package com.pearadmin.modules.sys.context;

import com.pearadmin.modules.sys.domain.\*;

import com.pearadmin.modules.sys.mapper.\*;

import com.pearadmin.modules.sys.service.ISysDictDataService;

import com.pearadmin.modules.sys.service.ISysLogService;

import com.pearadmin.modules.sys.domain.SysLog;

import com.pearadmin.common.plugin.system.domain.\*;

import com.pearadmin.common.plugin.system.service.SysContext;

import org.springframework.beans.BeanUtils;

import org.springframework.stereotype.Service;

import javax.annotation.Resource;

import java.util.ArrayList;

import java.util.List;

/\*\*

\* Describe: 对外开放的公用服务

\* Author: 就 眠 仪 式

\* CreateTime: 2019/10/23

\* \*/

@Service

public class SysContextService implements SysContext {

@Resource

private ISysLogService sysLogService;

@Resource

private SysUserMapper sysUserMapper;

@Resource

private SysRoleMapper sysRoleMapper;

@Resource

private SysPowerMapper sysPowerMapper;

@Resource

private SysConfigMapper sysConfigMapper;

@Resource

private ISysDictDataService iSysDictDataService;

@Resource

private SysDictDataMapper sysDictDataMapper;

@Override

public SysBaseUser getUserByName(String username) {

SysUser sysUser= sysUserMapper.selectByUsername(username);

SysBaseUser sysUserModel=null;

if(sysUser!=null) {

sysUserModel = new SysBaseUser();

BeanUtils.copyProperties(sysUser, sysUserModel);

List<SysPower> powerList = sysPowerMapper.selectByUsername(username);

if(powerList!=null&&powerList.size()>0){

List<SysBasePower> sysPowerModelList=new ArrayList<>();

for(SysPower sysPower:powerList){

try{

SysBasePower sysPowerModel = new SysBasePower();

BeanUtils.copyProperties(sysPower, sysPowerModel);

sysPowerModelList.add(sysPowerModel);

}catch (Exception ignored){

}

}

sysUserModel.setPowerList(sysPowerModelList);

}

}

return sysUserModel;

}

@Override

public SysBaseUser getUserById(String id) {

SysUser sysUser= sysUserMapper.selectById(id);

SysBaseUser sysUserModel=null;

if(sysUser!=null) {

sysUserModel = new SysBaseUser();

BeanUtils.copyProperties(sysUser, sysUserModel);

}

return sysUserModel;

}

@Override

public List<SysBaseRole> getRolesByUsername(String username) {

List<SysRole> roles= sysRoleMapper.selectByUsername(username);

List<SysBaseRole> sysRoleModelList=new ArrayList<>();

if(roles!=null&&roles.size()>0){

for(SysRole sysRole:roles){

try{

SysBaseRole sysRoleModel = new SysBaseRole();

BeanUtils.copyProperties(sysRole, sysRoleModel);

sysRoleModelList.add(sysRoleModel);

}catch (Exception e){

e.printStackTrace();

}

}

}

return sysRoleModelList;

}

@Override

public List<SysBaseDict> selectDictByCode(String typeCode) {

List<SysDictData> sysDictDataList= iSysDictDataService.selectByCode(typeCode);

return buildSysDictDataModel(sysDictDataList);

}

@Override

public List<SysBaseDict> queryTableDictItemsByCode(String table, String text, String code) {

return buildSysDictDataModel(sysDictDataMapper.queryTableDictItemsByCode(table,text,code));

}

@Override

public List<SysBaseDict> queryTableDictItemsByCodeAndFilter(String table, String text, String code, String filterSql) {

return buildSysDictDataModel(sysDictDataMapper.queryTableDictItemsByCodeAndFilter(table,text,code,filterSql));

}

@Override

public List<SysBaseDict> queryTableDictByKeys(String table, String text, String code, String[] keyArray) {

return buildSysDictDataModel(sysDictDataMapper.queryTableDictByKeys(table,text,code,keyArray));

}

private List<SysBaseDict> buildSysDictDataModel(List<SysDictData> sysDictDataList){

List<SysBaseDict> sysDictDataModelList=new ArrayList<>();

if(sysDictDataList!=null&&sysDictDataList.size()>0){

for(SysDictData sysDictData:sysDictDataList){

try{

SysBaseDict sysDictDataModel = new SysBaseDict();

BeanUtils.copyProperties(sysDictData, sysDictDataModel);

sysDictDataModelList.add(sysDictDataModel);

}catch (Exception e){

e.printStackTrace();

}

}

}

return sysDictDataModelList;

}

@Override

public String getConfig(String code) {

SysConfig config = sysConfigMapper.selectByCode(code);

return config!=null?config.getConfigValue():"";

}

@Override

public Boolean saveLog(SysBaseLog baseLog) {

SysLog log = new SysLog();

BeanUtils.copyProperties(baseLog,log);

return sysLogService.save(log);

}

}

package com.pearadmin.modules.sys.controller;

import com.pearadmin.common.web.domain.response.Result;

import com.wf.captcha.utils.CaptchaUtil;

import com.pearadmin.common.web.base.BaseController;

import io.swagger.annotations.Api;

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

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

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpServletRequest;

/\*\*

\* Describe: 验证码控制器

\* Author: 就 眠 仪 式

\* CreateTime: 2019/10/23

\* \*/

@RestController

@Api(tags = {"验证生成"})

@RequestMapping("system/captcha")

public class CaptchaController extends BaseController {

/\*\*

\* 验证码生成

\* @param request 请求报文

\* @param response 响应报文

\* \*/

@RequestMapping("generate")

public void generate(HttpServletRequest request, HttpServletResponse response) throws Exception {

CaptchaUtil.out(request, response);

}

/\*\*

\* 异步验证

\* @param request 请求报文

\* @param captcha 验证码

\* @return 验证结果

\* \*/

@RequestMapping("verify")

public Result verify(HttpServletRequest request, String captcha){

if(CaptchaUtil.ver(captcha,request)){

return success("验证成功");

}

return failure("验证失败");

}

}

package com.pearadmin.modules.sys.controller;

import com.pearadmin.common.secure.session.SecureSessionService;

import com.pearadmin.modules.sys.service.ISysLogService;

import io.swagger.annotations.Api;

import org.springframework.security.core.session.SessionRegistry;

import org.springframework.ui.Model;

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

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

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

import com.pearadmin.common.plugin.logging.aop.annotation.Logging;

import com.pearadmin.common.plugin.logging.aop.enums.BusinessType;

import com.pearadmin.common.tools.secure.SecurityUtil;

import com.pearadmin.common.web.base.BaseController;

import org.springframework.web.servlet.ModelAndView;

import javax.annotation.Resource;

import javax.servlet.http.HttpServletRequest;

/\*\*

\* Describe: 入 口 控 制 器

\* Author: 就 眠 仪 式

\* CreateTime: 2019/10/23

\* \*/

@RestController

@RequestMapping

@Api(tags = {"项目入口"})

public class EntranceController extends BaseController {

@Resource

private SessionRegistry sessionRegistry;

@Resource

private ISysLogService sysLogService;

/\*\*

\* Describe: 获取登录视图

\* Param: ModelAndView

\* Return: 登录视图

\* \*/

@GetMapping("login")

public ModelAndView login(HttpServletRequest request){

if (SecurityUtil.isAuthentication()) {

SecureSessionService.expiredSession(request, sessionRegistry);

return jumpPage("index");

}else{

return jumpPage("login");

}

}

/\*\*

\* Describe: 获取主页视图

\* Param: ModelAndView

\* Return: 登录视图

\* \*/

@GetMapping("index")

@Logging(title = "主页",describe = "返回 Index 主页视图",type = BusinessType.ADD)

public ModelAndView index( )

{

return jumpPage("index");

}

/\*\*

\* Describe: 获取主页视图

\* Param: ModelAndView

\* Return: 主页视图

\* \*/

@GetMapping("console")

public ModelAndView home(Model model)

{

return jumpPage("console/console");

}

/\*\*

\* Describe:无权限页面

\* Return:返回403页面

\* \*/

@GetMapping("error/403")

public ModelAndView noPermission(){

return jumpPage("error/403");

}

/\*\*

\* Describe:找不带页面

\* Return:返回404页面

\* \*/

@GetMapping("error/404")

public ModelAndView notFound(){

return jumpPage("error/404");

}

/\*\*

\* Describe:异常处理页

\* Return:返回500界面

\* \*/

@GetMapping("error/500")

public ModelAndView onException(){

return jumpPage("error/500");

}

}

package com.pearadmin.modules.sys.controller;

import com.pearadmin.common.tools.system.SystemUtil;

import com.pearadmin.common.tools.system.CpuInfo;

import com.pearadmin.common.constant.ControllerConstant;

import com.pearadmin.common.web.base.BaseController;

import io.swagger.annotations.Api;

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.ui.Model;

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

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

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

import org.springframework.web.servlet.ModelAndView;

@RestController

@Api(tags = {"服务监控"})

@RequestMapping(ControllerConstant.API\_SYSTEM\_PREFIX + "monitor")

public class MonitorController extends BaseController {

@GetMapping("main")

@PreAuthorize("hasPermission('/system/monitor/main','sys:monitor:main')")

public ModelAndView main(Model model){

CpuInfo cpu = SystemUtil.getCpu();

model.addAttribute("cpu", cpu);

return jumpPage("system/monitor/main");

}

}

package com.pearadmin.modules.sys.controller;

import com.github.pagehelper.PageInfo;

import com.pearadmin.common.mgcfx.main.main;

import com.pearadmin.common.plugin.logging.aop.annotation.Logging;

import com.pearadmin.common.plugin.logging.aop.enums.BusinessType;

import com.pearadmin.modules.sys.domain.NewE;

import com.pearadmin.common.tools.string.Convert;

import com.pearadmin.common.web.base.BaseController;

import com.pearadmin.common.web.domain.request.PageDomain;

import com.pearadmin.common.web.domain.response.Result;

import com.pearadmin.common.web.domain.response.module.ResultTable;

import com.pearadmin.common.tools.secure.SecurityUtil;

import com.pearadmin.modules.sys.domain.SysUser;

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

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.ui.ModelMap;

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

import org.springframework.web.servlet.ModelAndView;

import com.pearadmin.modules.sys.service.INewEService;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.time.LocalDateTime;

import java.util.ArrayList;

import java.util.Date;

import java.util.logging.SimpleFormatter;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

/\*\*

\* 获取文章Controller

\*

\* @author jmys

\* @date 2021-10-28

\*/

@RestController

@RequestMapping("/system/newe")

public class NewEController extends BaseController

{

private String prefix = "system/newe";

@Autowired

private INewEService newEService;

@GetMapping("/main")

@PreAuthorize("hasPermission('/system/newe/main','system:newe:main')")

public ModelAndView main()

{

return jumpPage(prefix + "/main");

}

/\*\*

\* 查询获取文章列表

\*/

@ResponseBody

@GetMapping("/data")

@PreAuthorize("hasPermission('/system/newe/data','system:newe:data')")

@Logging(title = "查询文章", describe = "查询文章", type = BusinessType.QUERY)

public ResultTable list(@ModelAttribute NewE newE, PageDomain pageDomain)

{

PageInfo<NewE> pageInfo = newEService.selectNewEPage(newE,pageDomain);

return pageTable(pageInfo.getList(),pageInfo.getTotal());

}

/\*\*

\* 新增获取文章

\*/

@GetMapping("/add")

@PreAuthorize("hasPermission('/system/newe/add','system:newe:add')")

public ModelAndView add()

{

return jumpPage(prefix + "/add");

}

/\*\*

\* 新增保存获取文章

\*/

@ResponseBody

@PostMapping("/save")

@PreAuthorize("hasPermission('/system/newe/add','system:newe:add')")

@Logging(title = "新增文章", describe = "新增文章", type = BusinessType.ADD)

public Result save(@RequestBody NewE newE) throws ParseException {

SysUser sysUser = (SysUser)SecurityUtil.currentUserObj();

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

String date=format.format(new Date());

newE.setCollecttime(format.parse(date));

newE.setCreateTime(LocalDateTime.now());//LocalDateTime.now()

newE.setCreateBy(sysUser.getUserId());

newE.setCreateName(sysUser.getUsername());

return decide(newEService.insertNewE(newE));

}

/\*\*

\* 修改获取文章

\*/

@GetMapping("/edit")

@PreAuthorize("hasPermission('/system/newe/edit','system:newe:edit')")

public ModelAndView edit(Long newid, ModelMap mmap)

{

NewE newE = newEService.selectNewEById(newid);

mmap.put("newE", newE);

return jumpPage(prefix + "/edit");

}

/\*\*

\* 修改保存获取文章

\*/

@ResponseBody

@PutMapping("/update")

@PreAuthorize("hasPermission('/system/newe/edit','system:newe:edit')")

@Logging(title = "修改文章", describe = "修改文章", type = BusinessType.EDIT)

public Result update(@RequestBody NewE newE)

{

SysUser sysUser = (SysUser)SecurityUtil.currentUserObj();

newE.setUpdateTime(LocalDateTime.now());

newE.setUpdateBy(sysUser.getUserId());

newE.setUpdateName(sysUser.getUsername());

return decide(newEService.updateNewE(newE));

}

/\*\*

\* 删除获取文章

\*/

@ResponseBody

@DeleteMapping( "/batchRemove")

@PreAuthorize("hasPermission('/system/newe/remove','system:newe:remove')")

@Logging(title = "批量删除文章", describe = "批量删除文章", type = BusinessType.REMOVE)

public Result batchRemove(String ids)

{

return decide(newEService.deleteNewEByIds(Convert.toStrArray(ids)));

}

/\*\*

\* 删除

\*/

@ResponseBody

@DeleteMapping("/remove/{newid}")

@PreAuthorize("hasPermission('/system/newe/remove','system:newe:remove')")

@Logging(title = "删除文章", describe = "删除文章", type = BusinessType.REMOVE)

public Result remove(@PathVariable("newid") Long newid)

{

return decide(newEService.deleteNewEById(newid));

}

/\*\*

\* 爬取文章

\*/

@GetMapping("/hq")

@PreAuthorize("hasPermission('/system/newe/hq','system:newe:hq')")

public ModelAndView hq()

{

return jumpPage(prefix + "/hq");

}

/\*\*

\* 爬取保存文章

\*/

@GetMapping("/newSave")

@PreAuthorize("hasPermission('/system/newe/hqSave','system:newe:hqSave')")

@Logging(title = "爬取保存文章", describe = "爬取保存文章", type = BusinessType.ADD)

public Result newSave(String newurl) throws Exception {

NewE n=new NewE();

main a=new main();

String str=a.getText(newurl);

//正则匹配

//标题

String title="(?<=正文)([\\s\\S]\*)?(?=来源)";

String titleNew=a.regexContext(title,str);

//来源

String ly="(?<=来源：)([\\s\\S]\*)?(?=作者)";

String lyNew=a.regexContext(ly,str);

//作者

String author="(?<=作者：)([\\s\\S]\*)?(?=编辑)";

String authorNew=a.regexContext(author,str);

//发布时间

String fbTime="(?<=发布时间：)([\\s\\S]\*)?(?=点击次数)";

String fbTimeNew=a.regexContext(fbTime,str);

//添加数据

n.setNewurl(newurl);

n.setNewtitle(titleNew);

n.setNewsource(lyNew);

n.setNewauthor(authorNew);

n.setNewcontent(str);

SimpleDateFormat format=new SimpleDateFormat("yyyy年MM月dd日");

n.setNewcreatetime(format.parse(fbTimeNew));

n.setCollecttime(format.parse(format.format(new Date())));

//ArrayList all= a.fixArrayList(a.mysqlValue("bad\_word"),a.getWebText(newurl));

//System.out.println("敏感词："+all);

//System.out.println("敏感词数："+all.size()+"个");

return decide(newEService.insertNewE(n));

}

/\*\*

\* 修改分析文章

\*/

@GetMapping("/fx")

@PreAuthorize("hasPermission('/system/newe/fx','system:newe:fx')")

@Logging(title = "修改分析文章", describe = "修改分析文章", type = BusinessType.EDIT)

public ModelAndView fx(Long newid) throws Exception {

ModelAndView mv=new ModelAndView();

NewE newE = newEService.selectNewEById(newid);

main a=new main();

ArrayList all= a.fixArrayList(a.mysqlValue("bad\_word"),a.getWebText(newE.getNewurl()));

// System.out.println("敏感词："+all);

// System.out.println("敏感词数："+all.size()+"个");

mv.addObject("newE",newE);

mv.addObject("mgc",all);

mv.setViewName(prefix + "/fx");

return mv;

}

}

package com.pearadmin.modules.sys.controller;

import com.github.pagehelper.PageInfo;

import com.pearadmin.common.tools.string.Convert;

import com.pearadmin.common.web.base.BaseController;

import com.pearadmin.common.web.domain.request.PageDomain;

import com.pearadmin.common.web.domain.response.Result;

import com.pearadmin.common.web.domain.response.module.ResultTable;

import com.pearadmin.common.tools.secure.SecurityUtil;

import com.pearadmin.modules.sys.domain.Notice;

import com.pearadmin.modules.sys.domain.SysUser;

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

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.ui.ModelMap;

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

import org.springframework.web.servlet.ModelAndView;

import com.pearadmin.modules.sys.service.INoticeService;

import java.text.SimpleDateFormat;

import java.time.LocalDateTime;

import java.util.Date;

import java.util.List;

/\*\*

\* 系统公告Controller

\*

\* @author jmys

\* @date 2021-10-16

\*/

@RestController

@RequestMapping("/system/notice")

public class NoticeController extends BaseController

{

private String prefix = "system/notice";

@Autowired

private INoticeService noticeService;

@GetMapping("/main")

@PreAuthorize("hasPermission('/system/notice/main','system:notice:main')")

public ModelAndView main()

{

return jumpPage(prefix + "/main");

}

/\*\*

\* 查询系统公告列表

\*/

@ResponseBody

@GetMapping("/data")

@PreAuthorize("hasPermission('/system/notice/data','system:notice:data')")

public ResultTable list(@ModelAttribute Notice notice, PageDomain pageDomain)

{

PageInfo<Notice> pageInfo = noticeService.selectNoticePage(notice,pageDomain);

return pageTable(pageInfo.getList(),pageInfo.getTotal());

}

/\*\*

\* 主页查询公告列表

\*/

@ResponseBody

@GetMapping("/data-list")

@PreAuthorize("hasPermission('/system/notice/data-list','system:notice:data')")

public List<Notice> data\_list(Notice notice)

{

return noticeService.selectNoticeList(notice);

}

/\*\*

\* 主页查询公告列表

\*/

@ResponseBody

@GetMapping("/data-list-id")

@PreAuthorize("hasPermission('/system/notice/data-list-id','system:notice:data')")

public Notice data\_list\_id(Long id)

{

return noticeService.selectNoticeById(id);

}

/\*\*

\* 新增系统公告

\*/

@GetMapping("/add")

@PreAuthorize("hasPermission('/system/notice/add','system:notice:add')")

public ModelAndView add()

{

return jumpPage(prefix + "/add");

}

/\*\*

\* 新增保存系统公告

\*/

@ResponseBody

@PostMapping("/save")

@PreAuthorize("hasPermission('/system/notice/add','system:notice:add')")

public Result save(@RequestBody Notice notice)

{

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

Date date=new Date();

notice.setNoticetime(format.format(date));

SysUser sysUser = (SysUser)SecurityUtil.currentUserObj();

notice.setCreateTime(LocalDateTime.now());

notice.setCreateBy(sysUser.getUserId());

notice.setCreateName(sysUser.getUsername());

return decide(noticeService.insertNotice(notice));

}

/\*\*

\* 修改系统公告

\*/

@GetMapping("/edit")

@PreAuthorize("hasPermission('/system/notice/edit','system:notice:edit')")

public ModelAndView edit(Long id, ModelMap mmap)

{

Notice notice = noticeService.selectNoticeById(id);

mmap.put("notice", notice);

return jumpPage(prefix + "/edit");

}

/\*\*

\* 修改保存系统公告

\*/

@ResponseBody

@PutMapping("/update")

@PreAuthorize("hasPermission('/system/notice/edit','system:notice:edit')")

public Result update(@RequestBody Notice notice)

{

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

Date date=new Date();

notice.setNoticetime(format.format(date));

SysUser sysUser = (SysUser)SecurityUtil.currentUserObj();

notice.setUpdateTime(LocalDateTime.now());

notice.setUpdateBy(sysUser.getUserId());

notice.setUpdateName(sysUser.getUsername());

return decide(noticeService.updateNotice(notice));

}

/\*\*

\* 删除系统公告

\*/

@ResponseBody

@DeleteMapping( "/batchRemove")

@PreAuthorize("hasPermission('/system/notice/remove','system:notice:remove')")

public Result batchRemove(String ids)

{

return decide(noticeService.deleteNoticeByIds(Convert.toStrArray(ids)));

}

/\*\*

\* 删除

\*/

@ResponseBody

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

@PreAuthorize("hasPermission('/system/notice/remove','system:notice:remove')")

public Result remove(@PathVariable("id") Long id)

{

return decide(noticeService.deleteNoticeById(id));

}

}

package com.pearadmin.modules.sys.controller;

import com.github.pagehelper.PageInfo;

import com.pearadmin.common.constant.ControllerConstant;

import com.pearadmin.common.tools.sequence.SequenceUtil;

import com.pearadmin.common.web.base.BaseController;

import com.pearadmin.common.web.domain.request.PageDomain;

import com.pearadmin.common.web.domain.response.Result;

import com.pearadmin.common.web.domain.response.module.ResultTable;

import com.pearadmin.modules.sys.domain.SysConfig;

import com.pearadmin.modules.sys.service.ISysConfigService;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.ui.Model;

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

import org.springframework.web.servlet.ModelAndView;

import javax.annotation.Resource;

import java.time.LocalDateTime;

/\*\*

\* Describe: 系 统 配 置 控 制 器

\* Author: 就 眠 仪 式

\* CreateTime: 2019/10/23

\* \*/

@RestController

@Api(tags = {"全局配置"})

@RequestMapping(ControllerConstant.API\_SYSTEM\_PREFIX + "config")

public class SysConfigController extends BaseController {

/\*\*

\* 引入服务

\* \*/

@Resource

private ISysConfigService sysConfigService;

/\*\*

\* 基础路径

\* \*/

private String MODULE\_PATH = "system/config/";

/\*\*

\* Describe: 数据字典列表视图

\* Param: ModelAndView

\* Return: ModelAndView

\* \*/

@GetMapping("main")

@PreAuthorize("hasPermission('/system/config/main','sys:config:main')")

public ModelAndView main(){

return jumpPage(MODULE\_PATH + "main");

}

/\*\*

\* Describe: 数据字典列表数据

\* Param: sysConfig

\* Return: ResultTable

\* \*/

@GetMapping("data")

@PreAuthorize("hasPermission('/system/config/data','sys:config:data')")

public ResultTable data(SysConfig param, PageDomain pageDomain){

PageInfo<SysConfig> pageInfo = sysConfigService.page(param,pageDomain);

return pageTable(pageInfo.getList(),pageInfo.getTotal());

}

/\*\*

\* Describe: 数据字典类型新增视图

\* Param: sysConfig

\* Return: ModelAndView

\* \*/

@GetMapping("add")

@PreAuthorize("hasPermission('/system/config/add','sys:config:add')")

public ModelAndView add(){

return jumpPage(MODULE\_PATH + "add");

}

/\*\*

\* Describe: 新增字典类型接口

\* Param: sysConfig

\* Return: ResultBean

\* \*/

@PostMapping("save")

@PreAuthorize("hasPermission('/system/config/add','sys:config:add')")

public Result save(@RequestBody SysConfig sysConfig){

sysConfig.setConfigId(SequenceUtil.makeStringId());

sysConfig.setCreateTime(LocalDateTime.now());

sysConfig.setConfigType("custom");

boolean result = sysConfigService.save(sysConfig);

return decide(result);

}

/\*\*

\* Describe: 数据字典类型修改视图

\* Param: sysConfig

\* Return: ModelAndView

\* \*/

@GetMapping("edit")

@PreAuthorize("hasPermission('/system/config/edit','sys:config:edit')")

public ModelAndView edit(Model model, String configId){

model.addAttribute("sysConfig",sysConfigService.getById(configId));

return jumpPage(MODULE\_PATH + "edit");

}

/\*\*

\* Describe: 数据字典类型修改视图

\* Param: sysConfig

\* Return: ModelAndView

\* \*/

@PutMapping("update")

@PreAuthorize("hasPermission('/system/config/edit','sys:config:edit')")

public Result update(@RequestBody SysConfig sysConfig){

sysConfig.setUpdateTime(LocalDateTime.now());

boolean result = sysConfigService.updateById(sysConfig);

return decide(result);

}

/\*\*

\* Describe: 数据字典删除

\* Param: sysConfig

\* Return: ModelAndView

\* \*/

@DeleteMapping("remove/{id}")

@PreAuthorize("hasPermission('/system/config/remove','sys:config:remove')")

public Result remove(@PathVariable("id")String id){

Boolean result = sysConfigService.remove(id);

return decide(result);

}

/\*\*

\* Describe: 系统配置批量删除接口

\* Param: ids

\* Return: Result

\* \*/

@DeleteMapping("batchRemove/{ids}")

@ApiOperation(value="批量删除系统配置数据")

@PreAuthorize("hasPermission('/system/config/remove','sys:config:remove')")

public Result batchRemove(@PathVariable String ids){

boolean result = sysConfigService.batchRemove(ids.split(","));

return decide(result);

}

}

package com.pearadmin.modules.sys.controller;

import com.pearadmin.common.constant.ControllerConstant;

import com.pearadmin.common.tools.sequence.SequenceUtil;

import com.pearadmin.common.web.base.BaseController;

import com.pearadmin.common.web.domain.response.module.ResultTree;

import com.pearadmin.common.web.domain.response.Result;

import com.pearadmin.common.web.domain.response.module.ResultTable;

import com.pearadmin.modules.sys.domain.SysDept;

import com.pearadmin.modules.sys.service.ISysDeptService;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.springframework.security.access.prepost.PreAuthorize;

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

import org.springframework.web.servlet.ModelAndView;

import javax.annotation.Resource;

import java.util.List;

/\*\*

\* Describe: 部门管理

\* Author: 就 眠 仪 式

\* CreateTime: 2019/10/23

\*/

@RestController

@Api(tags = {"组织部门"})

@RequestMapping(ControllerConstant.API\_SYSTEM\_PREFIX + "dept")

public class SysDeptController extends BaseController {

/\*\*

\* Describe: 基础路径

\*/

private static String MODULE\_PATH = "system/dept/";

/\*\*

\* Describe: 部门模块服务

\*/

@Resource

private ISysDeptService sysDeptService;

/\*\*

\* Describe: 获取部门列表视图

\* Param ModelAndView

\* Return 用户列表视图

\*/

@GetMapping("main")

@PreAuthorize("hasPermission('/system/dept/main','sys:dept:main')")

public ModelAndView main() {

return jumpPage(MODULE\_PATH + "main");

}

/\*\*

\* Describe: 获取部门列表数据

\* Param SysDept PageDomain

\* Return 部门列表数据

\*/

@GetMapping("data")

@PreAuthorize("hasPermission('/system/dept/data','sys:dept:data')")

public ResultTable data(SysDept param) {

List<SysDept> data = sysDeptService.list(param);

return dataTable(data);

}

/\*\*

\* Describe: 获取部门树状数据结构

\* Param ModelAndView

\* Return ModelAndView

\*/

@GetMapping("tree")

public ResultTree tree(SysDept param) {

List<SysDept> data = sysDeptService.list(param);

return dataTree(data);

}

/\*\*

\* Describe: 获取部门新增视图

\* Param ModelAndView

\* Return 部门新增视图

\*/

@GetMapping("add")

@PreAuthorize("hasPermission('/system/dept/add','sys:dept:add')")

public ModelAndView add() {

return jumpPage(MODULE\_PATH + "add");

}

/\*\*

\* Describe: 保存部门信息

\* Param SysDept

\* Return 执行结果

\*/

@PostMapping("save")

@ApiOperation(value = "保存部门数据")

@PreAuthorize("hasPermission('/system/dept/add','sys:dept:add')")

public Result save(@RequestBody SysDept sysDept) {

sysDept.setDeptId(SequenceUtil.makeStringId());

boolean result = sysDeptService.save(sysDept);

return decide(result);

}

/\*\*

\* Describe: 获取部门修改视图

\* Param ModelAndView

\* Return 部门修改视图

\*/

@GetMapping("edit")

@PreAuthorize("hasPermission('/system/dept/edit','sys:dept:edit')")

public ModelAndView edit(ModelAndView modelAndView, String deptId) {

modelAndView.addObject("sysDept", sysDeptService.getById(deptId));

modelAndView.setViewName(MODULE\_PATH + "edit");

return modelAndView;

}

/\*\*

\* Describe: 修改部门信息

\* Param SysDept

\* Return 执行结果

\*/

@PutMapping("update")

@PreAuthorize("hasPermission('/system/dept/edit','sys:dept:edit')")

public Result update(@RequestBody SysDept SysDept) {

boolean result = sysDeptService.update(SysDept);

return decide(result);

}

/\*\*

\* Describe: 部门删除接口

\* Param: id

\* Return: Result

\*/

@DeleteMapping("remove/{id}")

@PreAuthorize("hasPermission('/system/dept/remove','sys:dept:remove')")

public Result remove(@PathVariable String id) {

boolean result = sysDeptService.remove(id);

return decide(result);

}

/\*\*

\* Describe: 部门批量删除接口

\* Param: ids

\* Return: Result

\*/

@DeleteMapping("batchRemove/{ids}")

@PreAuthorize("hasPermission('/system/dept/remove','sys:dept:remove')")

public Result batchRemove(@PathVariable String ids) {

boolean result = sysDeptService.batchRemove(ids.split(","));

return decide(result);

}

/\*\*

\* Describe: 根据 Id 启用部门

\* Param: roleId

\* Return: Result

\*/

@PutMapping("enable")

@PreAuthorize("hasPermission('/system/dept/edit','sys:dept:edit')")

public Result enable(@RequestBody SysDept SysDept) {

SysDept.setStatus("0");

boolean result = sysDeptService.update(SysDept);

return decide(result);

}

/\*\*

\* Describe: 根据 Id 禁用部门

\* Param: roleId

\* Return: Result

\*/

@PutMapping("disable")

@PreAuthorize("hasPermission('/system/dept/edit','sys:dept:edit')")

public Result disable(@RequestBody SysDept SysDept) {

SysDept.setStatus("1");

boolean result = sysDeptService.update(SysDept);

return decide(result);

}

}

package com.pearadmin.modules.sys.controller;

import com.github.pagehelper.PageInfo;

import com.pearadmin.common.constant.ControllerConstant;

import com.pearadmin.common.plugin.system.domain.SysBaseDict;

import com.pearadmin.common.plugin.system.service.SysContext;

import com.pearadmin.common.tools.sequence.SequenceUtil;

import com.pearadmin.common.tools.database.SqlInjectionUtil;

import com.pearadmin.common.web.base.BaseController;

import com.pearadmin.common.web.domain.request.PageDomain;

import com.pearadmin.common.web.domain.response.Result;

import com.pearadmin.common.web.domain.response.module.ResultTable;

import com.pearadmin.modules.sys.domain.SysDictData;

import com.pearadmin.modules.sys.service.ISysDictDataService;

import io.swagger.annotations.Api;

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.ui.Model;

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

import org.springframework.web.servlet.ModelAndView;

import javax.annotation.Resource;

import javax.servlet.http.HttpServletRequest;

import java.util.List;

/\*\*

\* Describe: 数据字典控制器

\* Author: 就 眠 仪 式

\* CreateTime: 2019/10/23

\* \*/

@RestController

@Api(tags = {"字典类型"})

@RequestMapping(ControllerConstant.API\_SYSTEM\_PREFIX + "dictData")

public class SysDictDataController extends BaseController {

private String MODULE\_PATH = "system/dict/data/";

@Resource

private ISysDictDataService sysDictDataService;

@Resource

private SysContext iSysBaseAPI;

/\*\*

\* Describe: 数据字典列表视图

\* Param: ModelAndView

\* Return: ModelAndView

\* \*/

@GetMapping("main")

@PreAuthorize("hasPermission('/system/dictData/main','sys:dictData:main')")

public ModelAndView main(Model model, String typeCode){

model.addAttribute("typeCode",typeCode);

return jumpPage(MODULE\_PATH + "main");

}

/\*\*

\* Describe: 数据字典列表数据

\* Param: sysDictType

\* Return: Result

\* \*/

@GetMapping("data")

@PreAuthorize("hasPermission('/system/dictData/data','sys:dictData:data')")

public ResultTable data (SysDictData sysDictData, PageDomain pageDomain){

PageInfo<SysDictData> pageInfo = sysDictDataService.page(sysDictData,pageDomain);

return pageTable(pageInfo.getList(),pageInfo.getTotal());

}

/\*\*

\* Describe: 根据字典code获取数据字典列表数据

\* Param: typeCode

\* Return: Result

\* \*/

@GetMapping("selectByCode")

@PreAuthorize("hasPermission('/system/dictData/selectByCode','sys:dictData:selectByCode')")

public Result selectByCode (String typeCode){

List<SysDictData> list = sysDictDataService.selectByCode(typeCode);

return success(list);

}

/\*\*

\* Describe: 获取字典数据

\* Param: dictCode 字典code

\* Return: Result

\*/

@GetMapping(value = "/getDictItems/{dictCode}")

public Result<List<SysBaseDict>> getDictItems(@PathVariable String dictCode, @RequestParam(value = "sign",required = false) String sign, HttpServletRequest request) {

Result<List<SysBaseDict>> result = new Result<List<SysBaseDict>>();

List<SysBaseDict> ls = null;

try {

if(dictCode.contains(",")) {

String[] params = dictCode.split(",");

if(params.length<3) {

return Result.failure("字典Code格式不正确！");

}

final String[] sqlInjCheck = {params[0],params[1],params[2]};

SqlInjectionUtil.filterContent(sqlInjCheck);

if(params.length==4) {

SqlInjectionUtil.specialFilterContent(params[3]);

ls = iSysBaseAPI.queryTableDictItemsByCodeAndFilter(params[0],params[1],params[2],params[3]);

}else if (params.length==3) {

ls = iSysBaseAPI.queryTableDictItemsByCode(params[0],params[1],params[2]);

}else{

return Result.failure("字典Code格式不正确！");

}

}else {

ls = iSysBaseAPI.selectDictByCode(dictCode);

}

result.setSuccess(true);

result.setData(ls);

} catch (Exception e) {

e.printStackTrace();

return Result.failure("操作失败！");

}

return result;

}

/\*\*

\* Describe: 根据字典code加载字典text

\* Param: dictCode 字典code

\* Return: Result

\*/

@RequestMapping(value = "/loadDictItem/{dictCode}", method = RequestMethod.GET)

public Result<List<SysBaseDict>> loadDictItem(@PathVariable String dictCode, @RequestParam(name="key") String keys, @RequestParam(value = "sign",required = false) String sign, HttpServletRequest request) {

Result<List<SysBaseDict>> result = new Result<>();

try {

if(dictCode.contains(",")) {

String[] params = dictCode.split(",");

if(params.length!=3) {

return Result.failure("字典Code格式不正确！");

}

String[] keyArray = keys.split(",");

List<SysBaseDict> texts = iSysBaseAPI.queryTableDictByKeys(params[0], params[1], params[2], keyArray);

return Result.success(texts);

}else {

return Result.failure("字典Code格式不正确！");

}

} catch (Exception e) {

e.printStackTrace();

return Result.failure("操作失败！");

}

}

/\*\*

\* Describe: 数据字典类型新增视图

\* Param: sysDictType

\* Return: ModelAndView

\* \*/

@GetMapping("add")

@PreAuthorize("hasPermission('/system/dictData/add','sys:dictData:add')")

public ModelAndView add(Model model,String typeCode){

model.addAttribute("typeCode",typeCode);

return jumpPage(MODULE\_PATH+"add");

}

/\*\*

\* Describe: 新增字典类型接口

\* Param: sysDictType

\* Return: Result

\* \*/

@PostMapping("save")

@PreAuthorize("hasPermission('/system/dictData/add','sys:dictData:add')")

public Result save(@RequestBody SysDictData sysDictData){

sysDictData.setDataId(SequenceUtil.makeStringId());

Boolean result = sysDictDataService.save(sysDictData);

return decide(result);

}

/\*\*

\* Describe: 数据字典类型修改视图

\* Param: sysDictType

\* Return: ModelAndView

\* \*/

@GetMapping("edit")

@PreAuthorize("hasPermission('/system/dictData/edit','sys:dictData:edit')")

public ModelAndView edit(Model model,String dataId){

model.addAttribute("sysDictData",sysDictDataService.getById(dataId));

return jumpPage(MODULE\_PATH+"edit");

}

/\*\*

\* Describe: 数据字典类型修改视图

\* Param: sysDictData

\* Return: ModelAndView

\* \*/

@PutMapping("update")

@PreAuthorize("hasPermission('/system/dictData/edit','sys:dictData:edit')")

public Result update(@RequestBody SysDictData sysDictData){

boolean result = sysDictDataService.updateById(sysDictData);

return decide(result);

}

/\*\*

\* Describe: 数据字典删除

\* Param: id

\* Return: Result

\* \*/

@DeleteMapping("remove/{id}")

@PreAuthorize("hasPermission('/system/dictData/remove','sys:dictData:remove')")

public Result remove(@PathVariable("id")String id){

Boolean result = sysDictDataService.remove(id);

return decide(result);

}

/\*\*

\* Describe: 根据 Id 启用字典

\* Param dictId

\* Return ResuTree

\* \*/

@PutMapping("enable")

public Result enable(@RequestBody SysDictData sysDictData){

sysDictData.setEnable("0");

boolean result = sysDictDataService.updateById(sysDictData);

return decide(result);

}

/\*\*

\* Describe: 根据 Id 禁用字典

\* Param dictId

\* Return ResuTree

\* \*/

@PutMapping("disable")

public Result disable(@RequestBody SysDictData sysDictData){

sysDictData.setEnable("1");

boolean result = sysDictDataService.updateById(sysDictData);

return decide(result);

}

}

package com.pearadmin.modules.sys.controller;

import com.github.pagehelper.PageInfo;

import com.pearadmin.common.constant.ControllerConstant;

import com.pearadmin.common.tools.sequence.SequenceUtil;

import com.pearadmin.common.web.base.BaseController;

import com.pearadmin.common.web.domain.request.PageDomain;

import com.pearadmin.common.web.domain.response.Result;

import com.pearadmin.common.web.domain.response.module.ResultTable;

import com.pearadmin.modules.sys.domain.SysDictType;

import com.pearadmin.modules.sys.service.ISysDictDataService;

import com.pearadmin.modules.sys.service.ISysDictTypeService;

import io.swagger.annotations.Api;

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.ui.Model;

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

import org.springframework.web.servlet.ModelAndView;

import javax.annotation.Resource;

import java.util.List;

/\*\*

\* Describe: 数据字典控制器

\* Author: 就 眠 仪 式

\* CreateTime: 2019/10/23

\* \*/

@RestController

@Api(tags = {"数据字典"})

@RequestMapping(ControllerConstant.API\_SYSTEM\_PREFIX + "dictType")

public class SysDictTypeController extends BaseController {

@Resource

private ISysDictTypeService sysDictTypeService;

@Resource

private ISysDictDataService sysDictDataService;

private String MODULE\_PATH = "system/dict/";

/\*\*

\* Describe: 数据字典列表视图

\* Param: ModelAndView

\* Return: ModelAndView

\* \*/

@GetMapping("main")

@PreAuthorize("hasPermission('/system/dictType/main','sys:dictType:main')")

public ModelAndView main(){

return jumpPage(MODULE\_PATH + "main");

}

/\*\*

\* Describe: 数据字典列表数据

\* Param: sysDictType

\* Return: ResuTable

\* \*/

@GetMapping("data")

@PreAuthorize("hasPermission('/system/dictType/data','sys:dictType:data')")

public ResultTable data(SysDictType sysDictType, PageDomain pageDomain){

PageInfo<SysDictType> pageInfo = sysDictTypeService.page(sysDictType,pageDomain);

return pageTable(pageInfo.getList(),pageInfo.getTotal());

}

@GetMapping("list")

@PreAuthorize("hasPermission('/system/dictType/data','sys:dictType:data')")

public ResultTable list(SysDictType sysDictType, PageDomain pageDomain){

List<SysDictType> list = sysDictTypeService.list(sysDictType);

return dataTable(list);

}

/\*\*

\* Describe: 数据字典类型新增视图

\* Param: sysDictType

\* Return: ModelAndView

\* \*/

@GetMapping("add")

@PreAuthorize("hasPermission('/system/dictType/add','sys:dictType:add')")

public ModelAndView add(){

return jumpPage(MODULE\_PATH + "add");

}

/\*\*

\* Describe: 新增字典类型接口

\* Param: sysDictType

\* Return: ResuBean

\* \*/

@PostMapping("save")

@PreAuthorize("hasPermission('/system/dictType/add','sys:dictType:add')")

public Result save(@RequestBody SysDictType sysDictType){

sysDictType.setId(SequenceUtil.makeStringId());

boolean result = sysDictTypeService.save(sysDictType);

return decide(result);

}

/\*\*

\* Describe: 数据字典类型修改视图

\* Param: sysDictType

\* Return: ModelAndView

\* \*/

@GetMapping("edit")

@PreAuthorize("hasPermission('/system/dictType/edit','sys:dictType:edit')")

public ModelAndView edit(Model model, String dictTypeId){

model.addAttribute("sysDictType",sysDictTypeService.getById(dictTypeId));

return jumpPage(MODULE\_PATH + "edit");

}

/\*\*

\* Describe: 数据字典类型修改视图

\* Param: sysDictType

\* Return: ModelAndView

\* \*/

@PutMapping("update")

@PreAuthorize("hasPermission('/system/dictType/edit','sys:dictType:edit')")

public Result update(@RequestBody SysDictType sysDictType){

boolean result = sysDictTypeService.updateById(sysDictType);

return decide(result);

}

/\*\*

\* Describe: 数据字典删除

\* Param: sysDictType

\* Return: ModelAndView

\* \*/

@DeleteMapping("remove/{id}")

@PreAuthorize("hasPermission('/system/dictType/remove','sys:dictType:remove')")

public Result remove(@PathVariable("id")String id){

Boolean result = sysDictTypeService.remove(id);

return decide(result);

}

/\*\*

\* Describe: 根据 Id 启用字典

\* Param dictId

\* Return ResuTree

\* \*/

@PutMapping("enable")

public Result enable(@RequestBody SysDictType sysDictType){

sysDictType.setEnable("0");

boolean result = sysDictTypeService.updateById(sysDictType);

return decide(result);

}

/\*\*

\* Describe: 根据 Id 禁用字典

\* Param dictId

\* Return ResuTree

\* \*/

@PutMapping("disable")

public Result disable(@RequestBody SysDictType sysDictType){

sysDictType.setEnable("1");

boolean result = sysDictTypeService.updateById(sysDictType);

return decide(result);

}

}

package com.pearadmin.modules.sys.controller;

import com.pearadmin.common.constant.ControllerConstant;

import com.pearadmin.common.web.base.BaseController;

import io.swagger.annotations.Api;

import org.springframework.security.access.prepost.PreAuthorize;

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

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

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

import org.springframework.web.servlet.ModelAndView;

/\*\*

\* Describe: 接口文档控制器

\* Author: 就 眠 仪 式

\* CreateTime: 2019/10/23

\*/

@RestController

@Api(tags = {"接口文档"})

@RequestMapping(ControllerConstant.API\_SYSTEM\_PREFIX + "doc")

public class SysDocController extends BaseController {

private String MODULE\_PATH = "system/doc/";

@GetMapping("main")

@PreAuthorize("hasPermission('/system/doc/main','sys:doc:main')")

public ModelAndView main(){

return jumpPage(MODULE\_PATH + "main");

}

}

package com.pearadmin.modules.sys.controller;

import com.github.pagehelper.PageHelper;

import com.github.pagehelper.PageInfo;

import com.pearadmin.common.config.proprety.TemplateProperty;

import com.pearadmin.common.constant.ControllerConstant;

import com.pearadmin.modules.sys.domain.SysFile;

import com.pearadmin.modules.sys.service.ISysFileService;

import com.pearadmin.common.web.base.BaseController;

import com.pearadmin.common.web.domain.request.PageDomain;

import com.pearadmin.common.web.domain.response.Result;

import com.pearadmin.common.web.domain.response.module.ResultTable;

import io.swagger.annotations.Api;

import org.apache.logging.log4j.util.Strings;

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

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.transaction.annotation.Transactional;

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

import org.springframework.web.multipart.MultipartFile;

import org.springframework.web.servlet.ModelAndView;

import javax.annotation.Resource;

import java.util.Map;

/\*\*

\* Describe: 文件控制器

\* Author: 就 眠 仪 式

\* CreateTime: 2019/10/23

\*/

@RestController

@Api(tags = {"资源文件"})

@RequestMapping(ControllerConstant.API\_SYSTEM\_PREFIX + "file")

public class SysFileController extends BaseController {

/\*\*

\* 系 统 文 件

\*/

private String MODULE\_PATH = "system/file/";

/\*\*

\* 配置文件

\*/

@Resource

private TemplateProperty uploadProperty;

/\*\*

\* 移 除 服 务

\*/

@Autowired

private Map<String, ISysFileService> fileServiceMap;

/\*\*

\* 根据配置文件选择实现类

\* @return

\*/

private ISysFileService getFileService() {

ISysFileService fileService = null;

if (uploadProperty.isFtpUse()) {

fileService = this.fileServiceMap.get("SysFileFTPServiceImpl");

} else {

fileService = this.fileServiceMap.get("SysFileServiceImpl");

}

return fileService;

}

/\*\*

\* Describe: 文件管理页面

\* Param: null

\* Return: ModelAndView

\*/

@GetMapping("main")

@PreAuthorize("hasPermission('/system/file/main','sys:file:main')")

public ModelAndView main() {

return jumpPage(MODULE\_PATH + "main");

}

/\*\*

\* Describe: 文件资源数据

\* Param: PageDomain

\* Return: 文件资源列表

\*/

@GetMapping("data")

@PreAuthorize("hasPermission('/system/file/data','sys:file:data')")

public ResultTable data(PageDomain pageDomain) {

PageHelper.startPage(pageDomain.getPage(), pageDomain.getLimit());

PageInfo<SysFile> pageInfo = new PageInfo<>(getFileService().data());

return pageTable(pageInfo.getList(), pageInfo.getTotal());

}

/\*\*

\* Describe: 文件上传视图

\* Param: null

\* Return: 执行结果

\*/

@GetMapping("add")

@PreAuthorize("hasPermission('/system/file/add','sys:file:add')")

public ModelAndView add() {

return jumpPage(MODULE\_PATH + "add");

}

/\*\*

\* Describe: 文件上传接口

\* Param: SysUser

\* Return: Result

\*/

@PostMapping("upload")

public Result upload(@RequestParam("file") MultipartFile file) {

String result = getFileService().upload(file);

if (Strings.isNotBlank(result)) {

return Result.success(0, "上传成功", result);

} else {

return Result.failure("上传失败");

}

}

/\*\*

\* Describe: 文件获取接口

\* Param: id

\* Return: 文件流

\*/

@GetMapping("download/{id}")

public void download(@PathVariable("id") String id) {

getFileService().download(id);

}

/\*\*

\* Describe: 文件删除接口

\* Param: id

\* Return: 文件流

\*/

@DeleteMapping("remove/{id}")

@PreAuthorize("hasPermission('/system/file/remove','sys:file:remove')")

public Result remove(@PathVariable("id") String id) {

boolean result = getFileService().remove(id);

return Result.decide(result, "删除成功", "删除失败");

}

/\*\*

\* Describe: 文件删除接口

\* Param: id

\* Return: 文件流

\*/

@Transactional

@DeleteMapping("batchRemove/{ids}")

@PreAuthorize("hasPermission('/system/file/remove','sys:file:remove')")

public Result batchRemove(@PathVariable("ids") String ids) {

for (String id : ids.split(",")) {

getFileService().remove(id);

}

return Result.success("删除成功");

}

}

package com.pearadmin.modules.sys.controller;

import com.github.pagehelper.PageHelper;

import com.github.pagehelper.PageInfo;

import com.pearadmin.common.constant.ControllerConstant;

import com.pearadmin.modules.sys.domain.SysLog;

import com.pearadmin.common.plugin.logging.aop.enums.LoggingType;

import com.pearadmin.modules.sys.service.ISysLogService;

import com.pearadmin.common.web.base.BaseController;

import com.pearadmin.common.web.domain.request.PageDomain;

import com.pearadmin.common.web.domain.response.module.ResultTable;

import io.swagger.annotations.Api;

import org.springframework.security.access.prepost.PreAuthorize;

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

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

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

import org.springframework.web.servlet.ModelAndView;

import javax.annotation.Resource;

import java.time.LocalDateTime;

/\*\*

\* Describe: 日志控制器

\* Author: 就 眠 仪 式

\* CreateTime: 2019/10/23

\*/

@RestController

@Api(tags = {"系统日志"})

@RequestMapping(ControllerConstant.API\_SYSTEM\_PREFIX + "log")

public class SysLogController extends BaseController {

@Resource

private ISysLogService sysLogService;

/\*\*

\* Describe: 行为日志视图

\* Param: null

\* Return: ModelAndView

\*/

@GetMapping("main")

@PreAuthorize("hasPermission('/system/log/main','sys:log:main')")

public ModelAndView main() {

return jumpPage("system/log/main");

}

/\*\*

\* Describe: 操作日志数据

\* Param: null

\* Return: ResultTable

\*/

@GetMapping("operateLog")

@PreAuthorize("hasPermission('/system/log/operateLog','sys:log:operateLog')")

public ResultTable operateLog(PageDomain pageDomain, LocalDateTime startTime,LocalDateTime endTime) {

PageHelper.startPage(pageDomain.getPage(), pageDomain.getLimit());

PageInfo<SysLog> pageInfo = new PageInfo<>(sysLogService.data(LoggingType.OPERATE,startTime,endTime));

return pageTable(pageInfo.getList(), pageInfo.getTotal());

}

/\*\*

\* Describe: 登录日志数据

\* Param: null

\* Return: ModelAndView

\*/

@GetMapping("loginLog")

@PreAuthorize("hasPermission('/system/log/loginLog','sys:log:loginLog')")

public ResultTable loginLog(PageDomain pageDomain, LocalDateTime startTime, LocalDateTime endTime) {

PageHelper.startPage(pageDomain.getPage(), pageDomain.getLimit());

PageInfo<SysLog> pageInfo = new PageInfo<>(sysLogService.data(LoggingType.LOGIN, startTime, endTime));

return pageTable(pageInfo.getList(), pageInfo.getTotal());

}

/\*\*

\* Describe: 日志详情

\* Param: null

\* Return: ModelAndView

\*/

@GetMapping("/info")

@PreAuthorize("hasPermission('/system/log/info','sys:log:info')")

public ModelAndView details(){

return jumpPage("system/log/info");

}

}

package com.pearadmin.modules.sys.controller;

import cn.hutool.core.collection.CollectionUtil;

import com.github.pagehelper.PageInfo;

import com.pearadmin.common.constant.ControllerConstant;

import com.pearadmin.common.tools.string.StringUtil;

import com.pearadmin.common.web.base.BaseController;

import com.pearadmin.common.web.domain.request.PageDomain;

import com.pearadmin.common.web.domain.response.Result;

import com.pearadmin.common.web.domain.response.module.ResultTable;

import com.pearadmin.modules.sys.domain.SysMail;

import com.pearadmin.modules.sys.service.ISysMailService;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.springframework.security.access.prepost.PreAuthorize;

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

import org.springframework.web.servlet.ModelAndView;

import javax.annotation.Resource;

import java.util.ArrayList;

/\*\*

\* Describe: 邮箱控制器

\* Author: 就 眠 仪 式

\* CreateTime: 2019/10/23

\*/

@RestController

@Api(tags = {"邮箱管理"})

@RequestMapping(ControllerConstant.API\_SYSTEM\_PREFIX + "mail")

public class SysMailController extends BaseController {

/\*\*

\* 基 础 路 径

\*/

private String MODULE\_PATH = "system/mail/";

@Resource

private ISysMailService sysMailService;

/\*\*

\* Describe: 邮件管理页面

\* Return: ModelAndView

\*/

@GetMapping("/main")

@ApiOperation(value = "邮件管理页面")

@PreAuthorize("hasPermission('/system/mail/main','sys:mail:main')")

public ModelAndView main() {

return jumpPage(MODULE\_PATH + "main");

}

/\*\*

\* Describe: 邮件列表数据

\* Param: PageDomain

\* Return: 邮件列表

\*/

@GetMapping("/data")

@ApiOperation(value = "邮件列表数据")

@PreAuthorize("hasPermission('/system/mail/data','sys:mail:data')")

public ResultTable data(SysMail sysMail, PageDomain pageDomain) {

PageInfo<SysMail> page = sysMailService.page(sysMail, pageDomain);

return pageTable(page.getList(), page.getTotal());

}

/\*\*

\* Describe: 邮件发送页面

\* Return: ModelAndView

\*/

@GetMapping("/add")

@ApiOperation(value = "邮件发送页面")

@PreAuthorize("hasPermission('/system/mail/add','sys:mail:add')")

public ModelAndView add() {

return jumpPage(MODULE\_PATH + "add");

}

/\*\*

\* Describe: 邮件保存和发送

\* Param: SysMail

\* Return: 操作结果

\*/

@PostMapping("/save")

@ApiOperation(value = "邮件保存和发送")

@PreAuthorize("hasPermission('/system/mail/save','sys:mail:save')")

public Result save(@RequestBody SysMail sysMail) {

try {

return decide(sysMailService.save(sysMail));

} catch (Exception e) {

e.printStackTrace();

return failure("请检查邮箱配置");

}

}

/\*\*

\* Describe: 删除邮件

\* Param: String

\* Return: 操作结果

\*/

@DeleteMapping("/remove/{mailId}")

@ApiOperation(value = "删除邮件")

@PreAuthorize("hasPermission('/system/mail/remove','sys:mail:remove')")

public Result remove(@PathVariable String mailId) {

return decide(sysMailService.removeById(mailId));

}

/\*\*

\* Describe: 批量删除邮件

\* Param: String

\* Return: 操作结果

\*/

@DeleteMapping("/batchRemove/{ids}")

@ApiOperation(value = "批量删除邮件")

@PreAuthorize("hasPermission('/system/mail/remove','sys:mail:remove')")

public Result batchRemove(@PathVariable String ids) {

ArrayList<String> idList = CollectionUtil.newArrayList(StringUtil.split(ids, ","));

return decide(sysMailService.removeByIds(idList));

}

}

package com.pearadmin.modules.sys.controller;

import com.pearadmin.common.constant.ControllerConstant;

import com.pearadmin.common.web.base.BaseController;

import com.pearadmin.common.web.domain.response.Result;

import com.pearadmin.common.web.domain.response.module.ResultTable;

import com.pearadmin.common.web.session.HttpSessionContext;

import com.pearadmin.common.web.session.HttpSessionContextHolder;

import com.pearadmin.modules.sys.domain.SysOnlineUser;

import com.pearadmin.modules.sys.domain.SysUser;

import io.swagger.annotations.Api;

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.security.core.session.SessionInformation;

import org.springframework.security.core.session.SessionRegistry;

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

import org.springframework.web.servlet.ModelAndView;

import javax.annotation.Resource;

import java.time.Duration;

import java.time.LocalDateTime;

import java.util.ArrayList;

import java.util.List;

/\*\*

\* Describe: 在线用户控制器

\* Author: 就 眠 仪 式

\* CreateTime: 2019/10/23

\*/

@RestController

@Api(tags = {"在线用户"})

@RequestMapping(ControllerConstant.API\_SYSTEM\_PREFIX + "online")

public class SysOnlineController extends BaseController {

@Resource

private SessionRegistry sessionRegistry;

/\*\*

\* Describe: 在线用户列表

\* Param: username

\* Return: ModelAndView

\*/

@GetMapping("data")

@PreAuthorize("hasPermission('/system/online/data','sys:online:data')")

public ResultTable data(String username) {

List<Object> allPrincipalsUser = sessionRegistry.getAllPrincipals();

List<SysOnlineUser> onlineUser = new ArrayList<>();

for (Object obj : allPrincipalsUser) {

SysOnlineUser sysOnlineUser = new SysOnlineUser();

SysUser objs = (SysUser) obj;

sysOnlineUser.setUserId(objs.getUserId());

sysOnlineUser.setUsername(objs.getUsername());

sysOnlineUser.setRealName(objs.getRealName());

sysOnlineUser.setLastTime(objs.getLastTime());

sysOnlineUser.setOnlineTime(Duration.between(objs.getLastTime(), LocalDateTime.now()).toMinutes() + "分钟");

onlineUser.add(sysOnlineUser);

}

// TODO 使用 Stream 进行内存数据过滤

return dataTable(onlineUser);

}

/\*\*

\* Describe: 获取在线用户列表视图

\* Return: ModelAndView

\*/

@GetMapping("main")

@PreAuthorize("hasPermission('/system/online/main','sys:online:main')")

public ModelAndView main() {

return jumpPage("system/user/online");

}

/\*\*

\* Describe: 踢出用户（下线）

\* Param: onlineId

\* Return: ModelAndView

\*/

@DeleteMapping("/remove/{onlineId}")

@ResponseBody

public Result remove(@PathVariable String onlineId) {

// 从sessionRegistry中获取所有的用户信息

List<Object> principals = sessionRegistry.getAllPrincipals();

for (Object principal : principals) {

SysUser userDetails = (SysUser) principal;

String userId = userDetails.getUserId();

if (onlineId.equals(userId)) {

// 不允许操作admin用户下线

if ("admin".equals(userDetails.getUsername())) {

return failure("不允许操作超级管理员[admin]下线");

}

for (SessionInformation sessionInformation : sessionRegistry.getAllSessions(userDetails, false)) {

sessionInformation.expireNow();

// 从sessionRegistry中清除session信息

sessionRegistry.removeSessionInformation(sessionInformation.getSessionId());

HttpSessionContext sessionContext = HttpSessionContextHolder.currentSessionContext();

// 销毁session

sessionContext.getSession(sessionInformation.getSessionId()).invalidate();

}

return success(String.format("用户[%s]已下线", userDetails.getRealName()));

}

}

return failure();

}

}

package com.pearadmin.modules.sys.controller;

import com.pearadmin.common.constant.ControllerConstant;

import com.pearadmin.common.tools.sequence.SequenceUtil;

import com.pearadmin.common.web.base.BaseController;

import com.pearadmin.common.web.domain.response.module.ResultTree;

import com.pearadmin.common.web.domain.response.Result;

import com.pearadmin.common.web.domain.response.module.ResultTable;

import com.pearadmin.modules.sys.domain.SysPower;

import com.pearadmin.modules.sys.service.ISysPowerService;

import io.swagger.annotations.Api;

import org.apache.logging.log4j.util.Strings;

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.ui.Model;

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

import org.springframework.web.servlet.ModelAndView;

import javax.annotation.Resource;

import java.util.List;

/\*\*

\* Describe: 权 限 控 制 器

\* Author: 就 眠 仪 式

\* CreateTime: 2019/10/23

\* \*/

@RestController

@Api(tags = {"系统权限"})

@RequestMapping(ControllerConstant.API\_SYSTEM\_PREFIX + "power")

public class SysPowerController extends BaseController {

/\*\*

\* Describe: 基础路径

\* \*/

private static String MODULE\_PATH = "system/power/";

/\*\*

\* Describe: 权限模块服务

\* \*/

@Resource

private ISysPowerService sysPowerService;

/\*\*

\* Describe: 获取权限列表视图

\* Param ModelAndView

\* Return 权限列表视图

\* \*/

@GetMapping("main")

@PreAuthorize("hasPermission('/system/power/main','sys:power:main')")

public ModelAndView main(ModelAndView modelAndView){

return jumpPage(MODULE\_PATH + "main");

}

/\*\*

\* Describe: 获取权限列表数据

\* Param ModelAndView

\* Return 权限列表数据

\* \*/

@GetMapping("data")

@PreAuthorize("hasPermission('/system/power/data','sys:power:data')")

public ResultTable data(SysPower sysPower){

return treeTable(sysPowerService.list(sysPower));

}

/\*\*

\* Describe: 获取权限新增视图

\* Param ModelAndView

\* Return 权限新增视图

\* \*/

@GetMapping("add")

@PreAuthorize("hasPermission('/system/power/add','sys:power:add')")

public ModelAndView add(){

return jumpPage(MODULE\_PATH + "add");

}

/\*\*

\* Describe: 获取权限修改视图

\* Param ModelAndView

\* Return 权限修改视图

\* \*/

@GetMapping("edit")

@PreAuthorize("hasPermission('/system/power/edit','sys:power:edit')")

public ModelAndView edit(Model model, String powerId){

model.addAttribute("sysPower",sysPowerService.getById(powerId));

return jumpPage(MODULE\_PATH + "edit");

}

/\*\*

\* Describe: 保存权限信息

\* Param: SysPower

\* Return: ResuBean

\* \*/

@PostMapping("save")

@PreAuthorize("hasPermission('/system/power/add','sys:power:add')")

public Result save(@RequestBody SysPower sysPower){

if(Strings.isBlank(sysPower.getParentId())){

return failure("请选择上级菜单");

}

sysPower.setPowerId(SequenceUtil.makeStringId());

boolean result = sysPowerService.save(sysPower);

return decide(result);

}

/\*\*

\* Describe: 修改权限信息

\* Param SysPower

\* Return 执行结果

\* \*/

@PutMapping("update")

@PreAuthorize("hasPermission('/system/power/edit','sys:power:edit')")

public Result update(@RequestBody SysPower sysPower){

if(Strings.isBlank(sysPower.getParentId())){

return failure("请选择上级菜单");

}

boolean result = sysPowerService.update(sysPower);

return decide(result);

}

/\*\*

\* Describe: 根据 id 进行删除

\* Param id

\* Return ResuTree

\* \*/

@DeleteMapping("remove/{id}")

@PreAuthorize("hasPermission('/system/power/remove','sys:power:remove')")

public Result remove(@PathVariable String id){

boolean result = sysPowerService.remove(id);

return decide(result);

}

/\*\*

\* Describe: 获取父级权限选择数据

\* Param sysPower

\* Return ResuTree

\* \*/

@GetMapping("selectParent")

public ResultTree selectParent(SysPower sysPower){

List<SysPower> list = sysPowerService.list(sysPower);

SysPower basePower = new SysPower();

basePower.setPowerName("顶级权限");

basePower.setPowerId("0");

basePower.setParentId("-1");

list.add(basePower);

return dataTree(list);

}

/\*\*

\* Describe: 根据 Id 开启用户

\* Param powerId

\* Return ResuTree

\* \*/

@PutMapping("enable")

public Result enable(@RequestBody SysPower sysPower){

sysPower.setEnable(true);

boolean result = sysPowerService.update(sysPower);

return decide(result);

}

/\*\*

\* Describe: 根据 Id 禁用用户

\* Param powerId

\* Return ResuTree

\* \*/

@PutMapping("disable")

public Result disable(@RequestBody SysPower sysPower){

sysPower.setEnable(false);

boolean result = sysPowerService.update(sysPower);

return decide(result);

}

}

package com.pearadmin.modules.sys.controller;

import com.github.pagehelper.PageInfo;

import com.pearadmin.common.constant.ControllerConstant;

import com.pearadmin.common.tools.sequence.SequenceUtil;

import com.pearadmin.common.web.base.BaseController;

import com.pearadmin.common.web.domain.request.PageDomain;

import com.pearadmin.common.web.domain.response.module.ResultTree;

import com.pearadmin.common.web.domain.response.Result;

import com.pearadmin.common.web.domain.response.module.ResultTable;

import com.pearadmin.modules.sys.domain.SysRole;

import com.pearadmin.modules.sys.service.ISysRoleService;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.ui.Model;

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

import org.springframework.web.servlet.ModelAndView;

import javax.annotation.Resource;

import java.util.Arrays;

/\*\*

\* Describe: 角 色 控 制 器

\* Author: 就 眠 仪 式

\* CreateTime: 2019/10/23

\* \*/

@RestController

@Api(tags = {"系统角色"})

@RequestMapping(ControllerConstant.API\_SYSTEM\_PREFIX + "role")

public class SysRoleController extends BaseController {

/\*\*

\* Describe: 基础路径

\* \*/

private static String MODULE\_PATH = "system/role/";

/\*\*

\* Describe: 角色模块服务

\* \*/

@Resource

private ISysRoleService sysRoleService;

/\*\*

\* Describe: 获取角色列表视图

\* Param ModelAndView

\* Return 用户列表视图

\* \*/

@GetMapping("main")

@ApiOperation(value="获取角色列表视图")

@PreAuthorize("hasPermission('/system/role/main','sys:role:main')")

public ModelAndView main(){

return jumpPage(MODULE\_PATH + "main");

}

/\*\*

\* Describe: 获取角色列表数据

\* Param SysRole PageDomain

\* Return 角色列表数据

\* \*/

@GetMapping("data")

@ApiOperation(value="获取角色列表数据")

@PreAuthorize("hasPermission('/system/role/data','sys:role:data')")

public ResultTable data(PageDomain pageDomain, SysRole param){

PageInfo<SysRole> pageInfo = sysRoleService.page(param,pageDomain);

return pageTable(pageInfo.getList(),pageInfo.getTotal());

}

/\*\*

\* Describe: 获取角色新增视图

\* Param ModelAndView

\* Return 角色新增视图

\* \*/

@GetMapping("add")

@ApiOperation(value="获取角色新增视图")

@PreAuthorize("hasPermission('/system/role/add','sys:role:add')")

public ModelAndView add(){

return jumpPage(MODULE\_PATH + "add");

}

/\*\*

\* Describe: 保存角色信息

\* Param SysRole

\* Return 执行结果

\* \*/

@PostMapping("save")

@ApiOperation(value="保存角色数据")

@PreAuthorize("hasPermission('/system/role/add','sys:role:add')")

public Result save(@RequestBody SysRole sysRole){

sysRole.setRoleId(SequenceUtil.makeStringId());

boolean result = sysRoleService.save(sysRole);

return decide(result);

}

/\*\*

\* Describe: 获取角色修改视图

\* Param ModelAndView

\* Return 角色修改视图

\* \*/

@GetMapping("edit")

@ApiOperation(value="获取角色修改视图")

@PreAuthorize("hasPermission('/system/role/edit','sys:role:edit')")

public ModelAndView edit(ModelAndView modelAndView,String roleId){

modelAndView.addObject("sysRole",sysRoleService.getById(roleId));

modelAndView.setViewName(MODULE\_PATH + "edit");

return modelAndView;

}

/\*\*

\* Describe: 修改角色信息

\* Param SysRole

\* Return 执行结果

\* \*/

@PutMapping("update")

@ApiOperation(value="修改角色数据")

@PreAuthorize("hasPermission('/system/role/edit','sys:role:edit')")

public Result update(@RequestBody SysRole sysRole){

boolean result = sysRoleService.update(sysRole);

return decide(result);

}

/\*\*

\* Describe: 获取角色授权视图

\* Param ModelAndView

\* Return ModelAndView

\* \*/

@GetMapping("power")

@ApiOperation(value="获取分配角色权限视图")

@PreAuthorize("hasPermission('/system/role/power','sys:role:power')")

public ModelAndView power(Model model, String roleId){

model.addAttribute("roleId",roleId);

return jumpPage(MODULE\_PATH + "power");

}

/\*\*

\* Describe: 保存角色权限

\* Param RoleId PowerIds

\* Return ResuBean

\* \*/

@PutMapping("saveRolePower")

@ApiOperation(value="保存角色权限数据")

@PreAuthorize("hasPermission('/system/role/power','sys:role:power')")

public Result saveRolePower(String roleId, String powerIds){

boolean result = sysRoleService.saveRolePower(roleId, Arrays.asList(powerIds.split(",")));

return decide(result);

}

/\*\*

\* Describe: 获取角色权限

\* Param RoleId

\* Return ResuTree

\* \*/

@GetMapping("getRolePower")

@ApiOperation(value="获取角色权限数据")

@PreAuthorize("hasPermission('/system/role/power','sys:role:power')")

public ResultTree getRolePower(String roleId){

return dataTree(sysRoleService.getRolePower(roleId));

}

/\*\*

\* Describe: 用户删除接口

\* Param: id

\* Return: ResuBean

\* \*/

@DeleteMapping("remove/{id}")

@ApiOperation(value="删除角色数据")

@PreAuthorize("hasPermission('/system/role/remove','sys:role:remove')")

public Result remove(@PathVariable String id){

boolean result = sysRoleService.remove(id);

return decide(result);

}

/\*\*

\* Describe: 用户批量删除接口

\* Param: ids

\* Return: ResuBean

\* \*/

@DeleteMapping("batchRemove/{ids}")

@ApiOperation(value="批量删除角色数据")

@PreAuthorize("hasPermission('/system/role/remove','sys:role:remove')")

public Result batchRemove(@PathVariable String ids){

boolean result = sysRoleService.batchRemove(ids.split(","));

return decide(result);

}

/\*\*

\* Describe: 根据 Id 启用角色

\* Param: roleId

\* Return: ResuBean

\* \*/

@PutMapping("enable")

@ApiOperation(value="启用角色")

public Result enable(@RequestBody SysRole sysRole){

sysRole.setEnable("0");

boolean result = sysRoleService.update(sysRole);

return decide(result);

}

/\*\*

\* Describe: 根据 Id 禁用角色

\* Param: roleId

\* Return: ResuBean

\* \*/

@PutMapping("disable")

@ApiOperation(value="禁用角色")

public Result disable(@RequestBody SysRole sysRole){

sysRole.setEnable("1");

boolean result = sysRoleService.update(sysRole);

return decide(result);

}

}

package com.pearadmin.modules.sys.controller;

import cn.hutool.core.map.MapUtil;

import com.pearadmin.common.constant.ConfigurationConstant;

import com.pearadmin.common.tools.sequence.SequenceUtil;

import com.pearadmin.modules.sys.domain.SysConfig;

import com.pearadmin.common.listener.event.SetupEvent;

import io.swagger.annotations.Api;

import org.springframework.context.ApplicationEventPublisher;

import org.springframework.context.ApplicationEventPublisherAware;

import org.springframework.transaction.annotation.Transactional;

import org.springframework.ui.Model;

import com.pearadmin.common.constant.ControllerConstant;

import com.pearadmin.common.web.base.BaseController;

import com.pearadmin.common.web.domain.response.Result;

import com.pearadmin.modules.sys.domain.SysSetup;

import com.pearadmin.modules.sys.service.ISysConfigService;

import org.springframework.security.access.prepost.PreAuthorize;

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

import org.springframework.web.servlet.ModelAndView;

import javax.annotation.Resource;

import java.util.HashMap;

/\*\*

\* Describe: 设 置 控 制 器

\* Author: 就 眠 仪 式

\* CreateTime: 2019/10/23

\*/

@RestController

@Api(tags = {"系统设置"})

@RequestMapping(ControllerConstant.API\_SYSTEM\_PREFIX + "setup")

public class SysSetupController extends BaseController implements ApplicationEventPublisherAware {

private static String MODULE\_PATH = "system/setup/";

private ApplicationEventPublisher applicationEventPublisher;

@Resource

private ISysConfigService sysConfigService;

@GetMapping("main")

@PreAuthorize("hasPermission('/system/setup/main','sys:setup:main')")

public ModelAndView main(Model model) {

SysSetup sysSetup = new SysSetup();

SysConfig mailFromConfig = sysConfigService.getByCode(ConfigurationConstant.MAIN\_FROM);

SysConfig mailUserConfig = sysConfigService.getByCode(ConfigurationConstant.MAIN\_USER);

SysConfig mailPassConfig = sysConfigService.getByCode(ConfigurationConstant.MAIN\_PASS);

SysConfig mailHostConfig = sysConfigService.getByCode(ConfigurationConstant.MAIN\_HOST);

SysConfig mailPortConfig = sysConfigService.getByCode(ConfigurationConstant.MAIN\_PORT);

sysSetup.setMailFrom(mailFromConfig == null ? "" : mailFromConfig.getConfigValue());

sysSetup.setMailUser(mailUserConfig == null ? "" : mailUserConfig.getConfigValue());

sysSetup.setMailPass(mailPassConfig == null ? "" : mailPassConfig.getConfigValue());

sysSetup.setMailHost(mailHostConfig == null ? "" : mailHostConfig.getConfigValue());

sysSetup.setMailPort(mailPortConfig == null ? "" : mailPortConfig.getConfigValue());

SysConfig uploadKindConfig = sysConfigService.getByCode(ConfigurationConstant.UPLOAD\_KIND);

SysConfig uploadPathConfig = sysConfigService.getByCode(ConfigurationConstant.UPLOAD\_PATH);

sysSetup.setUploadKind(uploadKindConfig == null ? "" : uploadKindConfig.getConfigValue());

sysSetup.setUploadPath(uploadPathConfig == null ? "" : uploadPathConfig.getConfigValue());

model.addAttribute("setup", sysSetup);

return jumpPage(MODULE\_PATH + "main");

}

@Transactional

@PutMapping("save")

@PreAuthorize("hasPermission('/system/setup/add','sys:setup:add')")

public Result save(@RequestBody SysSetup sysSetup) {

String from = sysSetup.getMailFrom();

String user = sysSetup.getMailUser();

String pass = sysSetup.getMailPass();

String port = sysSetup.getMailPort();

String host = sysSetup.getMailHost();

String uploadKind = sysSetup.getUploadKind();

String uploadPath = sysSetup.getUploadPath();

updateSetup("邮箱来源", ConfigurationConstant.MAIN\_FROM, from);

updateSetup("邮箱用户", ConfigurationConstant.MAIN\_USER, user);

updateSetup("邮箱密码", ConfigurationConstant.MAIN\_PASS, pass);

updateSetup("邮箱端口", ConfigurationConstant.MAIN\_PORT, port);

updateSetup("邮箱主机", ConfigurationConstant.MAIN\_HOST, host);

updateSetup("上传方式", ConfigurationConstant.UPLOAD\_KIND, uploadKind);

updateSetup("上传路径", ConfigurationConstant.UPLOAD\_PATH, uploadPath);

HashMap<String, String> map = MapUtil.newHashMap(5);

map.put(ConfigurationConstant.MAIN\_FROM, from);

map.put(ConfigurationConstant.MAIN\_USER, user);

map.put(ConfigurationConstant.MAIN\_PASS, pass);

map.put(ConfigurationConstant.MAIN\_PORT, port);

map.put(ConfigurationConstant.MAIN\_HOST, host);

SetupEvent setupEvent = new SetupEvent(this, map);

applicationEventPublisher.publishEvent(setupEvent);

return success("保存成功");

}

private void updateSetup(String name, String code, String value) {

SysConfig config = sysConfigService.getByCode(code);

if (config != null) {

config.setConfigValue(value);

sysConfigService.updateById(config);

} else {

config = new SysConfig();

config.setConfigId(SequenceUtil.makeStringId());

config.setConfigName(name);

config.setConfigCode(code);

config.setConfigType("system");

config.setConfigValue(value);

sysConfigService.save(config);

}

}

@Override

public void setApplicationEventPublisher(ApplicationEventPublisher applicationEventPublisher) {

this.applicationEventPublisher = applicationEventPublisher;

}

}

package com.pearadmin.modules.sys.controller;

import com.github.pagehelper.PageInfo;

import com.pearadmin.common.constant.ControllerConstant;

import com.pearadmin.common.plugin.logging.aop.annotation.Logging;

import com.pearadmin.common.plugin.logging.aop.enums.BusinessType;

import com.pearadmin.modules.sys.service.ISysLogService;

import com.pearadmin.common.plugin.submit.annotation.RepeatSubmit;

import com.pearadmin.common.tools.secure.SecurityUtil;

import com.pearadmin.common.tools.sequence.SequenceUtil;

import com.pearadmin.common.tools.servlet.ServletUtil;

import com.pearadmin.common.web.base.BaseController;

import com.pearadmin.common.web.domain.request.PageDomain;

import com.pearadmin.common.web.domain.response.Result;

import com.pearadmin.common.web.domain.response.module.ResultTable;

import com.pearadmin.modules.sys.domain.SysUser;

import com.pearadmin.modules.sys.domain.SysMenu;

import com.pearadmin.modules.sys.param.EditPassword;

import com.pearadmin.modules.sys.service.ISysRoleService;

import com.pearadmin.modules.sys.service.ISysUserService;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.apache.logging.log4j.util.Strings;

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.transaction.annotation.Transactional;

import org.springframework.ui.Model;

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

import org.springframework.web.servlet.ModelAndView;

import javax.annotation.Resource;

import java.time.LocalDateTime;

import java.util.Arrays;

import java.util.List;

/\*\*

\* Describe: 用户控制器

\* Author: 就 眠 仪 式

\* CreateTime: 2019/10/23

\*/

@RestController

@Api(tags = {"用户管理"})

@RequestMapping(ControllerConstant.API\_SYSTEM\_PREFIX + "user")

public class SysUserController extends BaseController {

/\*\*

\* Describe: 基础路径

\*/

private static String MODULE\_PATH = "system/user/";

/\*\*

\* Describe: 用户模块服务

\*/

@Resource

private ISysUserService sysUserService;

/\*\*

\* Describe: 角色模块服务

\*/

@Resource

private ISysRoleService sysRoleService;

/\*\*

\* Describe: 日志模块服务

\*/

@Resource

private ISysLogService sysLogService;

/\*\*

\* Describe: 获取用户列表视图

\* Param ModelAndView

\* Return 用户列表视图

\*/

@GetMapping("main")

@ApiOperation(value = "获取用户列表视图")

@PreAuthorize("hasPermission('/system/user/main','sys:user:main')")

public ModelAndView main() {

return jumpPage(MODULE\_PATH + "main");

}

/\*\*

\* Describe: 获取用户列表数据

\* Param ModelAndView

\* Return 用户列表数据

\*/

@GetMapping("data")

@ApiOperation(value = "获取用户列表数据")

@PreAuthorize("hasPermission('/system/user/data','sys:user:data')")

@Logging(title = "查询用户", describe = "查询用户", type = BusinessType.QUERY)

public ResultTable data(PageDomain pageDomain, SysUser param) {

PageInfo<SysUser> pageInfo = sysUserService.page(param, pageDomain);

return pageTable(pageInfo.getList(), pageInfo.getTotal());

}

/\*\*

\* Describe: 用户新增视图

\* Param ModelAndView

\* Return 返回用户新增视图

\*/

@GetMapping("add")

@ApiOperation(value = "获取用户新增视图")

@PreAuthorize("hasPermission('/system/user/add','sys:user:add')")

public ModelAndView add(Model model) {

model.addAttribute("sysRoles", sysRoleService.list(null));

return jumpPage(MODULE\_PATH + "add");

}

/\*\*

\* Describe: 用户新增接口

\* Param ModelAndView

\* Return 操作结果

\*/

@RepeatSubmit

@PostMapping("save")

@ApiOperation(value = "保存用户数据")

@PreAuthorize("hasPermission('/system/user/add','sys:user:add')")

@Logging(title = "新增用户", describe = "新增用户", type = BusinessType.ADD)

public Result save(@RequestBody SysUser sysUser) {

sysUser.setLogin("0");

sysUser.setEnable("1");

sysUser.setStatus("1");

sysUser.setUserId(SequenceUtil.makeStringId());

sysUser.setCreateTime(LocalDateTime.now());

sysUser.setPassword(new BCryptPasswordEncoder().encode(sysUser.getPassword()));

sysUserService.saveUserRole(sysUser.getUserId(), Arrays.asList(sysUser.getRoleIds().split(",")));

Boolean result = sysUserService.save(sysUser);

return decide(result);

}

/\*\*

\* Describe: 用户修改视图

\* Param ModelAndView

\* Return 返回用户修改视图

\*/

@GetMapping("edit")

@ApiOperation(value = "获取用户修改视图")

@PreAuthorize("hasPermission('/system/user/edit','sys:user:edit')")

public ModelAndView edit(Model model, String userId) {

model.addAttribute("sysRoles", sysUserService.getUserRole(userId));

model.addAttribute("sysUser", sysUserService.getById(userId));

return jumpPage(MODULE\_PATH + "edit");

}

/\*\*

\* Describe: 用户密码修改视图

\* Param ModelAndView

\* Return 返回用户密码修改视图

\*/

@GetMapping("editPassword")

public ModelAndView editPasswordView() {

return jumpPage(MODULE\_PATH + "password");

}

/\*\*

\* Describe: 用户密码修改接口

\* Param editPassword

\* Return: Result

\*/

@PutMapping("editPassword")

public Result editPassword(@RequestBody EditPassword editPassword) {

String oldPassword = editPassword.getOldPassword();

String newPassword = editPassword.getNewPassword();

String confirmPassword = editPassword.getConfirmPassword();

SysUser sysUser = (SysUser) ServletUtil.getSession().getAttribute("currentUser");

SysUser editUser = sysUserService.getById(sysUser.getUserId());

if (Strings.isBlank(confirmPassword)

|| Strings.isBlank(newPassword)

|| Strings.isBlank(oldPassword)) {

return failure("输入不能为空");

}

if (!new BCryptPasswordEncoder().matches(oldPassword, editUser.getPassword())) {

return failure("密码验证失败");

}

if (!newPassword.equals(confirmPassword)) {

return failure("两次密码输入不一致");

}

editUser.setPassword(new BCryptPasswordEncoder().encode(newPassword));

boolean result = sysUserService.update(editUser);

return decide(result, "修改成功", "修改失败");

}

/\*\*

\* Describe: 用户修改接口

\* Param ModelAndView

\* Return 返回用户修改接口

\*/

@PutMapping("update")

@ApiOperation(value = "修改用户数据")

@PreAuthorize("hasPermission('/system/user/edit','sys:user:edit')")

@Logging(title = "修改用户", describe = "修改用户", type = BusinessType.EDIT)

public Result update(@RequestBody SysUser sysUser) {

sysUserService.saveUserRole(sysUser.getUserId(), Arrays.asList(sysUser.getRoleIds().split(",")));

boolean result = sysUserService.update(sysUser);

return decide(result);

}

/\*\*

\* Describe: 头像修改接口

\* Param: SysUser

\* Return: Result

\*/

@PutMapping("updateAvatar")

@ApiOperation(value = "修改用户头像")

@Logging(title = "修改头像", describe = "修改头像", type = BusinessType.EDIT)

public Result updateAvatar(@RequestBody SysUser sysUser) {

sysUser.setUserId(((SysUser)SecurityUtil.currentUserObj()).getUserId());

boolean result = sysUserService.update(sysUser);

return decide(result);

}

/\*\*

\* Describe: 用户批量删除接口

\* Param: ids

\* Return: Result

\*/

@DeleteMapping("batchRemove/{ids}")

@ApiOperation(value = "批量删除用户")

@PreAuthorize("hasPermission('/system/user/remove','sys:user:remove')")

@Logging(title = "删除用户", describe = "删除用户", type = BusinessType.REMOVE)

public Result batchRemove(@PathVariable String ids) {

boolean result = sysUserService.batchRemove(ids.split(","));

return decide(result);

}

/\*\*

\* Describe: 用户删除接口

\* Param: id

\* Return: Result

\*/

@Transactional

@DeleteMapping("remove/{id}")

@ApiOperation(value = "删除用户数据")

@PreAuthorize("hasPermission('/system/user/remove','sys:user:remove')")

@Logging(title = "删除用户", describe = "删除用户", type = BusinessType.REMOVE)

public Result remove(@PathVariable String id) {

boolean result = sysUserService.remove(id);

return decide(result);

}

/\*\*

\* Describe: 根据 username 获取菜单数据

\* Param SysRole

\* Return 执行结果

\*/

@GetMapping("menu")

@ApiOperation(value = "获取用户菜单数据")

public List<SysMenu> getUserMenu() {

SysUser sysUser = (SysUser)SecurityUtil.currentUserObj();

List<SysMenu> menus = sysUserService.getUserMenu(sysUser.getUsername());

return sysUserService.toUserMenu(menus, "0");

}

/\*\*

\* Describe: 根据 userId 开启用户

\* Param: SysUser

\* Return: 执行结果

\*/

@PutMapping("enable")

@ApiOperation(value = "开启用户登录")

public Result enable(@RequestBody SysUser sysUser) {

sysUser.setEnable("1");

boolean result = sysUserService.update(sysUser);

return decide(result);

}

/\*\*

\* Describe: 根据 userId 禁用用户

\* Param: SysUser

\* Return: 执行结果

\*/

@PutMapping("disable")

@ApiOperation(value = "禁用用户登录")

public Result disable(@RequestBody SysUser sysUser) {

sysUser.setEnable("0");

boolean result = sysUserService.update(sysUser);

return decide(result);

}

/\*\*

\* Describe: 个人资料

\* Param: null

\* Return: ModelAndView

\*/

@GetMapping("center")

@ApiOperation(value = "个人资料")

public ModelAndView center(Model model) {

SysUser sysUser = (SysUser) SecurityUtil.currentUserObj();

model.addAttribute("userInfo", sysUserService.getById(sysUser.getUserId()));

model.addAttribute("logs", sysLogService.selectTopLoginLog(sysUser.getUsername()));

return jumpPage(MODULE\_PATH + "center");

}

/\*\*

\* Describe: 用户修改接口

\* Param ModelAndView

\* Return 返回用户修改接口

\*/

@PutMapping("updateInfo")

@ApiOperation(value = "修改用户数据")

public Result updateInfo(@RequestBody SysUser sysUser) {

boolean result = sysUserService.update(sysUser);

return decide(result);

}

/\*\*

\* Describe: 更换头像

\* Param: null

\* Return: ModelAndView

\*/

@GetMapping("profile/{id}")

public ModelAndView profile(Model model,@PathVariable("id")String userId){

model.addAttribute("userId",userId);

return jumpPage(MODULE\_PATH + "profile");

}

}