**作业提交系统代码清单**

V1.0

**评审日期：2019年7月4日**

**编写人员：梁艺翔**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号 | 修改人/日期 | 摘要 | 审定人/日期 |
| V1.0 | 梁艺翔/2019.7.4 | 初始发布版本 | 韦杨琳/2019.7.4 |
|  |  |  |  |
|  |  |  |  |

目录

[一、 功能说明 4](#_Toc17082)

[二、 环境说明 4](#_Toc27298)

[2.1、硬件环境 4](#_Toc3645)

[2.2、软件环境 4](#_Toc23910)

[三、 设计说明 4](#_Toc24111)

[四、源码清单 5](#_Toc31865)

[4.1、登陆模块 5](#_Toc15712)

[4.11、Login.ejs 5](#_Toc28235)

[4.12、AuthApi 15](#_Toc27608)

[4.2、作业管理模块 23](#_Toc16570)

[4.21、Exam.ejs 23](#_Toc7719)

[4.22、ExamApi.java 34](#_Toc25492)

[4.3、学院管理模块 55](#_Toc25696)

[4.31、AcademyInfo.ejs 55](#_Toc28727)

[4.32、AcademyInfoApi.java 59](#_Toc23985)

[4.4、专业管理模块 69](#_Toc28471)

[4.41、MajorInfo.ejs 69](#_Toc17910)

[4.42、MajorInfoApi.java 73](#_Toc5171)

[4.5、题库管理模块 83](#_Toc5313)

[4.51、Question.ejs 83](#_Toc26936)

[4.52、QuestionApi.java 94](#_Toc24993)

[4.6、课程管理模块 118](#_Toc28266)

[4.61、Course.ejs 118](#_Toc8118)

[4.62、CourseApi.java 120](#_Toc6091)

[4.7、教师模块管理 124](#_Toc4819)

[4.71、Teacher.ejs 124](#_Toc8786)

[4.72、TeacherApi.java 127](#_Toc12989)

[4.8用户模块管理 131](#_Toc17081)

[4.81、User.ejs 131](#_Toc5442)

[4.82、SysUserApi.java 134](#_Toc929)

## 功能说明

本系统按照功能主要分为登陆模块、作业管理模块、学生管理模块、专业管理模块、题库管理模块、课程管理模块、教师模块管理、用户管理模块。

1. 登陆模块主要是管理员、老师和学生的登陆。
2. 作业管理模块主要是新增作业、修改作业、发布作业、删除作业、批改作业等内容。
3. 学生管理模块主要是对学生信息的增、删、查、改操作。
4. 专业管理模块主要是对专业信息的增、删、查、改操作。
5. 题库管理模块主要是对题库信息的增、删、查、改操作。
6. 课程管理模块主要是对课程信息的增、删、查、改操作还有给老师分配课程。
7. 教师管理模块主要是对教师身份信息的增、删、查、改操作

## 环境说明

### 2.1、硬件环境

PC机：一台

CPU：1G Hz以上

内存：1GB以上

硬盘：10GB上

处理器：100M Hz以上

### 2.2、软件环境

操作系统：支持在windows7以上 64位 环境下安全运行

服务器：Tomcat

开发平台：Intellij IDEA

数据库：mysql5.6

## 设计说明

表现层

用户看到并与客户进行交互的界面，使用js+ejs技术

控制层

进行业务逻辑判断和数据库存取，使用Spring Controller组件

业务逻辑层

将用户输入的指令和数据传递给业务模型，使用Spring Service组件

数据持久层

使用MySQL数据库引擎，使用MyBaties组件

域模型层

整个系统的核心层，维护一个使用面向对象技术实现的领域模型

## 四、源码清单

### 4.1、登陆模块

#### 4.11、Login.ejs

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>

<meta http-equiv="pragma" content="no-cache">

<meta http-equiv="Cache-Control" content="no-cache, must-revalidate">

<meta http-equiv="expires" content="0">

<title>作业管理系统登录</title>

<link rel="stylesheet" href="login/css/index\_style.css">

</head>

<body onLoad="javascript:document.getElementById('optForm').reset()">

<form way-data="optForm" id="optForm" name="optForm" autocomplete="off" way-persistent="true"

class="container offset1 loginform">

<div class="main">

<input type="hidden" name="\_csrf" value="9IAtUxV2CatyxHiK2LxzOsT6wtBE6h8BpzOmk=">

<div class="main-center">

<div class="main-txt">

账号登录

</div>

<div class="line">

<div class="line-img">

<img src="login/images/user.png" width="100%">

</div>

<input autocomplete="off" id="loginName" name="loginName" class="required"

style=" border-top: 1px solid #ccc;border-bottom: 1px solid #ccc;" placeholder="用户名" size="34"

type="text" tabindex="1">

</div>

<div class="line" style="position: relative">

<div class="line-img">

<img src="login/images/lock.png" width="100%">

</div>

<input autocomplete="off" id="password" name="password" class="required" placeholder="密码" size="34"

type="password" tabindex="2">

<div class="password-hidden" id="passwordeye"></div>

</div>

<div style="width: 306px;height: 40px">

<div class="line-cold">

<div class="line-img">

<img src="login/images/validation.png" width="100%">

</div>

<input autocomplete="off" id="code" class="cold" name="code" placeholder="验证码" size="34" type="text"

tabindex="3" maxlength="4" autocomplete="off">

</div>

<img src="assets/images/min-loading.gif" style="margin-top: 16px;" width="100" id="verify-loading"/>

<span id="verify-code" style="height: 40px">

<canvas width="100" height="35" id="canvas" class="yanzm"></canvas>

</span>

</div>

<div style="float: right;margin-top:10px;cursor:pointer;color:gray" onclick=""></div>

<div class="button">

<pre style="display: block;float: left;color:red;" id="login-message"></pre>

<button id="sys-login" type="button" autocomplete="off" data-loading-text="认证中..." tabindex="4"

class="button-style">登录

</button>

</div>

</div>

</div>

</form>

<script src="assets/js/jquery.min.js"></script>

<script>

//判断JS是否支持

//console.log(navigator);

if (!window.jQuery) {

alert("您当前正在使用" + navigator.appName + "浏览器，因安全考虑，本站暂不支持IE10以下内核浏览器访问；请使用IE10+内核浏览器、QQ浏览器、谷歌浏览器、火狐浏览器、Opera浏览器、360极速浏览器最新版进行访问！");

} else {

if (navigator.appName == "Microsoft Internet Explorer") {

if (navigator.appVersion.match(/7./i) == '6.' ||

navigator.appVersion.match(/7./i) == '7.' ||

navigator.appVersion.match(/7./i) == '8.' ||

navigator.appVersion.match(/7./i) == '9.') {

alert("您当前正在使用" + navigator.appName + "浏览器，因安全考虑，本站暂不支持IE10以下内核浏览器访问；请使用IE10+内核浏览器、QQ浏览器、谷歌浏览器、火狐浏览器、Opera浏览器、360极速浏览器最新版进行访问！");

}

}

}

</script>

<script src="assets/js/bootstrap.min.js?v=3.3.6"></script>

<script src="assets/js/config.js"></script>

<script src="assets/js/way.js"></script>

<script src="assets/js/util/store.min.js"></script>

<script src="assets/js/routes.js"></script>

<script src="assets/js/util/jsencrypt.min.js"></script>

<script>

var server\_root = getRootPath();

function getRootPath() {

if (server\_root != null && server\_root != "") {

return server\_root;

} else {

var host = window.location.host;

var origin = window.location.origin;

if (!origin) {

origin = window.location.protocol + "//" + window.location.hostname + (window.location.port ? ':' + window.location.port : '');

}

var root = '/';

switch (host) {

case "gxmddj.gxun.edu.cn":

case "10.240.16.116":

case "mengwp.gxun.edu.cn":

root = origin + '/party-admin/';

break;

default:

root = origin + '/';

break;

}

return root;

}

}

//显示隐藏对应的switchPwd()方法:

function switchPwd() {

var passwordeye = $('#passwordeye');

var showPwd = $("#password");

passwordeye.off('click').on('click', function () {

if (passwordeye.hasClass('password-hidden')) {

passwordeye.removeClass('password-hidden').addClass('password-show');//密码可见

showPwd.prop('type', 'text');

} else {

passwordeye.removeClass('password-show').addClass('password-hidden');//密码不可见

showPwd.prop('type', 'password');

}

;

});

}

switchPwd();

var verifyToken = "";

var $btnLogin;

$(function () {

document.getElementById('optForm').reset();

$('#loginName').val('');

$('#password').val('');

$('#code').val('');

if (getVal('password').length != 0) {

window.location.reload();

}

//验证码

verifyCode();

//todo:后台图片验证码

// verifyCodePic();

//密码遮挡效果

$('#login #password').focus(function () {

$('#owl-login').addClass('password');

}).blur(function () {

$('#owl-login').removeClass('password');

});

//回车事件

document.onkeydown = function (e) {

var ev = document.all ? window.event : e;

if (ev.keyCode == 13) {

login();

}

}

//自动登录

var object = store.get(cache\_user\_key);

if ((typeof object) == 'object') {

window.location.href = server\_root + 'sys';

}

$btnLogin = $('#sys-login');

$btnLogin.click(function () {

login();

})

});

function verifyCode() {

getVerifyCode();

$('#verify-code').click(function () {

getVerifyCode();

})

}

function verifyCodePic() {

getVerifyCodePic();

$('#verify-code').click(function () {

getVerifyCodePic();

})

}

function loading(flag) {

if (flag) {

$btnLogin.button('loading')

} else {

$btnLogin.button('reset')

}

}

//验证码效果处理

function verifyLoading(flag) {

if (flag) {

$('#verify-loading').show();

$('#verify-code').hide();

} else {

$('#verify-loading').hide();

$('#verify-code').show();

}

}

function getVerifyCode() {

verifyLoading(true);

$.ajax({

type: 'GET',

url: commonModule.verifyCodeApi + "?x=" + Date.parse(new Date()),

headers: {

"Access-Control-Allow-Origin": "\*",

"token": verifyToken

},

error: function (d) {

alert("验证码获取出错！"+d.message);

},

success: function (d) {

var data = d.result;

verifyToken = data.token;

drawPic(data.code)

setTimeout("verifyLoading(false);", 1000);

}

});

}

function getVerifyCodePic() {

verifyLoading(true);

$.ajax({

type: 'GET',

url: commonModule.verifyCodePicApi + "?x=" + Date.parse(new Date()),

headers: {

"Access-Control-Allow-Origin": "\*",

"token": verifyToken

},

dataType: "json",

error: function (d) {

alert("验证码获取出错！"+d.message);

},

success: function (d) {

var data = d.result;

var src = data.jpg;

verifyToken = data.token;

var img = document.createElement("img");

var canvas = document.getElementById("canvas");

var width = canvas.width;

var height = canvas.height;

img.width = width;

img.height = height;

document.body.appendChild(img);

img.onload = function () {

var ctx = canvas.getContext("2d")

ctx.drawImage(this, 0, 0, width, height)

setTimeout("verifyLoading(false);", 1000);

document.body.removeChild(img);

}

img.src = src;

}

});

}

function login() {

var data = way.get().optForm;

if (getVal('loginName').length == 0) {

setMessage('请输入登录名');

$('#loginName').focus();

return false;

}

if (getVal('password').length == 0) {

setMessage('请输入登录密码');

$('#password').focus();

return false;

}

if (getVal('code').length == 0) {

setMessage('请输入验证码');

$('#code').focus();

return false;

}

$.ajax({

type: 'GET',

url: commonModule.getPublicKeyApi + "?x=" + Date.parse(new Date()),

headers: {

"Access-Control-Allow-Origin": "\*",

"token": verifyToken

},

async: false,

error: function (d) {

setMessage(d.message);

return false;

},

success: function (d) {

var key = d.result;

var Encrypt = new JSEncrypt();

Encrypt.setPublicKey(key);

data['password'] = Encrypt.encrypt(data['password']);

}

});

loading(true);

var request = $.ajax({

type: 'POST',

url: commonModule.sysLoginApi,

headers: {token: verifyToken},

data: JSON.stringify(data),

contentType: "application/json; charset=utf-8",

dataType: "json",

error: function (e) {

var res = e.responseJSON;

setMessage(res.message);

loading(false);

getVerifyCode();

$('#password').focus();

$('#password').val('');

$('#code').val('');

}

});

request.done(function (res) {

loading(false);

way.clear();

$('#loginName').val('');

$('#password').val('');

$('#code').val('');

data['password'] = '';

store.set(cache\_user\_key, res.result);

window.location.href = server\_root + 'sys';

})

}

function getVal(\_id) {

var id = '#' + \_id;

return $(id).val().trim();

}

function setMessage(\_message) {

$('#login-message').html(\_message);

}

/\*\*\*\*\*\*\*\*验证码\*\*\*\*\*\*\*/

/\*\*生成一个随机数\*\*/

function randomNum(min, max) {

return Math.floor(Math.random() \* (max - min) + min);

}

/\*\*生成一个随机色\*\*/

function randomColor(min, max) {

var r = randomNum(min, max);

var g = randomNum(min, max);

var b = randomNum(min, max);

return "rgb(" + r + "," + g + "," + b + ")";

}

/\*\*绘制验证码图片\*\*/

function drawPic(\_verifyCode) {

var canvas = document.getElementById("canvas");

var width = canvas.width;

var height = canvas.height;

var ctx = canvas.getContext('2d');

ctx.textBaseline = 'bottom';

/\*\*绘制背景色\*\*/

ctx.fillStyle = randomColor(180, 240); //颜色若太深可能导致看不清

ctx.fillRect(0, 0, 100, 35);

/\*\*绘制文字\*\*/

var str = \_verifyCode;

for (var i = 0; i < \_verifyCode.length; i++) {

var txt = \_verifyCode.charAt(i);

ctx.fillStyle = randomColor(50, 160); //随机生成字体颜色

ctx.font = randomNum(30, 30) + 'px SimHei'; //随机生成字体大小

var x = 10 + i \* 20;

var y = randomNum(25, 30);

var deg = randomNum(-10, 45);

// 修改坐标原点和旋转角度

ctx.translate(x, y);

ctx.rotate(deg \* Math.PI / 180);

ctx.fillText(txt, 0, 0);

//恢复坐标原点和旋转角度

ctx.rotate(-deg \* Math.PI / 180);

ctx.translate(-x, -y);

}

/\*\*绘制干扰线\*\*/

for (var i = 0; i < 2; i++) {

ctx.strokeStyle = randomColor(40, 180);

ctx.beginPath();

ctx.moveTo(randomNum(0, width), randomNum(0, height));

ctx.lineTo(randomNum(0, width), randomNum(0, height));

ctx.stroke();

}

/\*\*绘制干扰点\*\*/

for (var i = 0; i < 30; i++) {

ctx.fillStyle = randomColor(0, 255);

ctx.beginPath();

ctx.arc(randomNum(0, width), randomNum(0, height), 1, 0, 2 \* Math.PI);

ctx.fill();

}

}

//读取权限

function auth(\_token) {

loading(true);

var request = $.ajax({

type: 'GET',

url: commonModule.sysAuthApi + '/' + client,

headers: {token: \_token},

contentType: "application/json; charset=utf-8",

dataType: "json",

error: function (e) {

loading(false);

var res = e.responseJSON;

setMessage(res.message);

}

});

request.done(function (res) {

loading(false);

store.set(cache\_user\_key, res.result);

window.location.href = server\_root + 'sys';

})

}

function forgetpassword() {

window.location.href = server\_root + 'forgetPassword';

}

</script>

</body>

</html>

#### 4.12、AuthApi

package team.work.api.unified;

import com.alibaba.fastjson.JSONObject;

import com.google.code.kaptcha.Producer;

import team.work.core.model.\*;

import team.work.core.service.impl.\*;

import team.work.utils.convert.\*;

import team.work.utils.tool.SMS\_func;

import team.work.doc.ForgetPassword;

import team.work.utils.tool.RSAEncrypt;

import team.work.utils.convert.\*;

import team.work.core.async.TokenAsync;

import team.work.core.service.impl.\*;

import team.work.doc.SysLogin;

import team.work.api.BaseApi;

import team.work.utils.kit.MD5Kit;

import team.work.utils.kit.TimeKit;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

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

import org.springframework.http.HttpEntity;

import org.springframework.http.HttpHeaders;

import org.springframework.http.HttpMethod;

import org.springframework.http.ResponseEntity;

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

import org.springframework.web.client.RestClientException;

import org.springframework.web.client.RestTemplate;

import org.springframework.web.context.request.RequestContextHolder;

import org.springframework.web.context.request.ServletRequestAttributes;

import sun.misc.BASE64Encoder;

import team.work.core.model.\*;

import team.work.doc.SysLogin;

import javax.annotation.Resource;

import javax.imageio.ImageIO;

import javax.servlet.http.HttpServletRequest;

import java.awt.image.BufferedImage;

import java.io.ByteArrayOutputStream;

import java.io.IOException;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

import java.util.Map;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

@Api(value = "系统授权管理")

@RestController

@RequestMapping("/v1/common")

@CrossOrigin //跨域服务注解

public class AuthApi extends BaseApi {

@Autowired

private TokenAsync tokenAsync;

@Autowired

public SysUserService userService;

@Autowired

public SysUserService sysUserService;

@Autowired

private SysRoleService sysRoleService;

@Autowired

private SysUserRoleService userRoleService;

@Autowired

private AcademyInfoService academyInfoService;

@Autowired

private SysOperationRecordService sysOperationRecordService;

@GetMapping("/getPublicKey")

@ApiOperation(value = "获取登陆公钥")

public Object getPublicKey(@RequestHeader("token") String token) {

Map<Integer, String> keyMap = RSAEncrypt.genKeyPair();

cacheKit.setVal(token + "\_privateKey", keyMap.get(1), 0);

return R.ok("获取登陆公钥成功", keyMap.get(0));

}

@GetMapping("/verify-code")

@ApiOperation(value = "获取登录验证码")

public Object getSysVerifyCode(HttpServletRequest request) {

JSONObject verify = new JSONObject();

String ip = getIpAddr(request);

String token = "";

String code = S.random(4);

token = S.getToken();

verify.put("ip", ip);

verify.put("code", code);

verify.put("token", token);

//cacheKit.setVal(ip, token, 120);

cacheKit.setVal(token, J.o2s(verify), 120);

return R.ok("获取验证码成功", verify);

}

@Autowired

private Producer captchaProducer;

@GetMapping("/verify-code/jpg")

@ApiOperation(value = "获取登录验证码")

public Object getSysVerifyCodePic(HttpServletRequest request) {

String capText = captchaProducer.createText();

JSONObject verify = new JSONObject();

String ip = getIpAddr(request);

String token = "";

token = S.getToken();

verify.put("ip", ip);

verify.put("code", capText);

verify.put("token", token);

// cacheKit.setVal(ip, token, 120);

cacheKit.setVal(token, J.o2s(verify), 120);

JSONObject res = new JSONObject();

res.put("token", token);

try {

BufferedImage bi = captchaProducer.createImage(capText);

ByteArrayOutputStream baos = new ByteArrayOutputStream();//io流

ImageIO.write(bi, "jpg", baos);//写入流中

byte[] bytes = baos.toByteArray();//转换成字节

BASE64Encoder encoder = new BASE64Encoder();

String png\_base64 = encoder.encodeBuffer(bytes).trim();//转换成base64串

png\_base64 = png\_base64.replaceAll("\n", "").replaceAll("\r", "");//删除 \r\n

res.put("jpg", "data:image/\*;base64," + png\_base64);

return R.ok("获取验证码成功", res);

} catch (IOException e) {

e.printStackTrace();

return R.error(e.getMessage());

}

}

@PostMapping("/login")

@ApiOperation(value = "用户统一登录")

public Object login(@RequestBody SysLogin object,

@RequestHeader("token") String token) {

// 操作日记初始化

ServletRequestAttributes attributes = (ServletRequestAttributes) RequestContextHolder.getRequestAttributes();

HttpServletRequest request = attributes.getRequest();

SysOperationRecord sysOperationRecord = new SysOperationRecord();

String ip = getIpAddr(request);

sysOperationRecord.setIpAddr(ip);

// 数据验证

String str = cacheKit.getVal(token);

cacheKit.deleteVal(token);

if (V.isEmpty(str))

return R.error("验证码已经过期请重新获取验证码");

JSONObject json = J.s2j(str);

String code = json.getString("code");

if (V.isEmpty(code))

return R.error("验证码已经过期请重新获取验证码2");

if (V.isEmpty(object.getCode()))

return R.error("请输入验证码");

if (!V.isEqual(code, object.getCode()))

return R.error("您输入的验证码不正确");

if (V.isEmpty(token))

return R.error("请先获取验证码");

if (V.isEmpty(object.getLoginName()))

return R.error("请输入登录名");

if (V.isEmpty(object.getPassword()))

return R.error("请输入登录密码");

String passwordStr = object.getPassword().replace(" ", "+");

String privateKey = cacheKit.getVal(token + "\_privateKey");

String password = RSAEncrypt.decrypt(passwordStr, privateKey);

cacheKit.deleteVal(token + "\_privateKey");

object.setPassword(password);

// 业务处理

JSONObject user = J.o2j(sysUserService.findByLoginName(object.getLoginName()));

if (user == null)

return R.error("登录名或密码错误");

//存在该用户

SysUser sysUser = sysUserService.findByUserName(object.getLoginName());

sysOperationRecord.setCreator(user.getString("id"));

sysOperationRecord.setCreateTime((int) TimeKit.getTimestamp());

sysOperationRecord.setControl("系统授权管理");

sysOperationRecord.setFunction("用户统一登录");

sysOperationRecord.setServer(adminServer);

//判断是否禁用

if (sysUser.getStatus() == 0) {

return R.error("您的账号当前已被禁用，请联系管理员");

}

String userId = user.getString("id");

// 数据处理

user.remove("password");

user.remove("salt");

String userToken = S.getToken();

user.put("token", userToken);

user.put("role", "");

user.put("lock", 0);

user.put("type", sysUser.getType());

user.put("auth", new ArrayList<>());

// 权限

SysUserRole ur = userRoleService.findByUserId(userId);

if (ur != null) {

user.put("auth", sysUserService.getUserAuth(user.getString("id")));

// user.put("auth", roleService.getAuthInfoByRoleId(ur.getRoleId()));

user.put("role", ur.getRoleId());

}

SysUserToken ut = new SysUserToken();

ut.setUserId(userId);

ut.setToken(userToken);

ut.setType("redis");

ut.setServer(adminServer);

tokenAsync.updateToken(ut);

List<JSONObject> sysUserRoles = sysRoleService.getUserRoleByUserId(userId);

cacheKit.deleteVal("verify\_" + object.getLoginName() + "\_count");

// 缓存处理

if (initCacheUser(user, userToken, 0)) {

sysUser.setLoginIp(ip);

sysUser.setLoginTime((int) TimeKit.getTimestamp());

sysUserService.update(sysUser);

sysOperationRecord.setStatus(0);

sysOperationRecordService.createSysOperationRecord(sysOperationRecord);

return R.ok(user);

}

return R.ok("缓存处理失败", user);

}

@PostMapping("/forget")

@ApiOperation(value = "忘记密码")

public Object forgetPassword(@RequestBody ForgetPassword object,

@RequestHeader("token") String token) {

// 操作日记初始化

ServletRequestAttributes attributes = (ServletRequestAttributes) RequestContextHolder.getRequestAttributes();

HttpServletRequest request = attributes.getRequest();

SysOperationRecord sysOperationRecord = new SysOperationRecord();

String ip = getIpAddr(request);

sysOperationRecord.setIpAddr(ip);

String passwordStr1 = object.getNewPassword1().replace(" ", "+");

String passwordStr2 = object.getNewPassword2().replace(" ", "+");

String privateKey = cacheKit.getVal(token + "\_privateKey");

String password1 = RSAEncrypt.decrypt(passwordStr1, privateKey);

String password2 = RSAEncrypt.decrypt(passwordStr2, privateKey);

cacheKit.deleteVal(token + "\_privateKey");

object.setNewPassword1(password1);

object.setNewPassword2(password2);

if (V.isEmpty(object.getPhone()))

return R.error("请输入手机号码");

if (V.isEmpty(object.getVerifyCode()))

return R.error("请输入验证码");

if (!object.getNewPassword1().equals(object.getNewPassword2())) {

return R.error("新密码不一致");

}

wheres = W.f(

W.and("phone", "eq", object.getPhone()),

W.and("verifyCode", "eq", object.getVerifyCode()),

W.and("isDel", "eq", "0")

);

List<SysUser> users = sysUserService.query(wheres);

if (users == null || users.size() == 0) {

return R.error("请检查手机号或验证码");

}

SysUser userInfo = users.get(0);

//存储操作日记

sysOperationRecord.setCreator(userInfo.getId());

sysOperationRecord.setCreateTime((int) TimeKit.getTimestamp());

sysOperationRecord.setControl("系统授权管理");

sysOperationRecord.setFunction("重置密码");

sysOperationRecord.setServer(pcServer);

JSONObject user = new JSONObject();

String salt = S.randomNum();

String password = MD5Kit.encode(S.apppend(object.getNewPassword1(), salt));

user.put("id", userInfo.getId());

user.put("salt", Integer.parseInt(salt));

user.put("password", password);

user.put("verifyCode", "");

user.put("errorCount", 0);

user.put("unlockTime", 0);

Object o = sysUserService.updateUser(user);

if (o == null) {

sysOperationRecord.setStatus(1);

sysOperationRecordService.createSysOperationRecord(sysOperationRecord);

return R.error("密码修改失败");

} else {

sysOperationRecord.setStatus(0);

sysOperationRecordService.createSysOperationRecord(sysOperationRecord);

SMS\_func.DirectSendSMS(object.getPhone(), "您已成功重置作业管理系统密码，若非本人操作，请尽快登录网站更改密码。" + "(作业管理)");

return R.ok("密码找回成功");

}

}

}

### 4.2、作业管理模块

#### 4.21、Exam.ejs

<% include ../inc/header.ejs %>

<style>

.date-table {

overflow-y: auto;

}

</style>

<% include ../inc/body.ejs %>

<!-- html主体代码 start -->

<div class="row">

<div class="col-sm-12">

<div class="ibox float-e-margins">

<div class="ibox-title" style="padding-top: 3px !important;">

<div style="width: 140px; display: block;float: left;padding-top: 10px;">

作业列表

</div>

<div class="ibox-tools">

<button style="margin-top: 8px;" class="btn btn-primary btn-xs" type="button"

onclick="exam.create(this);">

<i class="fa fa-plus"></i>&nbsp;新增

</button>

<!-- <button style="margin-top: 8px;" class="btn btn-primary btn-xs" type="button"-->

<!-- onclick="exam.importUserModel(this);">-->

<!-- <i class="fa fa-download"></i>&nbsp;下载学生信息模板-->

<!-- </button>-->

<!-- <button style="margin-top: 8px;" class="btn btn-primary btn-xs" type="button"-->

<!-- onclick="exam.importUser(this);">-->

<!-- <i class="fa fa-user"></i>&nbsp;导入学生信息-->

<!-- </button>-->

<!-- <button style="margin-top: 8px;" class="btn btn-primary btn-xs" type="button"-->

<!-- onclick="exam.randomExam(this);">-->

<!-- <i class="fa fa-random"></i>&nbsp;随机组卷-->

<!-- </button>-->

</div>

</div>

<div class="ibox-content data-view" style="overflow-y: auto;">

<div class="col-sm-12">

<div class=" col-sm-12 date-table">

<table class="table table-hover">

<thead>

<tr>

<th>序号</th>

<th>名称</th>

<th>状态</th>

<th>类型</th>

<th>总分值</th>

<th>开始时间</th>

<th>结束时间</th>

<th>时长(分钟)</th>

<th>发布状态</th>

<th>成绩公布状态</th>

<th></th>

</tr>

</thead>

<tbody id="item-list">

</tbody>

</table>

</div>

<div class="col-sm-12 pages M-box3 list-page">

</div>

</div>

</div>

</div>

</div>

</div>

</div>

<!-- html主体代码 end -->

<!-- 管理的作业模板 -->

<script id="exam-template" type="text/x-dot-template">

{{ if(it.length == 0) { }}

暂无数据

{{ }else{ }}

{{ for(var i=0;i < it.length;i++){ }}

<tr>

<td>

{{= ((index-1) \* size +i+1)}}

</td>

<td>

{{= it[i].name }}

</td>

<td>

{{= helper.status(it[i].status) }}

</td>

<td>

{{= helper.type(it[i].type) }}

</td>

<td>

{{= it[i].score}}

</td>

<td>

{{= it[i].startTimeStr||""}}

</td>

<td>

{{= it[i].endTimeStr||""}}

</td>

<td>

{{=helper.length(it[i].duration)}}

</td>

<td>

{{= helper.releaseStatus(it[i].releaseStatus) }}

</td>

<td>

{{= helper.publishStatus(it[i].publishStatus) }}

</td>

<td>

<div class="btn-group">

<!--<a class="btn btn-primary btn-xs dropdown-toggle" style="margin-right: 5px"-->

<!--onclick="examAdd()">组卷</a>-->

<button data-toggle="dropdown" class="btn btn-primary btn-xs dropdown-toggle">操作 <span class="caret"></span>

</button>

<ul class="dropdown-menu">

{{ if(it[i].status !=2) {}}

<!-- <li>-->

<!-- <a href="#" data-id="{{= it[i].id}}" onclick="exam.importUser(this);">导入学生</a>-->

<!-- </li>-->

{{ } }}

<!-- <li>-->

<!-- <a href="#" data-id="{{= it[i].id}}" onclick="exam.redirectExamRoom(this);">考场管理</a>-->

<!-- </li>-->

{{ if(it[i].releaseStatus == 1) {}}

<!-- <li>-->

<!-- <a href="#" data-id="{{= it[i].id}}" onclick="exam.randomExam(this);">随机组卷</a>-->

<!-- </li>-->

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="exam.redirectExamPaperGenerate(this);">手动组卷</a>

</li>

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="exam.preview(this);">作业预览</a>

</li>

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="exam.release(this);">发布作业</a>

</li>

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="exam.update(this);">修改作业</a>

</li>

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="exam.delete(this);">删除作业</a>

</li>

{{ }else if(it[i].releaseStatus == 2) {}}

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="exam.preview(this);">作业预览</a>

</li>

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="exam.redirectExamStudent(this);">学生管理</a>

</li>

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="exam.publish(this);">公布成绩</a>

</li>

{{ if(it[i].isNote ==1&&it[i].status !=2) { }}

<!-- <li>-->

<!-- <a href="#" data-id="{{= it[i].id}}" onclick="exam.sendNote(this);">短信推送</a>-->

<!-- </li>-->

{{ } }}

{{ if(it[i].status ==1) { }}

<!-- <li>-->

<!-- <a href="#" data-id="{{= it[i].id}}" onclick="exam.sync(this);">重置缓存</a>-->

<!-- </li>-->

<!-- <li>-->

<!-- <a href="#" data-id="{{= it[i].id}}" onclick="exam.finish(this);">结束本次作业</a>-->

<!-- </li>-->

{{ } }}

{{ } }}

</ul>

</div>

</td>

</tr>

{{ } }}

{{ } }}

</script>

<script id="choice-list-template" type="text/x-dot-template">

{{ if(it.length == 0) { }}

暂无数据

{{ }else{ }}

{{ for(var i=0;i < it.length;i++){ }}

{{ var answer = ""; }}

<tr>

<td>

<span class="question-number">{{=(i+1)}}</span><span class="highlight-color"> 【{{=it[i].score }}分】</span>

{{=it[i].questionName }}<br>

{{ for(var j = 0; j < it[i]['options'].length ; j++){ }}

{{ if(it[i]['options'][j].isAnswer == 1){ }}

{{ answer += String.fromCharCode(65+j);}}

{{ } }}

&emsp;&emsp;&emsp;&nbsp;

<span class="option-num">{{=String.fromCharCode(65+j) }}</span> {{=it[i]['options'][j].optionInfo }}<br>

{{ } }}

&emsp;&emsp;&emsp;

正确答案：<span class="highlight-color">{{=answer }}</span><br>

答案解析：

<div class="answer-main">{{=it[i].answerKeys }}</div>

</td>

</tr>

{{ } }}

{{ } }}

</script>

<!--填空题-->

<script id="gap-list-template" type="text/x-dot-template">

{{ if(it.length == 0) { }}

暂无数据

{{ }else{ }}

{{ for(var i=0;i < it.length;i++){ }}

<tr>

<td>

<span class="question-number">{{=(i+1)}}</span><span class="highlight-color"> 【{{=it[i].score }}分】</span>

{{=it[i].questionName }}<br>

{{ for(var j = 0; j < it[i]['options'].length ; j++){ }}

&emsp;&emsp;&emsp;&nbsp;

<span class="option-num">{{=(j+1) }}</span> {{=it[i]['options'][j].optionInfo }}<br>

{{ } }}

答案解析：

<div class="answer-main">{{=it[i].answerKeys }}</div>

</td>

</tr>

{{ } }}

{{ } }}

</script>

<!--客观题-->

<script id="subjective-list-template" type="text/x-dot-template">

{{ if(it.length == 0) { }}

暂无数据

{{ }else{ }}

{{ for(var i=0;i < it.length;i++){ }}

<tr>

<td>

<span class="question-number">{{=(i+1)}}</span><span class="highlight-color"> 【{{=it[i].score }}分】</span>

{{=it[i].questionName }}<br>

答案解析：

<div class="answer-main">{{=it[i].answerKeys }}</div>

</td>

</tr>

{{ } }}

{{ } }}

</script>

<script id="exam-select-template" type="text/x-dot-template">

{{ if(it.length == 0 ){ }}

<option value=" ">暂无未发布的作业</option>

{{ }else{ }}

<option value=" ">请选择作业</option>

{{ for(var i = 0 ; i < it.length ; i++){ }}

<option value="{{=it[i].id }}">{{=it[i].name }}</option>

{{ } }}

{{ } }}

</script>

<script id="course-template" type="text/x-dot-template">

{{ if(it.length == 0 ){ }}

<option value="">您还没有课程安排</option>

{{ }else{ }}

<option value="" selected="selected">请选择课程</option>

{{ for(var i = 0 ; i < it.length ; i++){ }}

<option value="{{=it[i].courseNumber }}">{{=it[i].courseName }}</option>

{{ } }}

{{ } }}

</script>

<script id="class-template" type="text/x-dot-template">

{{ if(it.length == 0 ){ }}

<option value="">没有可布置的班级</option>

{{ }else{ }}

<option value="" selected="selected">请选择班级</option>

{{ for(var i = 0 ; i < it.length ; i++){ }}

<option value="{{=it[i].classId }}">{{= (it[i].grade+" "+it[i].majorName+" "+it[i].classNo )}}</option>

{{ } }}

{{ } }}

</script>

<script id="exam-info-template" type="text/x-dot-template">

{{ if(it.length == 0 ){ }}

暂无数据

{{ }else{ }}

<tr>

<td style="width: 100px; ">题型</td>

<td style="width: 100px; ">所占分值</td>

<td style="width: 100px; ">题库总题量</td>

<td style="width: 100px; ">题量</td>

<td class="form-item-remark"><span></span></td>

</tr>

{{ for(var i = 0 ; i < it.length ; i++){ }}

<tr>

<td>判断题</td>

<td>{{=it[i].trueOrFalse }}</td>

<td id="trueOrFalseNumberQuestion"></td>

<td>

<input required score="{{=it[i].trueOrFalse }}" type="number" name="trueOrFalseNumber" class="form-control"

placeholder="请输入判断题题量" onblur="blurs(this)"/>

</td>

<td class="form-item-remark"><span id="trueOrFalseNumber"></span></td>

</tr>q

<tr>

<td>单选题</td>

<td>{{=it[i].single }}</td>

<td id="singleNumberQuestion"></td>

<td>

<input required score="{{=it[i].single }}" type="number" name="singleNumber" class="form-control"

placeholder="请输入单选题题量" onblur="blurs(this)"/>

</td>

<td class="form-item-remark"><span id="singleNumber"></span></td>

</tr>

<tr>

<td>多选题</td>

<td>{{=it[i].multiple }}</td>

<td id="multipleNumberQuestion"></td>

<td>

<input required score="{{=it[i].multiple }}" type="number" name="multipleNumber" class="form-control"

placeholder="请输入多选题题量" onblur="blurs(this)"/>

</td>

<td class="form-item-remark"><span id="multipleNumber"></span></td>

</tr>

<tr>

<td>填空题</td>

<td>{{=it[i].gap }}</td>

<td id="gapNumberQuestion"></td>

<td>

<input required score="{{=it[i].gap }}" type="number" name="gapNumber" class="form-control"

placeholder="请输入填空题题量" onblur="blurs(this)"/>

</td>

<td class="form-item-remark"><span id="gapNumber"></span></td>

</tr>

<tr>

<td class="form-item-title" style="text-align: center">

作业总分

</td>

<td>

<input disabled required type="text" name="score" maxlength="20" class="form-control"

style="text-align: center" value="{{= it[0].score }}">

</td>

<td id="allQuestion"></td>

<td class="form-item-remark">

<input disabled required type="text" id="allNumber" maxlength="20" class="form-control"

style="text-align: center">

</td>

<td class="form-item-remark"><span></span></td>

</tr>

{{ } }}

{{ } }}

</script>

<script>

function blurs(event) {

var score = parseInt( $(event).attr("score"));

var tipsId = $(event).attr("name");

var inputValue = parseInt($(event).val());

if (inputValue > parseInt($('#' + tipsId + "Question").html())) {

$('#' + tipsId).html("题库的题量不足" + inputValue + "题");

return false;

}

if (inputValue != 0 && !(score % inputValue == 0||score % inputValue == inputValue/2)) {

$('#' + tipsId).html("单题分数必须为整数或者0.5分结尾");

return false;

} else {

$('#' + tipsId).html(" ");

}

var valueArr = $('input[type="number"]');

var sumValue = 0;

if (valueArr.size() > 0 || valueArr != null) {

for (var i = 0; i < valueArr.size(); i++) {

if (isNaN(valueArr[i].value) || valueArr[i].value == null || valueArr[i].value == "") {

sumValue += 0;

} else {

sumValue += parseFloat(valueArr[i].value);//强制转换为数字

}

}

}

if (sumValue > 0) {

$('#allNumber').val(parseInt(sumValue));

} else {

$('#allNumber').val(0);

}

}

</script>

<script id="error-template" type="text/x-dot-template">

{{ if(it.length == 0) { }}

暂无数据

{{ }else{ }}

{{ for(var i=0;i < it.length;i++){ }}

<tr>

<td>

{{= i+1}}

</td>

<td>

{{= it[i].name|| ""}}

</td>

<td>

{{= it[i].myClass || ""}}

</td>

<td class="td-number">

{{= it[i].number|| ""}}

</td>

<td class="td-phone">

{{= it[i].phone || ""}}

</td>

<td>

{{= it[i].info || ""}}

</td>

</tr>

{{ } }}

{{ } }}

</script>

<% include ../inc/js.ejs %>

<!-- 私有脚本 start -->

<script src="../../js/baseModule/\_exam.js"></script>

<!-- 私有脚本 end -->

<% include ../inc/footer.ejs %>

#### 4.22、ExamApi.java

package team.work.api.admin;

import com.alibaba.fastjson.JSONArray;

import com.alibaba.fastjson.JSONObject;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.apache.poi.hssf.usermodel.HSSFWorkbook;

import org.apache.poi.ss.usermodel.Row;

import org.apache.poi.ss.usermodel.Sheet;

import org.apache.poi.ss.usermodel.Workbook;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

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

import org.springframework.transaction.annotation.Transactional;

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

import team.work.api.BaseApi;

import team.work.core.model.\*;

import team.work.core.service.impl.\*;

import team.work.doc.ExamCreate;

import team.work.doc.ExamUpdate;

import team.work.doc.ExamUserImport;

import team.work.utils.bean.Where;

import team.work.utils.convert.R;

import team.work.utils.convert.S;

import team.work.utils.convert.V;

import team.work.utils.convert.W;

import team.work.utils.kit.TimeKit;

import javax.servlet.http.HttpServletRequest;

import java.io.File;

import java.io.FileInputStream;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

@Api(value = "36\_作业管理")

@RestController

@RequestMapping("/v1/base")

public class ExamApi extends BaseApi {

@Autowired

private ExamService examService;

@Autowired

private StudentService studentService;

@Autowired

private ExamCandidateService examCandidateService;

@Autowired

private ExamRecordService examRecordService;

@Autowired

private SysRoleService roleService;

@Autowired

private ExamQuestionService examQuestionService;

@Transactional(rollbackFor = Exception.class)

@PostMapping("/exam")

@ApiOperation(value = "作业新增")

public Object createExam(@RequestBody ExamCreate object,

@RequestHeader("token") String token) throws Exception {

String teacherNumber = getNumberByCache(token);

//映射对象

Exam model = o2c(object, token, Exam.class);

//数据校验

JSONObject check = V.checkEmpty(verify(), object);

if (check.getBoolean("check"))

return R.error(check.getString("message"));

model.setStatus(0);

model.setTeacherNumber(teacherNumber);

double single = Double.parseDouble(object.getSingle());

double gap = Double.parseDouble(object.getGap());

double multiple = Double.parseDouble(object.getMultiple());

double trueOrFalse = Double.parseDouble(object.getTrueOrFalse());

double score = Double.parseDouble(object.getScore());

if (single + gap + multiple + trueOrFalse == 0 || score == 0) {

return R.error("总分值不能为0");

}

// Boolean exist = examService.exist(W.f(

// W.and("name", "eq", model.getName()),

// W.and("isDel", "eq", "0"))

// );

// if (exist)

// return R.error("该作业名称已存在，请更换！");

if (V.isEmpty(model.getStartTime()) || V.isEmpty(model.getEndTime())) {

return R.error("未设置开始时间和结束时间！");

}

Long now = TimeKit.getTimestamp();

if (model.getStartTime() > model.getEndTime()) {

return R.error("开始时间应小于结束时间");

}

if (model.getEndTime() <= now) {

return R.error("结束时间应大于现在时间");

}

String classId = object.getClassId();

if (V.isEmpty(classId)) {

return R.error("请正确选择完成该作业的班级");

}

model.setClassId(classId);

model = examService.createExam(model);

if (model == null) {

return R.error("新增失败");

}

return R.ok("新增成功，请去添加题目", fm2(model));

}

@Transactional(rollbackFor = Exception.class)

@PutMapping("/exam")

@ApiOperation(value = "修改作业")

public Object updateExam(@RequestBody ExamUpdate object,

@RequestHeader("token") String token) throws Exception {

String userId = getUserIdByCache(token);

String teacherNumber = getNumberByCache(token);

//映射对象

Exam model = o2c(object, token, Exam.class);

//数据校验

JSONObject check = V.checkEmpty(updateVerify(), object);

if (check.getBoolean("check"))

return R.error(check.getString("message"));

Exam data = examService.selectById(model.getId());

if (data == null) {

return R.error("该作业不存在，无法修改");

}

// if (!data.getName().equals(model.getName())) {

// Boolean exist = examService.exist(W.f(

// W.and("name", "eq", model.getName()),

// W.and("isDel", "eq", "0"))

// );

// if (exist)

// return R.error("该作业已存在请更换！");

// }

if (V.isEmpty(model.getStartTime()) || V.isEmpty(model.getEndTime())) {

return R.error("未设置开始时间和结束时间！");

}

Long now = TimeKit.getTimestamp();

if (model.getStartTime() > model.getEndTime()) {

return R.error("开始时间应小于结束时间");

}

if (model.getEndTime() <= now) {

return R.error("结束时间应大于现在时间");

}

String classId = object.getClassId();

if (V.isEmpty(classId)) {

return R.error("请正确选择完成该作业的班级");

}

model.setClassId(classId);

model.setTeacherNumber(teacherNumber);

model.setReviser(userId);

model = examService.updateExam(model);

if (model == null) {

return R.error("修改失败");

} else {

return R.ok("修改成功", fm2(model));

}

}

@DeleteMapping("/exam/{id}")

@ApiOperation(value = "作业删除")

public Object deleteExam(@PathVariable("id") String id,

@RequestHeader("token") String token) {

if (!examService.existId(id))

return R.error("Id数据异常");

if (examService.deleteExam(id, cacheKit.getUserId(token)))

return R.ok("删除成功");

return R.error("删除失败");

}

@GetMapping("/exam/{index}-{size}-{key}")

@ApiOperation(value = "读取作业分页列表")

public Object getExam(@PathVariable("index") int index,

@PathVariable("size") int size,

@PathVariable("key") String key,

@RequestHeader("token") String token) {

Integer typeByCache = getTypeByCache(token);

String wKey = "";

if (typeByCache == 1) {

String number = getNumberByCache(token);

wKey += " and teacherNumber = '" + number + "' ";

}

if (!V.isEmpty(key))

wKey += S.apppend(" and id like '%", key, "%' ");

return R.ok(examService.page(index, size, wKey));

}

@GetMapping("/exam/all/{status}")

@ApiOperation(value = "根据状态读取作业所有列表")

public Object getAllExamByStatus(@PathVariable("status") String status,

@RequestHeader("token") String token) {

String teacherNumber = getNumberByCache(token);

//若有status则说明不是全查，发布后的作业不再重新组卷

String where = " and teacherNumber = '" + teacherNumber + "' ";

if (!V.isEmpty(status)) {

where += " and releaseStatus = " + status + " ";

}

return R.ok(examService.queryAll(where));

}

@GetMapping("/exam/import/selectList")

@ApiOperation(value = "获取可导入作业列表")

public Object getImportExamList(@RequestHeader("token") String token) {

String where = " and status !=2 ";

return R.ok(examService.queryAll(where));

}

@GetMapping("/exam/ready/selectList")

@ApiOperation(value = "获取未开始作业列表")

public Object getReadyExamList(@RequestHeader("token") String token) {

String where = " and status =0 ";

return R.ok(examService.queryAll(where));

}

@GetMapping("/exam/all/selectList")

@ApiOperation(value = "获取全部作业列表")

public Object getAllExamList(@RequestHeader("token") String token) {

return R.ok(examService.queryAll(""));

}

@Transactional(noRollbackFor = Exception.class)

@GetMapping("/exam/info/{id}")

@ApiOperation(value = "读取作业信息")

public Object getExamById(@PathVariable("id") String id,

@RequestHeader("token") String token) {

if (!V.isEmpty(id))

return R.ok(examService.queryAll(" and id = '" + id + "' "));

return R.ok(examService.queryAll(""));

}

@Transactional

@PutMapping("/exam/release/{examId}")

@ApiOperation(value = "发布作业")

public Object releaseExam(@PathVariable("examId") String examId,

@RequestHeader("token") String token) throws Exception {

String userId = getUserIdByCache(token);

List<Exam> examList = examService.findByIds(examId);

if (examList == null || examList.size() == 0)

return R.error("id数据异常");

Exam exam = examList.get(0);

Boolean examQuestions = examQuestionService.exist(W.f(

W.and("examId", "eq", exam.getId()),

W.and("isDel", "eq", 0)

));

if (!examQuestions) {

return R.error("该作业还没有题目");

}

if (V.isEmpty(exam.getStartTime()) || V.isEmpty(exam.getEndTime())) {

return R.error("未设置开始时间和结束时间！");

}

Long now = TimeKit.getTimestamp();

if (exam.getStartTime() > exam.getEndTime()) {

return R.error("开始时间应小于结束时间");

}

if (exam.getEndTime() <= now) {

return R.error("结束时间应大于现在时间");

}

exam.setReleaseStatus(2);//2-已发布

exam.setReviser(userId);

//发布的时候 通过classId循环添加学生到作业学生关系表

//通过classId查询班级学生

List<JSONObject> stuList = studentService.queryAll(" and classId = '" + exam.getClassId() + "' ");

int stuSize = stuList.size();

if (stuSize == 0) {

return R.error("所布置的班级没有学生");

}

List<ExamCandidate> examCandidates = new ArrayList<>(stuSize);

for (JSONObject stu : stuList) {

ExamCandidate examCandidate = new ExamCandidate();

examCandidate.setExamId(exam.getId());

examCandidate.setNumber(stu.getString("number"));

examCandidate.setCreator(userId);

examCandidates.add(examCandidate);

// examCandidate = examCandidateService.createExamCandidate(examCandidate);

}

System.out.println(examCandidates);

int userCount = examCandidateService.insertRows(examCandidates);

exam = examService.updateExam(exam);

if (userCount != stuSize || exam == null) {

return R.error("发布失败");

} else {

return R.ok("发布成功！", fm2(exam));

}

}

@PutMapping("/exam/syncCache/{examid}")

@ApiOperation(value = "重置作业缓存")

@Transactional

public Object syncExamCache(@PathVariable("examid") String examid, @RequestHeader("token") String token) {

cacheKit.deleteVal("exam\_info\_" + examid);

cacheKit.deleteVals("exam\_question\_" + examid);

cacheKit.deleteVals("examQuestion\_type\_" + examid + "\*");

cacheKit.deleteVals("exam\_question\_score\_" + examid + "\*");

cacheKit.deleteVals("exam\_questionId\_options\_unans\_\*");

cacheKit.deleteVals("exam\_questionId\_Options\_\*");

cacheKit.deleteVals("exam\_questionId\_trueOptionIds\_\*");

return R.ok("重置作业缓存成功！");

}

@PutMapping("/exam/publish/{examId}")

@ApiOperation(value = "公布成绩")

public Object publishExam(@PathVariable("examId") String examId,

@RequestHeader("token") String token) {

String userId = getUserIdByCache(token);

Exam exam = examService.selectById(examId);

if (V.isEmpty(exam))

return R.error("作业不存在！");

exam.setEndTime((int) TimeKit.getTimestamp());

exam.setPublishStatus(2);

exam.setReviser(userId);

exam = examService.updateExam(exam);

if (exam == null) {

return R.error("发布作业失败！");

}

return R.ok("发布成绩成功！");

}

@PutMapping("/exam/finish/{examid}")

@ApiOperation(value = "结束本次作业")

@Transactional

public Object finishExam(@PathVariable("examid") String examid, @RequestHeader("token") String token) {

String userId = getUserIdByCache(token);

Exam exam = examService.selectById(examid);

if (V.isEmpty(exam))

return R.error("作业不存在！");

exam.setEndTime((int) TimeKit.getTimestamp());

exam.setStatus(2);

exam.setReviser(userId);

exam = examService.updateExam(exam);

if (exam == null) {

return R.error("结束作业失败！");

} else {

List<ExamCandidate> examCandidateList = examCandidateService.query(W.f(

W.and("examId", "eq", examid),

W.and("status", "eq", 1),

W.and("isDel", "eq", 0)

));

for (ExamCandidate examCandidate : examCandidateList) {

// examCandidate.setRemark("未按时作答考试结束");

examCandidate.setReviser(userId);

examCandidate.setScore(0d);

examCandidate.setStatus(2);

closeExamCandidate(examCandidate);

}

return R.ok("结束作业成功！");

}

}

public Boolean closeExamCandidate(ExamCandidate model) {

String[] examQuestions = null;

String examId = model.getExamId();

String number = model.getNumber();

if (V.isEmpty(examId) || V.isEmpty(number)) {

return false;

}

List<ExamRecord> examRecords = new ArrayList<>();

String tmp\_exam\_question = cacheKit.getVal("exam\_question\_" + examId);

if (!V.isEmpty(tmp\_exam\_question)) {

JSONArray examQuestionArray = JSONArray.parseArray(tmp\_exam\_question);

examQuestions = new String[examQuestionArray.size()];

examQuestions = examQuestionArray.toArray(examQuestions);

} else {

examQuestions = examQuestionService.queryPCQuestionIds(examId);

cacheKit.setVal("exam\_question\_" + examId, JSONObject.toJSONString(examQuestions), 1800);

}

for (int i = 0; examQuestions != null && i < examQuestions.length; i++) {

ExamRecord examRecord = new ExamRecord();

examRecord.setExamId(examId);

examRecord.setNumber(number);

examRecord.setQuestionId(examQuestions[i]);

examRecord.setSituation(2);

examRecord.setAnswerContent("");

examRecords.add(examRecord);

}

//记录答题

try {

if (!V.isEmpty(examRecords)) {

int userCount = examRecordService.insertRows(examRecords, model.getCreator());

}

model.setScore(0d);

model.setStatus(2);

model = examCandidateService.updateExamCandidate(model);

if (!V.isEmpty(model)) {

cacheKit.deleteVal("examQuestion\_submitTime\_" + number + "\_" + examId);

cacheKit.deleteVal("examQuestion\_Options\_" + examId + "\_" + number);

return true;

} else {

return false;

}

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

@GetMapping("/exam/grade/{index}-{size}/examid-{examId}/academyCode-{academyCode}/number-{number}")

@ApiOperation(value = "读取学生作业信息")

public Object getExamGrade(@PathVariable("index") int index,

@PathVariable("size") int size,

@PathVariable("examId") String examId,

@PathVariable("academyCode") String academyCode,

@PathVariable("number") String number,

@RequestHeader("token") String token) {

String teacherNumber = getNumberByCache(token);

String where = " and teacherNumber = '" + teacherNumber + "' ";

if (!V.isEmpty(examId))

where += S.apppend(" and examId = '" + examId + "' ");

if (!V.isEmpty(number))

where += S.apppend(" and number like '%" + number + "%' ");

if (!V.isEmpty(academyCode))

where += S.apppend(" and academyCode = '" + academyCode + "' ");

return R.ok(examCandidateService.page(index, size, where));

}

@GetMapping("/exam/grade/all/examid-{examid}/academyCode-{academyCode}/number-{number}")

@ApiOperation(value = "读取全部测试结果信息")

public Object getAllExamGrade(@PathVariable("examid") String examid, @PathVariable("academyCode") String academyCode, @PathVariable("number") String number, @RequestHeader("token") String token) {

String where = "";

String ubiWhere = "";

if (!V.isEmpty(examid))

where += S.apppend(" and examid = '" + examid + "' ");

if (!V.isEmpty(number))

where += S.apppend(" and number like '%" + number + "%' ");

if (!V.isEmpty(academyCode))

ubiWhere += S.apppend(" and academyCode = '" + academyCode + "' ");

return R.ok(examCandidateService.queryAllGrade(where, ubiWhere));

}

@PutMapping("/exam/unlockStudent/{number}")

@ApiOperation(value = "解除学生登陆锁")

@Transactional

public Object unlockStudent(@PathVariable("number") String number, @RequestHeader("token") String token) {

if (V.isEmpty(number)) {

return R.error("学号不存在！");

}

cacheKit.deleteVal("verify\_" + number.trim() + "\_count");

return R.ok("解除学生登陆锁成功！");

}

@PutMapping("/exam/syncStudentStatus/{examid}-{number}")

@ApiOperation(value = "重置学生状态")

@Transactional

public Object syncExamStudentStatus(@PathVariable("examid") String examid, @PathVariable("number") String number, @RequestHeader("token") String token) {

if (V.isEmpty(examid) || V.isEmpty(number)) {

return R.error("作业编号或学号不存在！");

}

number = number.trim();

Exam exam = examService.selectById(examid);

if (V.isEmpty(exam))

return R.error("考试不存在！");

Integer status = exam.getStatus();

if (2 == status) {

return R.error("考试已结束，无法重置学生！");

}

//删除考试回顾缓存

cacheKit.deleteVal("exam\_candidate\_" + examid + "\_" + number);

for (int i = 0; i < 5; i++) {

cacheKit.deleteVal("examRecord\_type\_" + examid + "\_" + (i + 1) + "\_" + number);

}

cacheKit.deleteVal("exam\_candidate\_examInfo\_" + examid + "\_" + number);

//删除考试记录

cacheKit.deleteVal("exam\_ans\_lock\_" + number + "\_" + examid);

cacheKit.deleteVal("examQuestion\_submitTime\_" + number + "\_" + examid);

cacheKit.deleteVal("examQuestion\_Options\_" + examid + "\_" + number);

examService.updateExamCandidate(examid, number);

examService.deleteExamRecord(examid, number);

return R.ok("重置学生状态成功！");

}

@PostMapping("/exam/ImportUserSetting")

@ApiOperation(value = "导入用户配置")

public Object ImportUserSetting(@RequestBody ExamUserImport object,

@RequestHeader("token") String token) {

if (object.getExamId() == null || object.getExamId() == "") {

return R.error("未选择作业!");

}

if (object.getAcademyCode() == null || object.getAcademyCode() == "") {

return R.error("未选择学院!");

}

if (object.getType() == null || object.getType() == 0) {

return R.error("未选择用户类型!");

}

Boolean exist = examService.exist(W.f(

W.and("id", "eq", object.getExamId()),

W.and("isDel", "eq", "0"))

);

if (!exist)

return R.error("作业数据异常");

String batch = S.getToken();

cacheKit.setVal(batch + "examId", object.getExamId(), 0);

cacheKit.setVal(batch + "academyCode", object.getAcademyCode(), 0);

cacheKit.setVal(batch + "type", object.getType() + "", 0);

return R.ok("配置成功", batch);

}

@PostMapping("/exam/ImportUser/{batch}")

@ApiOperation(value = "导入用户")

public Object ImportUser(

HttpServletRequest req,

@PathVariable("batch") String batch,

@RequestHeader("token") String token) {

if (V.isEmpty(batch)) {

return R.error("导入批次有误！");

}

if (V.isEmpty(cacheKit.getVal("exam-ImportUser-lock-" + batch))) {

cacheKit.setVal("exam-ImportUser-lock-" + batch, "true", 1800);

} else {

return R.error("导入操作正在进行中，请耐心等待操作结果");

}

if (cacheKit.getVal(batch + "examId") == null) {

return R.error("未选择作业!");

}

if (cacheKit.getVal(batch + "academyCode") == null) {

return R.error("未选择学院!");

}

if (cacheKit.getVal(batch + "type") == null) {

return R.error("未选择用户类型!");

}

String academyCode = cacheKit.getVal(batch + "academyCode");

String examId = cacheKit.getVal(batch + "examId");

String roleId = "";//学生角色Id 前端选择

List<Where> wheres = W.f(

W.and("name", "eq", "前端用户")

);

// String wKey = "";

// if (!V.isEmpty(name))

// wKey = S.apppend(" and (name like '%", name, "%')");

List<SysRole> roles = roleService.query(wheres);

if (!V.isEmpty(roles)) {

roleId = roles.get(0).getId();

} else {

return R.error("角色配置出错，请联系管理员处理！");

}

int type = Integer.parseInt(cacheKit.getVal(batch + "type")); //学生用户类型

String userId = getUserIdByCache(token);

String filepath = uploadFile(req);

if (V.isEmpty(filepath)) {

return R.error("文件上传失败！");

}

JSONObject result = new JSONObject();

JSONObject obj = new JSONObject();

obj.put("academyCode", academyCode);

obj.put("examId", examId);

obj.put("roleId", roleId);

obj.put("type", type);

obj.put("userid", userId);

obj.put("token", token);

obj.put("filepath", filepath);

List<Exam> examList = examService.findByIds(examId);

if (examList == null || examList.size() == 0)

return R.error("作业数据异常！");

JSONObject res = SolvedFile(obj, batch);

if (res == null) {

return R.error("学生导入异常！");

}

Boolean flag = res.getBoolean("flag");

if (!flag) {

return R.error(res.getString("message"));

}

result.put("total", res.getInteger("total"));

result.put("truecount", res.getInteger("truecount"));

result.put("errcount", res.getInteger("errcount"));

if (res.getInteger("truecount") == 0 && res.getInteger("errcount") == 0) {

return R.error("导入数据为空！");

}

if (res.getInteger("errcount") > 0) {

result.put("errlist", res.getJSONArray("errlist"));

result.put("success", 0);

} else {

result.put("success", 1);

}

cacheKit.deleteVal("exam-ImportUser-lock-" + batch);

return R.ok("导入成功", result);

}

@PostMapping("/exam/ImportUserStatus/{batch}")

@ApiOperation(value = "导入用户状态")

public Object getImportUserStatus(@PathVariable("batch") String batch) {

if (V.isEmpty(batch)) {

return R.error("导入批次有误！");

}

JSONObject obj = getImportStatusObj(batch);

if (V.isEmpty(batch)) {

return R.error("导入状态异常！");

}

return R.ok(obj);

}

public JSONObject SolvedFile(JSONObject obj, String batch) {

JSONObject res = new JSONObject();

JSONArray errors = new JSONArray();

if (obj.getString("filepath") == null || obj.getString("filepath") == "") {

res.put("flag", false);

res.put("message", "上传文件有误！");

return res;

}

try {

File excel = new File(obj.getString("filepath"));

if (excel.isFile() && excel.exists()) { //判断文件是否存在

String[] split = excel.getName().split("\\."); //.是特殊字符，需要转义！！！！！

Workbook wb;

//根据文件后缀（xls/xlsx）进行判断

if ("xls".equals(split[1])) {

FileInputStream fis = new FileInputStream(excel); //文件流对象

wb = new HSSFWorkbook(fis);

} else if ("xlsx".equals(split[1])) {

wb = new XSSFWorkbook(excel);

} else {

res.put("flag", false);

res.put("message", "上传文件有误！");

return res;

}

//开始解析

Sheet sheet = wb.getSheetAt(0); //读取sheet

int firstRowIndex = sheet.getFirstRowNum() + 1; //第一行是列名，所以不读

int lastRowIndex = sheet.getLastRowNum();

int userCount = 0, trueCount = 0;

for (int rIndex = firstRowIndex; rIndex <= lastRowIndex; rIndex++) { //遍历行

Row row = sheet.getRow(rIndex);

if (row != null) {

int lastCellIndex = row.getLastCellNum();

if (lastCellIndex != 5 && lastCellIndex != 4 && lastCellIndex != 3) {

continue;

} else {

String myclass = "", name = "", number = "", phone = "", room = "";

if (row.getCell(0) != null && row.getCell(1) != null && row.getCell(2) != null) {

JSONObject userobj = new JSONObject();

name = row.getCell(0).toString().trim();

myclass = row.getCell(1).toString().trim();

number = row.getCell(2).toString().trim();

if (row.getCell(3) != null) {

phone = row.getCell(3).toString().trim();

userobj.put("phone", phone);

}

if (row.getCell(4) != null) {

room = row.getCell(4).toString().trim();

userobj.put("room", room);

}

userobj.put("name", name);

userobj.put("myClass", myclass);

userobj.put("number", number);

if (number == null || number == "") {

continue;

}

userobj.put("academyCode", obj.getString("academyCode"));

userobj.put("examId", obj.getString("examId"));

userobj.put("roleId", obj.getString("roleId"));

userobj.put("type", obj.getInteger("type"));

userobj.put("userid", obj.getString("userid"));

userobj.put("token", obj.getString("token"));

JSONObject examUserObj = examService.ImportUser(userobj);

userobj.remove("academyCode");

userobj.remove("examId");

userobj.remove("roleId");

userobj.remove("type");

userobj.remove("userid");

userobj.remove("token");

userCount++;

if (examUserObj != null) {

boolean userFlag = examUserObj.getBoolean("flag");

if (!userFlag) {

userobj.put("info", examUserObj.getString("info"));

errors.add(userobj);

cacheKit.setVal(batch + "status", 2 + "", 0);

} else {

trueCount++;

cacheKit.setVal(batch + "status", 1 + "", 0);

}

} else {

cacheKit.setVal(batch + "status", 2 + "", 0);

userobj.put("info", "其他原因");

errors.add(userobj);

}

cacheKit.setVal(batch + "now", rIndex + "", 0);

}

cacheKit.setVal(batch + "total", lastRowIndex + "", 0);

if (rIndex != lastRowIndex) {

cacheKit.setVal(batch + "res", 2 + "", 0);

}

}

}

}

cacheKit.setVal(batch + "res", 1 + "", 0);

res.put("truecount", trueCount);

res.put("errcount", errors.size());

res.put("total", userCount);

res.put("errlist", errors);

res.put("flag", true);

return res;

} else {

res.put("flag", false);

res.put("message", "上传文件有误！");

System.out.println("找不到指定的文件");

return res;

}

} catch (Exception e) {

e.printStackTrace();

res.put("flag", false);

res.put("message", "学生处理出错！");

return res;

}

}

private Map<String, String> verify() {

Map<String, String> verify = new HashMap<>();

verify.put("name", "请输入作业名称");

verify.put("type", "请选择作业类型");

verify.put("score", "请输入作业总分值");

verify.put("trueOrFalse", "请输入判断题总分值");

verify.put("single", "请输入单选题总分值");

verify.put("multiple", "请输入多选题总分值");

verify.put("gap", "请输入填空题总分值");

verify.put("duration", "请输入考试时长");

return verify;

}

private Map<String, String> updateVerify() {

Map<String, String> verify = new HashMap<>();

verify.put("name", "请输入作业名称");

verify.put("type", "请选择作业类型");

verify.put("score", "请输入作业总分值");

verify.put("trueOrFalse", "请输入判断题总分值");

verify.put("single", "请输入单选题总分值");

verify.put("multiple", "请输入多选题总分值");

verify.put("gap", "请输入填空题总分值");

verify.put("duration", "请输入考试时长");

return verify;

}

}

### 4.3、学院管理模块

#### 4.31、AcademyInfo.ejs

<% include ../inc/header.ejs %>

<style>

.date-table {

overflow-y: auto;

}

</style>

<% include ../inc/body.ejs %>

<!-- html主体代码 start -->

<div class="row">

<div class="col-sm-12">

<div class="ibox float-e-margins">

<div class="ibox-title" style="padding-top: 3px !important;">

<div style="width: 140px; display: block;float: left;padding-top: 10px;">

学院部门信息管理

</div>

<div class="ibox-tools">

<button style="margin-top: 8px;" class="btn btn-primary btn-xs" type="button"

onclick="academyInfo.create(this);">

<i class="fa fa-plus"></i>&nbsp;新增

</button>

<!-- <button style="margin-top: 8px;" class="btn btn-primary btn-xs" type="button"-->

<!-- onclick="academyInfo.importModel(this);">-->

<!-- <i class="fa fa-download"></i>&nbsp;下载学院部门信息模板-->

<!-- </button>-->

<!-- <button style="margin-top: 8px;" class="btn btn-primary btn-xs" type="button"-->

<!-- onclick="academyInfo.import(this);">-->

<!-- <i class="fa fa-plus"></i>&nbsp;导入学院部门信息-->

<!-- </button>-->

</div>

</div>

<div class="ibox-content data-view" style="overflow-y: auto;">

<div class="input-group" style="width: 200px;display: flex">

<select style="min-width:200px;min-width: 200px;width: 200px;height: 34px" type="text"

id="select-campus"

class="input-sm form-control">

<option value=" ">全部校区</option>

<option value="1">东校区</option>

<option value="2">西校区</option>

<option value="3">武鸣校区</option>

</select>

<input style="min-width: 200px;width: 200px;" type="text" id="select-name" placeholder="请输入学院名称"

class="input-sm form-control"/>

<input style="min-width: 200px;width: 200px;" type="text" id="select-code" placeholder="请输入学院代码"

class="input-sm form-control"/>

<span class="input-group-btn">

<button onclick="academyInfo.select(this);" id="search\_button"

style="width: 50px;height: 34px" type="button"

class="btn btn-sm btn-primary"> 搜索</button>

</span>

</div>

<div class="ibox-content data-view" style="overflow-y: auto;">

<div class="col-sm-12">

<div class=" col-sm-12 date-table">

<table class="table table-striped">

<thead>

<tr>

<th>序号</th>

<th>学院部门代码</th>

<th>学院部门名称</th>

<td>所在校区</td>

<th>学院部门状态</th>

<th></th>

</tr>

</thead>

<tbody id="item-list">

</tbody>

</table>

</div>

<div class="col-sm-12 pages M-box3 list-page">

</div>

</div>

</div>

</div>

</div>

</div>

<!-- html主体代码 end -->

<!-- 管理的课程模板 -->

<script id="course-template" type="text/x-dot-template">

{{ if(it.length == 0) { }}

暂无数据

{{ }else{ }}

{{ for(var i=0;i

<it.length;i++){ }}

<tr>

<td>

{{= ((index-1) \* size +i+1)}}

</td>

<td>

{{= it[i].code}}

</td>

<td>

{{= it[i].name}}

</td>

<td>

{{= helper.campus(it[i].campus)}}

</td>

<td>

{{= helper.status(it[i].status)}}

</td>

<td>

<div class="btn-group">

<button data-toggle="dropdown" class="btn btn-primary btn-xs dropdown-toggle">操作 <span

class="caret"></span>

</button>

<ul class="dropdown-menu">

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="academyInfo.update(this);">修改</a>

</li>

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="academyInfo.delete(this);">删除</a>

</li>

</ul>

</div>

</td>

</tr>

{{ } }}

{{ } }}

</script>

<script id="error-template" type="text/x-dot-template">

{{ if(it.length == 0) { }}

暂无数据

{{ }else{ }}

{{ for(var i=0;i < it.length;i++){ }}

<tr>

<td>

{{=(i+1)}}

</td>

<td>

{{= it[i].code|| ""}}

</td>

<td>

{{= it[i].name || ""}}

</td>

<td>

{{= it[i].campus|| ""}}

</td>

<td>

{{= it[i].info || ""}}

</td>

</tr>

{{ } }}

{{ } }}

</script>

<% include ../inc/js.ejs %>

<!-- 私有脚本 start -->

<script src="../../js/baseModule/\_academyInfo.js"></script>

<!-- 私有脚本 end -->

<% include ../inc/footer.ejs %>

#### 4.32、AcademyInfoApi.java

package team.work.api.admin;

import com.alibaba.fastjson.JSONArray;

import com.alibaba.fastjson.JSONObject;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.apache.poi.hssf.usermodel.HSSFWorkbook;

import org.apache.poi.ss.usermodel.Row;

import org.apache.poi.ss.usermodel.Sheet;

import org.apache.poi.ss.usermodel.Workbook;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

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

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

import team.work.api.BaseApi;

import team.work.core.model.AcademyInfo;

import team.work.core.service.impl.AcademyInfoService;

import team.work.doc.AcademyInfoCreate;

import team.work.doc.AcademyInfoUpdate;

import team.work.utils.convert.R;

import team.work.utils.convert.S;

import team.work.utils.convert.V;

import team.work.utils.convert.W;

import javax.servlet.http.HttpServletRequest;

import java.io.File;

import java.io.FileInputStream;

import java.util.List;

@Api(value = "12\_学院信息管理")

@RestController

@RequestMapping("/v1/base")

public class AcademyInfoApi extends BaseApi {

@Autowired

private AcademyInfoService academyInfoService;

@PostMapping("/academyInfo")

@ApiOperation(value = "添加学院信息")

public Object createAcademyInfo(@RequestBody AcademyInfoCreate object

, @RequestHeader("token") String token) {

String userId = getUserIdByCache(token);//获取操作人id

//映射对象

AcademyInfo model = o2c(object, token, AcademyInfo.class);

Boolean exist = academyInfoService.exist(W.f(

W.and("code", "eq", model.getCode()),

W.and("isDel", "eq", "0"))

);

if (exist)

return R.error("代码已经存在请更换一个代码");

Boolean nameexist = academyInfoService.exist(W.f(

W.and("name", "eq", model.getName()),

W.and("isDel", "eq", "0"))

);

if (nameexist)

return R.error("名称已经存在请更换一个名称");

model = academyInfoService.createAcademyInfo(model);

if (model == null)

return R.error("添加失败");

return R.ok("添加成功", fm2(model));

}

@PutMapping("/academyInfo")

@ApiOperation(value = " 修改学院信息")

public Object updateAcademyInfo(@RequestBody AcademyInfoUpdate object, @RequestHeader("token") String token) {

String userId = getUserIdByCache(token);//获取操作人id

//映射对象

AcademyInfo model = o2c(object, token, AcademyInfo.class);

AcademyInfo data = academyInfoService.selectById(model.getId());

if (data == null) {

return R.error("该信息不存在，无法修改");

}

if (!model.getCode().equals(data.getCode())) {

Boolean exist = academyInfoService.exist(W.f(

W.and("code", "eq", model.getCode()),

W.and("isDel", "eq", "0"))

);

if (exist)

return R.error("代码已经存在请更换一个代码");

}

if (!model.getName().equals(data.getName())) {

Boolean nameexist = academyInfoService.exist(W.f(

W.and("name", "eq", model.getName()),

W.and("isDel", "eq", "0"))

);

if (nameexist)

return R.error("名称已经存在请更换一个名称");

}

model.setReviser(userId);

model = academyInfoService.updateAcademyInfo(model);

if (model == null)

return R.error("修改失败");

return R.ok("修改成功", fm2(model));

}

@DeleteMapping("/academyInfo/{id}")

@ApiOperation(value = "删除学院信息")

public Object deleteAcademyInfo(@PathVariable("id") String id, @RequestHeader("token") String token) {

if (!academyInfoService.existId(id))

return R.error("Id数据异常");

if (academyInfoService.delete(id, cacheKit.getUserId(token)))

return R.ok("删除成功");

return R.error("删除失败");

}

@GetMapping("/academyInfo/{index}-{size}-{name}-{code}-{campus}")

@ApiOperation(value = "读取学院信息分页列表")

public Object getAcademyInfo(@PathVariable("index") int index,

@PathVariable("size") int size,

@PathVariable("name") String name,

@PathVariable("code") String code,

@PathVariable("campus") String campus,

@RequestHeader("token") String token) {

//前端的搜索条件（学院代码，名称，校区）

String wKey = "";

if (!V.isEmpty(name))

wKey = S.apppend(" and (name like '%", name, "%')");

if (!V.isEmpty(code))

wKey += S.apppend(" and (code like '%", code, "%')");

if (!V.isEmpty(campus))

wKey += S.apppend(" and (campus = '", campus, "')");

return R.ok(academyInfoService.page(index, size, wKey));

}

@GetMapping("/academyInfo")

@ApiOperation(value = "读取学院所有列表")

public Object getAllAcademyInfo(@RequestHeader("token") String token) {

return R.ok(academyInfoService.queryAll(""));

}

//通过学院代码找所在校区 /\*需单独配置权限\*/

@GetMapping("/academyInfo/campus/{code}")

@ApiOperation(value = "通过学院代码找所在校区")

public Object getAcademyInfo(

@PathVariable("code") String code,

@RequestHeader("token") String token) {

String wKey = "";

if (!V.isEmpty(code))

wKey = S.apppend(" and code = '", code, "'");

return R.ok(academyInfoService.getCampusByCode(wKey));

}

@PostMapping("/academyInfo/import/{batch}")

@ApiOperation(value = "导入学院")

public Object Import(

HttpServletRequest req,

@PathVariable("batch") String batch,

@RequestHeader("token") String token) {

if (V.isEmpty(batch)) {

return R.error("导入批次有误！");

}

if(V.isEmpty( cacheKit.getVal("academy-import-lock-"+batch))) {

cacheKit.setVal("academy-import-lock-"+batch, "true", 1800);

}else{

return R.error("导入操作正在进行中，请耐心等待操作结果");

}

String filepath = uploadFile(req);

if (V.isEmpty(filepath)) {

return R.error("文件上传失败！");

}

JSONObject result = new JSONObject();

JSONObject obj = new JSONObject();

obj.put("token", token);

obj.put("filepath", filepath);

JSONObject res = SolvedFile(obj, batch);

if (res == null) {

return R.error("学院导入异常！");

}

Boolean flag =res.getBoolean("flag");

if(!flag){

return R.error(res.getString("message"));

}

result.put("total", res.getInteger("total"));

result.put("truecount", res.getInteger("truecount"));

result.put("errcount", res.getInteger("errcount"));

if (res.getInteger("truecount") == 0 && res.getInteger("errcount") == 0) {

return R.error("导入数据为空！");

}

if (res.getInteger("errcount") > 0) {

result.put("errlist", res.getJSONArray("errlist"));

result.put("success", 0);

} else {

result.put("success", 1);

}

cacheKit.deleteVal("academy-import-lock-"+batch);

return R.ok("导入成功", result);

}

@PostMapping("/academyInfo/importStatus/{batch}")

@ApiOperation(value = "导入学院状态")

public Object getImportStatus(@PathVariable("batch") String batch) {

if (V.isEmpty(batch)) {

return R.error("导入批次有误！");

}

JSONObject obj = getImportStatusObj(batch);

if (V.isEmpty(batch)) {

return R.error("导入状态异常！");

}

return R.ok(obj);

}

public JSONObject SolvedFile(JSONObject obj, String batch) {

JSONObject res = new JSONObject();

JSONArray errors = new JSONArray();

if (obj.getString("filepath") == null || obj.getString("filepath") == "") {

res.put("flag",false);

res.put("message","上传文件有误！");

return res;

}

if (obj.getString("token") == null || obj.getString("token") == "") {

res.put("flag",false);

res.put("message","用户授权失败！");

return res;

}

String token = obj.getString("token");

try {

File excel = new File(obj.getString("filepath"));

if (excel.isFile() && excel.exists()) { //判断文件是否存在

String[] split = excel.getName().split("\\."); //.是特殊字符，需要转义！！！！！

Workbook wb;

//根据文件后缀（xls/xlsx）进行判断

if ("xls".equals(split[1])) {

FileInputStream fis = new FileInputStream(excel); //文件流对象

wb = new HSSFWorkbook(fis);

} else if ("xlsx".equals(split[1])) {

wb = new XSSFWorkbook(excel);

} else {

res.put("flag",false);

res.put("message","上传文件有误！");

return res;

}

//开始解析

Sheet sheet = wb.getSheetAt(0); //读取sheet

int firstRowIndex = sheet.getFirstRowNum() + 1; //第一行是标题所以不读

int lastRowIndex = sheet.getLastRowNum();

int dealCount = 0, trueCount = 0;

for (int rIndex = firstRowIndex; rIndex <= lastRowIndex; rIndex++) { //遍历行

// System.out.println("rIndex: " + rIndex);

Row row = sheet.getRow(rIndex);

if (row != null) {

String name = "", code = "", campus = "";

if (row.getCell(0) != null && row.getCell(1) != null && row.getCell(2) != null) {

JSONObject academyobj = new JSONObject();

code = row.getCell(0).toString().trim();//学院代码

name = row.getCell(1).toString().trim();//学院名称

campus = row.getCell(2).toString().trim();//所在校区

academyobj.put("code", code);

academyobj.put("name", name);

academyobj.put("campus", campus);

boolean academyFlag = false;

String userId = getUserIdByCache(token);

dealCount++;

Boolean nameexist = academyInfoService.exist(W.f(

W.and("name", "eq", name),

W.and("isDel", "eq", "0"))

);

Boolean codeexist = academyInfoService.exist(W.f(

W.and("code", "eq", code),

W.and("isDel", "eq", "0"))

);

if (nameexist || codeexist) {

academyFlag = false;

if (nameexist) {

academyobj.put("info", "学院名称已存在");

}

if (codeexist) {

academyobj.put("info", "学院代码已存在");

}

} else {

if (campus.equals("东校区") || campus.equals("西校区") || campus.equals("武鸣校区")) {

int campus\_int = -1;

switch (campus) {

case "东校区":

campus\_int = 1;

break;

case "西校区":

campus\_int = 2;

break;

case "武鸣校区":

campus\_int = 3;

break;

}

AcademyInfo academyInfo = new AcademyInfo();

academyInfo.setCampus(campus\_int);

academyInfo.setCode(code);

academyInfo.setName(name);

academyInfo.setCreator(userId);

academyInfo.setStatus(0);

academyInfo = academyInfoService.createAcademyInfo(academyInfo);

if (academyInfo == null) {

academyFlag = false;

academyobj.put("info", "学院信息保存失败");

} else {

academyFlag = true;

}

} else {

academyFlag = false;

academyobj.put("info", "校区错误");

}

}

if (!academyFlag) {

if (V.isEmpty(academyobj.getString("info"))) {

academyobj.put("info", "学院信息创建失败");

}

errors.add(academyobj);

cacheKit.setVal(batch + "status", 2 + "", 0);

} else {

trueCount++;

cacheKit.setVal(batch + "status", 1 + "", 0);

}

cacheKit.setVal(batch + "now", rIndex + "", 0);

}

cacheKit.setVal(batch + "total", lastRowIndex + "", 0);

if (rIndex != lastRowIndex) {

cacheKit.setVal(batch + "res", 2 + "", 0);

}

}

}

cacheKit.setVal(batch + "res", 1 + "", 0);

res.put("truecount", trueCount);

res.put("errcount", errors.size());

res.put("total", dealCount);

res.put("errlist", errors);

res.put("flag",true);

return res;

} else {

res.put("flag", false);

res.put("message", "上传文件有误！");

System.out.println("找不到指定的文件");

return res;

}

} catch (Exception e) {

e.printStackTrace();

res.put("flag", false);

res.put("message", "学院处理出错！");

return res;

}

}

}

### 4.4、专业管理模块

#### 4.41、MajorInfo.ejs

<% include ../inc/header.ejs %>

<style>

.date-table {

overflow-y: auto;

}

</style>

<% include ../inc/body.ejs %>

<!-- html主体代码 start -->

<div class="row">

<div class="col-sm-12">

<div class="ibox float-e-margins">

<div class="ibox-title" style="padding-top: 3px !important;">

<div style="width: 140px; display: block;float: left;padding-top: 10px;">

专业信息管理

</div>

<div class="ibox-tools">

<button style="margin-top: 8px;" class="btn btn-primary btn-xs" type="button"

onclick="majorInfo.create(this);">

<i class="fa fa-plus"></i>&nbsp;新增

</button>

<!-- <button style="margin-top: 8px;" class="btn btn-primary btn-xs" type="button"-->

<!-- onclick="majorInfo.importModel(this);">-->

<!-- <i class="fa fa-download"></i>&nbsp;下载专业信息模板-->

<!-- </button>-->

<!-- <button style="margin-top: 8px;" class="btn btn-primary btn-xs" type="button"-->

<!-- onclick="majorInfo.import(this);">-->

<!-- <i class="fa fa-plus"></i>&nbsp;导入专业信息-->

<!-- </button>-->

</div>

</div>

<div class="ibox-content">

<div class="input-group" style="width: 200px; display: flex" id="search-key">

<input style="min-min-width: 200px;width: 200px;min-width: 200px;width: 200px;height:30px;"

type="text" id="name" placeholder="请输入专业名称" class="input-sm form-control">

<span class="input-group-btn" style="float:left;flex: 4">

<button onclick="majorInfo.select(this);" id="search\_button" style="width: 50px;"

type="button" class="btn btn-sm btn-primary"> 搜索</button>

</span>

</div>

</div>

<div class="ibox-content data-view" style="overflow-y: auto;">

<div class="col-sm-12">

<div class=" col-sm-12 date-table">

<table class="table table-striped">

<thead>

<tr>

<th>序号</th>

<th>专业代码</th>

<th>专业名称</th>

<th>所属学院</th>

<th>状态</th>

<th></th>

</tr>

</thead>

<tbody id="item-list">

</tbody>

</table>

</div>

<div class="col-sm-12 pages M-box3 list-page">

</div>

</div>

</div>

</div>

</div>

</div>

<!-- html主体代码 end -->

<!-- 管理的课程模板 -->

<script id="major-template" type="text/x-dot-template">

{{ if(it.length == 0) { }}

暂无数据

{{ }else{ }}

{{ for(var i=0;i < it.length;i++){ }}

<tr>

<td>

{{= ((index-1) \* size +i+1)}}

</td>

<td>

{{= it[i].code}}

</td>

<td>

{{= it[i].name }}

</td>

<td>

{{= it[i].academyName }}

</td>

<td>

{{= helper.status(it[i].status) }}

</td>

<td>

<div class="btn-group">

<button data-toggle="dropdown" class="btn btn-primary btn-xs dropdown-toggle">操作 <span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="majorInfo.update(this);">修改</a>

</li>

<!--<li>-->

<!--<a href="#" data-id="{{= it[i].id}}" onclick="majorInfo.read(this);">查询</a>-->

<!--</li>-->

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="majorInfo.delete(this);">删除</a>

</li>

</ul>

</div>

</td>

</tr>

{{ } }}

{{ } }}

</script>

<script id="academy-select-template" type="text/x-dot-template">

{{ if(it.length == 0 ){ }}

暂无数据

{{ }else{ }}

<option value=" ">请选择学院</option>

{{ for(var i = 0 ; i < it.length ; i++){ }}

<option value="{{=it[i].code }}">{{=it[i].name }}</option>

{{ } }}

{{ } }}

</script>

<script id="error-template" type="text/x-dot-template">

{{ if(it.length == 0) { }}

暂无数据

{{ }else{ }}

{{ for(var i=0;i < it.length;i++){ }}

<tr>

<td>

{{=(i+1)}}

</td>

<td>

{{= it[i].code|| ""}}

</td>

<td>

{{= it[i].name || ""}}

</td>

<td>

{{= it[i].academyName || ""}}

</td>

<td>

{{= it[i].info || ""}}

</td>

</tr>

{{ } }}

{{ } }}

</script>

<% include ../inc/js.ejs %>

<!-- 私有脚本 start -->

<script src="../../js/baseModule/\_majorInfo.js"></script>

<!-- 私有脚本 end -->

<% include ../inc/footer.ejs %>

#### 4.42、MajorInfoApi.java

package team.work.api.admin;

import com.alibaba.fastjson.JSONArray;

import com.alibaba.fastjson.JSONObject;

import team.work.api.BaseApi;

import team.work.api.BaseApi;

import team.work.core.model.MajorInfo;

import team.work.core.service.impl.AcademyInfoService;

import team.work.core.service.impl.MajorInfoService;

import team.work.doc.MajorInfoCreate;

import team.work.doc.MajorInfoUpdate;

import team.work.utils.bean.Page;

import team.work.utils.convert.R;

import team.work.utils.convert.S;

import team.work.utils.convert.V;

import team.work.utils.convert.W;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.apache.poi.hssf.usermodel.HSSFWorkbook;

import org.apache.poi.ss.usermodel.Row;

import org.apache.poi.ss.usermodel.Sheet;

import org.apache.poi.ss.usermodel.Workbook;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

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

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

import team.work.core.model.MajorInfo;

import team.work.doc.MajorInfoCreate;

import team.work.doc.MajorInfoUpdate;

import team.work.utils.convert.S;

import team.work.utils.convert.V;

import team.work.utils.convert.W;

import javax.servlet.http.HttpServletRequest;

import java.io.File;

import java.io.FileInputStream;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

@Api(value = "06\_专业信息管理")

@RestController

@RequestMapping("/v1/base")

public class MajorInfoApi extends BaseApi {

@Autowired

private MajorInfoService majorInfoService;

@Autowired

private AcademyInfoService academyInfoService;

@PostMapping("/majorInfo")

@ApiOperation(value = "专业信息新增")

public Object createMajorInfo(@RequestBody MajorInfoCreate object,

@RequestHeader("token") String token) {

String userId = getUserIdByCache(token);

//映射对象

MajorInfo model = o2c(object, token, MajorInfo.class);

//数据校验

JSONObject check = V.checkEmpty(verify(), object);

if (check.getBoolean("check"))

return R.error(check.getString("message"));

Boolean exist = majorInfoService.exist(W.f(

W.and("code", "eq", model.getCode()),

W.and("isDel", "eq", "0"))

);

if (exist)

return R.error("代码已经存在请更换一个代码");

Boolean nameexist = majorInfoService.exist(W.f(

W.and("name", "eq", model.getName()),

W.and("isDel", "eq", "0"))

);

if (nameexist)

return R.error("名称已经存在请更换一个名称");

model = majorInfoService.createMajorInfo(model);

if (model == null)

return R.error("新增失败");

return R.ok("新增成功", fm2(model));

}

@PutMapping("/majorInfo")

@ApiOperation(value = "修改专业信息")

public Object updateMajorInfo(@RequestBody MajorInfoUpdate object,

@RequestHeader("token") String token) {

String userId = getUserIdByCache(token);

//映射对象

MajorInfo model = o2c(object, token, MajorInfo.class);

//数据校验

JSONObject check = V.checkEmpty(updateVerify(), object);

if (check.getBoolean("check"))

return R.error(check.getString("message"));

MajorInfo data = majorInfoService.selectById(model.getId());

if (data == null) {

return R.error("该信息不存在，无法修改");

}

if (!model.getCode().equals(data.getCode())) {

Boolean exist = majorInfoService.exist(W.f(

W.and("code", "eq", model.getCode()),

W.and("isDel", "eq", "0"))

);

if (exist)

return R.error("代码已经存在请更换一个代码");

}

if (!model.getName().equals(data.getName())) {

Boolean nameexist = majorInfoService.exist(W.f(

W.and("name", "eq", model.getName()),

W.and("isDel", "eq", "0"))

);

if (nameexist)

return R.error("名称已经存在请更换一个名称");

}

model.setReviser(userId);

model = majorInfoService.updateMajorInfo(model);

if (model == null)

return R.error("修改失败");

return R.ok("修改成功", fm2(model));

}

@DeleteMapping("/majorInfo/{id}")

@ApiOperation(value = "专业信息删除")

public Object deleteMajorInfo(@PathVariable("id") String id,

@RequestHeader("token") String token) {

if (!majorInfoService.existId(id))

return R.error("Id数据异常");

if (majorInfoService.deleteMajorInfo(id, cacheKit.getUserId(token)))

return R.ok("删除成功");

return R.error("删除失败");

}

@GetMapping("/majorInfo/{index}-{size}-{name}")

@ApiOperation(value = "读取专业信息分页列表")

public Object getMajorInfo(@PathVariable("index") int index,

@PathVariable("size") int size,

@PathVariable("name") String name,

@RequestHeader("token") String token) {

String wKey = "";

if (!V.isEmpty(name))

wKey = S.apppend(" and a.name like '%", name, "%' ");

// if (!V.isEmpty(code))

// wKey += S.apppend(" and (code = '", code , "')");

return R.ok(majorInfoService.page(index, size, wKey));

}

@GetMapping("/majorInfo")

@ApiOperation(value = "读取专业信息所有列表")

public Object getAllMajorInfo(@RequestHeader("token") String token) {

return R.ok(majorInfoService.queryAll(""));

}

@GetMapping("/majorInfoByAcademy/{academyCode}")

@ApiOperation(value = "读取专业信息列表by学院code")

public Object getAllMajorInfoByAcademyCode(@PathVariable("academyCode") String academyCode,

@RequestHeader("token") String token) {

if (!V.isEmpty(academyCode))

academyCode = S.apppend(" and academyCode = '", academyCode, "' ");

return R.ok(majorInfoService.queryAll(academyCode));

}

@PostMapping("/majorInfo/import/{batch}")

@ApiOperation(value = "导入专业")

public Object Import(

HttpServletRequest req,

@PathVariable("batch") String batch,

@RequestHeader("token") String token) {

if (V.isEmpty(batch)) {

return R.error("导入批次有误！");

}

if(V.isEmpty( cacheKit.getVal("major-import-lock-"+batch))) {

cacheKit.setVal("major-import-lock-"+batch, "true", 1800);

}else{

return R.error("导入操作正在进行中，请耐心等待操作结果");

}

String filepath = uploadFile(req);

if (V.isEmpty(filepath)) {

return R.error("文件上传失败！");

}

JSONObject result = new JSONObject();

JSONObject obj = new JSONObject();

obj.put("token", token);

obj.put("filepath", filepath);

JSONObject res = SolvedFile(obj, batch);

if (res == null) {

return R.error("专业导入异常！");

}

Boolean flag =res.getBoolean("flag");

if(!flag){

return R.error(res.getString("message"));

}

result.put("total", res.getInteger("total"));

result.put("truecount", res.getInteger("truecount"));

result.put("errcount", res.getInteger("errcount"));

if (res.getInteger("truecount") == 0 && res.getInteger("errcount") == 0) {

return R.error("导入数据为空！");

}

if (res.getInteger("errcount") > 0) {

result.put("errlist", res.getJSONArray("errlist"));

result.put("success", 0);

} else {

result.put("success", 1);

}

cacheKit.deleteVal("major-import-lock-"+batch);

return R.ok("导入成功", result);

}

@PostMapping("/majorInfo/importStatus/{batch}")

@ApiOperation(value = "导入专业状态")

public Object getImportStatus(@PathVariable("batch") String batch) {

if (V.isEmpty(batch)) {

return R.error("导入批次有误！");

}

JSONObject obj = getImportStatusObj(batch);

if (V.isEmpty(batch)) {

return R.error("导入状态异常！");

}

return R.ok(obj);

}

public JSONObject SolvedFile(JSONObject obj, String batch) {

JSONObject res = new JSONObject();

JSONArray errors = new JSONArray();

if (obj.getString("filepath") == null || obj.getString("filepath") == "") {

res.put("flag",false);

res.put("message","上传文件有误！");

return res;

}

if (obj.getString("token") == null || obj.getString("token") == "") {

res.put("flag",false);

res.put("message","用户授权失败！");

return res;

}

String token = obj.getString("token");

try {

File excel = new File(obj.getString("filepath"));

if (excel.isFile() && excel.exists()) { //判断文件是否存在

String[] split = excel.getName().split("\\."); //.是特殊字符，需要转义！！！！！

Workbook wb;

//根据文件后缀（xls/xlsx）进行判断

if ("xls".equals(split[1])) {

FileInputStream fis = new FileInputStream(excel); //文件流对象

wb = new HSSFWorkbook(fis);

} else if ("xlsx".equals(split[1])) {

wb = new XSSFWorkbook(excel);

} else {

res.put("flag",false);

res.put("message","上传文件有误！");

return res;

}

//开始解析

Sheet sheet = wb.getSheetAt(0); //读取sheet

int firstRowIndex = sheet.getFirstRowNum() + 1; //第一行是标题所以不读

int lastRowIndex = sheet.getLastRowNum();

int dealCount = 0, trueCount = 0;

for (int rIndex = firstRowIndex; rIndex <= lastRowIndex; rIndex++) { //遍历行

// System.out.println("rIndex: " + rIndex);

Row row = sheet.getRow(rIndex);

if (row != null) {

String name = "", code = "", academyName = "";

if (row.getCell(0) != null && row.getCell(1) != null && row.getCell(2) != null) {

JSONObject importObj = new JSONObject();

code = row.getCell(0).toString().trim();//专业代码

name = row.getCell(1).toString().trim();//专业名称

academyName = row.getCell(2).toString().trim();//学院名称

importObj.put("code", code);

importObj.put("name", name);

importObj.put("academyName", academyName);

boolean importFlag = false;

String userId = getUserIdByCache(token);

dealCount++;

Boolean nameexist = majorInfoService.exist(W.f(

W.and("name", "eq", name),

W.and("isDel", "eq", "0"))

);

Boolean codeexist = majorInfoService.exist(W.f(

W.and("code", "eq", code),

W.and("isDel", "eq", "0"))

);

if (nameexist || codeexist) {

importFlag = false;

if (nameexist) {

importObj.put("info", "专业名称已存在");

}

if (codeexist) {

importObj.put("info", "专业代码已存在");

}

} else {

JSONObject academyObj = academyInfoService.queryByName(academyName);

if (V.isEmpty(academyObj)) {

importFlag = false;

importObj.put("info", "学院名称不存在");

} else {

MajorInfo info = new MajorInfo();

info.setCode(code);

info.setName(name);

info.setCreator(userId);

info.setAcademyCode(academyObj.getString("code"));

info = majorInfoService.createMajorInfo(info);

if (info == null) {

importFlag = false;

importObj.put("info", "专业信息保存失败");

} else {

importFlag = true;

}

}

}

if (!importFlag) {

if (V.isEmpty(importObj.getString("info"))) {

importObj.put("info", "专业信息创建失败");

}

errors.add(importObj);

cacheKit.setVal(batch + "status", 2 + "", 0);

} else {

trueCount++;

cacheKit.setVal(batch + "status", 1 + "", 0);

}

cacheKit.setVal(batch + "now", rIndex + "", 0);

}

cacheKit.setVal(batch + "total", lastRowIndex + "", 0);

if (rIndex != lastRowIndex) {

cacheKit.setVal(batch + "res", 2 + "", 0);

}

}

}

cacheKit.setVal(batch + "res", 1 + "", 0);

res.put("truecount", trueCount);

res.put("errcount", errors.size());

res.put("total", dealCount);

res.put("errlist", errors);

res.put("flag",true);

return res;

} else {

res.put("flag", false);

res.put("message", "上传文件有误！");

System.out.println("找不到指定的文件");

return res;

}

} catch (Exception e) {

e.printStackTrace();

res.put("flag", false);

res.put("message", "专业处理出错！");

return res;

}

}

private Map<String, String> verify() {

Map<String, String> verify = new HashMap<>();

verify.put("code", "请输入专业代码");

verify.put("academyCode", "请输入所属学院");

verify.put("name", "请输入专业名称");

verify.put("status", "请输入状态");

return verify;

}

private Map<String, String> updateVerify() {

Map<String, String> verify = new HashMap<>();

verify.put("code", "请输入专业代码");

verify.put("academyCode", "请选择所属学院");

verify.put("name", "请输入专业名称");

verify.put("status", "请输入状态");

return verify;

}

//通过学院代码找该学院所有专业

@GetMapping("/majorInfo/majors/{academyCode}")

@ApiOperation(value = "通过学院代码找该学院所有专业")

public Object getAcademyInfo(@PathVariable("academyCode") String academyCode,

@RequestHeader("token") String token) {

String wKey = "";

if (!V.isEmpty(academyCode))

wKey = S.apppend(" and academyCode = '", academyCode, "'");

return R.ok(majorInfoService.getMajorByAcademyCode(wKey));

}

}

### 4.5、题库管理模块

#### 4.51、Question.ejs

<% include ../inc/header.ejs %>

<style>

.date-table {

overflow-y: auto;

}

</style>

<link href="../../assets/js/summernote/summernote.css" rel="stylesheet">

<% include ../inc/body.ejs %>

<!-- html主体代码 start -->

<div class="row">

<div class="col-sm-12">

<div class="ibox float-e-margins">

<div class="ibox-title" style="padding-top: 3px !important;">

<div style="width: 140px; display: block;float: left;padding-top: 10px;">

题库列表

</div>

<div class="ibox-tools">

<!--<button style="margin-top: 8px;" class="btn btn-primary btn-xs" type="button"-->

<!--onclick="question.create(this);">-->

<!--<i class="fa fa-plus"></i>&nbsp;新增題目-->

<!--</button>-->

<button style="margin-top: 8px;" class="btn btn-primary btn-xs" type="button"

onclick="question.make(this);">

<i class="fa fa-plus"></i>&nbsp;创建

</button>

<!-- <button style="margin-top: 8px;" class="btn btn-primary btn-xs" type="button"-->

<!-- onclick="question.importModel(this);">-->

<!-- <i class="fa fa-download"></i>&nbsp;下载题库信息模板-->

<!-- </button>-->

<!-- <button style="margin-top: 8px;" class="btn btn-primary btn-xs" type="button"-->

<!-- onclick="question.import(this);">-->

<!-- <i class="fa fa-plus"></i>&nbsp;导入题库信息-->

<!-- </button>-->

</div>

</div>

<div class="ibox-content data-view" style="overflow-y: auto;">

<div class="input-group" style="width: 200px;display: flex">

<select style="min-width: 200px;width: 200px;height: 34px" type="text" id="select-type"

class="input-sm form-control">

<option value=" ">全部类型</option>

<option value="1">判断</option>

<option value="2">单选</option>

<option value="3">多选</option>

<option value="4">填空</option>

<option value="5">客观题</option>

</select>

<input style="min-width: 200px;width: 200px;" type="text" id="select-name" placeholder="请输入题目名称"

class="input-sm form-control"/>

<span class="input-group-btn">

<button onclick="question.select(this);" id="search\_button" style="width: 50px;height: 34px"

type="button"

class="btn btn-sm btn-primary"> 搜索</button>

</span>

</div>

<div class="col-sm-12">

<div class=" col-sm-12 date-table">

<table class="table table-hover">

<thead>

<tr>

<th>序号</th>

<th>题目名称</th>

<th>类型</th>

<!--<th>答案解析</th>-->

<th></th>

</tr>

</thead>

<tbody id="item-list">

</tbody>

</table>

</div>

<div class="col-sm-12 pages M-box3 list-page">

</div>

</div>

</div>

</div>

</div>

</div>

<!-- html主体代码 end -->

<!-- 管理的题库模板 -->

<script id="question-template" type="text/x-dot-template">

{{ if(it.length == 0) { }}

暂无数据

{{ }else{ }}

{{ for(var i=0;i < it.length;i++){ }}

<tr>

<td>

{{= ((index-1) \* size +i+1)}}

</td>

<td>

{{= it[i].questionName}}

</td>

<td>

{{= helper.type(it[i].type) }}

</td>

<!--<td>-->

<!--{{= it[i].answerKeys}}-->

<!--</td>-->

<td>

<div class="btn-group">

<button data-toggle="dropdown" class="btn btn-primary btn-xs dropdown-toggle">操作 <span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="question.update(this);">修改</a>

</li>

<!--<li>-->

<!--<a href="#" data-id="{{= it[i].id}}" onclick="question.delete(this);">删除</a>-->

<!--</li>-->

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="question.deleteQuestionAndOptions(this);">删除</a>

</li>

</ul>

</div>

</td>

</tr>

{{ } }}

{{ } }}

</script>

<% include ../inc/js.ejs %>

<script>

// 选项序号

var optionAlph = ['A', 'B', 'C', 'D', 'E', 'F', 'G'];

</script>

<!--判断题选项模板-->

<script id="true-or-false" type="text/x-dot-template">

<tr>

<td style="width: 120px; text-align: center">选项</td>

<td>选项内容</td>

<td style="width: 120px;">请选择正确答案</td>

</tr>

{{ if(typeof it == "undefined" || it.length == 0){ }}

<tr>

<td style="text-align: center">1</td>

<td>正确</td>

<td><input type="radio" name="answerNumber" value="1"></td>

</tr>

<tr>

<td style="text-align: center">2</td>

<td>错误</td>

<td><input type="radio" name="answerNumber" value="2"></td>

</tr>

{{ }else{ }}

{{ for(var i = 0; i < it.length; i++) { }}

<tr>

<td style="text-align: center">{{=(i+1) }}</td>

<td>{{=it[i].optionInfo }}</td>

{{ if(it[i].isAnswer == 1 ) }}

<td><input checked type="radio" name="answerNumber" value="{{=(i+1) }}"></td>

{{ else }}

<td><input type="radio" name="answerNumber" value="{{=(i+1) }}"></td>

{{ } }}

</tr>

{{ } }}

</script>

<!--单选题-->

<script id="single-choice" type="text/x-dot-template">

<tr>

<td class="addSingleOption" style="text-align: center"><span class="glyphicon glyphicon-plus-sign "

aria-hidden="true"></span></td>

<td style="width: 120px; text-align: center">选项</td>

<td>选项内容</td>

<td style="width: 120px;">请选择正确答案</td>

</tr>

{{ if(typeof it == "undefined" || it.length == 0){ }}

{{ for(var i = 0; i < 3; i++) { }}

<tr class="singleOption">

<td style="text-align: center"><span class="glyphicon minusSingleOption glyphicon-minus"

aria-hidden="true"></span></td>

<input type="hidden" name="optionNumber" value="{{=(i+1) }}">

<td class="singleNumber" style="text-align: center">{{= optionAlph[i] }}</td>

<td><input required type="text" name="optionInfo" class="form-control" placeholder="请输入选项"></td>

<td><input type="radio" name="answerNumber" value="{{=(i+1) }}"></td>

</tr>

{{ } }}

{{ }else{ }}

{{ for(var i = 0; i < it.length; i++) { }}

<tr class="singleOption">

<td style="text-align: center"><span class="glyphicon minusSingleOption glyphicon-minus"

aria-hidden="true"></span></td>

<input type="hidden" name="optionNumber" value="{{=(i+1) }}">

<td class="singleNumber" style="text-align: center">{{= optionAlph[i] }}</td>

<td><input required type="text" name="optionInfo" class="form-control" placeholder="请输入选项"

value="{{=it[i].optionInfo }}"></td>

{{ if(it[i].isAnswer == 1) }}

<td><input checked type="radio" name="answerNumber" value="{{=(i+1) }}"></td>

{{ else }}

<td><input type="radio" name="answerNumber" value="{{=(i+1) }}"></td>

{{ } }}

</tr>

{{ } }}

</script>

<!--多选题-->

<script id="multiple-choice" type="text/x-dot-template">

<tr>

<td class="addMultipleOption" style="text-align: center"><span class="glyphicon glyphicon-plus-sign "

aria-hidden="true"></span></td>

<td style="width: 120px; text-align: center">选项</td>

<td>选项内容</td>

<td style="width: 120px;">请选择正确答案</td>

</tr>

{{ if(typeof it == "undefined" || it.length == 0){ }}

{{ for(var i = 0; i < 4; i++) {}}

<tr class="multipleOption">

<td style="text-align: center"><span class="glyphicon minusMultipleOption glyphicon-minus"

aria-hidden="true"></span></td>

<input type="hidden" name="optionNumber" value="{{=(i+1) }}">

<td class="multipleNumber" style="text-align: center">{{= optionAlph[i] }}</td>

<td><input required type="text" name="optionInfo" class="form-control" placeholder="请输入选项"></td>

<td><input type="checkbox" name="answerNumber" value="{{=(i+1) }}"></td>

</tr>

{{ } }}

{{ }else{ }}

{{ for(var i = 0; i < 4; i++) {}}

<tr class="multipleOption">

<td style="text-align: center"><span class="glyphicon minusMultipleOption glyphicon-minus"

aria-hidden="true"></span></td>

<input type="hidden" name="optionNumber" value="{{=(i+1) }}">

<td class="multipleNumber" style="text-align: center">{{= optionAlph[i] }}</td>

<td><input required type="text" name="optionInfo" class="form-control" placeholder="请输入选项"

value="{{=it[i].optionInfo }}"></td>

{{ if(it[i].isAnswer == 1) }}

<td><input checked type="checkbox" name="answerNumber" value="{{=(i+1) }}"></td>

{{ else }}

<td><input type="checkbox" name="answerNumber" value="{{=(i+1) }}"></td>

{{ } }}

</tr>

{{ } }}

</script>

<!--填空题-->

<script id="gap-filling" type="text/x-dot-template">

<tr>

<td class="addGapOption" style="text-align: center"><span class="glyphicon glyphicon-plus-sign "

aria-hidden="true"></span></td>

<td style="text-align: center;width: 120px" class="addGapOption">对应空格</td>

<td>答案</td>

</tr>

{{ if(typeof it == "undefined" || it.length == 0){ }}

{{ for(var i = 0; i < 1; i++) {}}

<tr class="gapOption">

<td style="text-align: center"><span class="glyphicon minusGapOption glyphicon-minus" aria-hidden="true"></span>

</td>

<td class="gapNumber" style="text-align: center">{{=(i+1) }}</td>

<td><input required type="text" name="optionInfo" class="form-control" placeholder="请输入答案"></td>

</tr>

{{ } }}

{{ }else{ }}

{{ for(var i = 0; i < it.length; i++) {}}

<tr class="gapOption">

<td style="text-align: center"><span class="glyphicon minusGapOption glyphicon-minus" aria-hidden="true"></span>

</td>

<td class="gapNumber" style="text-align: center">{{=(i+1) }}</td>

<td><input required type="text" name="optionInfo" class="form-control" placeholder="请输入答案"

value="{{=it[i].optionInfo }}"></td>

</tr>

{{ } }}

{{ } }}

</script>

<script>

var optionAlph = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L'];

var singleOption = 3;//单选题初始选项数

//增删单选项

$(document).on('click', '.addSingleOption', function () {

var index = 0;

var len = $('.singleOption').length;

$('.singleOption').eq($('.singleOption').length - 1).after(

'<tr class="singleOption">'

+ '<td style="text-align: center"><span class="glyphicon minusSingleOption glyphicon-minus" aria-hidden="true" ></span></td>'

+ '<input type="hidden" name="optionNumber" value="' + (++singleOption) + '">'

+ '<td class="singleNumber" style="text-align: center">' + optionAlph[len] + '</td>'

+ '<td><input required type="text" name="optionInfo" class="form-control" placeholder="请输入选项"></td>\n'

+ '<td><input type="radio" name="answerNumber" value="' + (singleOption) + '"></td>'

+ '</tr>')

});

$(document).on('click', '.minusSingleOption', function () {

if ($('.singleOption').length - 1 == 0) {

return false;

}

$(this).parent().parent('.singleOption').remove();

//重新编号

$('.singleNumber').each(function (i) {

$(this).html(optionAlph[i]);

});

// singleOption--;

});

var multipleOption = 4;//多选题初始选项数

//增删多选项

$(document).on('click', '.addMultipleOption', function () {

var index = 0;

var len = $('.multipleOption').length;

$('.multipleOption').eq($('.multipleOption').length - 1).after(

'<tr class="multipleOption">'

+ '<td style="text-align: center" ><span class="glyphicon minusMultipleOption glyphicon-minus" aria-hidden="true" ></span></td>'

+ '<input type="hidden" name="optionNumber" value="' + (++multipleOption) + '">'

+ '<td class="multipleNumber" style="text-align: center">' + optionAlph[len] + '</td>'

+ '<td><input required type="text" name="optionInfo" class="form-control" placeholder="请输入选项"></td>\n'

+ '<td><input type="checkbox" name="answerNumber" value="' + (multipleOption) + '"></td>'

+ '</tr>')

});

$(document).on('click', '.minusMultipleOption', function () {

var len = $('.multipleOption').length;

if (len - 1 == 0) {

return false;

}

$(this).parent().parent('.multipleOption').remove();

//重新编号

$('.multipleNumber').each(function (i) {

$(this).html(optionAlph[i]);

});

// multipleOption--;

});

var gapOption = 1;//填空题初始选项数

//增删填空项

$(document).on('click', '.addGapOption', function () {

var index = 0;

var len = $('.gapOption').length;

$('.gapOption').eq($('.gapOption').length - 1).after(

'<tr class="gapOption">'

+ '<td style="text-align: center" ><span class="glyphicon minusGapOption glyphicon-minus" aria-hidden="true" ></span></td>'

+ '<td class="gapNumber" style="width: 120px; text-align: center">' + (len + 1) + '</td>'

+ '<td><input required type="text" name="optionInfo" class="form-control" placeholder="请输入答案"></td>'

+ '</tr>')

});

$(document).on('click', '.minusGapOption', function () {

var len = $('.gapOption').length;

if (len - 1 == 0) {

return false;

}

//获取当前点击的是第几个class

// console.log($(this).parent().parent('.gapOption').index())

$(this).parent().parent('.gapOption').remove();

//重新编号

$('.gapNumber').each(function (i) {

$(this).html(i + 1);

});

// gapOption--;

});

</script>

<script id="error-template" type="text/x-dot-template">

{{ if(it.length == 0) { }}

暂无数据

{{ }else{ }}

{{ for(var i=0;i < it.length;i++){ }}

<tr>

<td>

{{=(i+1)}}

</td>

<td>

{{= it[i].name|| ""}}

</td>

<td>

{{= it[i].type || ""}}

</td>

<td>

{{= it[i].info || ""}}

</td>

</tr>

{{ } }}

{{ } }}

</script>

<!-- 私有脚本 start -->

<script src="../../js/baseModule/\_question.js"></script>

<!-- 私有脚本 end -->

<% include ../inc/footer.ejs %>

#### 4.52、QuestionApi.java

package team.work.api.admin;

import com.alibaba.fastjson.JSONArray;

import com.alibaba.fastjson.JSONObject;

import team.work.api.BaseApi;

import team.work.core.model.Options;

import team.work.core.model.Question;

import team.work.core.service.impl.OptionsService;

import team.work.core.service.impl.QuestionService;

import team.work.doc.\*;

import team.work.utils.convert.\*;

import team.work.utils.kit.OSKit;

import team.work.utils.kit.TimeKit;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.apache.poi.hssf.usermodel.HSSFWorkbook;

import org.apache.poi.ss.usermodel.Row;

import org.apache.poi.ss.usermodel.Sheet;

import org.apache.poi.ss.usermodel.Workbook;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

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

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.jdbc.datasource.DataSourceTransactionManager;

import org.springframework.transaction.TransactionDefinition;

import org.springframework.transaction.TransactionStatus;

import org.springframework.transaction.annotation.Transactional;

import org.springframework.transaction.support.DefaultTransactionDefinition;

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

import org.springframework.web.multipart.MultipartHttpServletRequest;

import team.work.core.model.Options;

import team.work.core.model.Question;

import team.work.doc.MakeQuestionAndOptions;

import team.work.doc.QuestionCreate;

import team.work.doc.QuestionUpdate;

import team.work.doc.UpdateQuestionAndOptions;

import team.work.utils.convert.\*;

import javax.servlet.http.HttpServletRequest;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.\*;

@Api(value = "31\_题库管理")

@RestController

@RequestMapping("/v1/base")

public class QuestionApi extends BaseApi {

@Autowired

private QuestionService questionService;

@Autowired

private OptionsService optionsService;

@PostMapping("/question")

@ApiOperation(value = "题目新增")

public Object createQuestion(@RequestBody QuestionCreate object,

@RequestHeader("token") String token) {

//映射对象

Question model = o2c(object, token, Question.class);

//数据校验

JSONObject check = V.checkEmpty(verify(), object);

if (check.getBoolean("check"))

return R.error(check.getString("message"));

model = questionService.createQuestion(model);

if (model == null)

return R.error("新增失败");

return R.ok("新增成功", fm2(model));

}

@PutMapping("/question")

@ApiOperation(value = "修改题目")

public Object updateQuestion(@RequestBody QuestionUpdate object,

@RequestHeader("token") String token) {

String userId = getUserIdByCache(token);

//映射对象

Question model = o2c(object, token, Question.class);

//数据校验

JSONObject check = V.checkEmpty(updateVerify(), object);

if (check.getBoolean("check"))

return R.error(check.getString("message"));

model.setReviser(userId);

model = questionService.updateQuestion(model);

if (model == null)

return R.error("修改失败");

return R.ok("修改成功", fm2(model));

}

@DeleteMapping("/question/{id}")

@ApiOperation(value = "题目删除")

public Object deleteQuestion(@PathVariable("id") String id,

@RequestHeader("token") String token) {

if (!questionService.existId(id))

return R.error("Id数据异常");

if (questionService.deleteQuestion(id, cacheKit.getUserId(token)))

return R.ok("删除成功");

return R.error("删除失败");

}

@GetMapping("/question/{index}-{size}-{key}")

@ApiOperation(value = "读取题库分页列表")

public Object getQuestion(@PathVariable("index") int index,

@PathVariable("size") int size,

@PathVariable("key") String key,

@RequestHeader("token") String token) {

//前端的搜索条件（学院部门代码，名称，校区）

String wKey = "";

if (!V.isEmpty(key))

wKey += S.apppend(" and (id = '", key, "')");

return R.ok(questionService.page(index, size, wKey));

}

@GetMapping("/question")

@ApiOperation(value = "读取题库所有列表")

public Object getAllQuestion(@RequestHeader("token") String token) {

return R.ok(questionService.queryAll(""));

}

@PostMapping("/question/make")

@ApiOperation(value = "题目和选项创建")

@Transactional

public Object makeQuestions(@RequestBody MakeQuestionAndOptions object,

@RequestHeader("token") String token) {

System.out.println("题目和选项创建:" + object);

String userId = getUserIdByCache(token);

List<String> allOptionInfo = object.getOptionInfo();

List<Integer> allOptionNumber = object.getOptionNumber();

List<Integer> allAnswerNumber = object.getAnswerNumber();

Boolean exist = questionService.exist(W.f(

W.and("questionName", "eq", object.getQuestionName()),

W.and("type", "eq", object.getType()),

W.and("isDel", "eq", "0"))

);

if (exist) {

return R.error("题目名称已存在");

}

//新增题目

Question question = o2c(object, token, Question.class);

int type = question.getType();

question.setCreator(userId);

question = questionService.createQuestion(question);

if (question == null)

return R.error("题目" + question.getQuestionName() + "创建失败");

//判断题

if (type == 1) {

Options[] options = new Options[2];

//判断题的的optionNumber只有1和2，1-答案是“正确”，2-答案是“错误”

options[0] = new Options();

options[0].setQuestionId(question.getId());

options[0].setOptionInfo("正确");

options[0].setOptionNumber(1);

options[1] = new Options();

options[1].setQuestionId(question.getId());

options[1].setOptionInfo("错误");

options[1].setOptionNumber(2);

if (allAnswerNumber.get(0) == 1) {

//正确答案选择了第1个，则第1个选项“正确”为正确答案，第2个选项“错误”为错误答案

options[0].setIsAnswer(1);

options[1].setIsAnswer(2);

} else if (allAnswerNumber.get(0) == 2) {

//正确答案选择了第2个，则第1个选项“正确”为错误答案，第2个选项“错误”为正确答案

options[0].setIsAnswer(2);

options[1].setIsAnswer(1);

}

options[0].setCreator(userId);

optionsService.createOptions(options[0]);

options[1].setCreator(userId);

optionsService.createOptions(options[1]);

}

//单选题

else if (type == 2) {

Options[] options = new Options[allOptionInfo.size()];

int answerNumber = allAnswerNumber.get(0);

for (int i = 0; i < allOptionInfo.size(); i++) {

// Boolean optionExist = questionService.exist(W.f(

// W.and("questionId", "eq", question.getId()),

// W.and("optionInfo", "eq", allOptionInfo.get(i)),

// W.and("isDel", "eq", "0"))

// );

// if (optionExist) {

// return R.ok("选项已存在");

// }

int optionNumber = allOptionNumber.get(i);

options[i] = new Options();

options[i].setQuestionId(question.getId());

options[i].setOptionInfo(allOptionInfo.get(i));

options[i].setOptionNumber(i + 1);

if (optionNumber == answerNumber) {

options[i].setIsAnswer(1);

} else {

options[i].setIsAnswer(2);

}

options[i].setCreator(userId);

optionsService.createOptions(options[i]);

}

}

//3-多选题

else if (type == 3) {

Options[] options = new Options[allOptionInfo.size()];

// int optionNumberIndex = 0;

int answerNumberIndex = 0;

// int optionNumber = allOptionNumber.get(optionNumberIndex);

int answerNumber = allAnswerNumber.get(answerNumberIndex);

for (int i = 0; i < allOptionInfo.size(); i++) {

// Boolean optionExist = questionService.exist(W.f(

// W.and("questionId", "eq", question.getId()),

// W.and("optionInfo", "eq", allOptionInfo.get(i)),

// W.and("isDel", "eq", "0"))

// );

// if (optionExist) {

// return R.ok("选项已存在");

// }

int optionNumber = allOptionNumber.get(i);

options[i] = new Options();

options[i].setQuestionId(question.getId());

options[i].setOptionInfo(allOptionInfo.get(i));

options[i].setOptionNumber(i + 1);

if (answerNumber == optionNumber) {

options[i].setIsAnswer(1);

answerNumberIndex++;

if (answerNumberIndex < allAnswerNumber.size()) {

answerNumber = allAnswerNumber.get(answerNumberIndex);

}

} else {

options[i].setIsAnswer(2);

}

options[i].setCreator(userId);

optionsService.createOptions(options[i]);

}

}

//填空题

else if (type == 4) {

//填空题传来的optionInfo都是答案

for (int i = 0; i < allOptionInfo.size(); i++) {

// Boolean optionExist = questionService.exist(W.f(

// W.and("questionId", "eq", question.getId()),

// W.and("optionInfo", "eq", allOptionInfo.get(i)),

// W.and("isDel", "eq", "0"))

// );

// if (optionExist) {

// return R.ok("选项已存在");

// }

Options option = new Options();

option.setQuestionId(question.getId());

option.setOptionInfo(allOptionInfo.get(i));

option.setIsAnswer(1);

option.setOptionNumber(i + 1);

option.setCreator(userId);

optionsService.createOptions(option);

}

}

return R.ok("题库创建成功", fm2(question));

}

@GetMapping("/question/details/{index}-{size}-{type}-{name}")

@ApiOperation(value = "读取题目和选项分页列表")

public Object getQuestionsByType(@PathVariable("index") int index,

@PathVariable("size") int size,

@PathVariable("type") String type,

@PathVariable("name") String name,

@RequestHeader("token") String token) {

String wKey = "";

if (!V.isEmpty(name))

wKey = S.apppend(" and (questionName like '%", name, "%')");

if (!V.isEmpty(type))

wKey += S.apppend(" and (type = '", type, "')");

return R.ok(questionService.getQuestionDetails(index, size, wKey));

}

@PutMapping("/question/update")

@ApiOperation(value = "修改题目和选项")

@Transactional

public Object updateQuestionAndOptions(@RequestBody UpdateQuestionAndOptions object,

@RequestHeader("token") String token) {

String userId = getUserIdByCache(token);

List<Question> questions = questionService.findByIds(object.getQuestionId());

if (questions == null || questions.size() == 0)

return R.error("题目id异常");

Question question = questions.get(0);

if (!object.getQuestionName().equals(object.getQuestionName()) || !object.getType().equals(object.getType())) {

Boolean exist = questionService.exist(W.f(

W.and("questionName", "eq", object.getQuestionName()),

W.and("type", "eq", object.getType()),

W.and("isDel", "eq", "0"))

);

if (exist) {

return R.error("题目名称已存在");

}

}

question.setQuestionName(object.getQuestionName());

question.setType(object.getType());

question.setAnswerKeys(object.getAnswerKeys());

question.setReviser(userId);

if (question == questionService.updateQuestion(question))

return R.error("修改失败");

//删除所有旧选项数据

List<Options> optionsList = optionsService.queryByQuestionId(" and questionId = '" + object.getQuestionId() + "' ");

for (Options option : optionsList) {

optionsService.deleteByQuestionId(option.getId());

}

//新增新选项数据

List<String> allOptionInfo = object.getOptionInfo();

List<Integer> allOptionNumber = object.getOptionNumber();

List<Integer> allAnswerNumber = object.getAnswerNumber();

//新增题目

int type = object.getType();

//判断题

if (type == 1) {

Options[] options = new Options[2];

//判断题的的optionNumber只有1和2，1-答案是“正确”，2-答案是“错误”

options[0] = new Options();

options[0].setQuestionId(question.getId());

options[0].setOptionInfo("正确");

options[0].setOptionNumber(1);

options[1] = new Options();

options[1].setQuestionId(question.getId());

options[1].setOptionInfo("错误");

options[1].setOptionNumber(2);

if (allAnswerNumber.get(0) == 1) {

//正确答案选择了第1个，则第1个选项“正确”为正确答案，第2个选项“错误”为错误答案

options[0].setIsAnswer(1);

options[1].setIsAnswer(2);

} else if (allAnswerNumber.get(0) == 2) {

//正确答案选择了第2个，则第1个选项“正确”为错误答案，第2个选项“错误”为正确答案

options[0].setIsAnswer(2);

options[1].setIsAnswer(1);

}

options[0].setCreator(userId);

optionsService.createOptions(options[0]);

options[1].setCreator(userId);

optionsService.createOptions(options[1]);

}

//单选题

else if (type == 2) {

Options[] options = new Options[allOptionInfo.size()];

int answerNumber = allAnswerNumber.get(0);

for (int i = 0; i < allOptionInfo.size(); i++) {

// Boolean optionExist = optionsService.exist(W.f(

// W.and("questionId", "eq", question.getId()),

// W.and("optionInfo", "eq", allOptionInfo.get(i)),

// W.and("isDel", "eq", "0"))

// );

// if (optionExist) {

// return R.ok("选项已存在");

// }

int optionNumber = allOptionNumber.get(i);

options[i] = new Options();

options[i].setQuestionId(question.getId());

options[i].setOptionInfo(allOptionInfo.get(i));

options[i].setOptionNumber(i + 1);

if (optionNumber == answerNumber) {

options[i].setIsAnswer(1);

} else {

options[i].setIsAnswer(2);

}

options[i].setCreator(userId);

optionsService.createOptions(options[i]);

}

}

//3-多选题

else if (type == 3) {

Options[] options = new Options[allOptionInfo.size()];

// int optionNumberIndex = 0;

int answerNumberIndex = 0;

// int optionNumber = allOptionNumber.get(optionNumberIndex);

int answerNumber = allAnswerNumber.get(answerNumberIndex);

for (int i = 0; i < allOptionInfo.size(); i++) {

// Boolean optionExist = optionsService.exist(W.f(

// W.and("questionId", "eq", question.getId()),

// W.and("optionInfo", "eq", allOptionInfo.get(i)),

// W.and("isDel", "eq", "0"))

// );

// if (optionExist) {

// return R.ok("选项已存在");

// }

int optionNumber = allOptionNumber.get(i);

options[i] = new Options();

options[i].setQuestionId(question.getId());

options[i].setOptionInfo(allOptionInfo.get(i));

options[i].setOptionNumber(i + 1);

if (answerNumber == optionNumber) {

options[i].setIsAnswer(1);

answerNumberIndex++;

if (answerNumberIndex < allAnswerNumber.size()) {

answerNumber = allAnswerNumber.get(answerNumberIndex);

}

} else {

options[i].setIsAnswer(2);

}

options[i].setCreator(userId);

optionsService.createOptions(options[i]);

}

}

//填空题

else if (type == 4) {

//填空题传来的optionInfo都是答案

for (int i = 0; i < allOptionInfo.size(); i++) {

// Boolean optionExist = questionService.exist(W.f(

// W.and("questionId", "eq", question.getId()),

// W.and("optionInfo", "eq", allOptionInfo.get(i)),

// W.and("isDel", "eq", "0"))

// );

// if (optionExist) {

// return R.ok("选项已存在");

// }

Options option = new Options();

option.setQuestionId(question.getId());

option.setOptionInfo(allOptionInfo.get(i));

option.setIsAnswer(1);

option.setOptionNumber(i + 1);

option.setCreator(userId);

optionsService.createOptions(option);

}

}

return R.ok("题库修改成功");

}

@DeleteMapping("/question/delete/{id}")

@ApiOperation(value = "题目和选项删除")

public Object deleteQuestionAndOptions(@PathVariable("id") String id,

@RequestHeader("token") String token) {

if (!questionService.existId(id))

return R.error("Id数据异常");

if (!questionService.deleteQuestion(id, cacheKit.getUserId(token)))

return R.error("删除失败");

List<Options> options = optionsService.query(W.f(

W.and("questionId", "eq", id),

W.and("isDel", "eq", "0"))

);

if (options != null && options.size() > 0) {

for (Options option : options) {

if (!optionsService.deleteOptions(option.getId(), cacheKit.getUserId(token)))//isDel = 1

// if (!optionsService.deleteByQuestionId(option.getId()))//物理删除,delete

return R.error("删除选项失败");

}

}

return R.ok("删除成功");

}

@GetMapping("/question/list/{type}-{name}")

@ApiOperation(value = "通过题目和类型读取题库列表")

public Object getQuestionByType(@PathVariable("type") String type,

@PathVariable("name") String name,

@RequestHeader("token") String token) {

String wKey = "";

if (!V.isEmpty(type))

wKey = S.apppend(" and type = ", type, " ");

if (!V.isEmpty(name))

wKey += S.apppend(" and questionName like '%", name, "%' ");

return R.ok(questionService.queryAll(wKey));

}

@GetMapping("/question/get")

@ApiOperation(value = "读取所有题目和选项")

public Object getQuestionsByType(@RequestHeader("token") String token) {

String[] types = {"trueOrFalse", "singleChoice", "multipleChoice", "gapFilling", "subjective"};

Map<String, ArrayList<JSONObject>> back = new TreeMap<>();

for (int i = 0; i < 5; i++) {

List<JSONObject> questions = questionService.queryAll(" and type = '" + (i + 1) + "' ");

ArrayList<JSONObject> teamPapers = new ArrayList<>();

for (int j = 0; j < questions.size(); j++) {

JSONObject examQuestion = questions.get(j);

String questionId = examQuestion.getString("id");

//获取该题目的选项

if (i + 1 != 5) {//5-客观题没有选项

String where = " and questionId = '" + questionId + "' ";

List<Options> options = optionsService.queryByQuestionId(where);

List<JSONObject> optionsJSON = new ArrayList<>();

for (Options o : options) {

optionsJSON.add(J.o2j(o));

}

examQuestion.put("options", F.f2l(optionsJSON, "id", "questionId"));

}

examQuestion.put("questionId", questionId);

F.f2l(questions, "id");

teamPapers.add(examQuestion);

}

back.put(types[i], teamPapers);

}

return R.ok(back);

}

@PostMapping("/question/import/{batch}")

@ApiOperation(value = "导入题库")

public Object Import(

HttpServletRequest req,

@PathVariable("batch") String batch,

@RequestHeader("token") String token) {

if (V.isEmpty(batch)) {

return R.error("导入批次有误！");

}

if (V.isEmpty(cacheKit.getVal("question-import-lock-" + batch))) {

cacheKit.setVal("question-import-lock-" + batch, "true", 1800);

} else {

return R.error("导入操作正在进行中，请耐心等待操作结果");

}

String filepath = uploadFile(req);

if (V.isEmpty(filepath)) {

return R.error("文件上传失败！");

}

JSONObject result = new JSONObject();

JSONObject obj = new JSONObject();

obj.put("token", token);

obj.put("filepath", filepath);

JSONObject res = SolvedFile(obj, batch);

if (res == null) {

return R.error("题库导入异常！");

}

Boolean flag = res.getBoolean("flag");

if (!flag) {

return R.error(res.getString("message"));

}

result.put("total", res.getInteger("total"));

result.put("truecount", res.getInteger("truecount"));

result.put("errcount", res.getInteger("errcount"));

if (res.getInteger("truecount") == 0 && res.getInteger("errcount") == 0) {

return R.error("导入数据为空！");

}

if (res.getInteger("errcount") > 0) {

result.put("errlist", res.getJSONArray("errlist"));

result.put("success", 0);

} else {

result.put("success", 1);

}

cacheKit.deleteVal("question-import-lock-" + batch);

return R.ok("导入成功", result);

}

@PostMapping("/question/importStatus/{batch}")

@ApiOperation(value = "导入题库状态")

public Object getImportStatus(@PathVariable("batch") String batch) {

if (V.isEmpty(batch)) {

return R.error("导入批次有误！");

}

JSONObject obj = getImportStatusObj(batch);

if (V.isEmpty(batch)) {

return R.error("导入状态异常！");

}

return R.ok(obj);

}

public JSONObject SolvedFile(JSONObject obj, String batch) {

JSONObject res = new JSONObject();

JSONArray errors = new JSONArray();

if (obj.getString("filepath") == null || obj.getString("filepath") == "") {

res.put("flag", false);

res.put("message", "上传文件有误！");

return res;

}

if (obj.getString("token") == null || obj.getString("token") == "") {

res.put("flag", false);

res.put("message", "用户授权失败！");

return res;

}

String token = obj.getString("token");

try {

File excel = new File(obj.getString("filepath"));

if (excel.isFile() && excel.exists()) { //判断文件是否存在

String[] split = excel.getName().split("\\."); //.是特殊字符，需要转义！！！！！

Workbook wb;

//根据文件后缀（xls/xlsx）进行判断

if ("xls".equals(split[1])) {

FileInputStream fis = new FileInputStream(excel); //文件流对象

wb = new HSSFWorkbook(fis);

} else if ("xlsx".equals(split[1])) {

wb = new XSSFWorkbook(excel);

} else {

res.put("flag", false);

res.put("message", "上传文件有误！");

return res;

}

//开始解析

Sheet sheet = wb.getSheetAt(0); //读取sheet

int firstRowIndex = sheet.getFirstRowNum() + 6; //第一行是填写说明，第二行为标题,第三到六行为示例所以不读

int lastRowIndex = sheet.getLastRowNum();

int dealCount = 0, trueCount = 0;

for (int rIndex = firstRowIndex; rIndex <= lastRowIndex; rIndex++) { //遍历行

// System.out.println("rIndex: " + rIndex);

Row row = sheet.getRow(rIndex);

if (row != null) {

//String examId,String academyCode,String myclass,String name,String number,String phone,String token

String name = "", type = "", trueAnswer = "", detail = "";

int optionNum = -1;

if (row.getCell(0) != null && row.getCell(2) != null && row.getCell(3) != null) {

JSONObject questionobj = new JSONObject();

name = row.getCell(0).toString().trim();//题目名称

type = row.getCell(2).toString().trim();//题目类型

trueAnswer = row.getCell(3).toString().trim();//正确答案

questionobj.put("name", name);

if (!V.isEmpty(row.getCell(1))) {

detail = row.getCell(1).toString().trim();//题目解析（非必填）

questionobj.put("detail", detail);

}

if (!V.isEmpty(row.getCell(4))) {

optionNum = (int) row.getCell(4).getNumericCellValue();//题目选项（非必填）

}

questionobj.put("type", type);

questionobj.put("trueAnswer", trueAnswer);

boolean questionFlag = false;//创建题库是否成功标志

String userId = getUserIdByCache(token);

dealCount++;

JSONObject resObject = dealQuestionData(type, name, detail, userId, trueAnswer, optionNum, row);

if (!V.isEmpty(resObject)) {

questionFlag = resObject.getBoolean("questionFlag");

if (!questionFlag) {

questionFlag = false;

questionobj.put("info", resObject.getString("info"));

}

} else {

questionFlag = false;

questionobj.put("info", "题目导入失败");

}

if (!questionFlag) {

if (V.isEmpty(questionobj.getString("info"))) {

questionobj.put("info", "题库创建失败");

}

errors.add(questionobj);

cacheKit.setVal(batch + "status", 2 + "", 0);

} else {

trueCount++;

cacheKit.setVal(batch + "status", 1 + "", 0);

}

cacheKit.setVal(batch + "now", rIndex - 5 + "", 0);

}

cacheKit.setVal(batch + "total", lastRowIndex - 5 + "", 0);

if (rIndex != lastRowIndex) {

cacheKit.setVal(batch + "res", 2 + "", 0);

}

}

}

cacheKit.setVal(batch + "res", 1 + "", 0);

res.put("truecount", trueCount);

res.put("errcount", errors.size());

res.put("total", dealCount);

res.put("errlist", errors);

res.put("flag", true);

return res;

} else {

res.put("flag", false);

res.put("message", "上传文件有误！");

System.out.println("找不到指定的文件");

return res;

}

} catch (Exception e) {

e.printStackTrace();

res.put("flag", false);

res.put("message", "题库处理出错！");

return res;

}

}

public JSONObject dealQuestionData(String type, String name, String detail, String userId, String trueAnswer, int optionNum, Row row) {

//关闭事务自动提交

//2.获取事务定义

DefaultTransactionDefinition def = new DefaultTransactionDefinition();

//3.设置事务隔离级别，开启新事务

def.setPropagationBehavior(TransactionDefinition.PROPAGATION\_REQUIRES\_NEW);

//4.获得事务状态

TransactionStatus transactionStatus = transactionManager.getTransaction(def);

Boolean questionFlag = false;

JSONObject questionobj = new JSONObject();

if (type.equals("填空题") || type.equals("判断题") || type.equals("单选题") || type.equals("多选题")) {

int type\_int = -1;

switch (type) {

case "填空题":

type\_int = 4;

break;

case "判断题":

type\_int = 1;

break;

case "单选题":

type\_int = 2;

break;

case "多选题":

type\_int = 3;

break;

}

Boolean exist = questionService.exist(W.f(

W.and("questionName", "eq", name),

W.and("type", "eq", type\_int),

W.and("isDel", "eq", "0"))

);

if (exist) {

questionFlag = false;

questionobj.put("info", "题目名称已存在");

} else {

questionFlag = true;

}

;

if (questionFlag) {

Question question = new Question();

question.setType(type\_int);

question.setQuestionName(name);

question.setAnswerKeys(detail);

question.setCreator(userId);

question = questionService.createQuestion(question);

if (question == null) {

questionFlag = false;

questionobj.put("info", "题目保存失败");

} else {

if (type.equals("填空题") || type.equals("判断题")) {

//新增题目

if (type\_int == 1) {//判断题

Options[] options = new Options[2];

Options check = new Options();

//判断题的的optionNumber只有1和2，1-答案是“正确”，2-答案是“错误”

options[0] = new Options();

options[0].setQuestionId(question.getId());

options[0].setOptionInfo("正确");

options[0].setOptionNumber(1);

options[1] = new Options();

options[1].setQuestionId(question.getId());

options[1].setOptionInfo("错误");

options[1].setOptionNumber(2);

int trueAnswer\_int = (int) Double.parseDouble(trueAnswer);

if (trueAnswer\_int == 1) {

//正确答案选择了第1个，则第1个选项“正确”为正确答案，第2个选项“错误”为错误答案

options[0].setIsAnswer(1);

options[1].setIsAnswer(2);

} else if (trueAnswer\_int == 2) {

//正确答案选择了第2个，则第1个选项“正确”为错误答案，第2个选项“错误”为正确答案

options[0].setIsAnswer(2);

options[1].setIsAnswer(1);

}

questionFlag = false;

options[0].setCreator(userId);

check = optionsService.createOptions(options[0]);

if (check != null) {

options[1].setCreator(userId);

check = optionsService.createOptions(options[1]);

if (check != null) {

questionFlag = true;

}

}

if (!questionFlag) {

if (V.isEmpty(questionobj.getString("info"))) {

questionobj.put("info", "选项保存失败");

}

}

}

if (type\_int == 4) {//填空题

questionFlag = true;

String[] optionArr = trueAnswer.split("#|＃");

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

Boolean optionExist = optionsService.exist(W.f(

W.and("questionId", "eq", question.getId()),

W.and("optionInfo", "eq", optionArr[i]),

W.and("isDel", "eq", "0"))

);

if (optionExist) {

questionFlag = false;

questionobj.put("info", "选项已存在");

break;

}

Options option = new Options();

option.setQuestionId(question.getId());

option.setOptionInfo(optionArr[i].trim());

if (optionArr[i].length() > 50) {

questionFlag = false;

questionobj.put("info", "填空题答案长度大于50");

break;

}

option.setIsAnswer(1);

option.setOptionNumber(i + 1);

option.setCreator(userId);

option = optionsService.createOptions(option);

if (option == null) {

questionFlag = false;

questionobj.put("info", "选项保存失败");

break;

}

}

if (!questionFlag) {

if (V.isEmpty(questionobj.getString("info"))) {

questionobj.put("info", "选项保存失败");

}

}

}

}

if (type.equals("单选题") || type.equals("多选题")) {

if (!V.isEmpty(optionNum) && optionNum >= 2) {

questionFlag = true;

ArrayList<String> optionArray = new ArrayList<String>();

for (int i = 5; i < 5 + optionNum; i++) {

if (!V.isEmpty(row.getCell(i))) {

optionArray.add(row.getCell(i).toString().trim());

} else {

questionFlag = false;

questionobj.put("info", "选项保存失败");

break;

}

}

if (questionFlag) {

questionFlag = true;

String[] answerArr = trueAnswer.split("#|＃");

for (int i = 0; i < optionArray.size(); i++) {

Boolean optionExist = optionsService.exist(W.f(

W.and("questionId", "eq", question.getId()),

W.and("optionInfo", "eq", optionArray.get(i)),

W.and("isDel", "eq", "0"))

);

if (optionExist) {

questionFlag = false;

questionobj.put("info", "选项已存在");

break;

}

Options option = new Options();

option.setQuestionId(question.getId());

option.setOptionInfo(optionArray.get(i));

option.setIsAnswer(2);

for (int j = 0; j < answerArr.length; j++) {

int answerId = (int) Double.parseDouble(answerArr[j].trim()) - 1;

if (answerId == i) {

option.setIsAnswer(1);

}

}

option.setOptionNumber(i + 1);

option.setCreator(userId);

option = optionsService.createOptions(option);

if (option == null) {

questionFlag = false;

break;

}

}

if (!questionFlag) {

if (V.isEmpty(questionobj.getString("info"))) {

questionobj.put("info", "选项保存失败");

}

}

}

} else {

questionFlag = false;

questionobj.put("info", "选项数量有误");

}

}

}

}

} else {

questionFlag = false;

questionobj.put("info", "题目类型错误");

}

questionobj.put("questionFlag", questionFlag);

if (questionFlag) {

transactionManager.commit(transactionStatus);

} else {

transactionManager.rollback(transactionStatus);

}

return questionobj;

}

private Map<String, String> verify() {

Map<String, String> verify = new HashMap<>();

verify.put("questionName", "请输入题目名称");

verify.put("type", "请输入类型");

verify.put("answerKeys", "请输入答案解析");

return verify;

}

private Map<String, String> updateVerify() {

Map<String, String> verify = new HashMap<>();

verify.put("questionName", "请输入题目名称");

verify.put("type", "请输入类型");

verify.put("answerKeys", "请输入答案解析");

return verify;

}

}

### 4.6、课程管理模块

#### 4.61、Course.ejs

<% include ../inc/header.ejs%>

<style>

.date-table {

overflow-y: auto;

}

</style>

<% include ../inc/body.ejs%>

<!-- html主体代码 start -->

<div class="row">

<div class="col-sm-12">

<div class="ibox float-e-margins">

<div class="ibox-title" style="padding-top: 3px !important;">

<div style="width: 140px; display: block;float: left;padding-top: 10px;">

课程信息列表

</div>

<div class="ibox-tools">

<button style="margin-top: 8px;" class="btn btn-primary btn-xs" type="button"

onclick="course.create(this);">

<i class="fa fa-plus"></i>&nbsp;新增

</button>

</div>

</div>

<div class="ibox-content data-view" style="overflow-y: auto;">

<div class="input-group" style="width: 200px;display: flex">

<input style="min-width: 200px;width: 200px;" type="text" id="select-name" placeholder="请输入课程名称"

class="input-sm form-control"/>

<span class="input-group-btn">

<button onclick="course.select(this);" id="search\_button"

style="width: 50px;height: 34px" type="button"

class="btn btn-sm btn-primary"> 搜索</button>

</span>

</div>

<div class="col-sm-12">

<div class=" col-sm-12 date-table">

<table class="table table-striped">

<thead>

<tr>

<th>序号</th>

<th>课程号</th>

<th>课程名</th>

<th>学分</th>

<th></th>

</tr>

</thead>

<tbody id="item-list">

</tbody>

</table>

</div>

<div class="col-sm-12 pages M-box3 list-page">

</div>

</div>

</div>

</div>

</div>

</div>

<!-- html主体代码 end -->

<!-- 管理的课程信息模板 -->

<script id="course-template" type="text/x-dot-template">

{{ if(it.length == 0) { }}

暂无数据

{{ }else{ }}

{{ for(var i=0;i < it.length;i++){ }}

<tr>

<td>

{{=(i+1)}}

</td>

<td>

{{= it[i].number}}

</td>

<td>

{{= it[i].name}}

</td>

<td>

{{= it[i].credit}}

</td>

<td>

<div class="btn-group">

<button data-toggle="dropdown" class="btn btn-primary btn-xs dropdown-toggle">操作 <span

class="caret"></span>

</button>

<ul class="dropdown-menu">

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="course.update(this);">修改</a>

</li>

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="course.delete(this);">删除</a>

</li>

</ul>

</div>

</td>

</tr>

{{ } }}

{{ } }}

</script>

<% include ../inc/js.ejs %>

<!-- 私有脚本 start -->

<script src="/js/baseModule/\_course.js"></script>

<!-- 私有脚本 end -->

<% include ../inc/footer.ejs %>

#### 4.62、CourseApi.java

package team.work.api.admin;

import com.alibaba.fastjson.JSONObject;

import team.work.api.BaseApi;

import team.work.core.model.Course;

import team.work.core.service.impl.CourseService;

import team.work.doc.CourseCreate;

import team.work.doc.CourseUpdate;

import team.work.utils.bean.Page;

import team.work.utils.convert.R;

import team.work.utils.convert.S;

import team.work.utils.convert.V;

import team.work.utils.convert.W;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

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

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

import java.util.HashMap;

import java.util.List;

import java.util.Map;

/\*\*

\* Created by party on 2019/06/26

\*/

@Api(value = "304\_课程信息管理")

@RestController

@RequestMapping("/v1/base")

public class CourseApi extends BaseApi {

@Autowired

private CourseService courseService;

@PostMapping("/course")

@ApiOperation(value = "课程信息新增")

public Object createCourse(@RequestBody CourseCreate object,

@RequestHeader("token") String token) {

String userId = getUserIdByCache(token);

//映射对象

Course model = o2c(object, token, Course.class);

//数据校验

JSONObject check = V.checkEmpty(verify(), object);

if (check.getBoolean("check"))

return R.error(check.getString("message"));

Boolean exist = courseService.exist(W.f(

W.and("number", "eq", model.getNumber()),

W.and("isDel", "eq", "0"))

);

if (exist)

return R.error("课程号已经存在");

model = courseService.createCourse(model);

if (model == null)

return R.error("新增失败");

return R.ok("新增成功", fm2(model));

}

@PutMapping("/course")

@ApiOperation(value = "修改课程信息")

public Object updateCourse(@RequestBody CourseUpdate object,

@RequestHeader("token") String token) {

String userId = getUserIdByCache(token);

//映射对象

Course model = o2c(object, token, Course.class);

//数据校验

JSONObject check = V.checkEmpty(updateVerify(), object);

if (check.getBoolean("check"))

return R.error(check.getString("message"));

Course data = courseService.selectById(model.getId());

if (data == null) {

return R.error("该信息不存在，无法修改");

}

if (!model.getNumber().equals(data.getNumber())) {

Boolean exist = courseService.exist(W.f(

W.and("number", "eq", model.getNumber()),

W.and("id", "ne", model.getId()),

W.and("isDel", "eq", "0"))

);

if (exist)

return R.error("课程号已经存在");

}

model.setReviser(userId);

model = courseService.updateCourse(model);

if (model == null)

return R.error("修改失败");

return R.ok("修改成功", fm2(model));

}

@DeleteMapping("/course/{id}")

@ApiOperation(value = "课程信息删除")

public Object deleteCourse(@PathVariable("id") String id,

@RequestHeader("token") String token) {

if (!courseService.existId(id))

return R.error("Id数据异常");

if (courseService.deleteCourse(id, cacheKit.getUserId(token)))

return R.ok("删除成功");

return R.error("删除失败");

}

@GetMapping("/course/{index}-{size}-{name}")

@ApiOperation(value = "读取课程信息分页列表")

public Object getCourse(@PathVariable("index") int index,

@PathVariable("size") int size,

@PathVariable("name") String name,

@RequestHeader("token") String token) {

String wKey = "";

if (!V.isEmpty(name))

wKey = S.apppend(" and name like '%", name, "%' ");

return R.ok(courseService.page(index, size, wKey));

}

@GetMapping("/course")

@ApiOperation(value = "读取课程信息所有列表")

public Object getAllCourse(@RequestHeader("token") String token) {

return R.ok(courseService.queryAll(""));

}

private Map<String, String> verify() {

Map<String, String> verify = new HashMap<>();

verify.put("credit", "请输入学分");

verify.put("name", "请输入课程名");

verify.put("number", "请输入课程号");

return verify;

}

private Map<String, String> updateVerify() {

Map<String, String> verify = new HashMap<>();

verify.put("credit", "请输入学分");

verify.put("name", "请输入课程名");

verify.put("number", "请输入课程号");

return verify;

}

}

### 4.7、教师模块管理

#### 4.71、Teacher.ejs

<% include ../inc/header.ejs%>

<style>

.date-table {

overflow-y: auto;

}

</style>

<% include ../inc/body.ejs%>

<!-- html主体代码 start -->

<div class="row">

<div class="col-sm-12">

<div class="ibox float-e-margins">

<div class="ibox-title" style="padding-top: 3px !important;">

<div style="width: 140px; display: block;float: left;padding-top: 10px;">

教师信息列表

</div>

<div class="ibox-tools">

<button style="margin-top: 8px;" class="btn btn-primary btn-xs" type="button"

onclick="teacher.create(this);">

<i class="fa fa-plus"></i>&nbsp;新增

</button>

</div>

</div>

<div class="ibox-content data-view" style="overflow-y: auto;">

<div class="col-sm-12">

<div class=" col-sm-12 date-table">

<table class="table table-striped">

<thead>

<tr>

<th>序号</th>

<th>所在学院</th>

<th>性别</th>

<th>姓名</th>

<th>工号/账号</th>

<th>账号状态</th>

<th>上次登录ip</th>

<th>登录时间</th>

<th></th>

</tr>

</thead>

<tbody id="item-list">

</tbody>

</table>

</div>

<div class="col-sm-12 pages M-box3 list-page">

</div>

</div>

</div>

</div>

</div>

</div>

<!-- html主体代码 end -->

<!-- 管理的教师信息模板 -->

<script id="teacher-template" type="text/x-dot-template">

{{ if(it.length == 0) { }}

暂无数据

{{ }else{ }}

{{ for(var i=0;i < it.length;i++){ }}

<tr>

<td>

{{=(i+1)}}

</td>

<td>

{{= it[i].academyName}}

</td>

<td>

{{= helper.sex(it[i].sex) }}

</td>

<td>

{{= it[i].name}}

</td>

<td>

{{= it[i].number}}

</td>

<td>

{{= helper.status(it[i].status) }}

</td>

<td>

{{= it[i].loginIp || ""}}

</td>

<td>

{{= time.timestamp2time(it[i].loginTime) ||"" }}

</td>

<td>

<div class="btn-group">

<button data-toggle="dropdown" class="btn btn-primary btn-xs dropdown-toggle">操作 <span

class="caret"></span>

</button>

<ul class="dropdown-menu">

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="teacher.update(this);">修改</a>

</li>

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="teacher.delete(this);">删除</a>

</li>

</ul>

</div>

</td>

</tr>

{{ } }}

{{ } }}

</script>

<script id="academy-select-template" type="text/x-dot-template">

{{ if(it.length == 0 ){ }}

<option value=" ">暂无学院数据</option>

{{ }else{ }}

<option value=" ">请选择学院</option>

{{ for(var i = 0 ; i < it.length ; i++){ }}

<option value="{{=it[i].id }}">{{=it[i].name }}</option>

{{ } }}

{{ } }}

</script>

<% include ../inc/js.ejs %>

<!-- 私有脚本 start -->

<script src="/js/userModule/\_teacher.js"></script>

<!-- 私有脚本 end -->

<% include ../inc/footer.ejs %>

#### 4.72、TeacherApi.java

package team.work.api.admin;

import com.alibaba.fastjson.JSONObject;

import team.work.api.BaseApi;

import team.work.core.model.Teacher;

import team.work.core.service.impl.SysUserService;

import team.work.core.service.impl.TeacherService;

import team.work.doc.TeacherCreate;

import team.work.doc.TeacherUpdate;

import team.work.utils.bean.Page;

import team.work.utils.convert.\*;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

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

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

import team.work.utils.kit.MD5Kit;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

/\*\*

\* Created by party on 2019/06/26

\*/

@Api(value = "303\_教师信息管理")

@RestController

@RequestMapping("/v1/base")

public class TeacherApi extends BaseApi {

@Autowired

private TeacherService teacherService;

@Autowired

private SysUserService sysUserService;

@PostMapping("/teacher")

@ApiOperation(value = "教师信息新增")

public Object createTeacher(@RequestBody TeacherCreate object,

@RequestHeader("token") String token) {

String userId = getUserIdByCache(token);

//映射对象

Teacher teacher = o2c(object, token, Teacher.class);

//数据校验

JSONObject check = V.checkEmpty(verify(), object);

if (check.getBoolean("check"))

return R.error(check.getString("message"));

Boolean exist = teacherService.exist(W.f(

W.and("number", "eq", teacher.getNumber()),

W.and("isDel", "eq", "0"))

);

if (exist)

return R.error("工号已存在");

String number = teacher.getNumber();

if (sysUserService.existLoginName(number))

return R.error(F.s("当前工号[%s]已经被注册", number));

//工号后四位

String numLast4 = number.substring(number.length() - 4);

String salt = S.randomNum();

String pwd = MD5Kit.encode("gxun" + numLast4 + salt);

JSONObject sysUser = new JSONObject();

sysUser.put("loginName", number);

sysUser.put("password", pwd);

sysUser.put("salt", salt);

sysUser.put("type", 1);//1-教师

sysUser.put("creator", userId);

sysUser.put("roleId", object.getRoleId());

sysUser = (JSONObject) sysUserService.createUser(sysUser);

if (sysUser == null) {

return R.error("教师用户创建失败");

}

teacher = teacherService.createTeacher(teacher);

if (teacher == null)

return R.error("新增失败");

return R.ok("新增成功\n登录账号为" + number + "\n密码为gxun" + numLast4, fm2(teacher));

}

@PutMapping("/teacher")

@ApiOperation(value = "修改教师信息")

public Object updateTeacher(@RequestBody TeacherUpdate object,

@RequestHeader("token") String token) {

String userId = getUserIdByCache(token);

//映射对象

Teacher model = o2c(object, token, Teacher.class);

//数据校验

JSONObject check = V.checkEmpty(updateVerify(), object);

if (check.getBoolean("check"))

return R.error(check.getString("message"));

Teacher data = teacherService.selectById(model.getId());

if (data == null) {

return R.error("该信息不存在，无法修改");

}

if (!model.getNumber().equals(data.getNumber())) {

Boolean exist = teacherService.exist(W.f(

W.and("number", "eq", model.getNumber()),

W.and("id", "eq", model.getId()),

W.and("isDel", "eq", "0"))

);

if (exist)

return R.error("代码已经存在请更换一个代码");

}

model.setReviser(userId);

model = teacherService.updateTeacher(model);

if (model == null)

return R.error("修改失败");

return R.ok("修改成功", fm2(model));

}

@DeleteMapping("/teacher/{id}")

@ApiOperation(value = "教师信息删除")

public Object deleteTeacher(@PathVariable("id") String id,

@RequestHeader("token") String token) {

if (!teacherService.existId(id))

return R.error("Id数据异常");

if (teacherService.deleteTeacher(id, cacheKit.getUserId(token)))

return R.ok("删除成功");

return R.error("删除失败");

}

@GetMapping("/teacher/{index}-{size}-{key}")

@ApiOperation(value = "读取教师信息分页列表")

public Object getTeacher(@PathVariable("index") int index,

@PathVariable("size") int size,

@PathVariable("key") String key,

@RequestHeader("token") String token) {

String wKey = "";

if (!V.isEmpty(key))

//FIXME 修改id为需要查询的字段

wKey = S.apppend(" and id like '%", key, "%' ");

return R.ok(teacherService.page(index, size, wKey));

}

@GetMapping("/teacher")

@ApiOperation(value = "读取教师信息所有列表")

public Object getAllTeacher(@RequestHeader("token") String token) {

return R.ok(teacherService.queryAll(""));

}

private Map<String, String> verify() {

Map<String, String> verify = new HashMap<>();

verify.put("name", "请输入姓名");

verify.put("number", "请输入工号");

verify.put("academyId", "请输入所属学院id");

verify.put("sex", "请输入性别");

return verify;

}

private Map<String, String> updateVerify() {

Map<String, String> verify = new HashMap<>();

verify.put("name", "请输入姓名");

verify.put("number", "请输入工号");

verify.put("academyId", "请输入所属学院id");

verify.put("sex", "请输入性别");

return verify;

}

}

### 4.8用户模块管理

#### 4.81、User.ejs

<% include inc/header.ejs %>

<% include inc/body.ejs %>

<!-- html主体代码 start -->

<div class="row">

<div class="ibox float-e-margins">

<div class="ibox-title">

<h5>用户列表</h5>

<div class="ibox-tools">

<button class="btn btn-primary btn-xs" type="button" onclick="user.create(this);">

<i class="fa fa-plus"></i>&nbsp;创建

</button>

</div>

</div>

<div class="ibox-content">

<div class="input-group" style="width: 200px;">

<input style="width: 200px;" type="text" id="select-key" placeholder="请输入关键词"

class="input-sm form-control"> <span class="input-group-btn">

<button onclick="user.select(this);" id="search\_button" style="width: 50px;" type="button"

class="btn btn-sm btn-primary"> 搜索</button> </span>

</div>

<table class="table table-striped">

<thead>

<tr>

<th></th>

<th>登录名</th>

<th>状态</th>

<th>token</th>

<th>注册时间</th>

<th>上次登录时间</th>

<th></th>

</tr>

</thead>

<tbody id="user-list">

</tbody>

</table>

<div class="pages M-box3 list-page">

</div>

</div>

</div>

</div>

<!-- html主体代码 end -->

<!-- 用户数据模板 -->

<script id="user-tmpl" type="text/x-dot-template">

{{ if(it.length>0){ }}

{{ for(var i=0;i

<it.length;i++){ }}

<tr>

<td>

<input type="checkbox" class="i-checks" name="user-ck">

</td>

<td><a href="#" data-id="{{= it[i].id}}" onclick="user.info(this);">{{= string.cut(it[i].loginName,6)}}</a></td>

<td>{{= format\_status(it[i].status)}}</td>

<td><a href="#" data-id="{{= it[i].token}}" onclick="user.token(this);">{{= it[i].token}}</a></td>

<td>

{{= time.timestamp2time(it[i].regTime) }}

</td>

<td>

{{= time.timestamp2time(it[i].loginTime) }}

</td>

<td>

<div class="btn-group">

<button data-toggle="dropdown" class="btn btn-primary btn-xs dropdown-toggle">操作 <span class="caret"></span>

</button>

<ul class="dropdown-menu">

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="user.initPwd(this);">重置密码</a>

</li>

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="user.update(this);">修改个人信息</a>

</li>

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="user.changeRole(this);">修改角色</a>

</li>

<li>

<a href="#" data-id="{{= it[i].id}}" data-status="{{= it[i].status}}"

onclick="user.status(this);">

{{ if(it[i].status == 1){ }} 冻结 {{ }else{ }} 恢复 {{ } }}

</a>

</li>

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="user.setAuth(this);">子系统配置</a>

</li>

<li>

<a href="#" data-id="{{= it[i].id}}" onclick="user.delete(this);">删除用户</a>

</li>

</ul>

</div>

</td>

</tr>

{{ } }}

{{ } }}

</script>

<script id="system-auth-tmpl" type="text/x-dot-template">

{{ for(var i=0;i

<it.length;i++){ }}

<tr>

<td>

<div class="i-checks">

<label>

<input {{ if(it[i].check> 1) { }} checked {{ } }}

name="auth" type="checkbox" data-id="{{= it[i].systemId}}">

</label>

</div>

</td>

<td>

{{= it[i].systemName}}

</td>

<td>

{{= it[i].roleName}}

</td>

<td>

<select class="form-control" id="role{{= it[i].systemId}}">

{{ for(var j=0;j

<it

[i].role.length;j++){ }}

<option {{ if(it[i].role[j].isDefault== 1) { }} check {{ } }} value="{{= it[i].role[j].roleId}}">

{{= it[i].role[j].roleName}}

</option>

{{ } }}

</select>

</td>

</tr>

{{ } }}

</script>

<script id="roleId-template" type="text/x-dot-template">

{{ if(it.length == 0) { }}

<option value="\*">请选择角色</option>

<option value="\*">暂无数据</option>

{{ }else{ }}

<option value="\*">请选择角色</option>

{{ for(var i=0;i

<it.length;i++){ }}

<option value="{{= it[i].id}}">{{= it[i].name}}</option>

{{ } }}

{{ } }}

</script>

<% include inc/js.ejs %>

<!-- 私有脚本 start -->

<script src="../../js/\_user.js"></script>

<!-- 私有脚本 end -->

<% include inc/footer.ejs %>

#### 4.82、SysUserApi.java

package team.work.api.admin;

import com.alibaba.fastjson.JSONArray;

import com.alibaba.fastjson.JSONObject;

import team.work.core.service.impl.\*;

import team.work.doc.\*;

import team.work.utils.convert.\*;

import team.work.utils.tool.RSAEncrypt;

import team.work.utils.bean.Page;

import team.work.api.BaseApi;

import team.work.utils.kit.MD5Kit;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.apache.poi.hssf.usermodel.HSSFWorkbook;

import org.apache.poi.ss.usermodel.Row;

import org.apache.poi.ss.usermodel.Sheet;

import org.apache.poi.ss.usermodel.Workbook;

import org.apache.poi.xssf.usermodel.XSSFWorkbook;

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

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

import team.work.core.model.SysUser;

import team.work.core.model.SysUserRole;

import team.work.core.model.UserBaseInfo;

import javax.servlet.http.HttpServletRequest;

import java.io.File;

import java.io.FileInputStream;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

@Api(value = "SYS5\_用户管理")

@RestController

@RequestMapping("/v1/sys")

public class SysUserApi extends BaseApi {

@Autowired

private SysUserService userService;

@Autowired

private SysRoleService roleService;

@Autowired

private SysUserRoleService userRoleService;

@Autowired

private StudentService studentService;

@Autowired

private TeacherService teacherService;

@Autowired

private UserBaseInfoService userBaseInfoService;

@GetMapping("/user/{index}-{size}-{loginName}-{roleId}-{academyCode}")

@ApiOperation(value = "读取用户列表")

public Object getUserList(@PathVariable("index") int index,

@PathVariable("size") int size,

@PathVariable("loginName") String loginName,

@PathVariable("roleId") String roleId,

@PathVariable("academyCode") String academySelectCode,

@RequestHeader("token") String token) {

String userkey = "";

String rolekey = "";

String infokey = "";

if (!V.isEmpty(loginName))

userkey += S.apppend(" and loginName like '%", loginName.replace("\*",""), "%' ");

if (!V.isEmpty(roleId))

rolekey += S.apppend(" and roleId like '%", roleId.replace("\*",""), "%' ");

if (!V.isEmpty(academySelectCode))

infokey += S.apppend(" and academyCode like '%", academySelectCode.replace("\*",""), "%' ");

//获取数据可见性

Integer visible = getVisibleByCache(token);

if (V.isEmpty(visible)) {

return R.error("您没有权限查看");

}

Page empty = new Page(new ArrayList<JSONObject>(), 0, 0, 0, 1);

if (visible == 1) {//1-全部（组织部）

return R.ok(userService.page(index, size, userkey,rolekey,infokey));

} else if (visible == 2) {//2-院级

String academyCode = getAcademyCodeByCache(token);

if (V.isEmpty(academyCode)) {

return R.error("请先完善您的学院信息");

}

return R.ok(userService.page(index, size, userkey,rolekey,infokey));

} else if (visible == 3) {//3-个人(班主任、联络员)

//获取学号/工号

String loginNumber = getLoginNumberCache(token);

if (V.isEmpty(loginNumber)) {

return R.error("请先完善你的学号/工号");

}

return R.ok(userService.page(index, size, userkey,rolekey,infokey));

} else {

return R.error("您没有权限查看");

}

}

@GetMapping("/userInfo/{index}-{size}-{number}-{academyCode}")

@ApiOperation(value = "读取用户基础信息分页列表")

public Object getUserInfo(@PathVariable("index") int index,

@PathVariable("size") int size,

@PathVariable("number") String number,

@PathVariable("academyCode") String academySelectCode,

@RequestHeader("token") String token) {

String wKey = "";

if (!V.isEmpty(number) )

wKey += S.apppend(" and number = '", number, "' ");

if (!V.isEmpty(academySelectCode) )

wKey += S.apppend(" and academyCode = '",academySelectCode, "' ");

//获取数据可见性

Integer visible = getVisibleByCache(token);

if (V.isEmpty(visible)) {

return R.error("您没有权限查看");

}

Page empty = new Page(new ArrayList<JSONObject>(), 0, 0, 0, 1);

if (visible == 1) {//1-全部（组织部）

return R.ok(userBaseInfoService.page(index, size, wKey));

} else if (visible == 2) {//2-院级

String academyCode = getAcademyCodeByCache(token);

if (V.isEmpty(academyCode)) {

return R.error("请先完善您的学院信息");

}

return R.ok(userBaseInfoService.page(index, size, wKey));

} else if (visible == 3) {//3-个人(班主任、联络员)

//获取学号/工号

String loginNumber = getLoginNumberCache(token);

if (V.isEmpty(loginNumber)) {

return R.error("请先完善你的学号/工号");

}

return R.ok(userBaseInfoService.page(index, size, wKey));

} else {

return R.error("您没有权限查看");

}

}

@GetMapping("/userInfo/personal")

@ApiOperation(value = "读取用户个人信息列表")

public Object getPersonalInfo(@RequestHeader("token") String token) {

String number = getNumberByCache(token);

Integer type = getTypeByCache(token);

String wKey = "";

List<JSONObject> back = new ArrayList<>();

if (type == 1) {//教师

wKey += " and number = '" + number + "' ";

back = teacherService.queryAll(wKey);

} else if (type == 2) {//学生：不能进后台

// wKey += " and number = '" + number + "' ";

// back = studentService.queryAll(wKey);

}

return R.ok(back);

}

@PostMapping("/user")

@ApiOperation(value = "登记用户")

public Object createUser(@RequestBody SysUserReg object,

@RequestHeader("token") String token) {

JSONObject user = cacheKit.user(token);

if (V.isEmpty(object.getLoginName()))

return R.error("请输入登录名");

if (V.isEmpty(object.getPassword()))

return R.error("请输入登录密码");

if (V.isLength(object.getPassword(), 6))

return R.error("必须配置6位数以上密码");

if (userService.existLoginName(object.getLoginName()))

return R.error(F.s("当前登录名[%s]已经被注册", object.getLoginName()));

if (object.getNumber() == null || object.getNumber().length() == 0) {

return R.error("请输入学号/工号");

}

if (userService.existNumber(object.getNumber())) {

return R.error("该学号/工号已存在");

}

if (userService.existPhone(object.getAttr().getMobile())) {

return R.error("该手机号已存在");

}

String salt = S.randomNum();

String pwd = MD5Kit.encode(object.getPassword() + salt);

JSONObject model = new JSONObject();

model.put("loginName", object.getLoginName());

model.put("password", pwd);

model.put("salt", salt);

model.put("number", object.getNumber());

model.put("phone", object.getAttr().getMobile());

model.put("type", object.getType());

model.put("creator", user.getString("id"));

model.put("roleId", object.getRoleId());

model = (JSONObject) userService.createUser(model);

if (model == null) {

return R.error("用户创建失败");

} else {

String id = model.getString("id");

//新增到个人信息表

UserBaseInfo userBaseInfo = new UserBaseInfo();

userBaseInfo.setId(id);

userBaseInfo.setName(object.getAttr().getTrueName());

userBaseInfo.setNumber(object.getNumber());

userBaseInfo.setType(object.getType());

userBaseInfo.setPhone(object.getAttr().getMobile());

userBaseInfoService.createUserBaseInfo(userBaseInfo);

return R.ok(model);

}

}

@PostMapping("/updatePassword")

@ApiOperation(value = "密码修改")

public Object updatePassword(@RequestBody UpdatePassword object, @RequestHeader("token") String token) {

String uid = cacheKit.getUserId(token);

int salt = S.randomNums();

List<SysUser> userList = userService.findByIds(uid);

if (userList == null || userList.size() <= 0) {

return R.error("查不到用户信息");

}

SysUser user = userList.get(0);

String passwordStr1 = object.getOldPassword().replace(" ", "+");

String passwordStr2 = object.getPassword().replace(" ", "+");

String passwordStr3 = object.getPasswordCheck().replace(" ", "+");

String privateKey = cacheKit.getVal(token + "\_privateKey");

String password1 = RSAEncrypt.decrypt(passwordStr1, privateKey);

String password2 = RSAEncrypt.decrypt(passwordStr2, privateKey);

String password3 = RSAEncrypt.decrypt(passwordStr3, privateKey);

cacheKit.deleteVal(token + "\_privateKey");

object.setOldPassword(password1);

object.setPassword(password2);

object.setPasswordCheck(password3);

String oldPassword = MD5Kit.encode(object.getOldPassword() + user.getSalt());

if (!oldPassword.equals(user.getPassword())) {

return R.error("旧密码不正确");

}

if (!object.getPassword().equals(object.getPasswordCheck())) {

return R.error("两次密码不相同");

}

String password = MD5Kit.encode(object.getPassword() + salt);

user.setReviser(uid);

user.setSalt(salt);

user.setPassword(password);

Object o = userService.updateSysUser(user);

if (o == null)

return R.error("修改失败");

return R.ok("修改成功", o);

}

@PutMapping("/user-pwd/{id}")

@ApiOperation(value = "密码重置")

public Object initPwd(@PathVariable("id") String id, @RequestHeader("token") String token) {

if (!userService.existId(id))

return R.error("Id信息异常");

String uid = cacheKit.getUserId(token);

String salt = S.randomNum();

String pwd = S.random(8);

String password = MD5Kit.encode(pwd + salt);

JSONObject model = new JSONObject();

model.put("id", id);

model.put("password", password);

model.put("salt", salt);

model.put("reviser", uid);

if (!userService.initPwd(model))

return R.error("密码重置失败");

model.remove("password");

model.put("password", pwd);

return R.ok(model);

}

@PutMapping("/user-status/{userId}-{status}")

@ApiOperation(value = "冻结/启用")

public Object initStatus(@PathVariable("userId") String userId,

@PathVariable("status") String status,

@RequestHeader("token") String token) {

if (!userService.existId(userId))

return R.error("Id信息异常");

JSONObject model = new JSONObject();

model.put("id", userId);

model.put("status", status);

Object o = userService.updateUser(model);

if (o == null)

return R.error("状态配置失败");

return R.ok("状态配置成功");

}

@PutMapping("/change-role/{id}-{roleId}")

@ApiOperation(value = "修改角色")

public Object changeRole(@PathVariable("id") String id,

@PathVariable("roleId") String roleId,

@RequestHeader("token") String token) {

String userId = cacheKit.getUserId(token);

if (!userService.existId(id))

return R.error("Id信息异常");

SysUser sysUser = userService.selectById(id);

String number = sysUser.getLoginName();

if (!checkUserDataAuth(token, number)) {

return R.error("您没有权限修改" + number + "的角色");

}

//读取 被修改用户的用户角色关系

SysUserRole userRole = userRoleService.findByUserId(id);

if (userRole == null) {

SysUserRole newUserRole = new SysUserRole();

newUserRole.setStatus(1);

newUserRole.setRoleId(roleId);

newUserRole.setUserId(id);

newUserRole.setSystemId("0");

newUserRole.setCreator(userId);

newUserRole = userRoleService.createUserRole(newUserRole);

if (newUserRole == null)

return R.error("修改失败");

return R.ok("修改成功", fm2(newUserRole));

}

userRole.setRoleId(roleId);

userRole.setReviser(userId);

userRole = userRoleService.updateUserRole(userRole);

if (userRole == null)

return R.error("修改失败");

return R.ok("修改成功", fm2(userRole));

}

@GetMapping("/user-system/{userId}")

@ApiOperation(value = "根据用户Id读取开通子系统信息")

public Object getSubSystemByUserId(@PathVariable("userId") String userId) {

List<JSONObject> srs = userService.getSubSystemByUserId(userId);

List<JSONObject> roles = roleService.getRoleInfo();

roles = F.f2l(roles, "roleId");

roles = F.f2l(roles, "systemId");

if (srs.size() > 0) {

for (JSONObject sr : srs) {

sr.put("role", getRoleBySystemId(roles, sr.getString("systemId")));

}

}

return R.ok(srs);

}

@PutMapping("/user-system/{userId}")

@ApiOperation(value = "配置子系统信息")

public Object setUserSystemRole(@RequestBody List<SysUserSystemRole> usr,

@PathVariable("userId") String userId) {

List<SysUserRole> urs = J.o2l(usr, SysUserRole.class);

if (userRoleService.setUserRole(userId, urs))

return R.ok("配置成功");

return R.error("配置失败");

}

private List<JSONObject> getRoleBySystemId(List<JSONObject> roles, String systemId) {

List<JSONObject> r = new ArrayList<>();

if (roles.size() > 0) {

for (JSONObject role : roles) {

if (V.isEqual(role.getString("systemId"), systemId)) {

r.add(role);

}

}

}

return r;

}

@DeleteMapping("/user/{userId}")

@ApiOperation(value = "删除用户信息")

public Object deleteUser(@PathVariable("userId") String userId,

@RequestHeader("token") String token) {

String uid = cacheKit.getUserId(token);

if (!userService.existId(userId))

return R.error("Id信息异常");

SysUser sysUser = userService.selectById(userId);

String number = sysUser.getLoginName();

if (!checkUserDataAuth(token, number)) {

return R.error("您没有权限删除" + number + "");

}

if (userService.deleteUser(userId, uid)) {

userBaseInfoService.deleteUserBaseInfo(userId, uid);

return R.ok("删除成功");

} else {

return R.error("删除失败");

}

}

@GetMapping("/user/exist/{number}")

@ApiOperation(value = "学号查询用户是否存在")

public Object exsistNumber(@PathVariable("number") String number,

@RequestHeader("token") String token) {

if (!userService.existNumber(number))

return R.error("用户不存在");

return R.ok("用户存在");

}

@GetMapping("/getPersonBaseAndDetail")

@ApiOperation(value = "获取用户基本和详细信息")

public Object getPersonBaseAndDetail(@RequestHeader("token") String token) {

String userId = getUserIdByCache(token);

JSONObject back = new JSONObject();

List<SysUser> sysUserList = userService.findByIds(userId);

//学号/公号

String number = sysUserList.get(0).getLoginName();

List<JSONObject> userBaseInfoList = userBaseInfoService.queryAll("and number ='" + number + "'");

if (userBaseInfoList == null || userBaseInfoList.size() == 0)

userBaseInfoList = null;

back.put("userBaseInfoList", userBaseInfoList);

back.put("userDetailInfoList", userBaseInfoList);

return R.ok(back);

}

@GetMapping("/user/getUserInfoById/{id}")

@ApiOperation(value = "根据id获取用户信息")

public Object getUserInfoById(@PathVariable("id") String id, @RequestHeader("token") String token) {

if (V.isEmpty(id)) {

return R.error("该用户不存在！");

}

List<SysUser> sysUserList = userService.findByIds(id);

if (V.isEmpty(sysUserList)) {

return R.error("该用户不存在！");

}

//学号/公号

List<JSONObject> userBaseInfoList = userBaseInfoService.queryAll("and id ='" + id + "'");

if (userBaseInfoList == null || userBaseInfoList.size() == 0)

userBaseInfoList = null;

return R.ok(userBaseInfoList.get(0));

}

@PostMapping("/user/importSetting")

@ApiOperation(value = "导入用户配置")

public Object importSetting(@RequestBody UserImport object,

@RequestHeader("token") String token) {

if (object.getRoleId() == null || object.getRoleId() == "") {

return R.error("未选择角色!");

}

if (object.getAcademyCode() == null || object.getAcademyCode() == "") {

return R.error("未选择学院!");

}

if (object.getType() == null || object.getType() == 0) {

return R.error("未选择用户类型!");

}

String batch = S.getToken();

cacheKit.setVal(batch + "roleId", object.getRoleId(), 0);

cacheKit.setVal(batch + "academyCode", object.getAcademyCode(), 0);

cacheKit.setVal(batch + "type", object.getType() + "", 0);

return R.ok("配置成功", batch);

}

@PostMapping("/user/importStatus/{batch}")

@ApiOperation(value = "导入用户状态")

public Object getImportStatus(@PathVariable("batch") String batch) {

if (V.isEmpty(batch)) {

return R.error("导入批次有误！");

}

JSONObject obj = getImportStatusObj(batch);

if (V.isEmpty(batch)) {

return R.error("导入状态异常！");

}

return R.ok(obj);

}

@PostMapping("/user/import/{batch}")

@ApiOperation(value = "导入用户")

public Object Import(

HttpServletRequest req,

@PathVariable("batch") String batch,

@RequestHeader("token") String token) {

if (V.isEmpty(batch)) {

return R.error("导入批次有误！");

}

if(V.isEmpty( cacheKit.getVal("user-import-lock-"+batch))) {

cacheKit.setVal("user-import-lock-"+batch, "true", 1800);

}else{

return R.error("导入操作正在进行中，请耐心等待操作结果");

}

if (cacheKit.getVal(batch + "roleId") == null) {

return R.error("未选择角色!");

}

if (cacheKit.getVal(batch + "academyCode") == null) {

return R.error("未选择学院!");

}

if (cacheKit.getVal(batch + "type") == null) {

return R.error("未选择用户类型!");

}

String academyCode = cacheKit.getVal(batch + "academyCode");

String examId = cacheKit.getVal(batch + "examId");

String roleId = cacheKit.getVal(batch + "roleId");

int type = Integer.parseInt(cacheKit.getVal(batch + "type")); //学生用户类型

String userId = getUserIdByCache(token);

String filepath = uploadFile(req);

if (V.isEmpty(filepath)) {

return R.error("文件上传失败！");

}

JSONObject result = new JSONObject();

JSONObject obj = new JSONObject();

obj.put("academyCode", academyCode);

obj.put("examId", examId);

obj.put("roleId", roleId);

obj.put("type", type);

obj.put("userid", userId);

obj.put("token", token);

obj.put("filepath", filepath);

JSONObject res = SolvedFile(obj, batch);

if (res == null) {

return R.error("用户导入异常！");

}

Boolean flag =res.getBoolean("flag");

if(!flag){

return R.error(res.getString("message"));

}

result.put("total", res.getInteger("total"));

result.put("truecount", res.getInteger("truecount"));

result.put("errcount", res.getInteger("errcount"));

if (res.getInteger("truecount") == 0 && res.getInteger("errcount") == 0) {

return R.error("导入数据为空！");

}

if (res.getInteger("errcount") > 0) {

result.put("errlist", res.getJSONArray("errlist"));

result.put("success", 0);

} else {

result.put("success", 1);

}

cacheKit.deleteVal("user-import-lock-"+batch);

return R.ok("导入成功", result);

}

public JSONObject SolvedFile(JSONObject obj, String batch) {

JSONObject res = new JSONObject();

JSONArray errors = new JSONArray();

if (obj.getString("filepath") == null || obj.getString("filepath") == "") {

res.put("flag",false);

res.put("message","上传文件有误！");

return res;

}

try {

File excel = new File(obj.getString("filepath"));

if (excel.isFile() && excel.exists()) { //判断文件是否存在

String[] split = excel.getName().split("\\."); //.是特殊字符，需要转义！！！！！

Workbook wb;

//根据文件后缀（xls/xlsx）进行判断

if ("xls".equals(split[1])) {

FileInputStream fis = new FileInputStream(excel); //文件流对象

wb = new HSSFWorkbook(fis);

} else if ("xlsx".equals(split[1])) {

wb = new XSSFWorkbook(excel);

} else {

res.put("flag",false);

res.put("message","上传文件有误！");

return res;

}

//开始解析

Sheet sheet = wb.getSheetAt(0); //读取sheet

int firstRowIndex = sheet.getFirstRowNum() + 1; //第一行是列名，所以不读

int lastRowIndex = sheet.getLastRowNum();

int userCount = 0, trueCount = 0;

for (int rIndex = firstRowIndex; rIndex <= lastRowIndex; rIndex++) { //遍历行

// System.out.println("rIndex: " + rIndex);

Row row = sheet.getRow(rIndex);

if (row != null) {

//String examId,String academyCode,String myclass,String name,String number,String phone,String token

String name, number, phone;

if (row.getCell(0) != null && row.getCell(1) != null) {

JSONObject userobj = new JSONObject();

number = row.getCell(0).toString().trim();

name = row.getCell(1).toString().trim();

if (row.getCell(2) != null) {

phone = row.getCell(2).toString().trim();

userobj.put("phone", phone);

}

userobj.put("name", name);

userobj.put("number", number);

if (number == null || number == "") {

continue;

}

userobj.put("academyCode", obj.getString("academyCode"));

userobj.put("roleId", obj.getString("roleId"));

userobj.put("type", obj.getInteger("type"));

userobj.put("userid", obj.getString("userid"));

userobj.put("token", obj.getString("token"));

JSONObject importObj = userService.importUser(userobj);

userobj.remove("academyCode");

userobj.remove("roleId");

userobj.remove("type");

userobj.remove("userid");

userobj.remove("token");

userCount++;

if (importObj != null) {

boolean userFlag = importObj.getBoolean("flag");

if (!userFlag) {

userobj.put("info", importObj.getString("info"));

errors.add(userobj);

cacheKit.setVal(batch + "status", 2 + "", 0);

} else {

if (obj.getString("phone") != null || obj.getString("phone") != "") {

//发送消息提醒

StringBuffer content = new StringBuffer();

content.append("#{name}你好，" + "您的登陆账号为：" + number + ",初始密码为："

+ "gxun" + number.substring(number.length() - 6) + ",请尽快登陆系统修改密码！");

}

trueCount++;

cacheKit.setVal(batch + "status", 1 + "", 0);

}

} else {

cacheKit.setVal(batch + "status", 2 + "", 0);

userobj.put("info", "其他原因");

errors.add(userobj);

}

cacheKit.setVal(batch + "now", rIndex + "", 0);

}

cacheKit.setVal(batch + "total", lastRowIndex + "", 0);

if (rIndex != lastRowIndex) {

cacheKit.setVal(batch + "res", 2 + "", 0);

}

}

}

cacheKit.setVal(batch + "res", 1 + "", 0);

res.put("truecount", trueCount);

res.put("errcount", errors.size());

res.put("total", userCount);

res.put("errlist", errors);

res.put("flag",true);

return res;

} else {

res.put("flag", false);

res.put("message", "上传文件有误！");

System.out.println("找不到指定的文件");

return res;

}

} catch (Exception e) {

e.printStackTrace();

res.put("flag", false);

res.put("message", "学生处理出错！");

return res;

}

}

private Map<String, String> updateBaseInfoVerify() {

Map<String, String> verify = new HashMap<>();

verify.put("name", "姓名");

// verify.put("oldName", "曾用名");

verify.put("number", "学号/工号");

// verify.put("sex", "性别");

// verify.put("nationCode", "民族代码");

// verify.put("birthday", "出生日期");

// verify.put("graduate", "学历");

// verify.put("degree", "学位或职称");

// verify.put("nationality", "国籍");

// verify.put("originPlace", "籍贯");

// verify.put("birthPlace", "出生地");

// verify.put("presentPesidence", "现居住地");

verify.put("phone", "手机号");

// verify.put("email", "邮箱");

// verify.put("specialty", "特长");

verify.put("idCard", "身份证");

return verify;

}

}