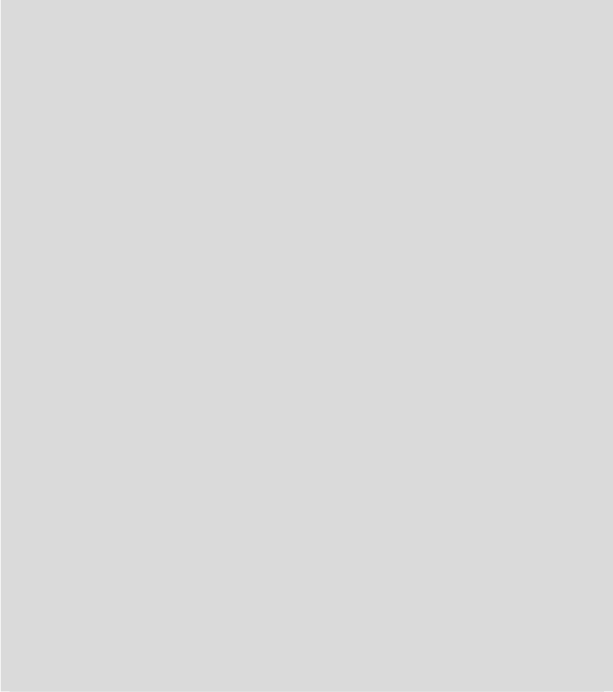
### 3.5.3 异步图片上传处理头像

*Controller*层代码

@Api(tags = "系统控制器")

@RestController

@RequestMapping("/sms/system")

public class SystemController {

@ApiOperation("头像上传统一入口")

@P("/headerImgUpload")

public Result headerImgUpload(

@ApiParam("文件二进制数据") @RequestPart("multipartFile")

MultipartFile multipartFile

){

//使用UUID随机生成文件名

String uuid = UUID.randomUUID().toString().replace("-", "").toLowerCase();

//生成新的文件名字

String filename =

uuid.concat(multipartFile.getOriginalFilename());

//生成文件的保存路径(实际生产环境这里会使用真正的文件存储服务器)

String portraitPath

="c:/code/zhxy/target/classes/public/upload/".concat(filename);

//保存文件

try {

multipartFile.transferTo(new File(portraitPath)); } catch (IOException e) {

e.printStackTrace();

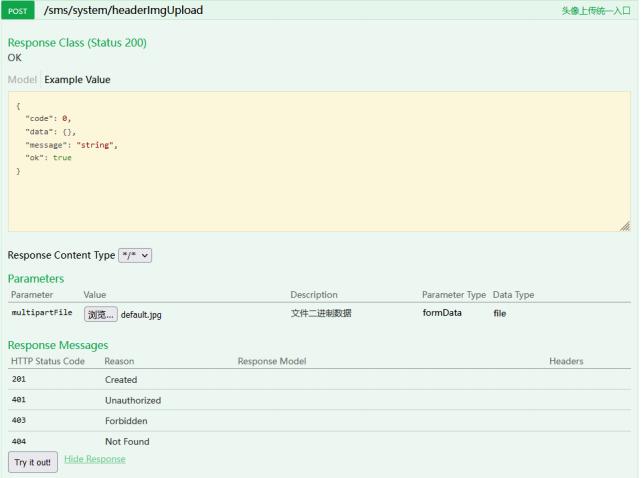
}

String headerImg ="upload/"+filename;

return Result.ok(headerImg);

}

}



### 3.5.4 添加和修改学生信息

 *Controller*层代码

@Api(tags = "学生控制器")

@RestController

@RequestMapping("/sms/studentController")

public class StudentController {

@Autowired

private StudentService studentService;

@ApiOperation("增加学生信息")

@PostMapping("/addOrUpdateStudent")

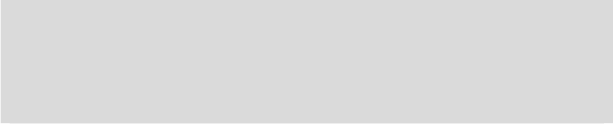
public Result addOrUpdateStudent(@RequestBody Student student){

//对学生的密码进行加密

if (!Strings.isEmpty(student.getPassword())) {

student.setPassword(MD5.encrypt(student.getPassword()));

}



//保存学生信息进入数据库

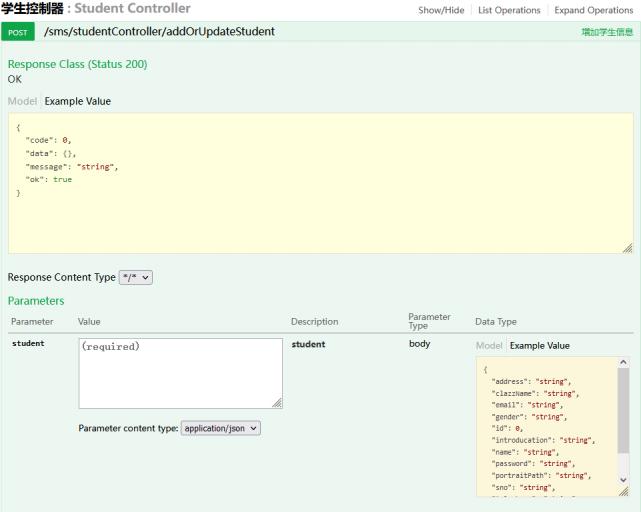
studentService.saveOrUpdate(student);

return Result.ok();

}

}

*swagger*测试接口



### 3.5.6 删除和批量删除学生信息

 *Controller*层代码

@Api(tags = "学生控制器")

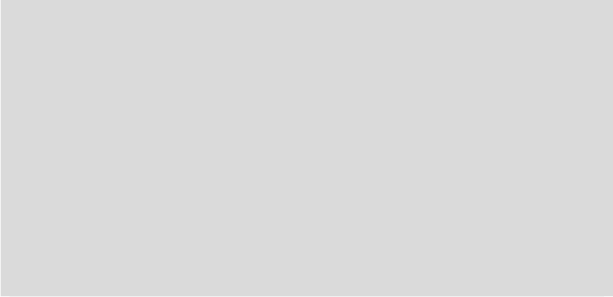
@RestController

@RequestMapping("/sms/studentController")

public class StudentController {

@Autowired

private StudentService studentService;



@ApiOperation("删除一个或者多个学生信息")

@DeleteMapping("/delStudentById")

public Result delStudentById(

@ApiParam("多个学生id的JSON") @RequestBody List<Integer>

ids

){

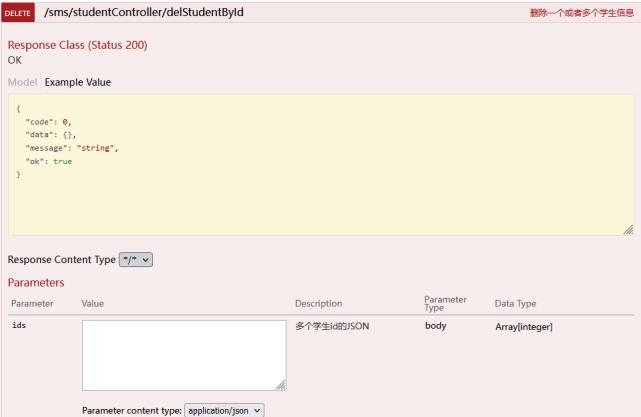
studentService.removeByIds(ids);

return Result.ok();

}

}

*swagger*测试接口



## 3.6 教师管理功能实现

3.6.1 查询教师信息【分页带条件】

@Api(tags = "教师信息管理控制器")

@RestController

@RequestMapping("/sms/teacherController")

public class TeacherController {

@Autowired

private TeacherService teacherService;

@ApiOperation("获取教师信息 ,分页带条件")

@GetMapping("/getTeachers/{pageNo}/{pageSize}")

public Result getTeachers(

@PathVariable("pageNo") Integer pageNo, @PathVariable("pageSize") Integer pageSize, Teacher teacher

){

Page<Teacher> pageParam = new Page<>(pageNo,pageSize); IPage<Teacher> page =

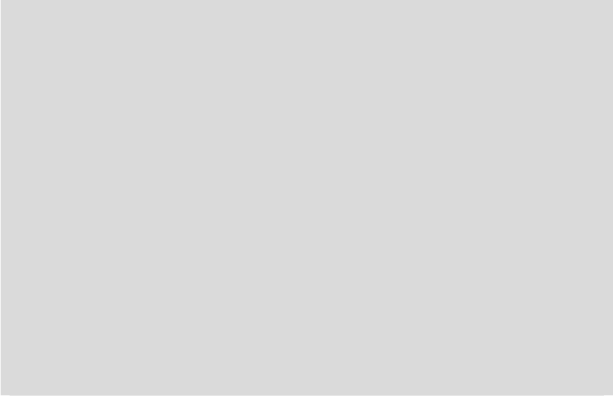
teacherService.getTeachersByOpr(pageParam, teacher);

return Result.ok(page);

}

}

*service*层代码



queryWrapper.eq("clazz\_name",clazzName); }

//教师名称条件

String teacherName = teacher.getName(); if(!StringUtils.isEmpty(teacherName)){

queryWrapper.like("name",teacherName); }

queryWrapper.orderByDesc("id");

queryWrapper.orderByAsc("name");

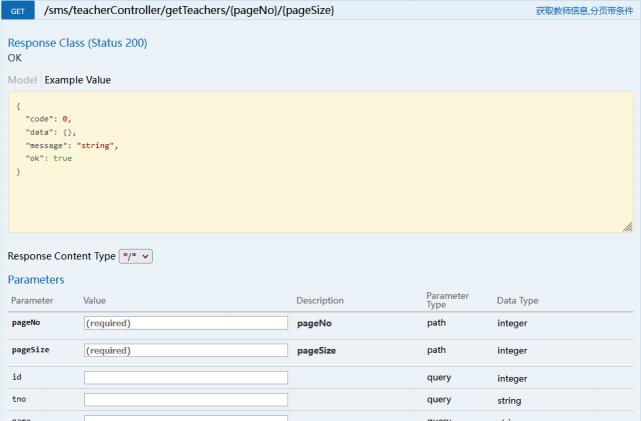
}

IPage<Teacher> page = baseMapper.selectPage(pageParam, queryWrapper);

return page;

}

*swagger*测试接口



### 3.6.2 添加和修改教师信息

*Controller*层代码

@ApiOperation("添加和修改教师信息")

@PostMapping("/saveOrUpdateTeacher")

public Result saveOrUpdateTeacher(

@RequestBody Teacher teacher

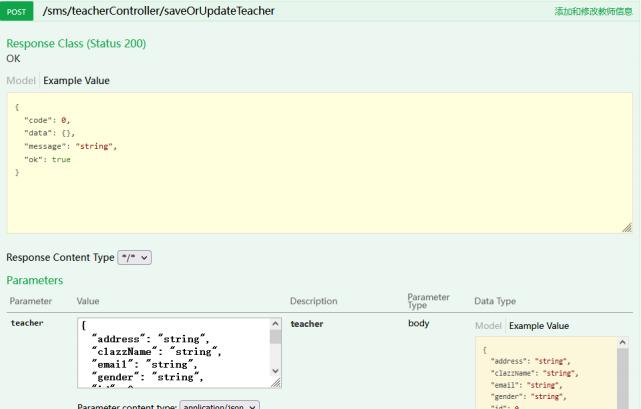
){

teacherService.saveOrUpdate(teacher);

return Result.ok();

}

*swagger*测试接口



### 3.6.3 删除和批量删除教师信息

 *Controller*层代码

@ApiOperation("删除一个或者多个教师信息")

@DeleteMapping("/deleteTeacher")

public Result deleteTeacher(

@RequestBody List<Integer> ids

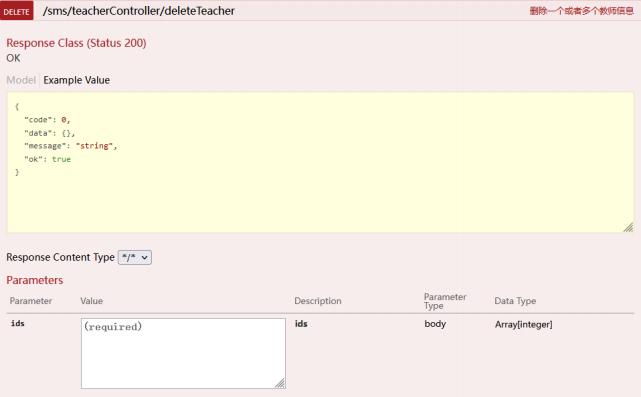
){

teacherService.removeByIds(ids);

return Result.ok();

}

*swagger*测试接口



## 3.7管理员管理功能实现

### 3.7.1 查询管理员信息【分页带条件】

*Controller*层代码

@Api(tags = "系统管理员控制器")

@RestController

@RequestMapping("/sms/adminController")

public class AdminController {

@Autowired

private AdminService adService;

@ApiOperation("分页获取所有Admin信息【带条件】")

@GetMapping("/getAllAdmin/{pageNo}/{pageSize}")

public Result getAllAdmin(@PathVariable Integer pageNo, @PathVariable Integer pageSize, String adminName){

Page<Admin> pageParam = new Page<>(pageNo,pageSize);

IPage<Admin> page = adService.getAdmins(pageParam, adminName);

return Result.ok(page);

}

}

*service*层代码

@Override

public IPage<Admin> getAdmins(Page<Admin> pageParam,String adminName) {

QueryWrapper queryWrapper = new QueryWrapper();

if(!StringUtils.isEmpty(adminName)){

queryWrapper.like("name",adminName);

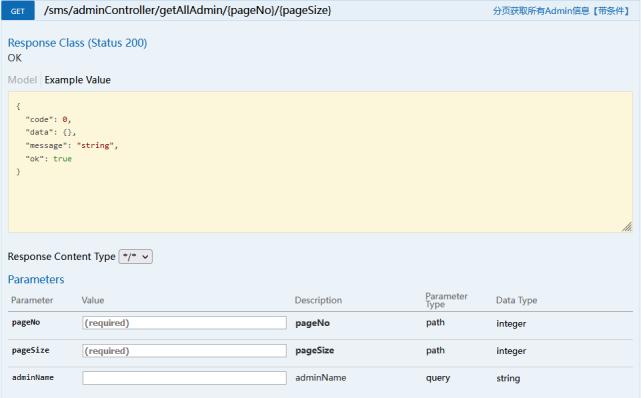
}

queryWrapper.orderByDesc("id");

queryWrapper.orderByAsc("name");

Page page = baseMapper.selectPage(pageParam, queryWrapper); return page;

}



### 3.7.2 添加和修改管理员信息

*Controller*层代码

@ApiOperation("添加或修改Admin信息")

@PostMapping("/saveOrUpdateAdmin")

public Result saveOrUpdateAdmin(@RequestBody Admin admin){ if (!Strings.isEmpty(admin.getPassword())) {

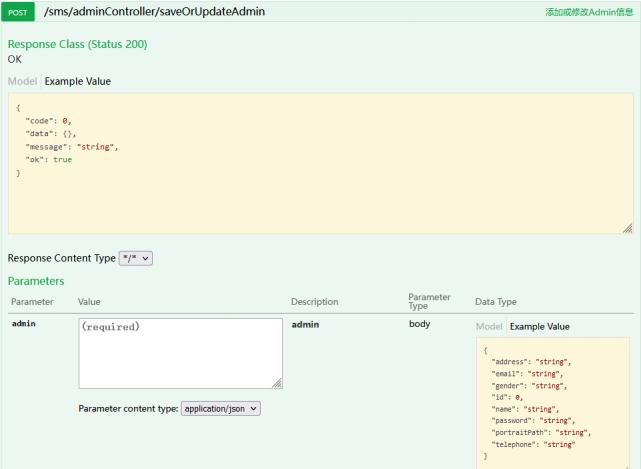
admin.setPassword(MD5.encrypt(admin.getPassword())); }

adService.saveOrUpdate(admin);

return Result.ok();

}

*swagger*测试接口



### 3.7.3 删除和批量删除管理员信息

 *Controller*层代码

@ApiOperation("删除Admin信息")

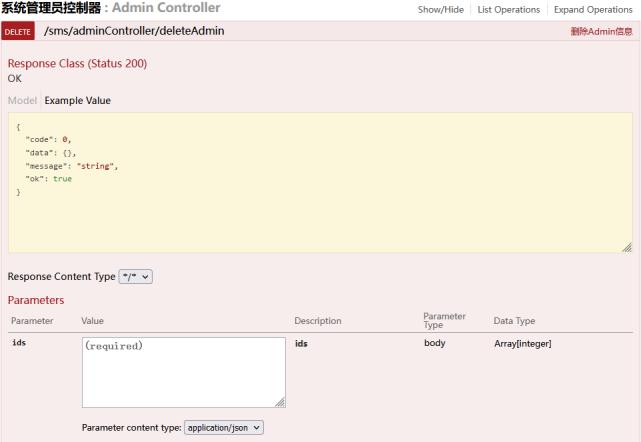
@DeleteMapping("/deleteAdmin")

public Result deleteAdmin(@RequestBody List<Integer> ids){ adService.removeByIds(ids);

return Result.ok();

}

*swagger*测试接口



### 3.7.4 管理员修改自己的密码

*Controller*层代码

@Api(tags = "系统控制器")

@RestController

@RequestMapping("/sms/system")

public class SystemController {

@ApiOperation("修改密码")

@PostMapping("/updatePwd/{oldPwd}/{newPwd}")

public Result updatePwd(@RequestHeader("token") String token, @PathVariable("oldPwd") String oldPwd, @PathVariable("newPwd") String newPwd){

boolean yOn = JwtHelper.isExpiration(token);

if(yOn){

//token过期

return Result.fail().message("token失效!");

}

//通过token获取当前登录的用户id

Long userId = JwtHelper.getUserId(token);

//通过token获取当前登录的用户类型

Integer userType = JwtHelper.getUserType(token);

// 将明文密码转换为暗文

oldPwd=MD5.encrypt(oldPwd);

newPwd= MD5.encrypt(newPwd);

if(userType == 1){

QueryWrapper<Admin> queryWrapper=new QueryWrapper<>();

queryWrapper.eq("id",userId.intValue()).eq("password",oldPwd); Admin admin = adService.getOne(queryWrapper); if (null !=admin) {

admin.setPassword(newPwd);

adService.saveOrUpdate(admin);

}else{

return Result.fail().message("原密码输入有误！"); }

}else if(userType == 2){

QueryWrapper<Student> queryWrapper=new QueryWrapper<>

();

queryWrapper.eq("id",userId.intValue()).eq("password",oldPwd); Student student = studentService.getOne(queryWrapper); if (null !=student) {

student.setPassword(newPwd);

studentService.saveOrUpdate(student);

}else{

return Result.fail().message("原密码输入有误！"); }

}

else if(userType == 3){

QueryWrapper<Teacher> queryWrapper=new QueryWrapper<>

();

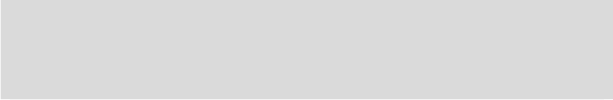
queryWrapper.eq("id",userId.intValue()).eq("password",oldPwd); Teacher teacher = teacherService.getOne(queryWrapper); if (null !=teacher) {

teacher.setPassword(newPwd);

teacherService.saveOrUpdate(teacher);

}else{

return Result.fail().message("原密码输入有误！"); }

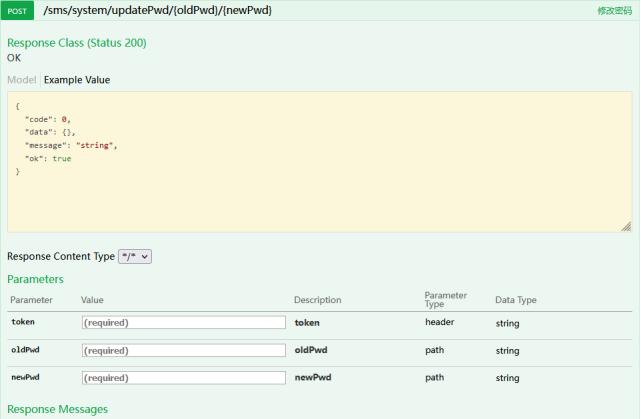


}

return Result.ok(); }

}

*swagger*测试接口



# 4.课程设计总结

在此次设计中遇到了很多问题，但经过一次又一次的思考，一遍又一遍的检查终于找出了原因所在，也暴露出了前期我在这方面的知识欠缺和经验不足。在今后社会的发展和学习实践过程中，一定要不懈努力，不能遇到问题就想到要退缩，一定要不厌其烦的发现问题所在，然后一一进行解决，只有这样，才能成功的做成想做的事，才能在今后的道路上劈荆斩棘，而不是知难而退。

此外，对程序的测试应该要仔细，根据模块的特点和测试阶段，采用各种软件测试方法对程序进行测试，确保各个模块的正确性和完整性，最后集成起来测试其是否正确和完整地实现了问题描述中要求的功能。

在这次课程设计中，我学到了很多的知识，在对Java框架的学习上是我学会了怎么将学到的理论知识用到实践中去，改变了过去的学习方法和学习态度，找到了一门自己很喜欢的学科。

