import request from '@/utils/request'

// 登录方法

export function login(username, password, code, uuid) {

const data = {

username,

password,

code,

uuid

}

return request({

url: '/login',

headers: {

isToken: false

},

method: 'post',

data: data

})

}

// 注册方法

export function register(data) {

return request({

url: '/register',

headers: {

isToken: false

},

method: 'post',

data: data

})

}

// 获取用户详细信息

export function getInfo() {

return request({

url: '/getInfo',

method: 'get'

})

}

// 退出方法

export function logout() {

return request({

url: '/logout',

method: 'post'

})

}

// 获取验证码

export function getCodeImg() {

return request({

url: '/captchaImage',

headers: {

isToken: false

},

method: 'get',

timeout: 20000

})

}

import request from '@/utils/request'

// 获取路由

export const getRouters = () => {

return request({

url: '/getRouters',

method: 'get'

})

}package com.ruoyi.system.controller;

import com.ruoyi.common.annotation.Log;

import com.ruoyi.common.core.controller.BaseController;

import com.ruoyi.common.core.domain.AjaxResult;

import com.ruoyi.common.core.page.TableDataInfo;

import com.ruoyi.common.enums.BusinessType;

import com.ruoyi.common.utils.StringUtils;

import com.ruoyi.common.utils.poi.ExcelUtil;

import com.ruoyi.system.domain.Dto.ProfitDto;

import com.ruoyi.system.domain.Dto.ReportQueryDto;

import com.ruoyi.system.domain.Dto.voucherQueryDto;

import com.ruoyi.system.domain.Voucher;

import com.ruoyi.system.domain.VoucherTotals;

import com.ruoyi.system.mapper.VoucherMapper;

import com.ruoyi.system.service.IVoucherService;

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

import org.springframework.cglib.beans.BeanMap;

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

import javax.annotation.Resource;

import javax.servlet.http.HttpServletResponse;

import java.text.DecimalFormat;

import java.time.LocalDate;

import java.util.List;

import java.util.stream.Collectors;

/\*\*

\* 凭证Controller

\*

\*/

@RestController

@RequestMapping("/system/voucher")

public class VoucherController extends BaseController {

@Autowired

private IVoucherService voucherService;

@Resource

private VoucherMapper voucherMapper;

/\*\*

\* 查询凭证列表

\*/

@GetMapping("/list")

public TableDataInfo list(Voucher voucher) {

startPage();

ProfitDto profitDto = null;

List<Voucher> list = voucherService.selectVoucherList(voucher);

if (voucher.getPayMethods() != null){

profitDto = voucherService.getYsYf(voucher);

}else {

profitDto = voucherService.getprofitData(voucher);

}

BeanMap beanMap = BeanMap.create(profitDto);

return getDataTable(list, beanMap);

}

/\*\*

\* 导出凭证列表

\*/

@Log(title = "凭证", businessType = BusinessType.EXPORT)

@PostMapping("/export")

public void export(HttpServletResponse response, Voucher voucher) {

List<Voucher> list = voucherService.selectVoucherList(voucher);

ExcelUtil<Voucher> util = new ExcelUtil<Voucher>(Voucher.class);

util.exportExcel(response, list, "凭证数据");

}

/\*\*

\* 获取凭证详细信息

\*/

@GetMapping(value = "/{voucherId}")

public AjaxResult getInfo(@PathVariable("voucherId") Long voucherId) {

return success(voucherService.selectVoucherByVoucherId(voucherId));

}

/\*\*

\* 新增凭证

\*/

@Log(title = "凭证", businessType = BusinessType.INSERT)

@PostMapping

public AjaxResult add(@RequestBody Voucher voucher) {

return toAjax(voucherService.insertVoucher(voucher));

}

/\*\*

\* 修改凭证

\*/

@Log(title = "凭证", businessType = BusinessType.UPDATE)

@PutMapping

public AjaxResult edit(@RequestBody Voucher voucher) {

return toAjax(voucherService.updateVoucher(voucher));

}

/\*\*

\* 删除凭证

\*/

@Log(title = "凭证", businessType = BusinessType.DELETE)

@DeleteMapping("/{voucherIds}")

public AjaxResult remove(@PathVariable Long[] voucherIds) {

return toAjax(voucherService.deleteVoucherByVoucherIds(voucherIds));

}

/\*\*

\* 根据传入日期查询对应月份的收支情况

\* @param date

\* @return

\*/

@PostMapping("/getallbytime")

public AjaxResult getInfoByTime(@RequestBody ReportQueryDto reportQueryDto){

return success(voucherService.selectVouchersByTime(reportQueryDto));

}

@PostMapping("/getvouchertotal")

public TableDataInfo getVoucherTotal(@RequestBody voucherQueryDto voucherQueryDto){

if (StringUtils.isNotEmpty(voucherQueryDto.getMonth()) && StringUtils.isEmpty(voucherQueryDto.getYear())){

voucherQueryDto.setYear(String.valueOf(LocalDate.now().getYear()));

}

List<VoucherTotals> voucherTotals = voucherService.getVoucherTotals(voucherQueryDto);

voucherQueryDto.setVoucherType(null);

List<VoucherTotals> TotalCountList = voucherService.getVoucherTotals(voucherQueryDto);

double total = TotalCountList.stream().mapToDouble(VoucherTotals::getMoney).sum();

DecimalFormat df = new DecimalFormat("#.##");

List<VoucherTotals> voucherTotalsList = voucherTotals.stream().map(n -> {

n.setPersent(Double.valueOf(df.format(n.getMoney() / total \* 100)));

return n;

}).collect(Collectors.toList());

return getDataTable(voucherTotalsList);

}

}

package com.ruoyi.system.controller;

import com.ruoyi.common.annotation.Log;

import com.ruoyi.common.core.controller.BaseController;

import com.ruoyi.common.core.domain.AjaxResult;

import com.ruoyi.common.core.page.TableDataInfo;

import com.ruoyi.common.enums.BusinessType;

import com.ruoyi.common.utils.poi.ExcelUtil;

import com.ruoyi.system.domain.Budget;

import com.ruoyi.system.service.IBudgetService;

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

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

import javax.servlet.http.HttpServletResponse;

import java.util.List;

/\*\*

\* 预算管理Controller

\*

\*/

@RestController

@RequestMapping("/system/budget")

public class BudgetController extends BaseController

{

@Autowired

private IBudgetService budgetService;

/\*\*

\* 查询预算管理列表

\*/

@GetMapping("/list")

public TableDataInfo list(Budget budget)

{

startPage();

List<Budget> list = budgetService.selectBudgetList(budget);

return getDataTable(list);

}

/\*\*

\* 导出预算管理列表

\*/

@Log(title = "预算管理", businessType = BusinessType.EXPORT)

@PostMapping("/export")

public void export(HttpServletResponse response, Budget budget)

{

List<Budget> list = budgetService.selectBudgetList(budget);

ExcelUtil<Budget> util = new ExcelUtil<Budget>(Budget.class);

util.exportExcel(response, list, "预算管理数据");

}

/\*\*

\* 获取预算管理详细信息

\*/

@GetMapping(value = "/{projiectId}")

public AjaxResult getInfo(@PathVariable("projiectId") Long projiectId)

{

return success(budgetService.selectBudgetByProjiectId(projiectId));

}

/\*\*

\* 新增预算管理

\*/

@Log(title = "预算管理", businessType = BusinessType.INSERT)

@PostMapping

public AjaxResult add(@RequestBody Budget budget)

{

return toAjax(budgetService.insertBudget(budget));

}

/\*\*

\* 修改预算管理

\*/

@Log(title = "预算管理", businessType = BusinessType.UPDATE)

@PutMapping

public AjaxResult edit(@RequestBody Budget budget)

{

return toAjax(budgetService.updateBudget(budget));

}

/\*\*

\* 删除预算管理

\*/

@Log(title = "预算管理", businessType = BusinessType.DELETE)

@DeleteMapping("/{projiectIds}")

public AjaxResult remove(@PathVariable Long[] projiectIds)

{

return toAjax(budgetService.deleteBudgetByProjiectIds(projiectIds));

}

}

package com.ruoyi.system.controller;

import com.ruoyi.common.annotation.Log;

import com.ruoyi.common.core.controller.BaseController;

import com.ruoyi.common.core.domain.AjaxResult;

import com.ruoyi.common.core.page.TableDataInfo;

import com.ruoyi.common.enums.BusinessType;

import com.ruoyi.common.utils.poi.ExcelUtil;

import com.ruoyi.system.domain.Operations;

import com.ruoyi.system.service.IOperationsService;

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

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

import javax.servlet.http.HttpServletResponse;

import java.util.List;

/\*\*

\* 工资操作记录Controller

\*

\*/

@RestController

@RequestMapping("/system/operations")

public class OperationsController extends BaseController {

@Autowired

private IOperationsService operationsService;

/\*\*

\* 查询工资操作记录列表

\*/

@GetMapping("/list")

public TableDataInfo list(Operations operations) {

startPage();

List<Operations> list = operationsService.selectOperationsList(operations);

return getDataTable(list);

}

/\*\*

\* 导出工资操作记录列表

\*/

@Log(title = "工资操作记录", businessType = BusinessType.EXPORT)

@PostMapping("/export")

public void export(HttpServletResponse response, Operations operations) {

List<Operations> list = operationsService.selectOperationsList(operations);

ExcelUtil<Operations> util = new ExcelUtil<Operations>(Operations.class);

util.exportExcel(response, list, "工资操作记录数据");

}

/\*\*

\* 获取工资操作记录详细信息

\*/

@GetMapping(value = "/{arctId}")

public AjaxResult getInfo(@PathVariable("arctId") Long arctId) {

return success(operationsService.selectOperationsByArctId(arctId));

}

/\*\*

\* 新增工资操作记录

\*/

@Log(title = "工资操作记录", businessType = BusinessType.INSERT)

@PostMapping

public AjaxResult add(@RequestBody Operations operations) {

return toAjax(operationsService.insertOperations(operations));

}

/\*\*

\* 修改工资操作记录

\*/

@Log(title = "工资操作记录", businessType = BusinessType.UPDATE)

@PutMapping

public AjaxResult edit(@RequestBody Operations operations) {

return toAjax(operationsService.updateOperations(operations));

}

/\*\*

\* 删除工资操作记录

\*/

@Log(title = "工资操作记录", businessType = BusinessType.DELETE)

@DeleteMapping("/{arctIds}")

public AjaxResult remove(@PathVariable Long[] arctIds) {

return toAjax(operationsService.deleteOperationsByArctIds(arctIds));

}

}

package com.ruoyi.system.domain;

import lombok.Data;

import org.apache.commons.lang3.builder.ToStringBuilder;

import org.apache.commons.lang3.builder.ToStringStyle;

import com.ruoyi.common.annotation.Excel;

import com.ruoyi.common.core.domain.BaseEntity;

/\*\*

\* 预算管理对象 budget

\*

\*/

@Data

public class Budget extends BaseEntity

{

private static final long serialVersionUID = 1L;

/\*\* 项目id \*/

private Long projiectId;

/\*\* 项目名称 \*/

@Excel(name = "项目名称")

private String projectName;

/\*\* 计划预算 \*/

@Excel(name = "计划预算")

private Long budget;

//已使用预算

private Double Rebudget;

//剩余预算

private Double Lsbudget;

/\*\* 状态 \*/

@Excel(name = "状态")

private Integer status;

public void setProjiectId(Long projiectId)

{

this.projiectId = projiectId;

}

public Long getProjiectId()

{

return projiectId;

}

public void setProjectName(String projectName)

{

this.projectName = projectName;

}

public String getProjectName()

{

return projectName;

}

public void setBudget(Long budget)

{

this.budget = budget;

}

public Long getBudget()

{

return budget;

}

public void setStatus(Integer status)

{

this.status = status;

}

public Integer getStatus()

{

return status;

}

@Override

public String toString() {

return new ToStringBuilder(this,ToStringStyle.MULTI\_LINE\_STYLE)

.append("projiectId", getProjiectId())

.append("projectName", getProjectName())

.append("budget", getBudget())

.append("status", getStatus())

.toString();

}

}

package com.ruoyi.system.domain;

import org.apache.commons.lang3.builder.ToStringBuilder;

import org.apache.commons.lang3.builder.ToStringStyle;

import com.ruoyi.common.annotation.Excel;

import com.ruoyi.common.core.domain.BaseEntity;

/\*\*

\* 工资操作记录对象 operations

\*

\*/

public class Operations extends BaseEntity {

private static final long serialVersionUID = 1L;

/\*\*

\* 操作Id

\*/

private Long arctId;

/\*\*

\* 员工工号(被操作人)

\*/

@Excel(name = "员工工号(被操作人)")

private Long empId;

/\*\*

\* 员工姓名

\*/

@Excel(name = "员工姓名")

private String empName;

/\*\*

\* 部门Id

\*/

@Excel(name = "部门Id")

private Long deptId;

/\*\*

\* 部门名称

\*/

@Excel(name = "部门名称")

private String deptName;

/\*\*

\* 操作类型

\*/

@Excel(name = "操作类型")

private Long arctTtpe;

/\*\*

\* 变动金额

\*/

@Excel(name = "变动金额")

private Long changeMoney;

/\*\*

\* 操作人id

\*/

@Excel(name = "操作人id")

private Long operaterId;

/\*\*

\* 操作人姓名

\*/

@Excel(name = "操作人姓名")

private String operaterName;

/\*\*

\* 变动原因

\*/

@Excel(name = "变动原因")

private String detail;

public void setArctId(Long arctId) {

this.arctId = arctId;

}

public Long getArctId() {

return arctId;

}

public void setEmpId(Long empId) {

this.empId = empId;

}

public Long getEmpId() {

return empId;

}

public void setEmpName(String empName) {

this.empName = empName;

}

public String getEmpName() {

return empName;

}

public void setDeptId(Long deptId) {

this.deptId = deptId;

}

public Long getDeptId() {

return deptId;

}

public void setDeptName(String deptName) {

this.deptName = deptName;

}

public String getDeptName() {

return deptName;

}

public void setArctTtpe(Long arctTtpe) {

this.arctTtpe = arctTtpe;

}

public Long getArctTtpe() {

return arctTtpe;

}

public void setChangeMoney(Long changeMoney) {

this.changeMoney = changeMoney;

}

public Long getChangeMoney() {

return changeMoney;

}

public void setOperaterId(Long operaterId) {

this.operaterId = operaterId;

}

public Long getOperaterId() {

return operaterId;

}

public void setOperaterName(String operaterName) {

this.operaterName = operaterName;

}

public String getOperaterName() {

return operaterName;

}

public void setDetail(String detail) {

this.detail = detail;

}

public String getDetail() {

return detail;

}

@Override

public String toString() {

return new ToStringBuilder(this, ToStringStyle.MULTI\_LINE\_STYLE)

.append("arctId", getArctId())

.append("empId", getEmpId())

.append("empName", getEmpName())

.append("deptId", getDeptId())

.append("deptName", getDeptName())

.append("arctTtpe", getArctTtpe())

.append("changeMoney", getChangeMoney())

.append("operaterId", getOperaterId())

.append("operaterName", getOperaterName())

.append("detail", getDetail())

.append("updateTime", getUpdateTime())

.toString();

}

}

package com.ruoyi.system.domain;

import org.apache.commons.lang3.builder.ToStringBuilder;

import org.apache.commons.lang3.builder.ToStringStyle;

import com.ruoyi.common.annotation.Excel;

import com.ruoyi.common.core.domain.BaseEntity;

import java.util.List;

/\*\*

\* 凭证对象 voucher

\*

\*/

public class Voucher extends BaseEntity

{

private static final long serialVersionUID = 1L;

/\*\* 凭证id \*/

private Long voucherId;

/\*\* 费用名称 \*/

@Excel(name = "费用名称")

private String voucherName;

/\*\* 科目类型 \*/

@Excel(name = "科目类型")

private Long voucherType;

/\*\* 收入还是支出 \*/

@Excel(name = "收入还是支出")

private Long voucherSZ;

/\*\* 金额 \*/

@Excel(name = "金额")

private Long money;

/\*\* 录入人id \*/

@Excel(name = "录入人id")

private Long inputId;

/\*\* 付款方式(字典值，现金，转账，付款，待付款，待收款) \*/

@Excel(name = "付款方式(字典值，现金，转账，付款，待付款，待收款)")

private Integer payMethod;

/\*\* 项目id \*/

@Excel(name = "项目id")

private String projectId;

/\*\* 备注 \*/

@Excel(name = "备注")

private String bz;

private String voucherImg;

private List<String> payMethods;

public void setVoucherId(Long voucherId)

{

this.voucherId = voucherId;

}

public Long getVoucherId()

{

return voucherId;

}

public void setVoucherName(String voucherName)

{

this.voucherName = voucherName;

}

public String getVoucherName()

{

return voucherName;

}

public void setVoucherType(Long voucherType)

{

this.voucherType = voucherType;

}

public Long getVoucherType()

{

return voucherType;

}

public void setVoucherSZ(Long voucherSZ)

{

this.voucherSZ = voucherSZ;

}

public Long getVoucherSZ()

{

return voucherSZ;

}

public void setMoney(Long money)

{

this.money = money;

}

public Long getMoney()

{

return money;

}

public void setInputId(Long inputId)

{

this.inputId = inputId;

}

public Long getInputId()

{

return inputId;

}

public void setPayMethod(Integer payMethod)

{

this.payMethod = payMethod;

}

public Integer getPayMethod()

{

return payMethod;

}

public void setProjectId(String projectId)

{

this.projectId = projectId;

}

public String getProjectId()

{

return projectId;

}

public void setBz(String bz)

{

this.bz = bz;

}

public String getBz()

{

return bz;

}

public List<String> getPayMethods() {

return payMethods;

}

public void setPayMethods(List<String> payMethods) {

this.payMethods = payMethods;

}

public String getVoucherImg() {

return voucherImg;

}

public void setVoucherImg(String voucherImg) {

this.voucherImg = voucherImg;

}

@Override

public String toString() {

return new ToStringBuilder(this,ToStringStyle.MULTI\_LINE\_STYLE)

.append("voucherId", getVoucherId())

.append("voucherName", getVoucherName())

.append("voucherType", getVoucherType())

.append("voucherSZ", getVoucherSZ())

.append("createTime", getCreateTime())

.append("updateTime", getUpdateTime())

.append("money", getMoney())

.append("inputId", getInputId())

.append("payMethod", getPayMethod())

.append("projectId", getProjectId())

.append("bz", getBz())

.toString();

}

}

package com.ruoyi.system.domain;

import lombok.Data;

import java.io.Serializable;

/\*\*

\* 科目汇总类

\* \*/

@Data

public class VoucherTotals implements Serializable {

private static final long serialVersionUID = 1L;

private Integer voucherType;

private String voucherName;

private Double money;

private Double persent;

public Integer getVoucherType() {

return voucherType;

}

public void setVoucherType(Integer voucherType) {

this.voucherType = voucherType;

}

public String getVoucherName() {

return voucherName;

}

public void setVoucherName(String voucherName) {

this.voucherName = voucherName;

}

public Double getMoney() {

return money;

}

public void setMoney(Double money) {

this.money = money;

}

@Override

public String toString() {

return "VoucherTotals{" +

"voucherType=" + voucherType +

", voucherName='" + voucherName + '\'' +

", money=" + money +

'}';

}

}

package com.ruoyi.system.mapper;

import java.util.List;

import com.ruoyi.system.domain.Budget;

/\*\*

\* 预算管理Mapper接口

\*

\*/

public interface BudgetMapper

{

/\*\*

\* 查询预算管理

\*

\* @param projiectId 预算管理主键

\* @return 预算管理

\*/

public Budget selectBudgetByProjiectId(Long projiectId);

/\*\*

\* 查询预算管理列表

\*

\* @param budget 预算管理

\* @return 预算管理集合

\*/

public List<Budget> selectBudgetList(Budget budget);

/\*\*

\* 新增预算管理

\*

\* @param budget 预算管理

\* @return 结果

\*/

public int insertBudget(Budget budget);

/\*\*

\* 修改预算管理

\*

\* @param budget 预算管理

\* @return 结果

\*/

public int updateBudget(Budget budget);

/\*\*

\* 删除预算管理

\*

\* @param projiectId 预算管理主键

\* @return 结果

\*/

public int deleteBudgetByProjiectId(Long projiectId);

/\*\*

\* 批量删除预算管理

\*

\* @param projiectIds 需要删除的数据主键集合

\* @return 结果

\*/

public int deleteBudgetByProjiectIds(Long[] projiectIds);

}

package com.ruoyi.system.mapper;

import java.util.List;

import com.ruoyi.system.domain.Operations;

/\*\*

\* 工资操作记录Mapper接口

\*

\*/

public interface OperationsMapper

{

/\*\*

\* 查询工资操作记录

\*

\* @param arctId 工资操作记录主键

\* @return 工资操作记录

\*/

public Operations selectOperationsByArctId(Long arctId);

/\*\*

\* 查询工资操作记录列表

\*

\* @param operations 工资操作记录

\* @return 工资操作记录集合

\*/

public List<Operations> selectOperationsList(Operations operations);

/\*\*

\* 新增工资操作记录

\*

\* @param operations 工资操作记录

\* @return 结果

\*/

public int insertOperations(Operations operations);

/\*\*

\* 修改工资操作记录

\*

\* @param operations 工资操作记录

\* @return 结果

\*/

public int updateOperations(Operations operations);

/\*\*

\* 删除工资操作记录

\*

\* @param arctId 工资操作记录主键

\* @return 结果

\*/

public int deleteOperationsByArctId(Long arctId);

/\*\*

\* 批量删除工资操作记录

\*

\* @param arctIds 需要删除的数据主键集合

\* @return 结果

\*/

public int deleteOperationsByArctIds(Long[] arctIds);

}

package com.ruoyi.system.mapper;

import com.ruoyi.system.domain.Dto.ReportQueryDto;

import com.ruoyi.system.domain.Dto.ResponsePie;

import com.ruoyi.system.domain.Dto.voucherQueryDto;

import com.ruoyi.system.domain.Voucher;

import com.ruoyi.system.domain.VoucherTotals;

import org.apache.ibatis.annotations.Param;

import java.util.List;

/\*\*

\* 凭证Mapper接口

\*

\*/

public interface VoucherMapper

{

/\*\*

\* 查询凭证

\*

\* @param voucherId 凭证主键

\* @return 凭证

\*/

public Voucher selectVoucherByVoucherId(Long voucherId);

/\*\*

\* 查询凭证列表

\*

\* @param voucher 凭证

\* @return 凭证集合

\*/

public List<Voucher> selectVoucherList(Voucher voucher);

/\*\*

\* 新增凭证

\*

\* @param voucher 凭证

\* @return 结果

\*/

public int insertVoucher(Voucher voucher);

/\*\*

\* 修改凭证

\*

\* @param voucher 凭证

\* @return 结果

\*/

public int updateVoucher(Voucher voucher);

/\*\*

\* 删除凭证

\*

\* @param voucherId 凭证主键

\* @return 结果

\*/

public int deleteVoucherByVoucherId(Long voucherId);

int deleteVoucherByProjectId(Long projectid);

int deleteVoucherByProjectIds(Long[] projectid);

/\*\*

\* 批量删除凭证

\*

\* @param voucherIds 需要删除的数据主键集合

\* @return 结果

\*/

public int deleteVoucherByVoucherIds(Long[] voucherIds);

public List<ResponsePie> selectVouchersByTime(@Param("reportQueryDto")ReportQueryDto reportQueryDto);

public List<VoucherTotals> selectTotalVoucher(@Param("voucherQueryDto")voucherQueryDto voucherQueryDto);

Double gettotalExpense(voucherQueryDto voucherQueryDto);

Double getProjetCost(@Param("projectId")Long projectId);

Double getcountData(Voucher voucher);

}

package com.ruoyi.system.service;

import java.util.List;

import com.ruoyi.system.domain.Budget;

/\*\*

\* 预算管理Service接口

\*

\*/

public interface IBudgetService

{

/\*\*

\* 查询预算管理

\*

\* @param projiectId 预算管理主键

\* @return 预算管理

\*/

public Budget selectBudgetByProjiectId(Long projiectId);

/\*\*

\* 查询预算管理列表

\*

\* @param budget 预算管理

\* @return 预算管理集合

\*/

public List<Budget> selectBudgetList(Budget budget);

/\*\*

\* 新增预算管理

\*

\* @param budget 预算管理

\* @return 结果

\*/

public int insertBudget(Budget budget);

/\*\*

\* 修改预算管理

\*

\* @param budget 预算管理

\* @return 结果

\*/

public int updateBudget(Budget budget);

/\*\*

\* 批量删除预算管理

\*

\* @param projiectIds 需要删除的预算管理主键集合

\* @return 结果

\*/

public int deleteBudgetByProjiectIds(Long[] projiectIds);

/\*\*

\* 删除预算管理信息

\*

\* @param projiectId 预算管理主键

\* @return 结果

\*/

public int deleteBudgetByProjiectId(Long projiectId);

}

package com.ruoyi.system.service;

import java.util.List;

import com.ruoyi.system.domain.Operations;

/\*\*

\* 工资操作记录Service接口

\*

\*/

public interface IOperationsService {

/\*\*

\* 查询工资操作记录

\*

\* @param arctId 工资操作记录主键

\* @return 工资操作记录

\*/

public Operations selectOperationsByArctId(Long arctId);

/\*\*

\* 查询工资操作记录列表

\*

\* @param operations 工资操作记录

\* @return 工资操作记录集合

\*/

public List<Operations> selectOperationsList(Operations operations);

/\*\*

\* 新增工资操作记录

\*

\* @param operations 工资操作记录

\* @return 结果

\*/

public int insertOperations(Operations operations);

/\*\*

\* 修改工资操作记录

\*

\* @param operations 工资操作记录

\* @return 结果

\*/

public int updateOperations(Operations operations);

/\*\*

\* 批量删除工资操作记录

\*

\* @param arctIds 需要删除的工资操作记录主键集合

\* @return 结果

\*/

public int deleteOperationsByArctIds(Long[] arctIds);

/\*\*

\* 删除工资操作记录信息

\*

\* @param arctId 工资操作记录主键

\* @return 结果

\*/

public int deleteOperationsByArctId(Long arctId);

}

package com.ruoyi.system.service;

import com.ruoyi.system.domain.Dto.ProfitDto;

import com.ruoyi.system.domain.Dto.ReportQueryDto;

import com.ruoyi.system.domain.Dto.ResponsePie;

import com.ruoyi.system.domain.Dto.voucherQueryDto;

import com.ruoyi.system.domain.Voucher;

import com.ruoyi.system.domain.VoucherTotals;

import java.util.List;

import java.util.Map;

/\*\*

\* 凭证Service接口

\*/

public interface IVoucherService

{

/\*\*

\* 查询凭证

\*

\* @param voucherId 凭证主键

\* @return 凭证

\*/

public Voucher selectVoucherByVoucherId(Long voucherId);

/\*\*

\* 查询凭证列表

\*

\* @param voucher 凭证

\* @return 凭证集合

\*/

public List<Voucher> selectVoucherList(Voucher voucher);

public ProfitDto getprofitData(Voucher voucher);

public ProfitDto getYsYf(Voucher voucher);

/\*\*

\* 新增凭证

\*

\* @param voucher 凭证

\* @return 结果

\*/

public int insertVoucher(Voucher voucher);

/\*\*

\* 修改凭证

\*

\* @param voucher 凭证

\* @return 结果

\*/

public int updateVoucher(Voucher voucher);

/\*\*

\* 批量删除凭证

\*

\* @param voucherIds 需要删除的凭证主键集合

\* @return 结果

\*/

public int deleteVoucherByVoucherIds(Long[] voucherIds);

/\*\*

\* 删除凭证信息

\*

\* @param voucherId 凭证主键

\* @return 结果

\*/

public int deleteVoucherByVoucherId(Long voucherId);

public Map<String, List<ResponsePie>> selectVouchersByTime(ReportQueryDto reportQueryDto);

public List<VoucherTotals> getVoucherTotals(voucherQueryDto voucherQueryDto);

}

package com.ruoyi.common.annotation;

import java.lang.annotation.Documented;

import java.lang.annotation.ElementType;

import java.lang.annotation.Retention;

import java.lang.annotation.RetentionPolicy;

import java.lang.annotation.Target;

/\*\*

\* 匿名访问不鉴权注解

\*

\*/

@Target({ ElementType.METHOD, ElementType.TYPE })

@Retention(RetentionPolicy.RUNTIME)

@Documented

public @interface Anonymous

{

}

package com.ruoyi.common.annotation;

import java.lang.annotation.Documented;

import java.lang.annotation.ElementType;

import java.lang.annotation.Retention;

import java.lang.annotation.RetentionPolicy;

import java.lang.annotation.Target;

/\*\*

\* 数据权限过滤注解

\*

\*/

@Target(ElementType.METHOD)

@Retention(RetentionPolicy.RUNTIME)

@Documented

public @interface DataScope

{

/\*\*

\* 部门表的别名

\*/

public String deptAlias() default "";

/\*\*

\* 用户表的别名

\*/

public String userAlias() default "";

/\*\*

\* 权限字符（用于多个角色匹配符合要求的权限）默认根据权限注解@ss获取，多个权限用逗号分隔开来

\*/

public String permission() default "";

}

package com.ruoyi.common.annotation;

import java.lang.annotation.Documented;

import java.lang.annotation.ElementType;

import java.lang.annotation.Inherited;

import java.lang.annotation.Retention;

import java.lang.annotation.RetentionPolicy;

import java.lang.annotation.Target;

import com.ruoyi.common.enums.DataSourceType;

/\*\*

\* 自定义多数据源切换注解

\*

\* 优先级：先方法，后类，如果方法覆盖了类上的数据源类型，以方法的为准，否则以类上的为准

\*/

@Target({ ElementType.METHOD, ElementType.TYPE })

@Retention(RetentionPolicy.RUNTIME)

@Documented

@Inherited

public @interface DataSource

{

/\*\*

\* 切换数据源名称

\*/

public DataSourceType value() default DataSourceType.MASTER;

}

package com.ruoyi.common.annotation;

import java.lang.annotation.ElementType;

import java.lang.annotation.Retention;

import java.lang.annotation.RetentionPolicy;

import java.lang.annotation.Target;

import java.math.BigDecimal;

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

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

import com.ruoyi.common.utils.poi.ExcelHandlerAdapter;

/\*\*

\* 自定义导出Excel数据注解

\*

\*/

@Retention(RetentionPolicy.RUNTIME)

@Target(ElementType.FIELD)

public @interface Excel

{

/\*\*

\* 导出时在excel中排序

\*/

public int sort() default Integer.MAX\_VALUE;

/\*\*

\* 导出到Excel中的名字.

\*/

public String name() default "";

/\*\*

\* 日期格式, 如: yyyy-MM-dd

\*/

public String dateFormat() default "";

/\*\*

\* 如果是字典类型，请设置字典的type值 (如: sys\_user\_sex)

\*/

public String dictType() default "";

/\*\*

\* 读取内容转表达式 (如: 0=男,1=女,2=未知)

\*/

public String readConverterExp() default "";

/\*\*

\* 分隔符，读取字符串组内容

\*/

public String separator() default ",";

/\*\*

\* BigDecimal 精度 默认:-1(默认不开启BigDecimal格式化)

\*/

public int scale() default -1;

/\*\*

\* BigDecimal 舍入规则 默认:BigDecimal.ROUND\_HALF\_EVEN

\*/

public int roundingMode() default BigDecimal.ROUND\_HALF\_EVEN;

/\*\*

\* 导出时在excel中每个列的高度 单位为字符

\*/

public double height() default 14;

/\*\*

\* 导出时在excel中每个列的宽 单位为字符

\*/

public double width() default 16;

/\*\*

\* 文字后缀,如% 90 变成90%

\*/

public String suffix() default "";

/\*\*

\* 当值为空时,字段的默认值

\*/

public String defaultValue() default "";

/\*\*

\* 提示信息

\*/

public String prompt() default "";

/\*\*

\* 设置只能选择不能输入的列内容.

\*/

public String[] combo() default {};

/\*\*

\* 是否需要纵向合并单元格,应对需求:含有list集合单元格)

\*/

public boolean needMerge() default false;

/\*\*

\* 是否导出数据,应对需求:有时我们需要导出一份模板,这是标题需要但内容需要用户手工填写.

\*/

public boolean isExport() default true;

/\*\*

\* 另一个类中的属性名称,支持多级获取,以小数点隔开

\*/

public String targetAttr() default "";

/\*\*

\* 是否自动统计数据,在最后追加一行统计数据总和

\*/

public boolean isStatistics() default false;

/\*\*

\* 导出类型（0数字 1字符串 2图片）

\*/

public ColumnType cellType() default ColumnType.STRING;

/\*\*

\* 导出列头背景色

\*/

public IndexedColors headerBackgroundColor() default IndexedColors.GREY\_50\_PERCENT;

/\*\*

\* 导出列头字体颜色

\*/

public IndexedColors headerColor() default IndexedColors.WHITE;

/\*\*

\* 导出单元格背景色

\*/

public IndexedColors backgroundColor() default IndexedColors.WHITE;

/\*\*

\* 导出单元格字体颜色

\*/

public IndexedColors color() default IndexedColors.BLACK;

/\*\*

\* 导出字段对齐方式

\*/

public HorizontalAlignment align() default HorizontalAlignment.CENTER;

/\*\*

\* 自定义数据处理器

\*/

public Class<?> handler() default ExcelHandlerAdapter.class;

/\*\*

\* 自定义数据处理器参数

\*/

public String[] args() default {};

/\*\*

\* 字段类型（0：导出导入；1：仅导出；2：仅导入）

\*/

Type type() default Type.ALL;

public enum Type

{

ALL(0), EXPORT(1), IMPORT(2);

private final int value;

Type(int value)

{

this.value = value;

}

public int value()

{

return this.value;

}

}

public enum ColumnType

{

NUMERIC(0), STRING(1), IMAGE(2);

private final int value;

ColumnType(int value)

{

this.value = value;

}

public int value()

{

return this.value;

}

}

}

package com.ruoyi.common.annotation;

import java.lang.annotation.ElementType;

import java.lang.annotation.Retention;

import java.lang.annotation.RetentionPolicy;

import java.lang.annotation.Target;

/\*\*

\* Excel注解集

\*

\*/

@Target(ElementType.FIELD)

@Retention(RetentionPolicy.RUNTIME)

public @interface Excels

{

public Excel[] value();

}

package com.ruoyi.common.annotation;

import java.lang.annotation.Documented;

import java.lang.annotation.ElementType;

import java.lang.annotation.Retention;

import java.lang.annotation.RetentionPolicy;

import java.lang.annotation.Target;

import com.ruoyi.common.enums.BusinessType;

import com.ruoyi.common.enums.OperatorType;

/\*\*

\* 自定义操作日志记录注解

\*

\*

\*/

@Target({ ElementType.PARAMETER, ElementType.METHOD })

@Retention(RetentionPolicy.RUNTIME)

@Documented

public @interface Log

{

/\*\*

\* 模块

\*/

public String title() default "";

/\*\*

\* 功能

\*/

public BusinessType businessType() default BusinessType.OTHER;

/\*\*

\* 操作人类别

\*/

public OperatorType operatorType() default OperatorType.MANAGE;

/\*\*

\* 是否保存请求的参数

\*/

public boolean isSaveRequestData() default true;

/\*\*

\* 是否保存响应的参数

\*/

public boolean isSaveResponseData() default true;

}

package com.ruoyi.common.annotation;

import java.lang.annotation.Documented;

import java.lang.annotation.ElementType;

import java.lang.annotation.Retention;

import java.lang.annotation.RetentionPolicy;

import java.lang.annotation.Target;

import com.ruoyi.common.constant.CacheConstants;

import com.ruoyi.common.enums.LimitType;

/\*\*

\* 限流注解

\*

\*/

@Target(ElementType.METHOD)

@Retention(RetentionPolicy.RUNTIME)

@Documented

public @interface RateLimiter

{

/\*\*

\* 限流key

\*/

public String key() default CacheConstants.RATE\_LIMIT\_KEY;

/\*\*

\* 限流时间,单位秒

\*/

public int time() default 60;

/\*\*

\* 限流次数

\*/

public int count() default 100;

/\*\*

\* 限流类型

\*/

public LimitType limitType() default LimitType.DEFAULT;

}

package com.ruoyi.common.annotation;

import java.lang.annotation.Documented;

import java.lang.annotation.ElementType;

import java.lang.annotation.Inherited;

import java.lang.annotation.Retention;

import java.lang.annotation.RetentionPolicy;

import java.lang.annotation.Target;

/\*\*

\* 自定义注解防止表单重复提交

\*

\*

\*/

@Inherited

@Target(ElementType.METHOD)

@Retention(RetentionPolicy.RUNTIME)

@Documented

public @interface RepeatSubmit

{

/\*\*

\* 间隔时间(ms)，小于此时间视为重复提交

\*/

public int interval() default 5000;

/\*\*

\* 提示消息

\*/

public String message() default "不允许重复提交，请稍候再试";

}

package com.ruoyi.common.config;

import org.springframework.boot.context.properties.ConfigurationProperties;

import org.springframework.stereotype.Component;

/\*\*

\* 读取项目相关配置

\*

\*/

@Component

@ConfigurationProperties(prefix = "ruoyi")

public class RuoYiConfig

{

/\*\* 项目名称 \*/

private String name;

/\*\* 版本 \*/

private String version;

/\*\* 版权年份 \*/

private String copyrightYear;

/\*\* 实例演示开关 \*/

private boolean demoEnabled;

/\*\* 上传路径 \*/

private static String profile;

/\*\* 获取地址开关 \*/

private static boolean addressEnabled;

/\*\* 验证码类型 \*/

private static String captchaType;

public String getName()

{

return name;

}

public void setName(String name)

{

this.name = name;

}

public String getVersion()

{

return version;

}

public void setVersion(String version)

{

this.version = version;

}

public String getCopyrightYear()

{

return copyrightYear;

}

public void setCopyrightYear(String copyrightYear)

{

this.copyrightYear = copyrightYear;

}

public boolean isDemoEnabled()

{

return demoEnabled;

}

public void setDemoEnabled(boolean demoEnabled)

{

this.demoEnabled = demoEnabled;

}

public static String getProfile()

{

return profile;

}

public void setProfile(String profile)

{

RuoYiConfig.profile = profile;

}

public static boolean isAddressEnabled()

{

return addressEnabled;

}

public void setAddressEnabled(boolean addressEnabled)

{

RuoYiConfig.addressEnabled = addressEnabled;

}

public static String getCaptchaType() {

return captchaType;

}

public void setCaptchaType(String captchaType) {

RuoYiConfig.captchaType = captchaType;

}

/\*\*

\* 获取导入上传路径

\*/

public static String getImportPath()

{

return getProfile() + "/import";

}

/\*\*

\* 获取头像上传路径

\*/

public static String getAvatarPath()

{

return getProfile() + "/avatar";

}

/\*\*

\* 获取下载路径

\*/

public static String getDownloadPath()

{

return getProfile() + "/download/";

}

/\*\*

\* 获取上传路径

\*/

public static String getUploadPath()

{

return getProfile() + "/upload";

}

}

package com.ruoyi.common.constant;

/\*\*

\* 缓存的key 常量

\*

\*/

public class CacheConstants

{

/\*\*

\* 登录用户 redis key

\*/

public static final String LOGIN\_TOKEN\_KEY = "login\_tokens:";

/\*\*

\* 验证码 redis key

\*/

public static final String CAPTCHA\_CODE\_KEY = "captcha\_codes:";

/\*\*

\* 参数管理 cache key

\*/

public static final String SYS\_CONFIG\_KEY = "sys\_config:";

/\*\*

\* 字典管理 cache key

\*/

public static final String SYS\_DICT\_KEY = "sys\_dict:";

/\*\*

\* 防重提交 redis key

\*/

public static final String REPEAT\_SUBMIT\_KEY = "repeat\_submit:";

/\*\*

\* 限流 redis key

\*/

public static final String RATE\_LIMIT\_KEY = "rate\_limit:";

/\*\*

\* 登录账户密码错误次数 redis key

\*/

public static final String PWD\_ERR\_CNT\_KEY = "pwd\_err\_cnt:";

}

package com.ruoyi.common.constant;

import io.jsonwebtoken.Claims;

/\*\*

\* 通用常量信息

\*

\*/

public class Constants

{

/\*\*

\* UTF-8 字符集

\*/

public static final String UTF8 = "UTF-8";

/\*\*

\* GBK 字符集

\*/

public static final String GBK = "GBK";

/\*\*

\* www主域

\*/

public static final String WWW = "www.";

/\*\*

\* http请求

\*/

public static final String HTTP = "http://";

/\*\*

\* https请求

\*/

public static final String HTTPS = "https://";

/\*\*

\* 通用成功标识

\*/

public static final String SUCCESS = "0";

/\*\*

\* 通用失败标识

\*/

public static final String FAIL = "1";

/\*\*

\* 登录成功

\*/

public static final String LOGIN\_SUCCESS = "Success";

/\*\*

\* 注销

\*/

public static final String LOGOUT = "Logout";

/\*\*

\* 注册

\*/

public static final String REGISTER = "Register";

/\*\*

\* 登录失败

\*/

public static final String LOGIN\_FAIL = "Error";

/\*\*

\* 验证码有效期（分钟）

\*/

public static final Integer CAPTCHA\_EXPIRATION = 2;

/\*\*

\* 令牌

\*/

public static final String TOKEN = "token";

/\*\*

\* 令牌前缀

\*/

public static final String TOKEN\_PREFIX = "Bearer ";

/\*\*

\* 令牌前缀

\*/

public static final String LOGIN\_USER\_KEY = "login\_user\_key";

/\*\*

\* 用户ID

\*/

public static final String JWT\_USERID = "userid";

/\*\*

\* 用户名称

\*/

public static final String JWT\_USERNAME = Claims.SUBJECT;

/\*\*

\* 用户头像

\*/

public static final String JWT\_AVATAR = "avatar";

/\*\*

\* 创建时间

\*/

public static final String JWT\_CREATED = "created";

/\*\*

\* 用户权限

\*/

public static final String JWT\_AUTHORITIES = "authorities";

/\*\*

\* 资源映射路径 前缀

\*/

public static final String RESOURCE\_PREFIX = "/profile";

/\*\*

\* RMI 远程方法调用

\*/

public static final String LOOKUP\_RMI = "rmi:";

/\*\*

\* LDAP 远程方法调用

\*/

public static final String LOOKUP\_LDAP = "ldap:";

/\*\*

\* LDAPS 远程方法调用

\*/

public static final String LOOKUP\_LDAPS = "ldaps:";

/\*\*

\* 定时任务白名单配置（仅允许访问的包名，如其他需要可以自行添加）

\*/

public static final String[] JOB\_WHITELIST\_STR = { "com.ruoyi" };

/\*\*

\* 定时任务违规的字符

\*/

public static final String[] JOB\_ERROR\_STR = { "java.net.URL", "javax.naming.InitialContext", "org.yaml.snakeyaml",

"org.springframework", "org.apache", "com.ruoyi.common.utils.file", "com.ruoyi.common.config" };

}

package com.ruoyi.common.constant;

/\*\*\*/

public class DictConstants {

/\*\* 收支类型 收入\*/

public static final int SZLX\_INCOME = 0;

/\*\* 收支类型 支出\*/

public static final int SZLX\_OUTCOME = 1;

public static final int FKFS\_DSK = 1;

public static final int FKFS\_DFK = 0;

}

package com.ruoyi.common.constant;

/\*\*

\* 代码生成通用常量

\*

\*/

public class GenConstants

{

/\*\* 单表（增删改查） \*/

public static final String TPL\_CRUD = "crud";

/\*\* 树表（增删改查） \*/

public static final String TPL\_TREE = "tree";

/\*\* 主子表（增删改查） \*/

public static final String TPL\_SUB = "sub";

/\*\* 树编码字段 \*/

public static final String TREE\_CODE = "treeCode";

/\*\* 树父编码字段 \*/

public static final String TREE\_PARENT\_CODE = "treeParentCode";

/\*\* 树名称字段 \*/

public static final String TREE\_NAME = "treeName";

/\*\* 上级菜单ID字段 \*/

public static final String PARENT\_MENU\_ID = "parentMenuId";

/\*\* 上级菜单名称字段 \*/

public static final String PARENT\_MENU\_NAME = "parentMenuName";

/\*\* 数据库字符串类型 \*/

public static final String[] COLUMNTYPE\_STR = { "char", "varchar", "nvarchar", "varchar2" };

/\*\* 数据库文本类型 \*/

public static final String[] COLUMNTYPE\_TEXT = { "tinytext", "text", "mediumtext", "longtext" };

/\*\* 数据库时间类型 \*/

public static final String[] COLUMNTYPE\_TIME = { "datetime", "time", "date", "timestamp" };

/\*\* 数据库数字类型 \*/

public static final String[] COLUMNTYPE\_NUMBER = { "tinyint", "smallint", "mediumint", "int", "number", "integer",

"bit", "bigint", "float", "double", "decimal" };

/\*\* 页面不需要编辑字段 \*/

public static final String[] COLUMNNAME\_NOT\_EDIT = { "id", "create\_by", "create\_time", "del\_flag" };

/\*\* 页面不需要显示的列表字段 \*/

public static final String[] COLUMNNAME\_NOT\_LIST = { "id", "create\_by", "create\_time", "del\_flag", "update\_by",

"update\_time" };

/\*\* 页面不需要查询字段 \*/

public static final String[] COLUMNNAME\_NOT\_QUERY = { "id", "create\_by", "create\_time", "del\_flag", "update\_by",

"update\_time", "remark" };

/\*\* Entity基类字段 \*/

public static final String[] BASE\_ENTITY = { "createBy", "createTime", "updateBy", "updateTime", "remark" };

/\*\* Tree基类字段 \*/

public static final String[] TREE\_ENTITY = { "parentName", "parentId", "orderNum", "ancestors", "children" };

/\*\* 文本框 \*/

public static final String HTML\_INPUT = "input";

/\*\* 文本域 \*/

public static final String HTML\_TEXTAREA = "textarea";

/\*\* 下拉框 \*/

public static final String HTML\_SELECT = "select";

/\*\* 单选框 \*/

public static final String HTML\_RADIO = "radio";

/\*\* 复选框 \*/

public static final String HTML\_CHECKBOX = "checkbox";

/\*\* 日期控件 \*/

public static final String HTML\_DATETIME = "datetime";

/\*\* 图片上传控件 \*/

public static final String HTML\_IMAGE\_UPLOAD = "imageUpload";

/\*\* 文件上传控件 \*/

public static final String HTML\_FILE\_UPLOAD = "fileUpload";

/\*\* 富文本控件 \*/

public static final String HTML\_EDITOR = "editor";

/\*\* 字符串类型 \*/

public static final String TYPE\_STRING = "String";

/\*\* 整型 \*/

public static final String TYPE\_INTEGER = "Integer";

/\*\* 长整型 \*/

public static final String TYPE\_LONG = "Long";

/\*\* 浮点型 \*/

public static final String TYPE\_DOUBLE = "Double";

/\*\* 高精度计算类型 \*/

public static final String TYPE\_BIGDECIMAL = "BigDecimal";

/\*\* 时间类型 \*/

public static final String TYPE\_DATE = "Date";

/\*\* 模糊查询 \*/

public static final String QUERY\_LIKE = "LIKE";

/\*\* 相等查询 \*/

public static final String QUERY\_EQ = "EQ";

/\*\* 需要 \*/

public static final String REQUIRE = "1";

}

package com.ruoyi.common.constant;

/\*\*

\* 返回状态码

\*

\*/

public class HttpStatus

{

/\*\*

\* 操作成功

\*/

public static final int SUCCESS = 200;

/\*\*

\* 对象创建成功

\*/

public static final int CREATED = 201;

/\*\*

\* 请求已经被接受

\*/

public static final int ACCEPTED = 202;

/\*\*

\* 操作已经执行成功，但是没有返回数据

\*/

public static final int NO\_CONTENT = 204;

/\*\*

\* 资源已被移除

\*/

public static final int MOVED\_PERM = 301;

/\*\*

\* 重定向

\*/

public static final int SEE\_OTHER = 303;

/\*\*

\* 资源没有被修改

\*/

public static final int NOT\_MODIFIED = 304;

/\*\*

\* 参数列表错误（缺少，格式不匹配）

\*/

public static final int BAD\_REQUEST = 400;

/\*\*

\* 未授权

\*/

public static final int UNAUTHORIZED = 401;

/\*\*

\* 访问受限，授权过期

\*/

public static final int FORBIDDEN = 403;

/\*\*

\* 资源，服务未找到

\*/

public static final int NOT\_FOUND = 404;

/\*\*

\* 不允许的http方法

\*/

public static final int BAD\_METHOD = 405;

/\*\*

\* 资源冲突，或者资源被锁

\*/

public static final int CONFLICT = 409;

/\*\*

\* 不支持的数据，媒体类型

\*/

public static final int UNSUPPORTED\_TYPE = 415;

/\*\*

\* 系统内部错误

\*/

public static final int ERROR = 500;

/\*\*

\* 接口未实现

\*/

public static final int NOT\_IMPLEMENTED = 501;

/\*\*

\* 系统警告消息

\*/

public static final int WARN = 601;

}

package com.ruoyi.common.constant;

/\*\*

\* 任务调度通用常量

\*

\*/

public class ScheduleConstants

{

public static final String TASK\_CLASS\_NAME = "TASK\_CLASS\_NAME";

/\*\* 执行目标key \*/

public static final String TASK\_PROPERTIES = "TASK\_PROPERTIES";

/\*\* 默认 \*/

public static final String MISFIRE\_DEFAULT = "0";

/\*\* 立即触发执行 \*/

public static final String MISFIRE\_IGNORE\_MISFIRES = "1";

/\*\* 触发一次执行 \*/

public static final String MISFIRE\_FIRE\_AND\_PROCEED = "2";

/\*\* 不触发立即执行 \*/

public static final String MISFIRE\_DO\_NOTHING = "3";

public enum Status

{

/\*\*

\* 正常

\*/

NORMAL("0"),

/\*\*

\* 暂停

\*/

PAUSE("1");

private String value;

private Status(String value)

{

this.value = value;

}

public String getValue()

{

return value;

}

}

}

package com.ruoyi.common.constant;

/\*\*

\* 用户常量信息

\*

\*/

public class UserConstants

{

/\*\*

\* 平台内系统用户的唯一标志

\*/

public static final String SYS\_USER = "SYS\_USER";

/\*\* 正常状态 \*/

public static final String NORMAL = "0";

/\*\* 异常状态 \*/

public static final String EXCEPTION = "1";

/\*\* 用户封禁状态 \*/

public static final String USER\_DISABLE = "1";

/\*\* 角色封禁状态 \*/

public static final String ROLE\_DISABLE = "1";

/\*\* 部门正常状态 \*/

public static final String DEPT\_NORMAL = "0";

/\*\* 部门停用状态 \*/

public static final String DEPT\_DISABLE = "1";

/\*\* 字典正常状态 \*/

public static final String DICT\_NORMAL = "0";

/\*\* 是否为系统默认（是） \*/

public static final String YES = "Y";

/\*\* 是否菜单外链（是） \*/

public static final String YES\_FRAME = "0";

/\*\* 是否菜单外链（否） \*/

public static final String NO\_FRAME = "1";

/\*\* 菜单类型（目录） \*/

public static final String TYPE\_DIR = "M";

/\*\* 菜单类型（菜单） \*/

public static final String TYPE\_MENU = "C";

/\*\* 菜单类型（按钮） \*/

public static final String TYPE\_BUTTON = "F";

/\*\* Layout组件标识 \*/

public final static String LAYOUT = "Layout";

/\*\* ParentView组件标识 \*/

public final static String PARENT\_VIEW = "ParentView";

/\*\* InnerLink组件标识 \*/

public final static String INNER\_LINK = "InnerLink";

/\*\* 校验返回结果码 \*/

public final static String UNIQUE = "0";

public final static String NOT\_UNIQUE = "1";

/\*\*

\* 用户名长度限制

\*/

public static final int USERNAME\_MIN\_LENGTH = 2;

public static final int USERNAME\_MAX\_LENGTH = 20;

/\*\*

\* 密码长度限制

\*/

public static final int PASSWORD\_MIN\_LENGTH = 5;

public static final int PASSWORD\_MAX\_LENGTH = 20;

}

package com.ruoyi.framework.aspectj;

import java.util.ArrayList;

import java.util.List;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

import org.springframework.stereotype.Component;

import com.ruoyi.common.annotation.DataScope;

import com.ruoyi.common.core.domain.BaseEntity;

import com.ruoyi.common.core.domain.entity.SysRole;

import com.ruoyi.common.core.domain.entity.SysUser;

import com.ruoyi.common.core.domain.model.LoginUser;

import com.ruoyi.common.core.text.Convert;

import com.ruoyi.common.utils.SecurityUtils;

import com.ruoyi.common.utils.StringUtils;

import com.ruoyi.framework.security.context.PermissionContextHolder;

/\*\*

\* 数据过滤处理

\*

\*/

@Aspect

@Component

public class DataScopeAspect

{

/\*\*

\* 全部数据权限

\*/

public static final String DATA\_SCOPE\_ALL = "1";

/\*\*

\* 自定数据权限

\*/

public static final String DATA\_SCOPE\_CUSTOM = "2";

/\*\*

\* 部门数据权限

\*/

public static final String DATA\_SCOPE\_DEPT = "3";

/\*\*

\* 部门及以下数据权限

\*/

public static final String DATA\_SCOPE\_DEPT\_AND\_CHILD = "4";

/\*\*

\* 仅本人数据权限

\*/

public static final String DATA\_SCOPE\_SELF = "5";

/\*\*

\* 数据权限过滤关键字

\*/

public static final String DATA\_SCOPE = "dataScope";

@Before("@annotation(controllerDataScope)")

public void doBefore(JoinPoint point, DataScope controllerDataScope) throws Throwable

{

clearDataScope(point);

handleDataScope(point, controllerDataScope);

}

protected void handleDataScope(final JoinPoint joinPoint, DataScope controllerDataScope)

{

// 获取当前的用户

LoginUser loginUser = SecurityUtils.getLoginUser();

if (StringUtils.isNotNull(loginUser))

{

SysUser currentUser = loginUser.getUser();

// 如果是超级管理员，则不过滤数据

if (StringUtils.isNotNull(currentUser) && !currentUser.isAdmin())

{

String permission = StringUtils.defaultIfEmpty(controllerDataScope.permission(), PermissionContextHolder.getContext());

dataScopeFilter(joinPoint, currentUser, controllerDataScope.deptAlias(),

controllerDataScope.userAlias(), permission);

}

}

}

/\*\*

\* 数据范围过滤

\*

\* @param joinPoint 切点

\* @param user 用户

\* @param deptAlias 部门别名

\* @param userAlias 用户别名

\* @param permission 权限字符

\*/

public static void dataScopeFilter(JoinPoint joinPoint, SysUser user, String deptAlias, String userAlias, String permission)

{

StringBuilder sqlString = new StringBuilder();

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

for (SysRole role : user.getRoles())

{

String dataScope = role.getDataScope();

if (!DATA\_SCOPE\_CUSTOM.equals(dataScope) && conditions.contains(dataScope))

{

continue;

}

if (StringUtils.isNotEmpty(permission) && StringUtils.isNotEmpty(role.getPermissions())

&& !StringUtils.containsAny(role.getPermissions(), Convert.toStrArray(permission)))

{

continue;

}

if (DATA\_SCOPE\_ALL.equals(dataScope))

{

sqlString = new StringBuilder();

break;

}

else if (DATA\_SCOPE\_CUSTOM.equals(dataScope))

{

sqlString.append(StringUtils.format(

" OR {}.dept\_id IN ( SELECT dept\_id FROM sys\_role\_dept WHERE role\_id = {} ) ", deptAlias,

role.getRoleId()));

}

else if (DATA\_SCOPE\_DEPT.equals(dataScope))

{

sqlString.append(StringUtils.format(" OR {}.dept\_id = {} ", deptAlias, user.getDeptId()));

}

else if (DATA\_SCOPE\_DEPT\_AND\_CHILD.equals(dataScope))

{

sqlString.append(StringUtils.format(

" OR {}.dept\_id IN ( SELECT dept\_id FROM sys\_dept WHERE dept\_id = {} or find\_in\_set( {} , ancestors ) )",

deptAlias, user.getDeptId(), user.getDeptId()));

}

else if (DATA\_SCOPE\_SELF.equals(dataScope))

{

if (StringUtils.isNotBlank(userAlias))

{

sqlString.append(StringUtils.format(" OR {}.user\_id = {} ", userAlias, user.getUserId()));

}

else

{

// 数据权限为仅本人且没有userAlias别名不查询任何数据

sqlString.append(StringUtils.format(" OR {}.dept\_id = 0 ", deptAlias));

}

}

conditions.add(dataScope);

}

if (StringUtils.isNotBlank(sqlString.toString()))

{

Object params = joinPoint.getArgs()[0];

if (StringUtils.isNotNull(params) && params instanceof BaseEntity)

{

BaseEntity baseEntity = (BaseEntity) params;

baseEntity.getParams().put(DATA\_SCOPE, " AND (" + sqlString.substring(4) + ")");

}

}

}

/\*\*

\* 拼接权限sql前先清空params.dataScope参数防止注入

\*/

private void clearDataScope(final JoinPoint joinPoint)

{

Object params = joinPoint.getArgs()[0];

if (StringUtils.isNotNull(params) && params instanceof BaseEntity)

{

BaseEntity baseEntity = (BaseEntity) params;

baseEntity.getParams().put(DATA\_SCOPE, "");

}

}

}

package com.ruoyi.framework.aspectj;

import java.util.Objects;

import org.aspectj.lang.ProceedingJoinPoint;

import org.aspectj.lang.annotation.Around;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Pointcut;

import org.aspectj.lang.reflect.MethodSignature;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.core.annotation.AnnotationUtils;

import org.springframework.core.annotation.Order;

import org.springframework.stereotype.Component;

import com.ruoyi.common.annotation.DataSource;

import com.ruoyi.common.utils.StringUtils;

import com.ruoyi.framework.datasource.DynamicDataSourceContextHolder;

/\*\*

\* 多数据源处理

\*

\*/

@Aspect

@Order(1)

@Component

public class DataSourceAspect

{

protected Logger logger = LoggerFactory.getLogger(getClass());

@Pointcut("@annotation(com.ruoyi.common.annotation.DataSource)"

+ "|| @within(com.ruoyi.common.annotation.DataSource)")

public void dsPointCut()

{

}

@Around("dsPointCut()")

public Object around(ProceedingJoinPoint point) throws Throwable

{

DataSource dataSource = getDataSource(point);

if (StringUtils.isNotNull(dataSource))

{

DynamicDataSourceContextHolder.setDataSourceType(dataSource.value().name());

}

try

{

return point.proceed();

}

finally

{

// 销毁数据源 在执行方法之后

DynamicDataSourceContextHolder.clearDataSourceType();

}

}

/\*\*

\* 获取需要切换的数据源

\*/

public DataSource getDataSource(ProceedingJoinPoint point)

{

MethodSignature signature = (MethodSignature) point.getSignature();

DataSource dataSource = AnnotationUtils.findAnnotation(signature.getMethod(), DataSource.class);

if (Objects.nonNull(dataSource))

{

return dataSource;

}

return AnnotationUtils.findAnnotation(signature.getDeclaringType(), DataSource.class);

}

}

package com.ruoyi.framework.aspectj;

import java.util.Collection;

import java.util.Map;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.annotation.AfterReturning;

import org.aspectj.lang.annotation.AfterThrowing;

import org.aspectj.lang.annotation.Aspect;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.stereotype.Component;

import org.springframework.validation.BindingResult;

import org.springframework.web.multipart.MultipartFile;

import com.alibaba.fastjson2.JSON;

import com.ruoyi.common.annotation.Log;

import com.ruoyi.common.core.domain.model.LoginUser;

import com.ruoyi.common.enums.BusinessStatus;

import com.ruoyi.common.enums.HttpMethod;

import com.ruoyi.common.filter.PropertyPreExcludeFilter;

import com.ruoyi.common.utils.SecurityUtils;

import com.ruoyi.common.utils.ServletUtils;

import com.ruoyi.common.utils.StringUtils;

import com.ruoyi.common.utils.ip.IpUtils;

import com.ruoyi.framework.manager.AsyncManager;

import com.ruoyi.framework.manager.factory.AsyncFactory;

import com.ruoyi.system.domain.SysOperLog;

/\*\*

\* 操作日志记录处理

\*

\*/

@Aspect

@Component

public class LogAspect

{

private static final Logger log = LoggerFactory.getLogger(LogAspect.class);

/\*\* 排除敏感属性字段 \*/

public static final String[] EXCLUDE\_PROPERTIES = { "password", "oldPassword", "newPassword", "confirmPassword" };

/\*\*

\* 处理完请求后执行

\*

\* @param joinPoint 切点

\*/

@AfterReturning(pointcut = "@annotation(controllerLog)", returning = "jsonResult")

public void doAfterReturning(JoinPoint joinPoint, Log controllerLog, Object jsonResult)

{

handleLog(joinPoint, controllerLog, null, jsonResult);

}

/\*\*

\* 拦截异常操作

\*

\* @param joinPoint 切点

\* @param e 异常

\*/

@AfterThrowing(value = "@annotation(controllerLog)", throwing = "e")

public void doAfterThrowing(JoinPoint joinPoint, Log controllerLog, Exception e)

{

handleLog(joinPoint, controllerLog, e, null);

}

protected void handleLog(final JoinPoint joinPoint, Log controllerLog, final Exception e, Object jsonResult)

{

try

{

// 获取当前的用户

LoginUser loginUser = SecurityUtils.getLoginUser();

// \*========数据库日志=========\*//

SysOperLog operLog = new SysOperLog();

operLog.setStatus(BusinessStatus.SUCCESS.ordinal());

// 请求的地址

String ip = IpUtils.getIpAddr(ServletUtils.getRequest());

operLog.setOperIp(ip);

operLog.setOperUrl(StringUtils.substring(ServletUtils.getRequest().getRequestURI(), 0, 255));

if (loginUser != null)

{

operLog.setOperName(loginUser.getUsername());

}

if (e != null)

{

operLog.setStatus(BusinessStatus.FAIL.ordinal());

operLog.setErrorMsg(StringUtils.substring(e.getMessage(), 0, 2000));

}

// 设置方法名称

String className = joinPoint.getTarget().getClass().getName();

String methodName = joinPoint.getSignature().getName();

operLog.setMethod(className + "." + methodName + "()");

// 设置请求方式

operLog.setRequestMethod(ServletUtils.getRequest().getMethod());

// 处理设置注解上的参数

getControllerMethodDescription(joinPoint, controllerLog, operLog, jsonResult);

// 保存数据库

AsyncManager.me().execute(AsyncFactory.recordOper(operLog));

}

catch (Exception exp)

{

// 记录本地异常日志

log.error("异常信息:{}", exp.getMessage());

exp.printStackTrace();

}

}

/\*\*

\* 获取注解中对方法的描述信息 用于Controller层注解

\*

\* @param log 日志

\* @param operLog 操作日志

\* @throws Exception

\*/

public void getControllerMethodDescription(JoinPoint joinPoint, Log log, SysOperLog operLog, Object jsonResult) throws Exception

{

// 设置action动作

operLog.setBusinessType(log.businessType().ordinal());

// 设置标题

operLog.setTitle(log.title());

// 设置操作人类别

operLog.setOperatorType(log.operatorType().ordinal());

// 是否需要保存request，参数和值

if (log.isSaveRequestData())

{

// 获取参数的信息，传入到数据库中。

setRequestValue(joinPoint, operLog);

}

// 是否需要保存response，参数和值

if (log.isSaveResponseData() && StringUtils.isNotNull(jsonResult))

{

operLog.setJsonResult(StringUtils.substring(JSON.toJSONString(jsonResult), 0, 2000));

}

}

/\*\*

\* 获取请求的参数，放到log中

\*

\* @param operLog 操作日志

\* @throws Exception 异常

\*/

private void setRequestValue(JoinPoint joinPoint, SysOperLog operLog) throws Exception

{

String requestMethod = operLog.getRequestMethod();

if (HttpMethod.PUT.name().equals(requestMethod) || HttpMethod.POST.name().equals(requestMethod))

{

String params = argsArrayToString(joinPoint.getArgs());

operLog.setOperParam(StringUtils.substring(params, 0, 2000));

}

else

{

Map<?, ?> paramsMap = ServletUtils.getParamMap(ServletUtils.getRequest());

operLog.setOperParam(StringUtils.substring(JSON.toJSONString(paramsMap, excludePropertyPreFilter()), 0, 2000));

}

}

/\*\*

\* 参数拼装

\*/

private String argsArrayToString(Object[] paramsArray)

{

String params = "";

if (paramsArray != null && paramsArray.length > 0)

{

for (Object o : paramsArray)

{

if (StringUtils.isNotNull(o) && !isFilterObject(o))

{

try

{

String jsonObj = JSON.toJSONString(o, excludePropertyPreFilter());

params += jsonObj.toString() + " ";

}

catch (Exception e)

{

}

}

}

}

return params.trim();

}

/\*\*

\* 忽略敏感属性

\*/

public PropertyPreExcludeFilter excludePropertyPreFilter()

{

return new PropertyPreExcludeFilter().addExcludes(EXCLUDE\_PROPERTIES);

}

/\*\*

\* 判断是否需要过滤的对象。

\*

\* @param o 对象信息。

\* @return 如果是需要过滤的对象，则返回true；否则返回false。

\*/

@SuppressWarnings("rawtypes")

public boolean isFilterObject(final Object o)

{

Class<?> clazz = o.getClass();

if (clazz.isArray())

{

return clazz.getComponentType().isAssignableFrom(MultipartFile.class);

}

else if (Collection.class.isAssignableFrom(clazz))

{

Collection collection = (Collection) o;

for (Object value : collection)

{

return value instanceof MultipartFile;

}

}

else if (Map.class.isAssignableFrom(clazz))

{

Map map = (Map) o;

for (Object value : map.entrySet())

{

Map.Entry entry = (Map.Entry) value;

return entry.getValue() instanceof MultipartFile;

}

}

return o instanceof MultipartFile || o instanceof HttpServletRequest || o instanceof HttpServletResponse

|| o instanceof BindingResult;

}

}

package com.ruoyi.framework.aspectj;

import java.lang.reflect.Method;

import java.util.Collections;

import java.util.List;

import org.aspectj.lang.JoinPoint;

import org.aspectj.lang.annotation.Aspect;

import org.aspectj.lang.annotation.Before;

import org.aspectj.lang.reflect.MethodSignature;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.data.redis.core.RedisTemplate;

import org.springframework.data.redis.core.script.RedisScript;

import org.springframework.stereotype.Component;

import com.ruoyi.common.annotation.RateLimiter;

import com.ruoyi.common.enums.LimitType;

import com.ruoyi.common.exception.ServiceException;

import com.ruoyi.common.utils.ServletUtils;

import com.ruoyi.common.utils.StringUtils;

import com.ruoyi.common.utils.ip.IpUtils;

/\*\*

\* 限流处理

\*

\*/

@Aspect

@Component

public class RateLimiterAspect

{

private static final Logger log = LoggerFactory.getLogger(RateLimiterAspect.class);

private RedisTemplate<Object, Object> redisTemplate;

private RedisScript<Long> limitScript;

@Autowired

public void setRedisTemplate1(RedisTemplate<Object, Object> redisTemplate)

{

this.redisTemplate = redisTemplate;

}

@Autowired

public void setLimitScript(RedisScript<Long> limitScript)

{

this.limitScript = limitScript;

}

@Before("@annotation(rateLimiter)")

public void doBefore(JoinPoint point, RateLimiter rateLimiter) throws Throwable

{

int time = rateLimiter.time();

int count = rateLimiter.count();

String combineKey = getCombineKey(rateLimiter, point);

List<Object> keys = Collections.singletonList(combineKey);

try

{

Long number = redisTemplate.execute(limitScript, keys, count, time);

if (StringUtils.isNull(number) || number.intValue() > count)

{

throw new ServiceException("访问过于频繁，请稍候再试");

}

log.info("限制请求'{}',当前请求'{}',缓存key'{}'", count, number.intValue(), combineKey);

}

catch (ServiceException e)

{

throw e;

}

catch (Exception e)

{

throw new RuntimeException("服务器限流异常，请稍候再试");

}

}

public String getCombineKey(RateLimiter rateLimiter, JoinPoint point)

{

StringBuffer stringBuffer = new StringBuffer(rateLimiter.key());

if (rateLimiter.limitType() == LimitType.IP)

{

stringBuffer.append(IpUtils.getIpAddr(ServletUtils.getRequest())).append("-");

}

MethodSignature signature = (MethodSignature) point.getSignature();

Method method = signature.getMethod();

Class<?> targetClass = method.getDeclaringClass();

stringBuffer.append(targetClass.getName()).append("-").append(method.getName());

return stringBuffer.toString();

}

}

package com.ruoyi.framework.datasource;

import java.util.Map;

import javax.sql.DataSource;

import org.springframework.jdbc.datasource.lookup.AbstractRoutingDataSource;

/\*\*

\* 动态数据源

\*

\*/

public class DynamicDataSource extends AbstractRoutingDataSource

{

public DynamicDataSource(DataSource defaultTargetDataSource, Map<Object, Object> targetDataSources)

{

super.setDefaultTargetDataSource(defaultTargetDataSource);

super.setTargetDataSources(targetDataSources);

super.afterPropertiesSet();

}

@Override

protected Object determineCurrentLookupKey()

{

return DynamicDataSourceContextHolder.getDataSourceType();

}

}

package com.ruoyi.framework.datasource;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

/\*\*

\* 数据源切换处理

\*

\*/

public class DynamicDataSourceContextHolder

{

public static final Logger log = LoggerFactory.getLogger(DynamicDataSourceContextHolder.class);

/\*\*

\* 使用ThreadLocal维护变量，ThreadLocal为每个使用该变量的线程提供独立的变量副本，

\* 所以每一个线程都可以独立地改变自己的副本，而不会影响其它线程所对应的副本。

\*/

private static final ThreadLocal<String> CONTEXT\_HOLDER = new ThreadLocal<>();

/\*\*

\* 设置数据源的变量

\*/

public static void setDataSourceType(String dsType)

{

log.info("切换到{}数据源", dsType);

CONTEXT\_HOLDER.set(dsType);

}

/\*\*

\* 获得数据源的变量

\*/

public static String getDataSourceType()

{

return CONTEXT\_HOLDER.get();

}

/\*\*

\* 清空数据源变量

\*/

public static void clearDataSourceType()

{

CONTEXT\_HOLDER.remove();

}

}

package com.ruoyi.framework.interceptor.impl;

import java.util.HashMap;

import java.util.Map;

import java.util.concurrent.TimeUnit;

import javax.servlet.http.HttpServletRequest;

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

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

import org.springframework.stereotype.Component;

import com.alibaba.fastjson2.JSON;

import com.ruoyi.common.annotation.RepeatSubmit;

import com.ruoyi.common.constant.CacheConstants;

import com.ruoyi.common.core.redis.RedisCache;

import com.ruoyi.common.filter.RepeatedlyRequestWrapper;

import com.ruoyi.common.utils.StringUtils;

import com.ruoyi.common.utils.http.HttpHelper;

import com.ruoyi.framework.interceptor.RepeatSubmitInterceptor;

/\*\*

\* 判断请求url和数据是否和上一次相同，

\* 如果和上次相同，则是重复提交表单。 有效时间为10秒内。

\*

\*/

@Component

public class SameUrlDataInterceptor extends RepeatSubmitInterceptor

{

public final String REPEAT\_PARAMS = "repeatParams";

public final String REPEAT\_TIME = "repeatTime";

// 令牌自定义标识

@Value("${token.header}")

private String header;

@Autowired

private RedisCache redisCache;

@SuppressWarnings("unchecked")

@Override

public boolean isRepeatSubmit(HttpServletRequest request, RepeatSubmit annotation)

{

String nowParams = "";

if (request instanceof RepeatedlyRequestWrapper)

{

RepeatedlyRequestWrapper repeatedlyRequest = (RepeatedlyRequestWrapper) request;

nowParams = HttpHelper.getBodyString(repeatedlyRequest);

}

// body参数为空，获取Parameter的数据

if (StringUtils.isEmpty(nowParams))

{

nowParams = JSON.toJSONString(request.getParameterMap());

}

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

nowDataMap.put(REPEAT\_PARAMS, nowParams);

nowDataMap.put(REPEAT\_TIME, System.currentTimeMillis());

// 请求地址（作为存放cache的key值）

String url = request.getRequestURI();

// 唯一值（没有消息头则使用请求地址）

String submitKey = StringUtils.trimToEmpty(request.getHeader(header));

// 唯一标识（指定key + url + 消息头）

String cacheRepeatKey = CacheConstants.REPEAT\_SUBMIT\_KEY + url + submitKey;

Object sessionObj = redisCache.getCacheObject(cacheRepeatKey);

if (sessionObj != null)

{

Map<String, Object> sessionMap = (Map<String, Object>) sessionObj;

if (sessionMap.containsKey(url))

{

Map<String, Object> preDataMap = (Map<String, Object>) sessionMap.get(url);

if (compareParams(nowDataMap, preDataMap) && compareTime(nowDataMap, preDataMap, annotation.interval()))

{

return true;

}

}

}

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

cacheMap.put(url, nowDataMap);

redisCache.setCacheObject(cacheRepeatKey, cacheMap, annotation.interval(), TimeUnit.MILLISECONDS);

return false;

}

/\*\*

\* 判断参数是否相同

\*/

private boolean compareParams(Map<String, Object> nowMap, Map<String, Object> preMap)

{

String nowParams = (String) nowMap.get(REPEAT\_PARAMS);

String preParams = (String) preMap.get(REPEAT\_PARAMS);

return nowParams.equals(preParams);

}

/\*\*

\* 判断两次间隔时间

\*/

private boolean compareTime(Map<String, Object> nowMap, Map<String, Object> preMap, int interval)

{

long time1 = (Long) nowMap.get(REPEAT\_TIME);

long time2 = (Long) preMap.get(REPEAT\_TIME);

if ((time1 - time2) < interval)

{

return true;

}

return false;

}

}

package com.ruoyi.framework.interceptor;

import java.lang.reflect.Method;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.springframework.stereotype.Component;

import org.springframework.web.method.HandlerMethod;

import org.springframework.web.servlet.HandlerInterceptor;

import com.alibaba.fastjson2.JSON;

import com.ruoyi.common.annotation.RepeatSubmit;

import com.ruoyi.common.core.domain.AjaxResult;

import com.ruoyi.common.utils.ServletUtils;

/\*\*

\* 防止重复提交拦截器

\*

\*/

@Component

public abstract class RepeatSubmitInterceptor implements HandlerInterceptor

{

@Override

public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception

{

if (handler instanceof HandlerMethod)

{

HandlerMethod handlerMethod = (HandlerMethod) handler;

Method method = handlerMethod.getMethod();

RepeatSubmit annotation = method.getAnnotation(RepeatSubmit.class);

if (annotation != null)

{

if (this.isRepeatSubmit(request, annotation))

{

AjaxResult ajaxResult = AjaxResult.error(annotation.message());

ServletUtils.renderString(response, JSON.toJSONString(ajaxResult));

return false;

}

}

return true;

}

else

{

return true;

}

}

/\*\*

\* 验证是否重复提交由子类实现具体的防重复提交的规则

\*

\* @param request

\* @return

\* @throws Exception

\*/

public abstract boolean isRepeatSubmit(HttpServletRequest request, RepeatSubmit annotation);

}