# 一、功能代码

1、发送短信接口

package com.example.dcloud.controller;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONObject;

import com.example.dcloud.annotation.NoToken;

import com.example.dcloud.service.MessageService;

import com.example.dcloud.util.ResultUtil;

import io.swagger.annotations.Api;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

import javax.annotation.Resource;

import java.util.Map;

import java.util.Random;

@Api(tags = "发送短信验证码接口")

@CrossOrigin

@RestController

public class MessageController {

@Autowired

MessageService messageService;

private static final Logger LOG = LoggerFactory.getLogger(MessageController.class);

@PostMapping(value = "/sendMessage")

public String sendMessage(@RequestParam(value="telephone")String telephone)

{

String checkCode = String.valueOf(new Random().nextInt(899999)+100000);

try {

LOG.info("======sendMessage======telephone：{}, checkCode:{}",telephone,checkCode);

messageService.sendMessage(telephone,checkCode);

return ResultUtil.error(checkCode);

}catch (Exception e){

//输出日志

LOG.info("======sendMessage======手机号错误，请输入真实邮箱");

e.printStackTrace();

return ResultUtil.error("请输入真实手机号");

}

}

}

1. 发送短信service层

package com.example.dcloud.service;

import com.github.qcloudsms.SmsSingleSender;

import com.github.qcloudsms.SmsSingleSenderResult;

import com.github.qcloudsms.httpclient.HTTPException;

import org.json.JSONException;

import com.github.qcloudsms.SmsSingleSender;

import com.github.qcloudsms.SmsSingleSenderResult;

import com.github.qcloudsms.httpclient.HTTPException;

import org.json.JSONException;

import org.springframework.stereotype.Service;

import java.io.IOException;

import java.io.IOException;

import java.util.Random;

@Service

public class MessageService {

public void sendMessage(String telephone,String checkCode){ // 短信应用SDK AppID

int appid =; // 1400开头

// 短信应用SDK AppKey

String appkey = "";

// 需要发送短信的手机号码

//签名参数

String smsSign = "";

// 短信模板ID，需要在短信应用中申请

int templateId =; // NOTE: 真实的模板ID需要在短信控制台中申请

//templateId7839对应的内容是"您的验证码是: {1}"

try {

String[] params = {checkCode};

// params[0] = checkCode;

SmsSingleSender ssender = new SmsSingleSender(appid, appkey);

SmsSingleSenderResult result = ssender.sendWithParam("86", telephone,

templateId, params,smsSign, "", "");

System.out.println(result);

} catch (HTTPException e) {

// HTTP 响应码错误

e.printStackTrace();

} catch (JSONException e) {

// JSON 解析错误

e.printStackTrace();

} catch (IOException e) {

// 网络 IO 错误

e.printStackTrace();

}

}

}

1. 注册接口

package com.example.dcloud.controller;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONObject;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;

import com.example.dcloud.annotation.NoToken;

import com.example.dcloud.entity.User;

import com.example.dcloud.entity.UserAuths;

import com.example.dcloud.service.UserService;

import com.example.dcloud.util.ResultUtil;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import io.swagger.annotations.ApiParam;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.stereotype.Controller;

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

import java.util.Map;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

@Api(tags = "注册接口")

@CrossOrigin

@RestController

public class RegisterController {

@Autowired

UserService userService;

private static final Logger LOG = LoggerFactory.getLogger(RegisterController.class);

@NoToken

@ApiOperation(value = "短信注册",notes = "get")

@PostMapping("/registerByMessage")

public String registerByMessage(@RequestParam(value="telephone")String telephone)

{

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

queryWrapper.eq("telephone",telephone);

int count = userService.count(queryWrapper);

if(count>0){

//输出日志

LOG.info("======register======telephone：{}===注册失败（该手机号已经注册过了）",telephone);

return ResultUtil.error("该手机号已经注册过了！");

}

else{

//输出日志

LOG.info("======register======telephone：{}",telephone);

User user1 = new User();

user1.setEmail("0");

user1.setExp(0);

user1.setIsDelete(0);

user1.setNickname("0");

user1.setSno("0");

user1.setName("0");

user1.setImage("0");

user1.setBirth("0");

user1.setTelephone(telephone);

user1.setSex(0);

user1.setEducation(0);

user1.setPassword("123456");

user1.setPowerId("0");

user1.setSchoolCode("0");

user1.setState(0);

user1.setRoleId(0);

userService.save(user1);

return ResultUtil.success();

}

}

@NoToken

@ApiOperation(value = "邮箱注册",notes = "get")

@PostMapping(value = "/registerByEmail")

public String registerByEmail(@RequestParam(value="email")String email,

@RequestParam(value="password")String password)

{

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

queryWrapper.eq("email",email);

int count = userService.count(queryWrapper);

System.out.println(count);

if(count>0){

return ResultUtil.error("该邮箱已经注册过了！");

}

else{

//输出日志

LOG.info("======register======email：{}, password：{}===注册成功",email,password);

User user1 = new User();

user1.setEmail(email);

user1.setExp(0);

user1.setIsDelete(0);

user1.setNickname("0");

user1.setSno("0");

user1.setName("0");

user1.setImage("0");

user1.setBirth("0");

user1.setTelephone("0");

user1.setSex(0);

user1.setEducation(0);

user1.setPassword(password);

user1.setPowerId("0");

user1.setSchoolCode("0");

user1.setState(0);

user1.setRoleId(0);

userService.save(user1);

return ResultUtil.success();

}

}

}

4、登录接口

package com.example.dcloud.controller;

import com.auth0.jwt.JWT;

import com.auth0.jwt.algorithms.Algorithm;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;

import com.example.dcloud.annotation.NoToken;

import com.example.dcloud.common.ServerResponse;

import com.example.dcloud.entity.User;

import com.example.dcloud.service.UserService;

import com.example.dcloud.vo.loginByMessageVo;

import com.example.dcloud.vo.loginByPasswordVo;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

import java.util.ArrayList;

import java.util.List;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

@Api(tags = "登录接口")

@CrossOrigin

@RestController

public class LoginController {

@Autowired

UserService userService;

private static final Logger LOG = LoggerFactory.getLogger(LoginController.class);

@NoToken

@ApiOperation(value = "账号密码登录",notes = "get")

@PostMapping(value = "/loginByPassword")

public ServerResponse<loginByPasswordVo> loginByPassword(@RequestParam(value="account")String account,

@RequestParam(value="password")String password)

{

String emailCheck = "^([a-z0-9A-Z]+[-|\\.]?)+[a-z0-9A-Z]@([a-z0-9A-Z]+(-[a-z0-9A-Z]+)?\\.)+[a-zA-Z]{2,}$";

Pattern emailRegex = Pattern.compile(emailCheck);

Matcher emailMatcher = emailRegex.matcher(account);

boolean emailIsMatched = emailMatcher.matches();

String phoneCheck = "^((13[0-9])|(15[^4,\\D])|(18[0,5-9]))\\d{8}$";

Pattern phoneRegex = Pattern.compile(phoneCheck);

Matcher phoneMatcher = phoneRegex.matcher(account);

boolean phoneIsMatched = phoneMatcher.matches();

System.out.println("是否是邮箱："+emailIsMatched);

System.out.println("是否是手机号："+phoneIsMatched);

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

queryWrapper.eq("is\_delete",0);

System.out.println(userService.count(queryWrapper));

if(emailIsMatched==true){

queryWrapper.eq("email",account);

}

if(phoneIsMatched==true){

queryWrapper.eq("telephone",account);

}

ServerResponse<loginByPasswordVo> response = new ServerResponse<>();

loginByPasswordVo loginVo = new loginByPasswordVo();

List<loginByPasswordVo> dataList = new ArrayList<>();

if(userService.count(queryWrapper)==0){

//输出日志

LOG.info("======loginByPassword======account：{}, password：{}===登录失败（账号或密码错误）",account,password);

response.setMsg("账号或密码错误");

response.setResult(false);

return response;

}else{

queryWrapper.eq("password",password);

if(userService.count(queryWrapper)==0){

//输出日志

LOG.info("======loginByPassword======account：{}, password：{}===登录失败（账号或密码错误）",account,password);

response.setMsg("账号或密码错误");

response.setResult(false);

return response;

}else{

//输出日志

LOG.info("======loginByPassword======account：{}, password：{}===登录成功",account,password);

User user = userService.getOne(queryWrapper);

String token = JWT.create().withAudience(user.getId() + "")

.sign(Algorithm.HMAC256(user.getPassword()));

loginVo.setRole(user.getRoleId());

if(emailIsMatched==true) {

loginVo.setAccount(user.getEmail());

}

else{

loginVo.setAccount(user.getTelephone());

}

loginVo.setToken(token);

dataList.add(loginVo);

response.setDataList(dataList);

response.setMsg("登录成功");

response.setResult(true);

return response;

}

}

}

@NoToken

@ApiOperation(value = "邮箱验证码登录",notes = "get")

@PostMapping(value = "/loginByCode")

public ServerResponse<loginByMessageVo> loginByCode(@RequestParam(value="email")String email)

{

ServerResponse<loginByMessageVo> response = new ServerResponse<>();

if(userService.loginByCode(email)==1){

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

queryWrapper.eq("email",email);

User user = userService.getOne(queryWrapper);

if(user.getIsDelete()==1){

response.setMsg("账号已被删除！");

response.setResult(false);

return response;

}

String token = JWT.create().withAudience(user.getId() + "")

.sign(Algorithm.HMAC256(user.getPassword()));

LOG.info("======loginByCode验证码登录======email：{}===登录成功",email);

List<loginByMessageVo> dataList = new ArrayList<>();

loginByMessageVo loginVo = new loginByMessageVo();

int role\_id = userService.selectRole(email);

loginVo.setRole(role\_id);

loginVo.setToken(token);

dataList.add(loginVo);

response.setDataList(dataList);

response.setMsg("登录成功！");

response.setResult(true);

return response;

}

else{

//输出日志

LOG.info("======loginByCode验证码登录======email：{}===登录失败（账号不存在）",email);

response.setResult(false);

response.setMsg("账号不存在!");

return response;

}

}

@NoToken

@ApiOperation(value = "短信验证码登录",notes = "get")

@PostMapping(value = "/loginByMessage")

public ServerResponse<loginByMessageVo> loginByMessage(@RequestParam(value="telephone")String telephone) {

ServerResponse<loginByMessageVo> response = new ServerResponse<>();

if (userService.loginByMessage(telephone) == 1) {

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

queryWrapper.eq("telephone", telephone);

User user = userService.getOne(queryWrapper);

if (user.getIsDelete() == 1) {

response.setMsg("账号已被删除！");

response.setResult(false);

// return response;

}

String token = JWT.create().withAudience(user.getId() + "")

.sign(Algorithm.HMAC256(user.getPassword()));

//输出日志

LOG.info("======loginByMessage短信验证码登录======telephone：{}===登录成功", telephone);

int role\_id = userService.selectRoleByTelephone(telephone);

loginByMessageVo loginVo = new loginByMessageVo();

loginVo.setRole(role\_id);

loginVo.setToken(token);

List<loginByMessageVo> datalist = new ArrayList<>();

datalist.add(loginVo);

response.setDataList(datalist);

response.setMsg("登录成功");

response.setResult(true);

} else {

//输出日志

LOG.info("======loginByMessage短信验证码登录======telephone：{}===登录失败（账号不存在）", telephone);

response.setResult(false);

response.setMsg("账号不存在");

// return response;

}

return response;

}

}

5、签到相关接口

package com.example.dcloud.controller;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONArray;

import com.alibaba.fastjson.JSONObject;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;

import com.example.dcloud.common.ServerResponse;

import com.example.dcloud.entity.\*;

import com.example.dcloud.service.\*;

import com.example.dcloud.util.ResultUtil;

import com.example.dcloud.vo.CourseVo;

import com.example.dcloud.vo.getAttendenceVo;

import com.example.dcloud.vo.historyAttendenceVo;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.stereotype.Controller;

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

import java.text.SimpleDateFormat;

import java.util.\*;

@Api(tags = "签到接口")

@Controller

@CrossOrigin

@RequestMapping("/attendence")

public class AttendenceController {

private static final Logger LOG = LoggerFactory.getLogger(AttendenceController.class);

@Autowired

private AttendenceService attendenceService;

@Autowired

private AttendenceResultService attendenceResultService;

@Autowired

private UserService userService;

@Autowired

private CourseStudentService courseStudentService;

@Autowired

private CourseService courseService;

@Autowired

private SystemManageService systemManageService;

@ResponseBody

//查看这个签到是否结束

//0表示未结束

@ApiOperation(value = "查看这个签到是否结束",notes = "get")

@RequestMapping(value = "/isEnd",method = RequestMethod.POST)

public String hasAttendence(@RequestParam(value="code")String code) {

QueryWrapper<Course> queryCourse = new QueryWrapper<>();

queryCourse.eq("code",code)

.eq("isDelete",0);

Course course = courseService.getOne(queryCourse);

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

queryWrapper.eq("course\_id",course.getId())

.eq("is\_delete",0);

int count = attendenceService.count(queryWrapper);

if(count > 0){

//有未结束签到

return ResultUtil.error("签到未结束！");

}else{

return ResultUtil.error("签到已结束！");

}

}

//结束签到

@ResponseBody

@ApiOperation(value = "结束这个签到",notes = "get")

@RequestMapping(value = "/end",method = RequestMethod.POST)

public String endPriorAttendence(@RequestParam(value="code")String code) {

//Attendence attendence = new Attendence();

QueryWrapper<Course> queryCourse = new QueryWrapper<>();

queryCourse.eq("code",code)

.eq("isDelete",0);

Course course = courseService.getOne(queryCourse);

//判断是否有签到正在进行

QueryWrapper<Attendence> queryWrapper1 = new QueryWrapper<>();

queryWrapper1.eq("course\_id", course.getId())

.eq("is\_delete",0);

int count = attendenceService.count(queryWrapper1);

if (count == 0)

{

return ResultUtil.error("签到均已结束！");

}

Attendence attendence = attendenceService.getOne(queryWrapper1);

attendence.setIsDelete(1);

attendenceService.update(attendence,queryWrapper1);

return ResultUtil.success();

}

@ResponseBody

@ApiOperation(value = "发起签到。返回一个签到id，根据签到id去结束签到。返回0说明有签到正在进行，不能再次发起签到",notes = "get")

@RequestMapping(value = "/createAttendence",method = RequestMethod.POST)

public int createAttendence(@RequestParam(value="code")String code,

@RequestParam(value="start\_time")String start\_time,

@RequestParam(value="end\_time",required = false)String end\_time,

@RequestParam(value="count",required = false)Integer count,

@RequestParam(value="longitude")Double longitude,

@RequestParam(value="latitude")Double latitude,

@RequestParam(value="attendance\_type")Integer attendance\_type,

@RequestParam(value="telephone")String telephone

) {

Attendence attendence = new Attendence();

QueryWrapper<Course> queryCourse = new QueryWrapper<>();

queryCourse.eq("code",code)

.eq("isDelete",0);

Course course = courseService.getOne(queryCourse);

//判断是否有签到正在进行

QueryWrapper<Attendence> queryWrapper1 = new QueryWrapper<>();

queryWrapper1.eq("course\_id", course.getId())

.eq("is\_delete",0);

int count1 = attendenceService.count(queryWrapper1);

if(count1>0)

{

return 0;

}

attendence.setLongitude(longitude);

attendence.setLatitude(latitude);

attendence.setAttendanceType(attendance\_type);

QueryWrapper<User> queryUser = new QueryWrapper<>();

queryUser.eq("telephone",telephone)

.eq("is\_delete",0);

User user = userService.getOne(queryUser);

attendence.setCreater(user.getName());

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

try{

Date date = sdf.parse(start\_time);

attendence.setStartTime(date);

attendence.setCreateTime(date);

}catch (Exception ex)

{

System.out.println(ex.getMessage());

}

if(end\_time!=null)

{

try{

Date date = sdf.parse(end\_time);

attendence.setEndTime(date);

}catch (Exception ex)

{

System.out.println(ex.getMessage());

}

}

if (count!=null)

{

attendence.setCount(count);

}

attendence.setIsDelete(0);

attendence.setCourseId(course.getId());

attendenceService.saveOrUpdate(attendence);

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

queryWrapper.eq("course\_id", course.getId())

// .eq("start\_time", start\_time)

.eq("is\_delete",0);

Attendence attendence1 = attendenceService.getOne(queryWrapper);

return attendence1.getId();

}

//一开始签到距离过远，之后再次签到

@ResponseBody

@RequestMapping(value = "/checkIn",method = RequestMethod.PUT)

@ApiOperation(value = "学生签到",notes = "get")

public String checkIn(@RequestParam(value="code")String code,

@RequestParam(value="telephone")String telephone,

@RequestParam(value="longitude")Double longitude,

@RequestParam(value="latitude")Double latitude,

@RequestParam(value="attend\_time")String attend\_time

){

//ban

//查看签到是否结束

QueryWrapper<Course> queryCourse = new QueryWrapper<>();

queryCourse.eq("code",code)

.eq("isDelete",0);

Course course = courseService.getOne(queryCourse);

QueryWrapper<Attendence> queryAttendence = new QueryWrapper<>();

queryAttendence.eq("course\_id",course.getId())

.eq("is\_delete",0);

//不管签到成功与否，都要新增一条签到的记录

AttendenceResult attendenceResult = new AttendenceResult();

attendenceResult.setCourseId(course.getId());

QueryWrapper<User> queryUser = new QueryWrapper<>();

queryUser.eq("telephone",telephone)

.eq("is\_delete",0);

User user = userService.getOne(queryUser);

attendenceResult.setStudentId(user.getId());

int count = attendenceService.count(queryAttendence);

if (count == 0)

{

attendenceResult.setLongitude(longitude);

attendenceResult.setLatitude(latitude);

attendenceResult.setIsDelete(1);//1表示未签到

attendenceResultService.save(attendenceResult);

return ResultUtil.error("签到已结束！");

}

//未结束

Attendence attendence = attendenceService.getOne(queryAttendence);

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

queryWrapper.eq("student\_id",user.getId())

.eq("attend\_id",attendence.getId())

.eq("is\_delete",0);

int count2 = attendenceResultService.count(queryWrapper);

if(count2>0)

{

return ResultUtil.error("已签到，请勿重复签到！");

}

//求距离

Double distance1 = Math.sqrt((attendence.getLongitude()-longitude)\*(attendence.getLongitude()-longitude)+(attendence.getLatitude()-latitude)\*(attendence.getLatitude()-latitude));

//获取系统参数

QueryWrapper<SystemManage> querySystem = new QueryWrapper<>();

querySystem.eq("key\_name","distance")

.eq("is\_delete",0);

SystemManage systemManage = systemManageService.getOne(querySystem);

if (distance1>systemManage.getKeyValue())

{

attendenceResult.setLongitude(longitude);

attendenceResult.setLatitude(latitude);

attendenceResult.setDistance(distance1);

attendenceResult.setIsDelete(1);//1表示未签到

attendenceResultService.save(attendenceResult);

return ResultUtil.error("签到距离过远，无法签到！");

}

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

try{

Date date = sdf.parse(attend\_time);

attendenceResult.setAttendTime(date);

}catch (Exception ex)

{

System.out.println(ex.getMessage());

}

//保存

attendenceResult.setIsDelete(0);

attendenceResult.setLongitude(longitude);

attendenceResult.setLatitude(latitude);

attendenceResult.setDistance(distance1);

attendenceResult.setAttendId(attendence.getId());

attendenceResultService.saveOrUpdate(attendenceResult);

//对应课程加经验值，课程学生表course\_student

QueryWrapper<CourseStudent> queryCourseStudent = new QueryWrapper<>();

queryCourseStudent.eq("course\_id",course.getId())

.eq("student\_id",user.getId())

.eq("is\_delete",0);

CourseStudent courseStudent = courseStudentService.getOne(queryCourseStudent);

//获取系统参数

QueryWrapper<SystemManage> querySystem1 = new QueryWrapper();

querySystem1.eq("key\_name","experience")

.eq("is\_delete",0);

SystemManage systemManage1 = systemManageService.getOne(querySystem1);

//增加该门课程的经验值

courseStudent.setExp(courseStudent.getExp()+systemManage1.getKeyValue());

courseStudentService.update(courseStudent,queryCourseStudent);

//增加总的经验值

user.setExp(user.getExp()+systemManage1.getKeyValue());

userService.update(user,queryUser);

return ResultUtil.success();

}

@ResponseBody

@RequestMapping(value = "/checkHelp",method = RequestMethod.PUT)

@ApiOperation(value = "老师直接帮学生补签",notes = "get")

public String checkHelp(@RequestParam(value="code")String code,

@RequestParam(value="student\_id")Integer student\_id,

@RequestParam(value="attend\_time")String attend\_time,

@RequestParam(value="attend\_id")Integer attend\_id

){

QueryWrapper<Course> queryCourse = new QueryWrapper<>();

queryCourse.eq("code",code)

.eq("isDelete",0);

Course course = courseService.getOne(queryCourse);

AttendenceResult attendenceResult = new AttendenceResult();

attendenceResult.setCourseId(course.getId());

attendenceResult.setStudentId(student\_id);

attendenceResult.setAttendId(attend\_id);

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

try{

Date date = sdf.parse(attend\_time);

attendenceResult.setAttendTime(date);

}catch (Exception ex)

{

System.out.println(ex.getMessage());

}

attendenceResult.setIsDelete(0);

attendenceResultService.save(attendenceResult);

//对应课程加经验值，课程学生表course\_student

QueryWrapper<CourseStudent> queryCourseStudent = new QueryWrapper<>();

queryCourseStudent.eq("course\_id",course.getId())

.eq("student\_id",student\_id)

.eq("is\_delete",0);

CourseStudent courseStudent = courseStudentService.getOne(queryCourseStudent);

//获取系统参数

QueryWrapper<SystemManage> querySystem1 = new QueryWrapper();

querySystem1.eq("key\_name","experience")

.eq("is\_delete",0);

SystemManage systemManage1 = systemManageService.getOne(querySystem1);

//增加该门课程的经验值

courseStudent.setExp(courseStudent.getExp()+systemManage1.getKeyValue());

courseStudentService.update(courseStudent,queryCourseStudent);

//增加总的经验值

QueryWrapper<User> queryUser = new QueryWrapper<>();

queryUser.eq("id",student\_id)

.eq("is\_delete",0);

User user = userService.getOne(queryUser);

user.setExp(user.getExp()+systemManage1.getKeyValue());

userService.update(user,queryUser);

return ResultUtil.success();

}

@ResponseBody

@ApiOperation(value = "查看某门课，某次学生的签到情况",notes = "get")

@RequestMapping(value = "/getAttendence",method = RequestMethod.PUT)

public ServerResponse<getAttendenceVo> getAttendence(@RequestParam(value="code")String code,

@RequestParam(value="attend\_id")Integer attend\_id) {

ServerResponse<getAttendenceVo> response = new ServerResponse<>();

// QueryWrapper<AttendenceResult> queryWrapper = new QueryWrapper<>();

QueryWrapper<Course> queryWrapper2= new QueryWrapper<>();

queryWrapper2.eq("code",code);

Course course = courseService.getOne(queryWrapper2);

//queryWrapper.eq("attend\_id",attend\_id)

//.eq("course\_id",course.getId());

// QueryWrapper<Attendence> queryAttendence = new QueryWrapper<>();

// queryAttendence.eq("id",attend\_id);

// Attendence attendence = attendenceService.getOne(queryAttendence);

// List<AttendenceResult> list = attendenceResultService.list(queryWrapper);

List<getAttendenceVo> dataList = new ArrayList<>();

QueryWrapper<CourseStudent> courseStudentQueryWrapper = new QueryWrapper<>();

courseStudentQueryWrapper.eq("course\_id",course.getId());

//获取学生列表

List<CourseStudent> listStudent = courseStudentService.list(courseStudentQueryWrapper);

for(int i=0;i<listStudent.size();i++){

getAttendenceVo getAttendenceVo = new getAttendenceVo();

int studentId = listStudent.get(i).getStudentId();

QueryWrapper<User> queryWrapper1 = new QueryWrapper<>();

queryWrapper1.eq("id",studentId);

User user = userService.getOne(queryWrapper1);

getAttendenceVo.setStudentId(studentId);

getAttendenceVo.setName(user.getName());

//看下签到结果表里是否有签到记录

QueryWrapper<AttendenceResult> attendenceResultQueryWrapper = new QueryWrapper<>();

attendenceResultQueryWrapper.eq("attend\_id",attend\_id)

.eq("student\_id",studentId);

AttendenceResult attendenceResult = attendenceResultService.getOne(attendenceResultQueryWrapper);

if(attendenceResult!=null) {

getAttendenceVo.setAttendTime(attendenceResult.getAttendTime());

getAttendenceVo.setDistance(attendenceResult.getDistance());

getAttendenceVo.setHasAttendence(attendenceResult.getIsDelete());//0代表已签到，1代表未签到

getAttendenceVo.setLongitude(attendenceResult.getLongitude());

getAttendenceVo.setLatitude(attendenceResult.getLatitude());

}

//如果attendenceResult为空，说明还没签到

else {

getAttendenceVo.setHasAttendence(1);

}

dataList.add(getAttendenceVo);

}

response.setResult(true);

response.setDataList(dataList);

response.setTotal((long)listStudent.size());

response.setMsg("hasAttendence值为0表示已签到，1为未签到。total值为该门课的学生人数。");

return response;

}

@ResponseBody

@ApiOperation(value = "教师发起签到的历史记录",notes = "get")

@RequestMapping(value = "/historyAttendence",method = RequestMethod.PUT)

public ServerResponse<historyAttendenceVo> historyAttendence(@RequestParam(value="code")String code) {

ServerResponse<historyAttendenceVo> response = new ServerResponse<>();

QueryWrapper<Course> queryWrapper31= new QueryWrapper<>();

queryWrapper31.eq("code",code);

Course course1 = courseService.getOne(queryWrapper31);

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

queryWrapper.eq("course\_id",course1.getId())

.eq("is\_delete",1)

.orderByDesc("id");

List<Attendence> list = attendenceService.list(queryWrapper);

//通过课程号查找课程id

QueryWrapper<Course> queryWrapper1= new QueryWrapper<>();

queryWrapper1.eq("code",code);

Course course = courseService.getOne(queryWrapper1);

long courseId = course.getId();

//通过该课程总学生数

QueryWrapper<CourseStudent> queryWrapper2 = new QueryWrapper<>();

queryWrapper2.eq("course\_id",courseId)

.eq("is\_delete",0);

int total = courseStudentService.count(queryWrapper2);

List<historyAttendenceVo> dataList = new ArrayList<>();

for(int i=0;i<list.size();i++){

historyAttendenceVo historyAttendenceVo = new historyAttendenceVo();

historyAttendenceVo.setStartTime(list.get(i).getStartTime());

historyAttendenceVo.setAttendanceType(list.get(i).getAttendanceType());

int attendId = list.get(i).getId();

QueryWrapper<AttendenceResult> queryWrapper3 = new QueryWrapper<>();

queryWrapper3.eq("attend\_id",attendId)

.eq("is\_delete",0);

int count = attendenceResultService.count(queryWrapper3);

historyAttendenceVo.setCount(count);

historyAttendenceVo.setTotal(total);

historyAttendenceVo.setAttendId(attendId);

dataList.add(historyAttendenceVo);

}

response.setTotal((long)total);

response.setDataList(dataList);

response.setResult(true);

response.setMsg("total表示课程学生数，count表示签到的人数");

return response;

}

}

6、班课相关接口

package com.example.dcloud.controller;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONArray;

import com.alibaba.fastjson.JSONException;

import com.alibaba.fastjson.JSONObject;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;

import com.baomidou.mybatisplus.core.conditions.update.UpdateWrapper;

import com.example.dcloud.common.ServerResponse;

import com.example.dcloud.dto.CourseDto;

import com.example.dcloud.dto.UpdateCourseDto;

import com.example.dcloud.entity.\*;

import com.example.dcloud.service.\*;

import com.example.dcloud.util.QRCodeGenUtil;

import com.example.dcloud.util.ResultUtil;

import com.example.dcloud.vo.CourseStudentVo;

import com.example.dcloud.vo.CourseVo;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.apache.tomcat.util.json.JSONParser;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

import org.springframework.stereotype.Controller;

import javax.jws.soap.SOAPBinding;

import java.util.\*;

import static java.lang.Integer.parseInt;

import static java.lang.Long.parseLong;

@Api(tags = "班课管理接口")

@CrossOrigin

@Controller

@RequestMapping("/courses")

public class CourseController {

@Autowired

private CourseService courseService;

@Autowired

private AttendenceResultService attendenceResultService;

@Autowired

private CourseStudentService courseStudentService;

@Autowired

private UserService userService;

private static final Logger LOG = LoggerFactory.getLogger(CourseController.class);

@ResponseBody

@RequestMapping(value = "/courseInfo",method = RequestMethod.GET)

@ApiOperation(value = "根据班课号，获取具体班课信息",notes = "get")

public ServerResponse<Course> courseInfo(@RequestParam(value="code")String code) {

ServerResponse<Course> response = new ServerResponse<>();

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

queryWrapper.eq("code",code);

int count = courseService.count(queryWrapper);

if(count == 0)

{

response.setMsg("不存在此班课");

response.setResult(false);

return response;

}

Course course = courseService.getOne(queryWrapper);

List<Course> dataList = new ArrayList<>();

dataList.add(course);

response.setDataList(dataList);

response.setResult(true);

response.setMsg("查询成功");

return response;

}

@ResponseBody

@RequestMapping(value = "/hasJoined",method = RequestMethod.GET)

@ApiOperation(value = "根据手机号，查询用户已经加入的课程",notes = "get")

public ServerResponse<CourseVo> hasJoined(@RequestParam(value="telephone")String telephone) {

ServerResponse<CourseVo> response = new ServerResponse<>();

QueryWrapper<User> queryUser = new QueryWrapper<>();

queryUser.eq("telephone",telephone)

.eq("is\_delete",0);

User user = userService.getOne(queryUser);

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

queryWrapper.eq("student\_id",user.getId())

.eq("is\_delete",0);

int count = courseStudentService.count(queryWrapper);

if(count == 0)

{

response.setMsg("没有加入任何班课！");

response.setResult(false);

return response;

}

//找到对应的课程id

List<CourseStudent> list = courseStudentService.list(queryWrapper);

List<CourseVo> dataList = new ArrayList<>();

for (int i = 0; i < list.size(); i++) {

CourseVo courseVo = new CourseVo();

QueryWrapper<Course> queryWrapper1 = new QueryWrapper<>();

queryWrapper1.eq("id", list.get(i).getCourseId());

Course course = courseService.getOne(queryWrapper1);

courseVo.setName(course.getName());

//根据教师id找教师名字

QueryWrapper<User> queryWrapper2 = new QueryWrapper<>();

queryWrapper2.eq("id", course.getTeacherId());

User user1 = userService.getOne(queryWrapper2);

courseVo.setTeacherName(user1.getName());

courseVo.setClassName(course.getClassName());

courseVo.setSemester(course.getSemester());

courseVo.setCode(course.getCode());

dataList.add(courseVo);

}

response.setDataList(dataList);

response.setMsg("查询成功");

response.setResult(true);

response.setTotal((long)list.size());

return response;

}

@ResponseBody

@RequestMapping(value = "/hasCreated",method = RequestMethod.GET)

@ApiOperation(value = "根据手机号，查询用户已经创建的课程",notes = "get")

public ServerResponse<CourseVo> hasCreated(@RequestParam(value="telephone")String telephone) {

ServerResponse<CourseVo> response = new ServerResponse<>();

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

queryWrapper.eq("telephone",telephone);

User user = userService.getOne(queryWrapper);

//找到该userid对应的课程

QueryWrapper<Course> courseQueryWrapper = new QueryWrapper<>();

courseQueryWrapper.eq("teacher\_id",user.getId())

.eq("isDelete",0);

int count = courseService.count(courseQueryWrapper);

if(count == 0)

{

response.setMsg("没有创建任何班课！");

response.setResult(false);

return response;

}

List<Course> list = courseService.list(courseQueryWrapper);

List<CourseVo> dataList = new ArrayList<>();

for(int i = 0 ; i < list.size(); i++) {

CourseVo courseVo = new CourseVo();

courseVo.setCode(list.get(i).getCode());

courseVo.setSemester(list.get(i).getSemester());

courseVo.setClassName(list.get(i).getClassName());

courseVo.setTeacherName(user.getName());

courseVo.setName(list.get(i).getName());

dataList.add(courseVo);

}

response.setDataList(dataList);

response.setTotal((long)list.size());

response.setResult(true);

response.setMsg("查询成功!");

return response;

}

@ResponseBody

@ApiOperation(value = "获取班课成员",notes = "get")

@RequestMapping(value = "/getMember",method = RequestMethod.GET)

//获取班课成员

public ServerResponse<CourseStudentVo> getMember(@RequestParam(value="code")String code

//@RequestParam(value="order",required = false)String order,

//@RequestParam(value="search",required = false)String search,

// @RequestParam(value="email",required = false)String email

){

ServerResponse<CourseStudentVo> response = new ServerResponse<CourseStudentVo>();

QueryWrapper<Course> courseQuery = new QueryWrapper<>();

courseQuery.eq("code",code);

Course course = courseService.getOne(courseQuery);

QueryWrapper<CourseStudent> courseId = new QueryWrapper<>();

courseId.eq("course\_id",course.getId())

.eq("is\_delete",0)

.orderByDesc("exp");

//获得课程的所有学生id

List<CourseStudent> list = courseStudentService.list(courseId);

if(list.size()==0){

response.setMsg("该课程暂无学生！");

response.setResult(false);

return response;

}

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

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

List<CourseStudentVo> dataList= new ArrayList<>();

for(int i = 0 ; i < list.size(); i++){

CourseStudentVo courseStudentVo = new CourseStudentVo();

QueryWrapper<User> queryUser = new QueryWrapper<>();

queryUser.eq("id",list.get(i).getStudentId());

User user = userService.getOne(queryUser);

if(list.get(i).getExp() == null){

courseStudentVo.setExp(0);

}else{

courseStudentVo.setExp(list.get(i).getExp());

}

courseStudentVo.setName(user.getName());

courseStudentVo.setSno(user.getSno());

courseStudentVo.setSex(user.getSex());

dataList.add(courseStudentVo);

}

response.setResult(true);

response.setMsg("查询成功");

response.setDataList(dataList);

response.setTotal((long)list.size());

return response;

}

@ApiOperation(value = "学生使用班课号加入班课",notes = "get")

@ResponseBody

@RequestMapping(method = RequestMethod.POST)

public String JoinClass(@RequestParam(value="code")String code,

@RequestParam(value="telephone")String telephone) {

//学生使用班课号加入班课

QueryWrapper<Course> queryWrapper = new QueryWrapper();

queryWrapper.eq("code", code)

.eq("isDelete", 0);

int count1 = courseService.count(queryWrapper);

if(count1 == 0)

return ResultUtil.error("班课号不存在！");

Course course = courseService.getOne(queryWrapper);

if(course.getFlag()==1)

return ResultUtil.error("班课已经结束！");

if(course.getIsJoin()==0)

return ResultUtil.error("班课不允许加入！");

if (course != null) {

// if (email != null) {

//不能加入自己创建的课程

QueryWrapper<User> queryUser = new QueryWrapper();

queryUser.eq("telephone", telephone)

.eq("is\_delete", 0);

User user1 = userService.getOne(queryUser);

if(user1.getId()==course.getTeacherId())

{

return ResultUtil.error("不能加入自己创建的班课！");

}

// QueryWrapper<CourseStudent> queryMyCourse = new QueryWrapper();

// queryMyCourse.eq("course\_id", course.getId())

// .eq("teacher\_email", email);

//

// int count1 = courseStudentService.count(queryMyCourse);

// if (count1 > 0) {

// return ResultUtil.error("不能加入自己创建的班课！");

// }

//判断是否已经加入过

QueryWrapper<CourseStudent> queryJoined = new QueryWrapper();

queryJoined.eq("course\_id", course.getId())

.eq("student\_id", user1.getId())

.eq("is\_delete", 0);

int count = courseStudentService.count(queryJoined);

if (count > 0) {

return ResultUtil.error("您已加入本班课，请勿重复加入！");

} else {

//courseStudent更新

CourseStudent courseStudent = new CourseStudent();

courseStudent.setCourseId(course.getId());

courseStudent.setStudentId(user1.getId());

courseStudentService.save(courseStudent);

return ResultUtil.success();

}

}

else {

return ResultUtil.error("班课号不存在！");

}

}

@ApiOperation(value = "创建班课",notes = "get")

@ResponseBody

@RequestMapping(value = "/CreateClass",method = RequestMethod.POST)

public String CreateClass(@RequestParam(value="className")String className,

@RequestParam(value="name")String name,

@RequestParam(value="school",required = false)String school,

@RequestParam(value="isFinish",required = false)Integer isFinish,

@RequestParam(value="require",required = false)String require,

@RequestParam(value="term",required = false)String term,

@RequestParam(value="examination",required = false)String examination,

@RequestParam(value="telephone")String telephone,

@RequestParam(value="process",required = false)String process

) {

//老师 创建班课

Course course = new Course();

course.setClassName(className);

course.setName(name);

if(school!=null)

course.setSchoolCode(school);

if(isFinish!=null)

course.setFlag(isFinish);

//随机生成课程号

// course.setCode("11111111");

int count = 1;

String code = "";

do{

StringBuilder str = new StringBuilder();

for (int i = 0; i < 7; i++) {

if (i == 0 && 8 > 1){

str.append(new Random().nextInt(9) + 1);

}else {

str.append(new Random().nextInt(10));

}

}

code = str.toString();

//查看数据库中是否存在 若存在则重新生成

QueryWrapper codeQuery = new QueryWrapper();

codeQuery.eq("code",code);

count = courseService.count(codeQuery);

}while(count > 0);

course.setCode(code);

if(require!=null)

course.setLearnRequire(require);

if(examination!=null)

course.setExamSchedule(examination);

if(term!=null)

course.setSemester(term);

//通过邮箱找userid

QueryWrapper queryWrapper = new QueryWrapper();

queryWrapper.eq("telephone",telephone);

User user = userService.getOne(queryWrapper);

course.setTeacherId(parseInt(user.getId().toString()));

if(process!=null)

course.setTeachProgress(process);

course.setIsJoin(1);

course.setIsDelete(0);

courseService.save(course);

return code;

}

//编辑

@ApiOperation(value = "编辑班课信息",notes = "get")

@ResponseBody

@RequestMapping(method = RequestMethod.PATCH)

public String update(

//@RequestBody UpdateCourseDto updateCourseDto

@RequestParam(value="code")String code,

@RequestParam(value="className",required = false)String className,

@RequestParam(value="name",required = false)String name,

@RequestParam(value="isJoin",required = false)Integer isJoin,

@RequestParam(value="school",required = false)String school,

@RequestParam(value="isFinish",required = false)Integer isFinish,

@RequestParam(value="require",required = false)String require,

@RequestParam(value="term",required = false)String term,

@RequestParam(value="examination",required = false)String examination,

@RequestParam(value="process",required = false)String process

){

// course.setCode(no);

QueryWrapper<Course> queryWrapper = new QueryWrapper();

queryWrapper.eq("code",code);

Course course = courseService.getOne(queryWrapper);

if(course == null)

return ResultUtil.error("没有这门课程!");

if( name != null){

course.setName(name);

}

if(term != null){

course.setSemester(term);

}

if(isJoin != null){

course.setIsJoin(isJoin);

}

if(school!= null){

course.setSchoolCode(school);

}

if(require != null){

course.setLearnRequire(require);

}

if(process != null){

course.setTeachProgress(process);

}

if(examination != null){

course.setExamSchedule(examination);

}

if(className != null){

course.setClassName(className);

}

if(isFinish != null){

course.setFlag(isFinish);

}

courseService.updateById(course);

return ResultUtil.success();

}

//删除

@ResponseBody

@ApiOperation(value = "退出、删除班课",notes = "get")

@RequestMapping(method = RequestMethod.DELETE)

public String delete(@RequestParam(value="code")String code,

@RequestParam(value="telephone",required = false)String telephone

){

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

queryWrapper.eq("code",code);

Course course1 = courseService.getOne(queryWrapper);

if(telephone != null){//退出班课

QueryWrapper<User> queryUser = new QueryWrapper<>();

queryUser.eq("telephone",telephone)

.eq("is\_delete",0);

User user = userService.getOne(queryUser);

QueryWrapper<CourseStudent> queryWrapper1 = new QueryWrapper<>();

queryWrapper1.eq("student\_id",user.getId())

.eq("course\_id",course1.getId());

CourseStudent courseStudent = new CourseStudent();

courseStudent.setIsDelete(1);

courseStudentService.update(courseStudent,queryWrapper1);

//清空签到历史记录

QueryWrapper<AttendenceResult> queryHistory = new QueryWrapper<>();

queryHistory.eq("course\_id",course1.getId())

.eq("student\_id",user.getId());

AttendenceResult attendenceResult = attendenceResultService.getOne(queryHistory);

if(attendenceResult != null)

attendenceResultService.removeById(attendenceResult.getId());

// attendenceResultService.update(attendenceResult,queryHistory);

return ResultUtil.success();

}

Course course = new Course();

course.setIsDelete(1);

boolean flag = courseService.update(course,queryWrapper);

//coursestudent表中对应的课程设置为删除状态

if(flag){

QueryWrapper<CourseStudent> queryWrapper1 = new QueryWrapper();

queryWrapper1.eq("course\_id",course1.getId());

CourseStudent courseStudent = new CourseStudent();

courseStudent.setIsDelete(1);

courseStudentService.update(courseStudent,queryWrapper1);

}

return ResultUtil.success();

}

}

7、用户相关接口

package com.example.dcloud.controller;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONObject;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;

import com.example.dcloud.annotation.NoToken;

import com.example.dcloud.common.ServerResponse;

import com.example.dcloud.dto.UpdateInfoDto;

import com.example.dcloud.dto.UpdatePasswordDto;

import com.example.dcloud.dto.UserBasicDto;

import com.example.dcloud.dto.UserDto;

import com.example.dcloud.entity.User;

import com.example.dcloud.service.UserService;

import com.example.dcloud.util.ResultUtil;

import com.example.dcloud.vo.UserListVo;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

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

import org.springframework.stereotype.Controller;

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

import java.util.List;

import java.util.Map;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

@Api(tags = "用户接口")

@CrossOrigin

@Controller

@RequestMapping("/user")

public class UserController {

@Autowired

UserService userService;

@ResponseBody

@RequestMapping(method = RequestMethod.GET)

public ServerResponse<UserListVo> userList(//通过一些参数查找符合条件的用户信息 返回的是用户列表

@RequestParam(value="page",required = false)Integer page,

@RequestParam(value="size",required = false)Integer size,

@RequestParam(value="state",required = false)String state,

@RequestParam(value="name",required = false)String name,

@RequestParam(value="roleId",required = false)Integer roleId

){

return userService.userList(state,name,page,size,roleId);//返回的是一个数组，每一行代表一条用户信息

}

@ResponseBody

@RequestMapping(method = RequestMethod.POST)

public String addUser(@RequestBody UserBasicDto userBasicDto){//新增用户

String password = "123456";//密码是默认

// int roleId = Integer.parseInt(map.get("roleId").toString());//角色自己可以设置

JSONObject jsonObject1 = new JSONObject();

if(userService.addUser(userBasicDto.getName(),userBasicDto.getSex(),userBasicDto.getTelephone(),password,userBasicDto.getRoleId())==1){//增加用户

jsonObject1.put("respCode","1");

return jsonObject1.toString();

}else{

jsonObject1.put("respCode","账号已存在");

return jsonObject1.toString();

}

}

@ResponseBody

@RequestMapping(method = RequestMethod.PUT)

//通过 管理员 更新用户信息

public String updateUserByAdmin(@RequestBody UserBasicDto userBasicDto){

userService.updateUserByAdmin(userBasicDto.getName(),userBasicDto.getSex(),userBasicDto.getRoleId(),userBasicDto.getTelephone());

JSONObject jsonObject1 = new JSONObject();

jsonObject1.put("respCode","1");

return jsonObject1.toString();

}

@ResponseBody

@RequestMapping(method = RequestMethod.DELETE)

//删除用户信息 可以删除多条记录

public String deleteUser(@RequestParam(value = "del\_list") List list){

for(int i=0;i<list.size();i++){

String telephone = list.get(i).toString();

userService.deleteUser(telephone);

}

JSONObject jsonObject1 = new JSONObject();

jsonObject1.put("respCode","1");

return jsonObject1.toString();

}

@ResponseBody

@RequestMapping(method = RequestMethod.PATCH)

//改变用户状态

public String changeUserState( @RequestParam(value="telephone")String telephone){

userService.changeUserStateService(telephone);

JSONObject jsonObject1 = new JSONObject();

jsonObject1.put("respCode","1");

return jsonObject1.toString();

}

@ResponseBody

@RequestMapping(value = "/updatePassword",method = RequestMethod.POST)

//更新密码

public String updatePassword(@RequestBody UpdatePasswordDto updatePasswordDto) {

//Map map = JSON.toJavaObject(jsonObject, Map.class);

String telephone = updatePasswordDto.getTelephone();

String newPassword = updatePasswordDto.getNewpassword1();

String repeatNewPassword = updatePasswordDto.getNewpasswor2();

String oldPassword = updatePasswordDto.getOldpassword();

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

queryWrapper.eq("telephone",telephone);

User user = userService.getOne(queryWrapper);

String password = user.getPassword();

int id = user.getId();

if(oldPassword.equals("")){

if(newPassword.equals(repeatNewPassword)){

User user1 = new User();

user1.setId(id);

user1.setPassword(newPassword);

userService.updateById(user1);

return ResultUtil.success();

}

else{

return ResultUtil.error("两次输入的密码不一致");

}

}

else {

if(oldPassword.equals(password)){

if(newPassword.equals(repeatNewPassword)){

User user1 = new User();

user1.setId(id);

user1.setPassword(newPassword);

userService.updateById(user1);

return ResultUtil.success();

}

else{

return ResultUtil.error("两次输入的密码不一致");

}

}

else{

return ResultUtil.error("原密码错误");

}

}

}

@ResponseBody

@NoToken

@RequestMapping(value = "/forgetPassword",method = RequestMethod.POST)

//忘记密码

public String forgetPassword(@RequestParam(value="telephone")String telephone,

@RequestParam(value="newpassword")String newPassword) {

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

queryWrapper.eq("telephone",telephone);

User user = userService.getOne(queryWrapper);

if(user!=null){

if(user.getIsDelete()==0){

int id = user.getId();

User user1 = new User();

user1.setId(id);

user1.setPassword(newPassword);

//自带的函数 根据id来更新数据库

userService.updateById(user1);

return ResultUtil.success();

}

else{

return ResultUtil.error("账号已被删除");

}

}

else{

return ResultUtil.error("账号不存在");

}

}

@ResponseBody

@RequestMapping(value = "/resetPassword",method = RequestMethod.PUT)

public String resetPassword(@RequestParam(value="telephone",required = false)String telephone) {

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

//获取到电话直接重置

queryWrapper.eq("telephone",telephone);

User user = new User();

user.setPassword("123456");

try{

userService.update(user,queryWrapper);

return ResultUtil.success();

} catch (Exception e){

return ResultUtil.error("重置失败");

}

}

@ResponseBody

@RequestMapping(value = "/info",method = RequestMethod.GET)

//显示个人信息 根据邮箱来查找用户 并显示到前端个人信息页面

public String getInfo(@RequestParam(value="account")String account) {

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

queryWrapper.eq("telephone",account);

try{

User user = userService.getOne(queryWrapper);

JSONObject jsonObject = new JSONObject();

jsonObject.put("nickname",user.getNickname());

jsonObject.put("sno",user.getSno());

jsonObject.put("birth",user.getBirth());

jsonObject.put("sex",user.getSex());

jsonObject.put("school",user.getSchoolCode());

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

jsonObject.put("name",user.getName());

jsonObject.put("telephone",user.getTelephone());

jsonObject.put("exp",user.getExp());

jsonObject.put("image",user.getImage());

return jsonObject.toString();

} catch (Exception e){

return ResultUtil.error("获取失败");

}

}

@ResponseBody

@RequestMapping(value = "/info",method = RequestMethod.PUT)

//修改个人信息

public String updateInfo(@RequestBody UpdateInfoDto updateInfoDto) {

// Map map = JSON.toJavaObject(jsonObject,Map.class);

//新建一个user对象， 对前端传来的数据进行判断，符合的放入user中 最后更新

User user = new User();

if(updateInfoDto.getSex()!=null){

if(!updateInfoDto.getSex().equals("")){

user.setSex(updateInfoDto.getSex());

}

}

//这里感觉有点奇怪 没有调用setEmail方法

String telephone = updateInfoDto.getTelephone();

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

queryWrapper.eq("telephone",telephone);

User user1 = userService.getOne(queryWrapper);

if(updateInfoDto.getSchoolCode()!=null)

if(!updateInfoDto.getSchoolCode().equals(""))

user.setSchoolCode(updateInfoDto.getSchoolCode());

if(updateInfoDto.getSno()!=null)

if(!updateInfoDto.getSno().equals(""))

user.setSno(updateInfoDto.getSno());

if(updateInfoDto.getName()!=null)

if(!updateInfoDto.getName().equals(""))

user.setName(updateInfoDto.getName());

if(updateInfoDto.getNickname()!=null){

if(!updateInfoDto.getNickname().equals("")){

if(!user1.getNickname().equals(updateInfoDto.getNickname())){

//判断一下这个昵称有没有被其他人使用

QueryWrapper<User> queryWrapper2 = new QueryWrapper<>();

queryWrapper2.eq("nickname",updateInfoDto.getNickname());

if(userService.count(queryWrapper2)>0)

return ResultUtil.error("该昵称已被使用");

user.setNickname(updateInfoDto.getNickname());

}

}

}

if(updateInfoDto.getTelephone()!=null){

//和昵称一样

if(!updateInfoDto.getTelephone().equals("")){

if(!user1.getTelephone().equals(updateInfoDto.getTelephone())){

QueryWrapper<User> queryWrapper1 = new QueryWrapper<>();

queryWrapper1.eq("telephone",updateInfoDto.getTelephone());

if(userService.count(queryWrapper1)>0)

return ResultUtil.error("该手机号已被使用");

String phoneCheck = "^[1](([3|5|8][\\d])|([4][4,5,6,7,8,9])|([6][2,5,6,7])|([7][^9])|([9][1,8,9]))[\\d]{8}$";

Pattern phoneRegex = Pattern.compile(phoneCheck);

Matcher phoneMatcher = phoneRegex.matcher(updateInfoDto.getTelephone());

boolean phoneIsMatched = phoneMatcher.matches();

if(phoneIsMatched==true)

user.setTelephone(updateInfoDto.getTelephone());

else

return ResultUtil.error("请输入真实手机号");

}

}

}

if(updateInfoDto.getBirth()!=null)

if(!updateInfoDto.getBirth().equals(""))

user.setBirth(updateInfoDto.getBirth());

if(updateInfoDto.getImage()!=null)

if(!updateInfoDto.getImage().equals(""))

user.setImage(updateInfoDto.getImage());

try{

//自带的更新函数

userService.update(user,queryWrapper);

return ResultUtil.success();

} catch (Exception e){

return ResultUtil.error("更新失败");

}

}

}

8、系统参数管理功能

package com.example.dcloud.controller;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONArray;

import com.alibaba.fastjson.JSONObject;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;

import com.example.dcloud.entity.Dictionary;

import com.example.dcloud.entity.DictionaryDetail;

import com.example.dcloud.entity.SystemManage;

import com.example.dcloud.service.SystemManageService;

import com.example.dcloud.util.ResultUtil;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

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

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

import org.springframework.stereotype.Controller;

import sun.awt.LightweightFrame;

import java.util.\*;

import static java.lang.Integer.parseInt;

import static java.lang.Long.parseLong;

@Api(tags = "系统参数管理接口")

@Controller

@RequestMapping("/systems")

public class SystemManageController {

@Autowired

private SystemManageService systemManageService;

//获取系统参数列表

@ApiOperation(value = "获取系统列表",notes = "get")

@ResponseBody

@RequestMapping(method = RequestMethod.GET)

public String getAll(@RequestParam(value="page")Integer page){

return systemManageService.pageList(page);

}

@ApiOperation(value = "新增系统参数",notes = "get")

@ResponseBody

@RequestMapping(method = RequestMethod.POST)

public String addOne(@RequestParam(value="para\_name")String para\_name,

@RequestParam(value="key\_name")String key\_name,

@RequestParam(value="key\_value")Integer key\_value) {

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

queryWrapper

.eq("para\_name", para\_name)

.eq("is\_delete",0);

int count = systemManageService.count(queryWrapper);

if(count > 0){

return ResultUtil.error("参数名称重复！");

}

QueryWrapper<SystemManage> queryName = new QueryWrapper<>();

queryName

.eq("key\_name", key\_name)

.eq("is\_delete",0);

int countName = systemManageService.count(queryName);

if(countName > 0){

return ResultUtil.error("关键字重复！");

}

if(key\_value == null){

return ResultUtil.error("值为空！");

}

SystemManage systemManage = new SystemManage();

systemManage.setParaName(para\_name);

systemManage.setKeyName(key\_name);

systemManage.setKeyValue(key\_value);

systemManage.setIsDelete(0);

systemManageService.saveOrUpdate(systemManage);

return ResultUtil.success();

}

@ApiOperation(value = "判断是否重复",notes = "get")

@ResponseBody

@RequestMapping(value = "/isRepeat",method = RequestMethod.POST)

public int isRepeat(

@RequestParam(value="key\_name")String key\_name

) {

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

queryWrapper

.eq("key\_name", key\_name)

.eq("is\_delete",0);

int count = systemManageService.count(queryWrapper);

if(count > 0){

return 0;

}

else

return 1;

}

@ApiOperation(value = "编辑系统参数",notes = "get")

@ResponseBody

@RequestMapping(method = RequestMethod.PATCH)

public String update(

@RequestParam(value="para\_name",required = false)String para\_name,

@RequestParam(value="key\_name",required = false)String key\_name,

@RequestParam(value="key\_value",required = false)Integer key\_value) {

System.out.println("编辑");

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

queryWrapper.eq("para\_name", para\_name)

.eq("is\_delete", 0);

SystemManage systemManage = new SystemManage();

// if (para\_name!=null)

systemManage.setParaName(para\_name);

// if(key\_name!=null)

systemManage.setKeyName(key\_name);

if(key\_value!=null)

systemManage.setKeyValue(key\_value);

// systemManage.setIsDelete(0);

systemManageService.update(systemManage,queryWrapper);

return ResultUtil.success();

}

@ApiOperation(value = "删除单条系统参数",notes = "get")

@ResponseBody

@RequestMapping(method = RequestMethod.DELETE)

public String delete(@RequestParam(value="key\_name")String key\_name) {

SystemManage systemManage = new SystemManage();

QueryWrapper<SystemManage> queryName = new QueryWrapper<>();

queryName

.eq("key\_name", key\_name)

.eq("is\_delete", 0);

systemManage.setIsDelete(1);

int count = systemManageService.count(queryName);

if(count == 0)

return ResultUtil.error("删除失败");

systemManageService.update(systemManage,queryName);

return ResultUtil.success();

}

}

9、数据字典管理接口

package com.example.dcloud.controller;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONArray;

import com.alibaba.fastjson.JSONObject;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;

import com.example.dcloud.entity.Dictionary;

import com.example.dcloud.entity.DictionaryDetail;

import com.example.dcloud.service.DictionaryDetailService;

import com.example.dcloud.service.DictionaryService;

import com.example.dcloud.util.ResultUtil;

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.\*;

import static java.lang.Integer.parseInt;

import static java.lang.Long.parseLong;

@Api(tags = "数据字典管理接口")

@RestController

@RequestMapping("/dictionaries")

public class DictionaryController {

@Autowired

private DictionaryService dictionaryService;

@Autowired

private DictionaryDetailService dictionaryDetailService;

@ResponseBody

@RequestMapping(method = RequestMethod.GET)

@ApiOperation(value = "获取数据字典列表",notes = "get")

public String getList(

@RequestParam(value="page",required = false)Integer page,

@RequestParam(value="code",required = false)String code,

@RequestParam(value="name",required = false)String name

) {

//如果有code，表示查询条件

if(code!=null){

//name查询条件

if(name!=null){

System.out.println("111");

//根据code和name进行查询

return dictionaryService.pageListforQuery(code,name,page);

}else{

System.out.println("222");

//根据code查询

return JSON.toJSONString(dictionaryDetailService.getDetail(code));

}

}else{

System.out.println("333");

//获取数据字典列表

return dictionaryService.pageList(page);

}

}

//获取数据字典的列表

//public String getList(

// public ServerResponse<Dictionary> getList(

// @RequestParam(value="page",required = false)Integer page,

// @RequestParam(value="code",required = false)String code,

// @RequestParam(value="name",required = false)String name

// ) {

// }

@ResponseBody

@RequestMapping(method = RequestMethod.PATCH)

@ApiOperation(value = "修改数据字典",notes = "get")

public String update(@RequestBody JSONObject jsonObject) {

Map map = JSON.toJavaObject(jsonObject, Map.class);

List<DictionaryDetail> detailList = new ArrayList<>();

//检查是否唯一

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

queryWrapper.eq("code", map.get("new\_code").toString())

.eq("is\_delete",0);

int count = dictionaryService.count(queryWrapper);

if(!map.get("new\_code").toString().equals(map.get("old\_code").toString())){

if(count > 0){

return ResultUtil.error("英文标识符已存在!");

}

}

//判断是否唯一

QueryWrapper<Dictionary> queryWrapper1 = new QueryWrapper<>();

queryWrapper1.eq("name", map.get("new\_name").toString())

.eq("is\_delete",0);

int count1 = dictionaryService.count(queryWrapper1);

if(!map.get("new\_name").toString().equals(map.get("old\_name").toString())){

if(count1 > 0){

return ResultUtil.error("中文标识符已存在!");

}

}

JSONArray detail=(JSONArray) JSONArray.toJSON(map.get("detail"));

int flag = 0;

Set valueSet = new HashSet<>();

Set nameSet = new HashSet<>();

int beforeSize = valueSet.size();

int beforeSize1 = nameSet.size();

//进一步判断

for(int i = 0; i < detail.size(); i++){

JSONObject tempObject=(JSONObject) JSONObject.toJSON(detail.get(i));

Map temp = JSON.toJavaObject(tempObject, Map.class);

valueSet.add(temp.get("value").toString());

int afterSize = valueSet.size();

if(beforeSize == afterSize){

return ResultUtil.error("数据项值重复！");

}else {

beforeSize = afterSize;

}

nameSet.add(temp.get("name").toString());

int afterSize1 = nameSet.size();

if(beforeSize1 == afterSize1){

return ResultUtil.error("数据项文本重复！");

}else {

beforeSize1 = afterSize1;

}

if(parseInt(temp.get("is\_default").toString()) == 1){

flag ++;

if(flag > 1){

return ResultUtil.error("多个数据项设置为默认值！");

}

}

DictionaryDetail dictionaryDetail = new DictionaryDetail();

if(temp.containsKey("id")){

dictionaryDetail.setId(parseLong(temp.get("id").toString()));

}

dictionaryDetail.setCode(temp.get("code").toString());

dictionaryDetail.setName(temp.get("name").toString());

dictionaryDetail.setValue(temp.get("value").toString());

//字典顺序

dictionaryDetail.setDictOrder(parseInt(temp.get("dict\_order").toString()));

//是否默认

dictionaryDetail.setIsDefault(parseInt(temp.get("is\_default").toString()));

dictionaryDetail.setTypeCode(map.get("new\_code").toString());

dictionaryDetail.setIsDelete(0);

detailList.add(dictionaryDetail);

}

Dictionary dictionary = new Dictionary();

dictionary.setId(parseLong(map.get("id").toString()));

if(map.get("description").toString() == ""){

dictionary.setDescription(" ");

}else{

dictionary.setDescription(map.get("description").toString());

}

dictionary.setName(map.get("new\_name").toString());

dictionary.setCode(map.get("new\_code").toString());

dictionary.setIsDelete(0);

//更新数据字典

dictionaryService.updateById(dictionary);

for(int i = 0; i < detailList.size(); i++){

//更新数据值和文本

dictionaryDetailService.saveOrUpdate(detailList.get(i));

}

return ResultUtil.success();

}

@ResponseBody

@RequestMapping(method = RequestMethod.DELETE)

@ApiOperation(value = "删除数据字典。可批量删除。",notes = "get")

public String delDict(@RequestParam(value = "del\_list") List list) {

//删除一条记录

if(list.size() == 1){

DictionaryDetail dictionaryDetail = new DictionaryDetail();

dictionaryDetail.setId(parseLong(list.get(0).toString()));

//软删除

dictionaryDetail.setIsDelete(1);

dictionaryDetailService.updateById(dictionaryDetail);

return ResultUtil.success();

}

list.remove(0);

//删除多天记录

for(int i = 0; i < list.size(); i++){

String code = list.get(i).toString();

QueryWrapper dictWrapper = new QueryWrapper();

dictWrapper.eq("code",code);

Dictionary dictionary = new Dictionary();

//软删除

dictionary.setIsDelete(1);

dictionaryService.update(dictionary,dictWrapper);

//删除数据项

QueryWrapper detailWrapper = new QueryWrapper();

detailWrapper.eq("type\_code",code);

DictionaryDetail dictionaryDetail = new DictionaryDetail();

//软删除

dictionaryDetail.setIsDelete(1);

dictionaryDetailService.update(dictionaryDetail,detailWrapper);

}

return ResultUtil.success();

}

@ResponseBody

@RequestMapping(method = RequestMethod.POST)

@ApiOperation(value = "新增数据字典",notes = "get")

public String addOne(@RequestBody JSONObject jsonObject) {

Map map = JSON.toJavaObject(jsonObject, Map.class);

List<DictionaryDetail> detailList = new ArrayList<>();

//检查英文标识是否唯一

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

queryWrapper

.eq("code", map.get("code").toString())

.eq("is\_delete",0);

int count = dictionaryService.count(queryWrapper);

//不唯一直接返回

if(count > 0){

return ResultUtil.error("英文标识符已存在！");

}

//进一步检查中文标识是否唯一

QueryWrapper<Dictionary> queryCode = new QueryWrapper<>();

queryCode

.eq("name", map.get("name").toString())

.eq("is\_delete",0);

int count1 = dictionaryService.count(queryWrapper);

if(count1 > 0){

return ResultUtil.error("中文标识符已存在！");

}

//数据项（文本名称，值，数据项标识符，是否默认值，排序）

JSONArray detail=(JSONArray) JSONArray.toJSON(map.get("detail"));

int flag1 = 0;

Set set = new HashSet<>();

int front = set.size();

//新增数据项

for(int i = 0; i < detail.size(); i++){

JSONObject tempObject=(JSONObject) JSONObject.toJSON(detail.get(i));

Map temp = JSON.toJavaObject(tempObject, Map.class);

set.add(temp.get("value").toString());

int back = set.size();

if(front == back){

return ResultUtil.error("数据项值不允许重复！");

}else {

front = back;

}

if(parseInt(temp.get("is\_default").toString()) == 1){

flag1 ++;

if(flag1 > 1){

return ResultUtil.error("多个数据项设置为默认值！");

}

}

//保存数据字典明细

DictionaryDetail dictionaryDetail = new DictionaryDetail();

dictionaryDetail.setCode(temp.get("code").toString());

dictionaryDetail.setName(temp.get("name").toString());

dictionaryDetail.setValue(temp.get("value").toString());

dictionaryDetail.setDictOrder(parseInt(temp.get("dict\_order").toString()));

dictionaryDetail.setIsDefault(parseInt(temp.get("is\_default").toString()));

dictionaryDetail.setTypeCode(map.get("code").toString());

dictionaryDetail.setIsDelete(0);

detailList.add(dictionaryDetail);

}

//保存数据字典

Dictionary dictionary = new Dictionary();

dictionary.setCode(map.get("code").toString());

dictionary.setName(map.get("name").toString());

if(map.get("description").toString()==("")){

dictionary.setDescription(" ");

}else{

dictionary.setDescription(map.get("description").toString());

}

dictionary.setIsDelete(0);

boolean flag = dictionaryService.saveOrUpdate(dictionary);

//如果数据字典保存成功，则去保存数据字典明细

if(flag){

dictionaryDetailService.saveBatch(detailList);

}

return ResultUtil.success();

}

}

10、角色管理接口

package com.example.dcloud.controller;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONObject;

import com.example.dcloud.entity.Role;

import com.example.dcloud.service.RoleService;

import com.example.dcloud.util.ResultUtil;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.stereotype.Controller;

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

import java.util.Map;

@Api(tags = "角色管理接口")

@CrossOrigin

@Controller

@RequestMapping("/role")

public class RoleController {

@Autowired

RoleService roleService;

private static final Logger LOG = LoggerFactory.getLogger(RoleController.class);

@ResponseBody

@RequestMapping(method = RequestMethod.POST)

@ApiOperation(value = "创建角色",notes = "get")

public String createRole(@RequestBody JSONObject jsonObject){

Map map = JSON.toJavaObject(jsonObject, Map.class);

Role role = new Role();

//设置默认值

role.setIsDelete(0);

role.setIsStudent(0);

role.setPowerId("0");

//状态设置可用,1为可用，0表示不可用

role.setState(1);

role.setDescription(map.get("description").toString());

role.setName(map.get("name").toString());

try {

roleService.save(role);

return ResultUtil.success();

}catch (Exception e){

return ResultUtil.error("角色创建失败！");

}

}

//获取用户列表

@ResponseBody

@RequestMapping(method = RequestMethod.GET)

@ApiOperation(value = "获取角色列表",notes = "get")

public String roleList(@RequestParam(value="page",required = false)Integer page,

@RequestParam(value="state",required = false)String state,

@RequestParam(value="name",required = false)String name) {

return roleService.roleList(state,name,page);

}

@ResponseBody

@RequestMapping(method = RequestMethod.PUT)

@ApiOperation(value = "更新角色信息",notes = "get")

public String updateRole(@RequestBody JSONObject jsonObject){

Map map = JSON.toJavaObject(jsonObject, Map.class);

Role role = new Role();

//获取id，根据id更新

role.setId(Integer.parseInt(map.get("id").toString()));

role.setState(Integer.parseInt(map.get("state").toString()));

role.setDescription(map.get("description").toString());

role.setName(map.get("name").toString());

try{

roleService.updateById(role);

return ResultUtil.success();

}catch (Exception e){

return ResultUtil.error("更新失败");

}

}

@ResponseBody

@RequestMapping(method = RequestMethod.DELETE)

@ApiOperation(value = "删除角色",notes = "get")

public String deleteRole(@RequestParam(value="id",required = false)Integer id){

//根据id删除

Role role = new Role();

role.setId(id);

role.setIsDelete(1);

try{

roleService.updateById(role);

return ResultUtil.success();

}catch (Exception e){

return ResultUtil.error("删除失败");

}

}

@ResponseBody

@RequestMapping(method = RequestMethod.PATCH)

@ApiOperation(value = "修改角色的状态",notes = "get")

public String updateRoleState(

@RequestParam(value="id",required = false)Integer id,

@RequestParam(value="state",required = false)Integer state){

Role role = new Role();

role.setId(id);

//取反

state = state == 1?0:1;

role.setState(state);

try{

//根据id更新

roleService.updateById(role);

return ResultUtil.success();

}catch (Exception e){

return ResultUtil.error("修改失败！");

}

}

}

11、角色权限接口

package com.example.dcloud.controller;

import com.example.dcloud.entity.Role;

import com.example.dcloud.service.RolePowerService;

import com.example.dcloud.service.RoleService;

import com.example.dcloud.util.ResultUtil;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

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

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RequestMethod;

import org.springframework.web.bind.annotation.RequestParam;

import org.springframework.web.bind.annotation.ResponseBody;

@Api(tags = "角色权限接口")

@Controller

@RequestMapping("/rolePower")

public class RolePowerController {

@Autowired

RolePowerService rolePowerService;

@Autowired

RoleService roleService;

@ResponseBody

@RequestMapping(method = RequestMethod.GET)

@ApiOperation(value = "返回权限id",notes = "get")

public String[] rolePower(@RequestParam(value="role\_id",required = false)Integer role\_id) {

Role role = roleService.getById(role\_id);

String power\_id = role.getPowerId();

String[] power = power\_id.split(",");

return power;

}

@ResponseBody

@RequestMapping(method = RequestMethod.PUT)

@ApiOperation(value = "更新角色权限",notes = "get")

public String updateRolePower(@RequestParam(value="role\_id",required = false)Integer role\_id,

@RequestParam(value="power\_id",required = false)String power\_id) {

Role role = new Role();

role.setId(role\_id);

role.setPowerId(power\_id);

try{

roleService.updateById(role);

return ResultUtil.success();

}catch (Exception e){

return ResultUtil.error("更新失败！");

}

}

}

12、机构管理接口

package com.example.dcloud.controller;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONObject;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;

import com.example.dcloud.common.ServerResponse;

import com.example.dcloud.entity.School;

import com.example.dcloud.service.SchoolService;

import com.example.dcloud.util.ResultUtil;

import com.example.dcloud.vo.SchoolVo;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

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

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

import org.springframework.stereotype.Controller;

import javax.annotation.Resource;

import java.util.ArrayList;

import java.util.List;

import java.util.Map;

import static java.awt.SystemColor.menu;

import static java.lang.Integer.parseInt;

@Api(tags = "学校管理接口")

@CrossOrigin

@Controller

@RequestMapping("/schools")

public class SchoolController {

@Autowired

private SchoolService schoolService;

@ResponseBody

@RequestMapping(value = "/getCode",method = RequestMethod.GET)

@ApiOperation(value = "获取学校名称",notes = "get")

public String getCode(@RequestParam(value="code",required = false)String code){

if(code == null || code.equals("0")){

return "未设置";

}

QueryWrapper<School> queryWrapper = new QueryWrapper();

queryWrapper.eq("code",code)

.eq("is\_delete",0);

School school = schoolService.getOne(queryWrapper);

return school.getName();

}

@ResponseBody

@RequestMapping(value = "/getCode1",method = RequestMethod.GET)

@ApiOperation(value = "获取学校、学院名称",notes = "get")

public ServerResponse<SchoolVo> getCode1(@RequestParam(value="code")String code){

ServerResponse<SchoolVo> response = new ServerResponse<>();

if(code == null || code.equals("0")){

response.setMsg("为设置学校、学院!");

response.setResult(false);

return response;

}

QueryWrapper<School> queryWrapper = new QueryWrapper();

queryWrapper.eq("code",code)

.eq("is\_delete",0);

School school = schoolService.getOne(queryWrapper);

SchoolVo schoolVo = new SchoolVo();

Integer parentId = parseInt(school.getParentId());

if(parentId!=0){

schoolVo.setDepartment(school.getName());

QueryWrapper<School> queryWrapper1 = new QueryWrapper();

queryWrapper1.eq("id",parentId)

.eq("is\_delete",0);

int count = schoolService.count(queryWrapper1);

if(count!=0){

School school1 = schoolService.getOne(queryWrapper1);

schoolVo.setUniversity(school1.getName());

}

}

else {

schoolVo.setUniversity(school.getName());

}

List<SchoolVo> dataList = new ArrayList<>();

dataList.add(schoolVo);

response.setDataList(dataList);

response.setMsg("查询成功！");

response.setResult(true);

return response;

}

@ResponseBody

@ApiOperation(value = "获取学校列表",notes = "get")

@RequestMapping(value = "/getSchools",method = RequestMethod.GET)

public ServerResponse<School> getSchools(){

return schoolService.getSchools1();

}

@ResponseBody

@ApiOperation(value = "获取学校的代码获取该学校的学院列表",notes = "get")

@RequestMapping(value = "/getAcademies",method = RequestMethod.GET)

public ServerResponse<School> Academies(@RequestParam(value="schoolCode")String schoolCode){

return schoolService.getAcademiesByCode1(schoolCode);

}

// 新增

@ResponseBody

@RequestMapping(method = RequestMethod.POST)

@ApiOperation(value = "新增学校或学院",notes = "get")

public String add(@RequestBody JSONObject jsonObject) {

Map map = JSON.toJavaObject(jsonObject, Map.class);

//判断学校编码是否重复

QueryWrapper<School> queryWrapper = new QueryWrapper();

queryWrapper.eq("code",map.get("code").toString())

.eq("is\_delete",0);

int count = schoolService.count(queryWrapper);

if(count > 0){

return ResultUtil.error("编码不能重复！");

}

//判断学校名称是否重复

QueryWrapper<School> queryName = new QueryWrapper();

queryName.eq("name",map.get("name").toString())

.eq("is\_delete",0);

int count1 = schoolService.count(queryName);

if(count1 > 0){

return ResultUtil.error("学校或学院名称不能重复！");

}

School school = new School();

school.setCode(map.get("code").toString());

school.setName(map.get("name").toString());

//保存父级机构

school.setParentId(map.get("parent\_id").toString());

school.setIsDelete(0);

schoolService.save(school);

return ResultUtil.success();

}

//编辑

@ResponseBody

@RequestMapping(method = RequestMethod.PATCH)

@ApiOperation(value = "编辑学校或学院",notes = "get")

public String edit(@RequestBody JSONObject jsonObject) {

Map map = JSON.toJavaObject(jsonObject, Map.class);

School school = new School();

//判断是否重复

QueryWrapper<School> queryWrapper = new QueryWrapper();

queryWrapper.eq("code",map.get("name").toString())

.eq("is\_delete",0)

.ne("id",parseInt(map.get("id").toString()));

int count = schoolService.count(queryWrapper);

if(count > 0){

return ResultUtil.error("该名称已存在！");

}

school.setId(parseInt(map.get("id").toString()));

school.setName(map.get("name").toString());

//根据id更新值

schoolService.updateById(school);

return ResultUtil.success();

}

//删除 传ids

@ResponseBody

@RequestMapping(method = RequestMethod.DELETE)

@ApiOperation(value = "删除学校或学院",notes = "get")

public String delByIds(@RequestParam(value = "ids") List list) {

boolean flag = false;

for(int i = 0; i < list.size(); i++){

QueryWrapper<School> queryWrapper = new QueryWrapper();

//判断是否有父级机构

queryWrapper.eq("parent\_id",list.get(i))

.eq("is\_delete",0);

int count = schoolService.count(queryWrapper);

if(count > 0){

flag = true;

}else{

School school = new School();

school.setId(parseInt(list.get(i).toString()));

//软删除

school.setIsDelete(1);

schoolService.updateById(school);

}

}

if(flag){

return ResultUtil.error("选中的节点中存在父节点，请先删除他们的子节点！");

}else{

return ResultUtil.success();

}

}

@ResponseBody

@RequestMapping(method = RequestMethod.GET)

@ApiOperation(value = "获取学校、学院列表",notes = "get")

public String getlist(

@RequestParam(value="page",required = false)Integer page,

@RequestParam(value="name",required = false)String name,

@RequestParam(value="id",required = false)Integer id,

@RequestParam(value="info",required = false)Integer info,

@RequestParam(value="school",required = false)Integer school,

@RequestParam(value="schoolCode",required = false)String schoolCode,

@RequestParam(value="academy",required = false)Integer parentId

){

//获取学院列表

if(id != null){

info = 0;

return schoolService.getChildList(page,id,info);

//根据学校名称查询

}else if(name != null){

return schoolService.queryList(page,name);

//分页获取列表

}else if(page != null){

return schoolService.queryListforAll(page);

}else if(info != null) {

return schoolService.getAll(info);

//获取学校列表

}else if(school != null) {

return schoolService.getSchools();

//获取学院列表

}else if(schoolCode !=null){

return schoolService.getAcademiesByCode(schoolCode);

}else if(parentId != null){

return schoolService.getAcademies(parentId);

}else{

info = 0;

return schoolService.getAll(info);

}

}

}

13、菜单管理接口

package com.example.dcloud.controller;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONObject;

import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;

import com.example.dcloud.entity.Menu;

import com.example.dcloud.service.MenuService;

import com.example.dcloud.util.ResultUtil;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

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

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

import org.springframework.stereotype.Controller;

import java.util.\*;

import static java.lang.Integer.parseInt;

import static java.lang.Long.parseLong;

@Api(tags = "菜单管理接口")

@Controller

@RequestMapping("/menus")

public class MenuController {

@Autowired

private MenuService menuService;

@ResponseBody

@RequestMapping(method = RequestMethod.PATCH)

@ApiOperation(value = "编辑菜单",notes = "get")

public String updateOne(@RequestBody JSONObject jsonObject){

Map map = JSON.toJavaObject(jsonObject, Map.class);

Menu menu = new Menu();

//如果没有父级菜单，parentid设为0

if(map.get("parent\_name").toString().equals("无")){

int a = 0;

long b = (int)a;

menu.setParentId(b);

}else{

//根据父级菜单名字查找parentid

QueryWrapper<Menu> queryWrapper1 = new QueryWrapper();

queryWrapper1.eq("name",map.get("parent\_name").toString())

.eq("is\_deleted",0);

Menu menu1 = menuService.getOne(queryWrapper1);

menu.setParentId(menu1.getId());

}

//更新

menu.setId(parseLong(map.get("id").toString()));

menu.setName(map.get("name").toString());

menu.setIcon(map.get("icon").toString());

menu.setIsPage(parseInt(map.get("is\_page").toString()));

menu.setUrl(map.get("url").toString());

menu.setIsVisible(parseInt(map.get("is\_visible").toString()));

menu.setIsMenu(parseInt(map.get("is\_menu").toString()));

menu.setMenuOrder(parseInt(map.get("menu\_order").toString()));

//为按钮

menu.setType(1);

menu.setIsDeleted(0);

//根据id更新菜单

menuService.updateById(menu);

return ResultUtil.success();

}

@ResponseBody

@RequestMapping(method = RequestMethod.GET)

@ApiOperation(value = "获取菜单列表",notes = "get")

public String getList(@RequestParam(value="id",required = false)Integer id,

@RequestParam(value="name",required = false)String name,

@RequestParam(value="page",required = false)Integer page,

@RequestParam(value="parent",required = false)Integer parent,

@RequestParam(value="is\_visible",required = false)Integer is\_visible){

if(id!=null){

//返回菜单详情，根据id查

return JSON.toJSONString(menuService.getById(id));

}else if(name!=null){

//查找菜单（搜索）

return menuService.queryList(name,page,is\_visible);

}else if(page!=null){

//获取菜单列表

return menuService.pageList(page);

}else if(parent != null) {

//获取其子菜单

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

//查找没被删除的

columnMap.put("is\_deleted",0);

Collection<Map<String,Object>> collection = menuService.listMaps();

List<String> parent1 = new ArrayList<>();

Iterator<Map<String,Object>> it = collection.iterator();

//遍历

while (it.hasNext()) {

Map<String,Object> s = it.next();

if(s.get("isDeleted").equals(0)){

parent1.add(s.get("name").toString());

}

}

return JSON.toJSONString(parent1);

}else{

//获取所有菜单

return menuService.getAll();

}

}

@ResponseBody

@RequestMapping(method = RequestMethod.POST)

@ApiOperation(value = "新增一个菜单",notes = "get")

public String saveOne(@RequestBody JSONObject jsonObject){

Map map = JSON.toJavaObject(jsonObject, Map.class);

Menu menu = new Menu();

if(map.containsKey("id")){

//如果有父级菜单，则指向父级菜单

menu.setParentId(parseLong(map.get("id").toString()));

}

//查询是否有重复的菜单名称

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

queryWrapper.eq("name",map.get("name").toString())

.eq("is\_deleted",0);

int count = menuService.count(queryWrapper);

if(count > 0){

return ResultUtil.error("菜单重复！");

}

//查找父级菜单，如果有父级菜单设置parentid，没有则设置0

QueryWrapper<Menu> queryWrapper1 = new QueryWrapper();

queryWrapper1.eq("name",map.get("parent\_name").toString())

.eq("is\_deleted",0);

Menu menu1 = menuService.getOne(queryWrapper1);

if(menu1 == null){

menu.setParentId(parseLong("0"));

}else{

menu.setParentId(menu1.getId());

}

//相关的信息

menu.setName(map.get("name").toString());

menu.setIcon(map.get("icon").toString());

menu.setIsPage(parseInt(map.get("is\_page").toString()));

menu.setUrl(map.get("url").toString());

menu.setIsVisible(parseInt(map.get("is\_visible").toString()));

menu.setIsMenu(parseInt(map.get("is\_menu").toString()));

menu.setMenuOrder(parseInt(map.get("menu\_order").toString()));

menu.setMenuLevel(parseInt(map.get("menu\_level").toString()));

//表示按钮

menu.setType(1);

//启用

menu.setIsDeleted(0);

menuService.save(menu);

return ResultUtil.success();

}

@ResponseBody

@RequestMapping(method = RequestMethod.DELETE)

@ApiOperation(value = "删除菜单",notes = "get")

public String delMenu(@RequestParam(value = "ids") List list) {

for(int i = 0; i < list.size(); i++){

Menu menu = new Menu();

menu.setId(parseLong(list.get(i).toString()));

//软删除

menu.setIsDeleted(1);

menuService.updateById(menu);

}

return ResultUtil.success();

}

}

# 项目配置文件

#spring.datasource.url=jdbc:mysql://localhost:3306/daoyun?useUnicode=true&characterEncoding=utf-8&useSSL=true&serverTimezone=UTC

spring.datasource.url=jdbc:mysql://120.77.220.113:3306/daoyun?useUnicode=true&characterEncoding=utf-8&useSSL=true&serverTimezone=Asia/Shanghai

spring.datasource.username=root

spring.datasource.password=

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.type=com.zaxxer.hikari.HikariDataSource

spring.datasource.hikari.minimum-idle=5

spring.datasource.hikari.maximum-pool-size=15

spring.datasource.hikari.auto-commit=true

spring.datasource.hikari.idle-timeout=30000

spring.datasource.hikari.pool-name=DatebookHikariCP

spring.datasource.hikari.max-lifetime=500000

spring.datasource.hikari.connection-timeout=30000

spring.datasource.hikari.connection-test-query=SELECT 1

spring.mail.default-encoding=UTF-8

spring.mail.host=smtp.qq.com

spring.mail.username=xxxxx@qq.com

spring.mail.password=

spring.mail.properties.mail.smtp.auth=true

spring.mail.properties.mail.smtp.starttls.enable=true

spring.mail.properties.mail.smtp.starttls.required=true

spring.mail.properties.mail.smtp.socketFactory.class=javax.net.ssl.SSLSocketFactory

spring.mail.properties.mail.smtp.socketFactory.port=465

###日志

logging.path=./logs