package com.erp.controller;

import com.alibaba.fastjson.JSONArray;

import com.alibaba.fastjson.JSONObject;

import com.erp.constants.BusinessConstants;

import com.erp.constants.ExceptionConstants;

import com.erp.datasource.entities.Depot;

import com.erp.datasource.entities.DepotEx;

import com.erp.datasource.entities.MaterialCurrentStock;

import com.erp.datasource.entities.MaterialInitialStock;

import com.erp.exception.BusinessRunTimeException;

import com.erp.service.depot.DepotService;

import com.erp.service.material.MaterialService;

import com.erp.service.systemConfig.SystemConfigService;

import com.erp.service.user.UserService;

import com.erp.service.userBusiness.UserBusinessService;

import com.erp.utils.\*;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.dao.DataAccessException;

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

import javax.annotation.Resource;

import javax.servlet.http.HttpServletRequest;

import java.io.IOException;

import java.math.BigDecimal;

import java.util.\*;

import static com.erp.utils.ResponseJsonUtil.returnJson;

/\*\*

\* 仓库管理

\*/

@RestController

@RequestMapping(value = "/depot")

@Api(tags = {"仓库管理"})

public class DepotController {

private Logger logger = LoggerFactory.getLogger(DepotController.class);

@Resource

private DepotService depotService;

@Resource

private UserBusinessService userBusinessService;

@Resource

private MaterialService materialService;

/\*\*

\* 仓库列表

\* @param request

\* @return

\* @throws Exception

\*/

@GetMapping(value = "/getAllList")

@ApiOperation(value = "仓库列表")

public BaseResponseInfo getAllList(HttpServletRequest request) throws Exception{

BaseResponseInfo res = new BaseResponseInfo();

try {

List<Depot> depotList = depotService.getAllList();

res.code = 200;

res.data = depotList;

} catch(Exception e){

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 用户对应仓库显示

\* @param type

\* @param keyId

\* @param request

\* @return

\*/

@GetMapping(value = "/findUserDepot")

@ApiOperation(value = "用户对应仓库显示")

public JSONArray findUserDepot(@RequestParam("UBType") String type, @RequestParam("UBKeyId") String keyId,

HttpServletRequest request) throws Exception{

JSONArray arr = new JSONArray();

try {

//获取权限信息

String ubValue = userBusinessService.getUBValueByTypeAndKeyId(type, keyId);

List<Depot> dataList = depotService.findUserDepot();

//开始拼接json数据

JSONObject outer = new JSONObject();

outer.put("id", 0);

outer.put("key", 0);

outer.put("value", 0);

outer.put("title", "仓库列表");

outer.put("attributes", "仓库列表");

//存放数据json数组

JSONArray dataArray = new JSONArray();

if (null != dataList) {

for (Depot depot : dataList) {

JSONObject item = new JSONObject();

item.put("id", depot.getId());

item.put("key", depot.getId());

item.put("value", depot.getId());

item.put("title", depot.getName());

item.put("attributes", depot.getName());

Boolean flag = ubValue.contains("[" + depot.getId().toString() + "]");

if (flag) {

item.put("checked", true);

}

dataArray.add(item);

}

}

outer.put("children", dataArray);

arr.add(outer);

} catch (Exception e) {

e.printStackTrace();

}

return arr;

}

/\*\*

\* 获取当前用户拥有权限的仓库列表

\* @param request

\* @return

\* @throws Exception

\*/

@GetMapping(value = "/findDepotByCurrentUser")

@ApiOperation(value = "获取当前用户拥有权限的仓库列表")

public BaseResponseInfo findDepotByCurrentUser(HttpServletRequest request) throws Exception{

BaseResponseInfo res = new BaseResponseInfo();

try {

JSONArray arr = depotService.findDepotByCurrentUser();

res.code = 200;

res.data = arr;

} catch (Exception e) {

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 更新默认仓库

\* @param object

\* @param request

\* @return

\* @throws Exception

\*/

@PostMapping(value = "/updateIsDefault")

@ApiOperation(value = "更新默认仓库")

public String updateIsDefault(@RequestBody JSONObject object,

HttpServletRequest request) throws Exception{

Long depotId = object.getLong("id");

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

int res = depotService.updateIsDefault(depotId);

if(res > 0) {

return returnJson(objectMap, ErpInfo.OK.name, ErpInfo.OK.code);

} else {

return returnJson(objectMap, ErpInfo.ERROR.name, ErpInfo.ERROR.code);

}

}

/\*\*

\* 仓库列表-带库存

\* @param mId

\* @param request

\* @return

\*/

@GetMapping(value = "/getAllListWithStock")

@ApiOperation(value = "仓库列表-带库存")

public BaseResponseInfo getAllList(@RequestParam("mId") Long mId,

HttpServletRequest request) {

BaseResponseInfo res = new BaseResponseInfo();

try {

List<Depot> list = depotService.getAllList();

List<DepotEx> depotList = new ArrayList<DepotEx>();

for(Depot depot: list) {

DepotEx de = new DepotEx();

if(mId!=0) {

BigDecimal initStock = materialService.getInitStock(mId, depot.getId());

BigDecimal currentStock = materialService.getCurrentStockByMaterialIdAndDepotId(mId, depot.getId());

de.setInitStock(initStock);

de.setCurrentStock(currentStock);

MaterialInitialStock materialInitialStock = materialService.getSafeStock(mId, depot.getId());

de.setLowSafeStock(materialInitialStock.getLowSafeStock());

de.setHighSafeStock(materialInitialStock.getHighSafeStock());

} else {

de.setInitStock(BigDecimal.ZERO);

de.setCurrentStock(BigDecimal.ZERO);

}

de.setId(depot.getId());

de.setName(depot.getName());

depotList.add(de);

}

res.code = 200;

res.data = depotList;

} catch(Exception e){

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 批量设置状态-启用或者禁用

\* @param jsonObject

\* @param request

\* @return

\*/

@PostMapping(value = "/batchSetStatus")

@ApiOperation(value = "批量设置状态")

public String batchSetStatus(@RequestBody JSONObject jsonObject,

HttpServletRequest request)throws Exception {

Boolean status = jsonObject.getBoolean("status");

String ids = jsonObject.getString("ids");

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

int res = depotService.batchSetStatus(status, ids);

if(res > 0) {

return returnJson(objectMap, ErpInfo.OK.name, ErpInfo.OK.code);

} else {

return returnJson(objectMap, ErpInfo.ERROR.name, ErpInfo.ERROR.code);

}

}

}

package com.erp.controller;

import com.alibaba.fastjson.JSONArray;

import com.alibaba.fastjson.JSONObject;

import com.erp.constants.BusinessConstants;

import com.erp.constants.ExceptionConstants;

import com.erp.datasource.entities.\*;

import com.erp.datasource.vo.DepotItemStockWarningCount;

import com.erp.datasource.vo.DepotItemVoBatchNumberList;

import com.erp.exception.BusinessRunTimeException;

import com.erp.service.depot.DepotService;

import com.erp.service.depotHead.DepotHeadService;

import com.erp.service.materialExtend.MaterialExtendService;

import com.erp.service.depotItem.DepotItemService;

import com.erp.service.material.MaterialService;

import com.erp.service.redis.RedisService;

import com.erp.service.role.RoleService;

import com.erp.service.unit.UnitService;

import com.erp.utils.\*;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import javax.annotation.Resource;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.File;

import java.math.BigDecimal;

import java.text.DecimalFormat;

import java.text.SimpleDateFormat;

import java.util.\*;

import static com.erp.utils.ResponseJsonUtil.returnJson;

import static com.erp.utils.Tools.getCenternTime;

@RestController

@RequestMapping(value = "/depotItem")

@Api(tags = {"单据明细"})

public class DepotItemController {

private Logger logger = LoggerFactory.getLogger(DepotItemController.class);

@Resource

private DepotHeadService depotHeadService;

@Resource

private DepotItemService depotItemService;

@Resource

private MaterialService materialService;

@Resource

private UnitService unitService;

@Resource

private DepotService depotService;

@Resource

private RoleService roleService;

/\*\*

\* 根据仓库和商品查询单据列表

\* @param mId

\* @param request

\* @return

\*/

@GetMapping(value = "/findDetailByDepotIdsAndMaterialId")

@ApiOperation(value = "根据仓库和商品查询单据列表")

public String findDetailByDepotIdsAndMaterialId(

@RequestParam(value = Constants.PAGE\_SIZE, required = false) Integer pageSize,

@RequestParam(value = Constants.CURRENT\_PAGE, required = false) Integer currentPage,

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

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

@RequestParam("materialId") Long mId,

HttpServletRequest request)throws Exception {

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

List<DepotItemVo4DetailByTypeAndMId> list = depotItemService.findDetailByDepotIdsAndMaterialIdList(depotIds, sku, mId, (currentPage-1)\*pageSize, pageSize);

JSONArray dataArray = new JSONArray();

if (list != null) {

for (DepotItemVo4DetailByTypeAndMId d: list) {

JSONObject item = new JSONObject();

item.put("number", d.getNumber()); //编号

item.put("barCode", d.getBarCode()); //条码

item.put("materialName", d.getMaterialName()); //名称

String type = d.getType();

String subType = d.getSubType();

if(("其它").equals(type)) {

item.put("type", subType); //进出类型

} else {

item.put("type", subType + type); //进出类型

}

item.put("depotName", d.getDepotName()); //仓库名称

item.put("basicNumber", d.getBnum()); //数量

item.put("operTime", Tools.getCenternTime(d.getOtime())); //时间

dataArray.add(item);

}

}

if (list == null) {

objectMap.put("rows", new ArrayList<Object>());

objectMap.put("total", BusinessConstants.DEFAULT\_LIST\_NULL\_NUMBER);

return returnJson(objectMap, "查找不到数据", ErpInfo.OK.code);

}

objectMap.put("rows", dataArray);

objectMap.put("total", depotItemService.findDetailByDepotIdsAndMaterialIdCount(depotIds, sku, mId));

return returnJson(objectMap, ErpInfo.OK.name, ErpInfo.OK.code);

}

/\*\*

\* 根据商品条码和仓库id查询库存数量

\* @param depotId

\* @param barCode

\* @param request

\* @return

\* @throws Exception

\*/

@GetMapping(value = "/findStockByDepotAndBarCode")

@ApiOperation(value = "根据商品条码和仓库id查询库存数量")

public BaseResponseInfo findStockByDepotAndBarCode(

@RequestParam(value = "depotId",required = false) Long depotId,

@RequestParam("barCode") String barCode,

HttpServletRequest request) throws Exception{

BaseResponseInfo res = new BaseResponseInfo();

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

try {

BigDecimal stock = BigDecimal.ZERO;

List<MaterialVo4Unit> list = materialService.getMaterialByBarCode(barCode);

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

MaterialVo4Unit materialVo4Unit = list.get(0);

if(StringUtil.isNotEmpty(materialVo4Unit.getSku())){

stock = depotItemService.getSkuStockByParam(depotId,materialVo4Unit.getMeId(),null,null);

} else {

stock = depotItemService.getStockByParam(depotId,materialVo4Unit.getId(),null,null);

if(materialVo4Unit.getUnitId()!=null) {

Unit unit = unitService.getUnit(materialVo4Unit.getUnitId());

String commodityUnit = materialVo4Unit.getCommodityUnit();

stock = unitService.parseStockByUnit(stock, unit, commodityUnit);

}

}

}

map.put("stock", stock);

res.code = 200;

res.data = map;

} catch (Exception e) {

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 单据明细列表

\* @param headerId

\* @param mpList

\* @param request

\* @return

\* @throws Exception

\*/

@GetMapping(value = "/getDetailList")

@ApiOperation(value = "单据明细列表")

public BaseResponseInfo getDetailList(@RequestParam("headerId") Long headerId,

@RequestParam("mpList") String mpList,

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

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

HttpServletRequest request)throws Exception {

BaseResponseInfo res = new BaseResponseInfo();

try {

List<DepotItemVo4WithInfoEx> dataList = new ArrayList<DepotItemVo4WithInfoEx>();

if(headerId != 0) {

dataList = depotItemService.getDetailList(headerId);

}

String[] mpArr = mpList.split(",");

JSONObject outer = new JSONObject();

outer.put("total", dataList.size());

//存放数据json数组

JSONArray dataArray = new JSONArray();

if (null != dataList) {

BigDecimal totalOperNumber = BigDecimal.ZERO;

BigDecimal totalAllPrice = BigDecimal.ZERO;

BigDecimal totalTaxMoney = BigDecimal.ZERO;

BigDecimal totalTaxLastMoney = BigDecimal.ZERO;

for (DepotItemVo4WithInfoEx diEx : dataList) {

JSONObject item = new JSONObject();

item.put("id", diEx.getId());

item.put("materialExtendId", diEx.getMaterialExtendId() == null ? "" : diEx.getMaterialExtendId());

item.put("barCode", diEx.getBarCode());

item.put("name", diEx.getMName());

item.put("standard", diEx.getMStandard());

item.put("model", diEx.getMModel());

item.put("color", diEx.getMColor());

item.put("materialOther", getOtherInfo(mpArr, diEx));

BigDecimal stock;

Unit unitInfo = materialService.findUnit(diEx.getMaterialId()); //查询计量单位信息

String materialUnit = diEx.getMaterialUnit();

if(StringUtil.isNotEmpty(diEx.getSku())){

stock = depotItemService.getSkuStockByParam(diEx.getDepotId(),diEx.getMaterialExtendId(),null,null);

} else {

stock = depotItemService.getStockByParam(diEx.getDepotId(),diEx.getMaterialId(),null,null);

if (StringUtil.isNotEmpty(unitInfo.getName())) {

stock = unitService.parseStockByUnit(stock, unitInfo, materialUnit);

}

}

item.put("stock", stock);

item.put("unit", diEx.getMaterialUnit());

item.put("snList", diEx.getSnList());

item.put("batchNumber", diEx.getBatchNumber());

item.put("expirationDate", Tools.parseDateToStr(diEx.getExpirationDate()));

item.put("sku", diEx.getSku());

item.put("enableSerialNumber", diEx.getEnableSerialNumber());

item.put("enableBatchNumber", diEx.getEnableBatchNumber());

item.put("operNumber", diEx.getOperNumber());

item.put("basicNumber", diEx.getBasicNumber());

item.put("preNumber", diEx.getOperNumber()); //原数量

item.put("finishNumber", depotItemService.getFinishNumber(diEx.getMaterialExtendId(), diEx.getId(), diEx.getHeaderId(), unitInfo, materialUnit, linkType)); //已入库|已出库

item.put("purchaseDecimal", diEx.getPurchaseDecimal()); //采购价

if("basic".equals(linkType)) {

//正常情况显示金额，而以销定购的情况不能显示金额

item.put("unitPrice", diEx.getUnitPrice());

item.put("taxUnitPrice", diEx.getTaxUnitPrice());

item.put("allPrice", diEx.getAllPrice());

item.put("taxRate", diEx.getTaxRate());

item.put("taxMoney", diEx.getTaxMoney());

item.put("taxLastMoney", diEx.getTaxLastMoney());

}

item.put("remark", diEx.getRemark());

item.put("linkId", diEx.getLinkId());

item.put("depotId", diEx.getDepotId() == null ? "" : diEx.getDepotId());

item.put("depotName", diEx.getDepotId() == null ? "" : diEx.getDepotName());

item.put("anotherDepotId", diEx.getAnotherDepotId() == null ? "" : diEx.getAnotherDepotId());

item.put("anotherDepotName", diEx.getAnotherDepotId() == null ? "" : diEx.getAnotherDepotName());

item.put("mType", diEx.getMaterialType());

item.put("op", 1);

dataArray.add(item);

//合计数据汇总

totalOperNumber = totalOperNumber.add(diEx.getOperNumber()==null?BigDecimal.ZERO:diEx.getOperNumber());

totalAllPrice = totalAllPrice.add(diEx.getAllPrice()==null?BigDecimal.ZERO:diEx.getAllPrice());

totalTaxMoney = totalTaxMoney.add(diEx.getTaxMoney()==null?BigDecimal.ZERO:diEx.getTaxMoney());

totalTaxLastMoney = totalTaxLastMoney.add(diEx.getTaxLastMoney()==null?BigDecimal.ZERO:diEx.getTaxLastMoney());

}

if(StringUtil.isNotEmpty(isReadOnly) && "1".equals(isReadOnly)) {

JSONObject footItem = new JSONObject();

footItem.put("operNumber", totalOperNumber);

footItem.put("allPrice", totalAllPrice);

footItem.put("taxMoney", totalTaxMoney);

footItem.put("taxLastMoney", totalTaxLastMoney);

dataArray.add(footItem);

}

}

outer.put("rows", dataArray);

res.code = 200;

res.data = outer;

} catch (Exception e) {

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 获取扩展信息

\*

\* @return

\*/

public String getOtherInfo(String[] mpArr, DepotItemVo4WithInfoEx diEx)throws Exception {

String materialOther = "";

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

if (mpArr[i].equals("制造商")) {

materialOther = materialOther + ((diEx.getMMfrs() == null || diEx.getMMfrs().equals("")) ? "" : "(" + diEx.getMMfrs() + ")");

}

if (mpArr[i].equals("自定义1")) {

materialOther = materialOther + ((diEx.getMOtherField1() == null || diEx.getMOtherField1().equals("")) ? "" : "(" + diEx.getMOtherField1() + ")");

}

if (mpArr[i].equals("自定义2")) {

materialOther = materialOther + ((diEx.getMOtherField2() == null || diEx.getMOtherField2().equals("")) ? "" : "(" + diEx.getMOtherField2() + ")");

}

if (mpArr[i].equals("自定义3")) {

materialOther = materialOther + ((diEx.getMOtherField3() == null || diEx.getMOtherField3().equals("")) ? "" : "(" + diEx.getMOtherField3() + ")");

}

}

return materialOther;

}

/\*\*

\* 进销存统计

\* @param currentPage

\* @param pageSize

\* @param depotIds

\* @param monthTime

\* @param materialParam

\* @param mpList

\* @param request

\* @return

\* @throws Exception

\*/

@GetMapping(value = "/findByAll")

@ApiOperation(value = "查找所有的明细")

public BaseResponseInfo findByAll(@RequestParam("currentPage") Integer currentPage,

@RequestParam("pageSize") Integer pageSize,

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

@RequestParam("monthTime") String monthTime,

@RequestParam("materialParam") String materialParam,

@RequestParam("mpList") String mpList,

HttpServletRequest request)throws Exception {

BaseResponseInfo res = new BaseResponseInfo();

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

try {

String timeA = Tools.firstDayOfMonth(monthTime) + BusinessConstants.DAY\_FIRST\_TIME;

String timeB = Tools.lastDayOfMonth(monthTime) + BusinessConstants.DAY\_LAST\_TIME;

List<Long> depotList = parseListByDepotIds(depotIds);

List<DepotItemVo4WithInfoEx> dataList = depotItemService.findByAll(StringUtil.toNull(materialParam),

timeB,(currentPage-1)\*pageSize, pageSize);

String[] mpArr = mpList.split(",");

int total = depotItemService.findByAllCount(StringUtil.toNull(materialParam), timeB);

map.put("total", total);

//存放数据json数组

JSONArray dataArray = new JSONArray();

if (null != dataList) {

for (DepotItemVo4WithInfoEx diEx : dataList) {

JSONObject item = new JSONObject();

Long mId = diEx.getMId();

item.put("barCode", diEx.getBarCode());

item.put("materialName", diEx.getMName());

item.put("materialModel", diEx.getMModel());

item.put("materialStandard", diEx.getMStandard());

//扩展信息

String materialOther = getOtherInfo(mpArr, diEx);

item.put("materialOther", materialOther);

item.put("materialColor", diEx.getMColor());

item.put("unitName", diEx.getMaterialUnit());

BigDecimal prevSum = depotItemService.getStockByParamWithDepotList(depotList,mId,null,timeA);

Map<String,BigDecimal> intervalMap = depotItemService.getIntervalMapByParamWithDepotList(depotList,mId,timeA,timeB);

BigDecimal inSum = intervalMap.get("inSum");

BigDecimal outSum = intervalMap.get("outSum");

BigDecimal thisSum = prevSum.add(inSum).subtract(outSum);

item.put("prevSum", prevSum);

item.put("inSum", inSum);

item.put("outSum", outSum);

item.put("thisSum", thisSum);

item.put("unitPrice", diEx.getPurchaseDecimal());

item.put("thisAllPrice", thisSum.multiply(diEx.getPurchaseDecimal()));

dataArray.add(item);

}

}

map.put("rows", dataArray);

res.code = 200;

res.data = map;

} catch (BusinessRunTimeException e) {

res.code = e.getCode();

res.data = e.getData().get("message");

} catch(Exception e){

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 进销存统计总计金额

\* @param depotIds

\* @param monthTime

\* @param materialParam

\* @param request

\* @return

\*/

@GetMapping(value = "/totalCountMoney")

@ApiOperation(value = "统计总计金额")

public BaseResponseInfo totalCountMoney(@RequestParam(value = "depotIds",required = false) String depotIds,

@RequestParam("monthTime") String monthTime,

@RequestParam("materialParam") String materialParam,

HttpServletRequest request) throws Exception{

BaseResponseInfo res = new BaseResponseInfo();

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

try {

String endTime = Tools.lastDayOfMonth(monthTime) + BusinessConstants.DAY\_LAST\_TIME;

List<Long> depotList = parseListByDepotIds(depotIds);

List<DepotItemVo4WithInfoEx> dataList = depotItemService.findByAll(StringUtil.toNull(materialParam),

endTime, null, null);

BigDecimal thisAllPrice = BigDecimal.ZERO;

if (null != dataList) {

for (DepotItemVo4WithInfoEx diEx : dataList) {

Long mId = diEx.getMId();

BigDecimal thisSum = depotItemService.getStockByParamWithDepotList(depotList,mId,null,endTime);

BigDecimal unitPrice = diEx.getPurchaseDecimal();

if(unitPrice == null) {

unitPrice = BigDecimal.ZERO;

}

thisAllPrice = thisAllPrice.add(thisSum.multiply(unitPrice));

}

}

map.put("totalCount", thisAllPrice);

res.code = 200;

res.data = map;

} catch (BusinessRunTimeException e) {

res.code = e.getCode();

res.data = e.getData().get("message");

} catch(Exception e){

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

private List<Long> parseListByDepotIds(@RequestParam("depotIds") String depotIds) throws Exception {

List<Long> depotList = new ArrayList<>();

if(StringUtil.isNotEmpty(depotIds)) {

depotList = StringUtil.strToLongList(depotIds);

} else {

//未选择仓库时默认为当前用户有权限的仓库

JSONArray depotArr = depotService.findDepotByCurrentUser();

for(Object obj: depotArr) {

JSONObject object = JSONObject.parseObject(obj.toString());

depotList.add(object.getLong("id"));

}

//如果有权限的仓库数量太多则提示要选择仓库

if(depotList.size()>20) {

throw new BusinessRunTimeException(ExceptionConstants.REPORT\_TWO\_MANY\_DEPOT\_FAILED\_CODE,

ExceptionConstants.REPORT\_TWO\_MANY\_DEPOT\_FAILED\_MSG);

}

}

return depotList;

}

/\*\*

\* 进货统计

\* @param currentPage

\* @param pageSize

\* @param beginTime

\* @param endTime

\* @param materialParam

\* @param mpList

\* @param request

\* @return

\*/

@GetMapping(value = "/buyIn")

@ApiOperation(value = "进货统计")

public BaseResponseInfo buyIn(@RequestParam("currentPage") Integer currentPage,

@RequestParam("pageSize") Integer pageSize,

@RequestParam("beginTime") String beginTime,

@RequestParam("endTime") String endTime,

@RequestParam("materialParam") String materialParam,

@RequestParam("mpList") String mpList,

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

HttpServletRequest request)throws Exception {

BaseResponseInfo res = new BaseResponseInfo();

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

beginTime = Tools.parseDayToTime(beginTime, BusinessConstants.DAY\_FIRST\_TIME);

endTime = Tools.parseDayToTime(endTime,BusinessConstants.DAY\_LAST\_TIME);

try {

String [] creatorArray = depotHeadService.getCreatorArray(roleType);

List<DepotItemVo4WithInfoEx> dataList = depotItemService.getListWithBugOrSale(StringUtil.toNull(materialParam),

"buy", beginTime, endTime, creatorArray, (currentPage-1)\*pageSize, pageSize);

String[] mpArr = mpList.split(",");

int total = depotItemService.getListWithBugOrSaleCount(StringUtil.toNull(materialParam),

"buy", beginTime, endTime, creatorArray);

map.put("total", total);

//存放数据json数组

JSONArray dataArray = new JSONArray();

if (null != dataList) {

for (DepotItemVo4WithInfoEx diEx : dataList) {

JSONObject item = new JSONObject();

BigDecimal InSum = depotItemService.buyOrSale("入库", "采购", diEx.getMId(), beginTime, endTime, creatorArray, "number");

BigDecimal OutSum = depotItemService.buyOrSale("出库", "采购退货", diEx.getMId(), beginTime, endTime, creatorArray, "number");

BigDecimal InSumPrice = depotItemService.buyOrSale("入库", "采购", diEx.getMId(), beginTime, endTime, creatorArray, "price");

BigDecimal OutSumPrice = depotItemService.buyOrSale("出库", "采购退货", diEx.getMId(), beginTime, endTime, creatorArray, "price");

BigDecimal InOutSumPrice = InSumPrice.subtract(OutSumPrice);

item.put("barCode", diEx.getBarCode());

item.put("materialName", diEx.getMName());

item.put("materialModel", diEx.getMModel());

item.put("materialStandard", diEx.getMStandard());

//扩展信息

String materialOther = getOtherInfo(mpArr, diEx);

item.put("materialOther", materialOther);

item.put("materialColor", diEx.getMColor());

item.put("materialUnit", diEx.getMaterialUnit());

item.put("unitName", diEx.getUnitName());

item.put("inSum", InSum);

item.put("outSum", OutSum);

item.put("inSumPrice", InSumPrice);

item.put("outSumPrice", OutSumPrice);

item.put("inOutSumPrice",InOutSumPrice);//实际采购金额

dataArray.add(item);

}

}

map.put("rows", dataArray);

res.code = 200;

res.data = map;

} catch(Exception e){

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 销售统计

\* @param currentPage

\* @param pageSize

\* @param beginTime

\* @param endTime

\* @param materialParam

\* @param mpList

\* @param request

\* @return

\*/

@GetMapping(value = "/saleOut")

@ApiOperation(value = "销售统计")

public BaseResponseInfo saleOut(@RequestParam("currentPage") Integer currentPage,

@RequestParam("pageSize") Integer pageSize,

@RequestParam("beginTime") String beginTime,

@RequestParam("endTime") String endTime,

@RequestParam("materialParam") String materialParam,

@RequestParam("mpList") String mpList,

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

HttpServletRequest request)throws Exception {

BaseResponseInfo res = new BaseResponseInfo();

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

beginTime = Tools.parseDayToTime(beginTime, BusinessConstants.DAY\_FIRST\_TIME);

endTime = Tools.parseDayToTime(endTime,BusinessConstants.DAY\_LAST\_TIME);

try {

String [] creatorArray = depotHeadService.getCreatorArray(roleType);

List<DepotItemVo4WithInfoEx> dataList = depotItemService.getListWithBugOrSale(StringUtil.toNull(materialParam),

"sale", beginTime, endTime, creatorArray, (currentPage-1)\*pageSize, pageSize);

String[] mpArr = mpList.split(",");

int total = depotItemService.getListWithBugOrSaleCount(StringUtil.toNull(materialParam),

"sale", beginTime, endTime, creatorArray);

map.put("total", total);

//存放数据json数组

JSONArray dataArray = new JSONArray();

if (null != dataList) {

for (DepotItemVo4WithInfoEx diEx : dataList) {

JSONObject item = new JSONObject();

BigDecimal OutSumRetail = depotItemService.buyOrSale("出库", "零售", diEx.getMId(), beginTime, endTime, creatorArray,"number");

BigDecimal OutSum = depotItemService.buyOrSale("出库", "销售", diEx.getMId(), beginTime, endTime, creatorArray,"number");

BigDecimal InSumRetail = depotItemService.buyOrSale("入库", "零售退货", diEx.getMId(), beginTime, endTime, creatorArray,"number");

BigDecimal InSum = depotItemService.buyOrSale("入库", "销售退货", diEx.getMId(), beginTime, endTime, creatorArray,"number");

BigDecimal OutSumRetailPrice = depotItemService.buyOrSale("出库", "零售", diEx.getMId(), beginTime, endTime, creatorArray,"price");

BigDecimal OutSumPrice = depotItemService.buyOrSale("出库", "销售", diEx.getMId(), beginTime, endTime, creatorArray,"price");

BigDecimal InSumRetailPrice = depotItemService.buyOrSale("入库", "零售退货", diEx.getMId(), beginTime, endTime, creatorArray,"price");

BigDecimal InSumPrice = depotItemService.buyOrSale("入库", "销售退货", diEx.getMId(), beginTime, endTime, creatorArray,"price");

BigDecimal OutInSumPrice = (OutSumRetailPrice.add(OutSumPrice)).subtract(InSumRetailPrice.add(InSumPrice));

item.put("barCode", diEx.getBarCode());

item.put("materialName", diEx.getMName());

item.put("materialModel", diEx.getMModel());

item.put("materialStandard", diEx.getMStandard());

//扩展信息

String materialOther = getOtherInfo(mpArr, diEx);

item.put("materialOther", materialOther);

item.put("materialColor", diEx.getMColor());

item.put("materialUnit", diEx.getMaterialUnit());

item.put("unitName", diEx.getUnitName());

item.put("outSum", OutSumRetail.add(OutSum));

item.put("inSum", InSumRetail.add(InSum));

item.put("outSumPrice", OutSumRetailPrice.add(OutSumPrice));

item.put("inSumPrice", InSumRetailPrice.add(InSumPrice));

item.put("outInSumPrice",OutInSumPrice);//实际销售金额

dataArray.add(item);

}

}

map.put("rows", dataArray);

res.code = 200;

res.data = map;

} catch(Exception e){

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 获取单位

\* @param materialUnit

\* @param uName

\* @return

\*/

public String getUName(String materialUnit, String uName) {

String unitName = null;

if(StringUtil.isNotEmpty(materialUnit)) {

unitName = materialUnit;

} else if(StringUtil.isNotEmpty(uName)) {

unitName = uName;

}

return unitName;

}

/\*\*

\* 库存预警报表

\* @param currentPage

\* @param pageSize

\* @return

\*/

@GetMapping(value = "/findStockWarningCount")

@ApiOperation(value = "库存预警报表")

public BaseResponseInfo findStockWarningCount(@RequestParam("currentPage") Integer currentPage,

@RequestParam("pageSize") Integer pageSize,

@RequestParam("materialParam") String materialParam,

@RequestParam(value = "depotId", required = false) Long depotId,

@RequestParam("mpList") String mpList)throws Exception {

BaseResponseInfo res = new BaseResponseInfo();

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

try {

List<Long> depotList = new ArrayList<>();

if(depotId != null) {

depotList.add(depotId);

} else {

//未选择仓库时默认为当前用户有权限的仓库

JSONArray depotArr = depotService.findDepotByCurrentUser();

for(Object obj: depotArr) {

JSONObject object = JSONObject.parseObject(obj.toString());

depotList.add(object.getLong("id"));

}

}

String[] mpArr = mpList.split(",");

List<DepotItemStockWarningCount> list = depotItemService.findStockWarningCount((currentPage-1)\*pageSize, pageSize, materialParam, depotList);

//存放数据json数组

if (null != list) {

for (DepotItemStockWarningCount disw : list) {

DepotItemVo4WithInfoEx diEx = new DepotItemVo4WithInfoEx();

diEx.setMMfrs(disw.getMMfrs());

diEx.setMOtherField1(disw.getMOtherField1());

diEx.setMOtherField2(disw.getMOtherField2());

diEx.setMOtherField3(disw.getMOtherField3());

disw.setMaterialOther(getOtherInfo(mpArr, diEx));

disw.setMaterialUnit(getUName(disw.getMaterialUnit(), disw.getUnitName()));

if(null!=disw.getLowSafeStock() && disw.getCurrentNumber().compareTo(disw.getLowSafeStock())<0) {

disw.setLowCritical(disw.getLowSafeStock().subtract(disw.getCurrentNumber()));

}

if(null!=disw.getHighSafeStock() && disw.getCurrentNumber().compareTo(disw.getHighSafeStock())>0) {

disw.setHighCritical(disw.getCurrentNumber().subtract(disw.getHighSafeStock()));

}

}

}

int total = depotItemService.findStockWarningCountTotal(materialParam, depotList);

map.put("total", total);

map.put("rows", list);

res.code = 200;

res.data = map;

} catch(Exception e){

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 统计采购、销售、零售的总金额

\* @param request

\* @param response

\* @return

\* @throws Exception

\*/

@GetMapping(value = "/buyOrSalePrice")

@ApiOperation(value = "统计采购、销售、零售的总金额")

public BaseResponseInfo buyOrSalePrice(@RequestParam(value = "roleType", required = false) String roleType,

HttpServletRequest request, HttpServletResponse response)throws Exception {

BaseResponseInfo res = new BaseResponseInfo();

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

try {

List<String> list = Tools.getLastMonths(6);

JSONArray buyPriceList = new JSONArray();

for(String month: list) {

JSONObject obj = new JSONObject();

BigDecimal outPrice = depotItemService.inOrOutPrice("入库", "采购", month, roleType);

BigDecimal inPrice = depotItemService.inOrOutPrice("出库", "采购退货", month, roleType);

obj.put("x", month);

obj.put("y", roleService.parsePriceByLimit(outPrice.subtract(inPrice), "buy", "\*\*\*", request));

buyPriceList.add(obj);

}

map.put("buyPriceList", buyPriceList);

JSONArray salePriceList = new JSONArray();

for(String month: list) {

JSONObject obj = new JSONObject();

BigDecimal outPrice = depotItemService.inOrOutPrice("出库", "销售", month, roleType);

BigDecimal inPrice = depotItemService.inOrOutPrice("入库", "销售退货", month, roleType);

obj.put("x", month);

obj.put("y", roleService.parsePriceByLimit(outPrice.subtract(inPrice), "sale", "\*\*\*", request));

salePriceList.add(obj);

}

map.put("salePriceList", salePriceList);

JSONArray retailPriceList = new JSONArray();

for(String month: list) {

JSONObject obj = new JSONObject();

BigDecimal outPrice = depotItemService.inOrOutRetailPrice("出库", "零售", month, roleType);

BigDecimal inPrice = depotItemService.inOrOutRetailPrice("入库", "零售退货", month, roleType);

obj.put("x", month);

obj.put("y", roleService.parsePriceByLimit(outPrice.subtract(inPrice), "retail", "\*\*\*", request));

retailPriceList.add(obj);

}

map.put("retailPriceList", retailPriceList);

res.code = 200;

res.data = map;

} catch (Exception e) {

e.printStackTrace();

res.code = 500;

res.data = "统计失败";

}

return res;

}

/\*\*

\* 获取批次商品列表信息

\* @param request

\* @return

\*/

@GetMapping(value = "/getBatchNumberList")

@ApiOperation(value = "获取批次商品列表信息")

public BaseResponseInfo getBatchNumberList(@RequestParam("name") String name,

@RequestParam("depotItemId") Long depotItemId,

@RequestParam("depotId") Long depotId,

@RequestParam("barCode") String barCode,

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

HttpServletRequest request) throws Exception{

BaseResponseInfo res = new BaseResponseInfo();

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

try {

String number = "";

if(depotItemId != null) {

DepotItem depotItem = depotItemService.getDepotItem(depotItemId);

number = depotHeadService.getDepotHead(depotItem.getHeaderId()).getNumber();

}

List<DepotItemVoBatchNumberList> reslist = new ArrayList<>();

List<DepotItemVoBatchNumberList> list = depotItemService.getBatchNumberList(number, name, depotId, barCode, batchNumber);

for(DepotItemVoBatchNumberList bn: list) {

if(bn.getTotalNum()!=null && bn.getTotalNum().compareTo(BigDecimal.ZERO)>0) {

reslist.add(bn);

}

bn.setExpirationDateStr(Tools.parseDateToStr(bn.getExpirationDate()));

}

map.put("rows", reslist);

map.put("total", reslist.size());

res.code = 200;

res.data = map;

} catch (Exception e) {

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

}

package com.erp.controller;

import com.alibaba.fastjson.JSONArray;

import com.alibaba.fastjson.JSONObject;

import com.erp.constants.BusinessConstants;

import com.erp.constants.ExceptionConstants;

import com.erp.datasource.entities.Depot;

import com.erp.datasource.entities.DepotEx;

import com.erp.datasource.entities.MaterialCurrentStock;

import com.erp.datasource.entities.MaterialInitialStock;

import com.erp.exception.BusinessRunTimeException;

import com.erp.service.depot.DepotService;

import com.erp.service.material.MaterialService;

import com.erp.service.systemConfig.SystemConfigService;

import com.erp.service.user.UserService;

import com.erp.service.userBusiness.UserBusinessService;

import com.erp.utils.\*;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.dao.DataAccessException;

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

import javax.annotation.Resource;

import javax.servlet.http.HttpServletRequest;

import java.io.IOException;

import java.math.BigDecimal;

import java.util.\*;

import static com.erp.utils.ResponseJsonUtil.returnJson;

@RestController

@RequestMapping(value = "/depot")

@Api(tags = {"仓库管理"})

public class DepotController {

private Logger logger = LoggerFactory.getLogger(DepotController.class);

@Resource

private DepotService depotService;

@Resource

private UserBusinessService userBusinessService;

@Resource

private MaterialService materialService;

/\*\*

\* 仓库列表

\* @param request

\* @return

\* @throws Exception

\*/

@GetMapping(value = "/getAllList")

@ApiOperation(value = "仓库列表")

public BaseResponseInfo getAllList(HttpServletRequest request) throws Exception{

BaseResponseInfo res = new BaseResponseInfo();

try {

List<Depot> depotList = depotService.getAllList();

res.code = 200;

res.data = depotList;

} catch(Exception e){

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 用户对应仓库显示

\* @param type

\* @param keyId

\* @param request

\* @return

\*/

@GetMapping(value = "/findUserDepot")

@ApiOperation(value = "用户对应仓库显示")

public JSONArray findUserDepot(@RequestParam("UBType") String type, @RequestParam("UBKeyId") String keyId,

HttpServletRequest request) throws Exception{

JSONArray arr = new JSONArray();

try {

//获取权限信息

String ubValue = userBusinessService.getUBValueByTypeAndKeyId(type, keyId);

List<Depot> dataList = depotService.findUserDepot();

//开始拼接json数据

JSONObject outer = new JSONObject();

outer.put("id", 0);

outer.put("key", 0);

outer.put("value", 0);

outer.put("title", "仓库列表");

outer.put("attributes", "仓库列表");

//存放数据json数组

JSONArray dataArray = new JSONArray();

if (null != dataList) {

for (Depot depot : dataList) {

JSONObject item = new JSONObject();

item.put("id", depot.getId());

item.put("key", depot.getId());

item.put("value", depot.getId());

item.put("title", depot.getName());

item.put("attributes", depot.getName());

Boolean flag = ubValue.contains("[" + depot.getId().toString() + "]");

if (flag) {

item.put("checked", true);

}

dataArray.add(item);

}

}

outer.put("children", dataArray);

arr.add(outer);

} catch (Exception e) {

e.printStackTrace();

}

return arr;

}

/\*\*

\* 获取当前用户拥有权限的仓库列表

\* @param request

\* @return

\* @throws Exception

\*/

@GetMapping(value = "/findDepotByCurrentUser")

@ApiOperation(value = "获取当前用户拥有权限的仓库列表")

public BaseResponseInfo findDepotByCurrentUser(HttpServletRequest request) throws Exception{

BaseResponseInfo res = new BaseResponseInfo();

try {

JSONArray arr = depotService.findDepotByCurrentUser();

res.code = 200;

res.data = arr;

} catch (Exception e) {

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 更新默认仓库

\* @param object

\* @param request

\* @return

\* @throws Exception

\*/

@PostMapping(value = "/updateIsDefault")

@ApiOperation(value = "更新默认仓库")

public String updateIsDefault(@RequestBody JSONObject object,

HttpServletRequest request) throws Exception{

Long depotId = object.getLong("id");

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

int res = depotService.updateIsDefault(depotId);

if(res > 0) {

return returnJson(objectMap, ErpInfo.OK.name, ErpInfo.OK.code);

} else {

return returnJson(objectMap, ErpInfo.ERROR.name, ErpInfo.ERROR.code);

}

}

/\*\*

\* 仓库列表-带库存

\* @param mId

\* @param request

\* @return

\*/

@GetMapping(value = "/getAllListWithStock")

@ApiOperation(value = "仓库列表-带库存")

public BaseResponseInfo getAllList(@RequestParam("mId") Long mId,

HttpServletRequest request) {

BaseResponseInfo res = new BaseResponseInfo();

try {

List<Depot> list = depotService.getAllList();

List<DepotEx> depotList = new ArrayList<DepotEx>();

for(Depot depot: list) {

DepotEx de = new DepotEx();

if(mId!=0) {

BigDecimal initStock = materialService.getInitStock(mId, depot.getId());

BigDecimal currentStock = materialService.getCurrentStockByMaterialIdAndDepotId(mId, depot.getId());

de.setInitStock(initStock);

de.setCurrentStock(currentStock);

MaterialInitialStock materialInitialStock = materialService.getSafeStock(mId, depot.getId());

de.setLowSafeStock(materialInitialStock.getLowSafeStock());

de.setHighSafeStock(materialInitialStock.getHighSafeStock());

} else {

de.setInitStock(BigDecimal.ZERO);

de.setCurrentStock(BigDecimal.ZERO);

}

de.setId(depot.getId());

de.setName(depot.getName());

depotList.add(de);

}

res.code = 200;

res.data = depotList;

} catch(Exception e){

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 批量设置状态-启用或者禁用

\* @param jsonObject

\* @param request

\* @return

\*/

@PostMapping(value = "/batchSetStatus")

@ApiOperation(value = "批量设置状态")

public String batchSetStatus(@RequestBody JSONObject jsonObject,

HttpServletRequest request)throws Exception {

Boolean status = jsonObject.getBoolean("status");

String ids = jsonObject.getString("ids");

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

int res = depotService.batchSetStatus(status, ids);

if(res > 0) {

return returnJson(objectMap, ErpInfo.OK.name, ErpInfo.OK.code);

} else {

return returnJson(objectMap, ErpInfo.ERROR.name, ErpInfo.ERROR.code);

}

}

}

package com.erp.controller;

import com.alibaba.fastjson.JSONArray;

import com.alibaba.fastjson.JSONObject;

import com.erp.constants.BusinessConstants;

import com.erp.constants.ExceptionConstants;

import com.erp.datasource.entities.DepotHead;

import com.erp.datasource.entities.DepotHeadVo4Body;

import com.erp.datasource.entities.Supplier;

import com.erp.datasource.vo.DepotHeadVo4InDetail;

import com.erp.datasource.vo.DepotHeadVo4InOutMCount;

import com.erp.datasource.vo.DepotHeadVo4List;

import com.erp.datasource.vo.DepotHeadVo4StatementAccount;

import com.erp.exception.BusinessParamCheckingException;

import com.erp.service.accountHead.AccountHeadService;

import com.erp.service.depot.DepotService;

import com.erp.service.depotHead.DepotHeadService;

import com.erp.service.log.LogService;

import com.erp.service.redis.RedisService;

import com.erp.service.supplier.SupplierService;

import com.erp.utils.\*;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

import javax.annotation.Resource;

import javax.servlet.http.HttpServletRequest;

import java.math.BigDecimal;

import java.sql.Date;

import java.text.DecimalFormat;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import static com.erp.utils.ResponseJsonUtil.returnJson;

import static com.erp.utils.Tools.getNow3;

@RestController

@RequestMapping(value = "/depotHead")

@Api(tags = {"单据管理"})

public class DepotHeadController {

private Logger logger = LoggerFactory.getLogger(DepotHeadController.class);

@Resource

private DepotHeadService depotHeadService;

@Resource

private DepotService depotService;

@Resource

private RedisService redisService;

/\*\*

\* 批量设置状态-审核或者反审核

\* @param jsonObject

\* @param request

\* @return

\*/

@PostMapping(value = "/batchSetStatus")

@ApiOperation(value = "批量设置状态-审核或者反审核")

public String batchSetStatus(@RequestBody JSONObject jsonObject,

HttpServletRequest request) throws Exception{

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

String status = jsonObject.getString("status");

String ids = jsonObject.getString("ids");

int res = depotHeadService.batchSetStatus(status, ids);

if(res > 0) {

return returnJson(objectMap, ErpInfo.OK.name, ErpInfo.OK.code);

} else {

return returnJson(objectMap, ErpInfo.ERROR.name, ErpInfo.ERROR.code);

}

}

/\*\*

\* 入库出库明细接口

\* @param currentPage

\* @param pageSize

\* @param oId

\* @param number

\* @param materialParam

\* @param depotId

\* @param beginTime

\* @param endTime

\* @param type

\* @param request

\* @return

\*/

@GetMapping(value = "/findInOutDetail")

@ApiOperation(value = "入库出库明细接口")

public BaseResponseInfo findInOutDetail(@RequestParam("currentPage") Integer currentPage,

@RequestParam("pageSize") Integer pageSize,

@RequestParam(value = "organId", required = false) Integer oId,

@RequestParam("number") String number,

@RequestParam("materialParam") String materialParam,

@RequestParam(value = "depotId", required = false) Long depotId,

@RequestParam("beginTime") String beginTime,

@RequestParam("endTime") String endTime,

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

@RequestParam("type") String type,

@RequestParam("remark") String remark,

HttpServletRequest request)throws Exception {

BaseResponseInfo res = new BaseResponseInfo();

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

try {

List<Long> depotList = new ArrayList<>();

if(depotId != null) {

depotList.add(depotId);

} else {

//未选择仓库时默认为当前用户有权限的仓库

JSONArray depotArr = depotService.findDepotByCurrentUser();

for(Object obj: depotArr) {

JSONObject object = JSONObject.parseObject(obj.toString());

depotList.add(object.getLong("id"));

}

}

List<DepotHeadVo4InDetail> resList = new ArrayList<DepotHeadVo4InDetail>();

String [] creatorArray = depotHeadService.getCreatorArray(roleType);

String subType = "出库".equals(type)? "销售" : "";

String [] organArray = depotHeadService.getOrganArray(subType, "");

beginTime = Tools.parseDayToTime(beginTime, BusinessConstants.DAY\_FIRST\_TIME);

endTime = Tools.parseDayToTime(endTime,BusinessConstants.DAY\_LAST\_TIME);

List<DepotHeadVo4InDetail> list = depotHeadService.findInOutDetail(beginTime, endTime, type, creatorArray, organArray,

StringUtil.toNull(materialParam), depotList, oId, StringUtil.toNull(number), remark, (currentPage-1)\*pageSize, pageSize);

int total = depotHeadService.findInOutDetailCount(beginTime, endTime, type, creatorArray, organArray,

StringUtil.toNull(materialParam), depotList, oId, StringUtil.toNull(number), remark);

map.put("total", total);

//存放数据json数组

if (null != list) {

for (DepotHeadVo4InDetail dhd : list) {

resList.add(dhd);

}

}

map.put("rows", resList);

res.code = 200;

res.data = map;

} catch(Exception e){

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 入库出库统计接口

\* @param currentPage

\* @param pageSize

\* @param oId

\* @param materialParam

\* @param depotId

\* @param beginTime

\* @param endTime

\* @param type

\* @param request

\* @return

\*/

@GetMapping(value = "/findInOutMaterialCount")

@ApiOperation(value = "入库出库统计接口")

public BaseResponseInfo findInOutMaterialCount(@RequestParam("currentPage") Integer currentPage,

@RequestParam("pageSize") Integer pageSize,

@RequestParam(value = "organId", required = false) Integer oId,

@RequestParam("materialParam") String materialParam,

@RequestParam(value = "depotId", required = false) Long depotId,

@RequestParam("beginTime") String beginTime,

@RequestParam("endTime") String endTime,

@RequestParam("type") String type,

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

HttpServletRequest request)throws Exception {

BaseResponseInfo res = new BaseResponseInfo();

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

try {

List<Long> depotList = new ArrayList<>();

if(depotId != null) {

depotList.add(depotId);

} else {

//未选择仓库时默认为当前用户有权限的仓库

JSONArray depotArr = depotService.findDepotByCurrentUser();

for(Object obj: depotArr) {

JSONObject object = JSONObject.parseObject(obj.toString());

depotList.add(object.getLong("id"));

}

}

beginTime = Tools.parseDayToTime(beginTime,BusinessConstants.DAY\_FIRST\_TIME);

endTime = Tools.parseDayToTime(endTime,BusinessConstants.DAY\_LAST\_TIME);

List<DepotHeadVo4InOutMCount> list = depotHeadService.findInOutMaterialCount(beginTime, endTime, type, StringUtil.toNull(materialParam),

depotList, oId, roleType, (currentPage-1)\*pageSize, pageSize);

int total = depotHeadService.findInOutMaterialCountTotal(beginTime, endTime, type, StringUtil.toNull(materialParam),

depotList, oId, roleType);

map.put("total", total);

map.put("rows", list);

res.code = 200;

res.data = map;

} catch(Exception e){

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 调拨明细统计

\* @param currentPage

\* @param pageSize

\* @param number

\* @param materialParam

\* @param depotIdF 调出仓库

\* @param depotId 调入仓库

\* @param beginTime

\* @param endTime

\* @param subType

\* @param request

\* @return

\*/

@GetMapping(value = "/findAllocationDetail")

@ApiOperation(value = "调拨明细统计")

public BaseResponseInfo findallocationDetail(@RequestParam("currentPage") Integer currentPage,

@RequestParam("pageSize") Integer pageSize,

@RequestParam("number") String number,

@RequestParam("materialParam") String materialParam,

@RequestParam(value = "depotId", required = false) Long depotId,

@RequestParam(value = "depotIdF", required = false) Long depotIdF,

@RequestParam("beginTime") String beginTime,

@RequestParam("endTime") String endTime,

@RequestParam("subType") String subType,

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

@RequestParam("remark") String remark,

HttpServletRequest request)throws Exception {

BaseResponseInfo res = new BaseResponseInfo();

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

try {

List<Long> depotList = new ArrayList<>();

List<Long> depotFList = new ArrayList<>();

if(depotId != null) {

depotList.add(depotId);

} else {

//未选择仓库时默认为当前用户有权限的仓库

JSONArray depotArr = depotService.findDepotByCurrentUser();

for(Object obj: depotArr) {

JSONObject object = JSONObject.parseObject(obj.toString());

depotList.add(object.getLong("id"));

}

}

if(depotIdF != null) {

depotFList.add(depotIdF);

} else {

//未选择仓库时默认为当前用户有权限的仓库

JSONArray depotArr = depotService.findDepotByCurrentUser();

for(Object obj: depotArr) {

JSONObject object = JSONObject.parseObject(obj.toString());

depotFList.add(object.getLong("id"));

}

}

String [] creatorArray = depotHeadService.getCreatorArray(roleType);

beginTime = Tools.parseDayToTime(beginTime, BusinessConstants.DAY\_FIRST\_TIME);

endTime = Tools.parseDayToTime(endTime,BusinessConstants.DAY\_LAST\_TIME);

List<DepotHeadVo4InDetail> list = depotHeadService.findAllocationDetail(beginTime, endTime, subType, StringUtil.toNull(number),

creatorArray, StringUtil.toNull(materialParam), depotList, depotFList, remark, (currentPage-1)\*pageSize, pageSize);

int total = depotHeadService.findAllocationDetailCount(beginTime, endTime, subType, StringUtil.toNull(number),

creatorArray, StringUtil.toNull(materialParam), depotList, depotFList, remark);

map.put("rows", list);

map.put("total", total);

res.code = 200;

res.data = map;

} catch(Exception e){

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 对账单接口

\* @param currentPage

\* @param pageSize

\* @param beginTime

\* @param endTime

\* @param organId

\* @param supplierType

\* @param request

\* @return

\*/

@GetMapping(value = "/getStatementAccount")

@ApiOperation(value = "对账单接口")

public BaseResponseInfo getStatementAccount(@RequestParam("currentPage") Integer currentPage,

@RequestParam("pageSize") Integer pageSize,

@RequestParam("beginTime") String beginTime,

@RequestParam("endTime") String endTime,

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

@RequestParam("supplierType") String supplierType,

HttpServletRequest request) throws Exception{

BaseResponseInfo res = new BaseResponseInfo();

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

try {

String type = "";

String subType = "";

if (("供应商").equals(supplierType)) {

type = "入库";

subType = "采购";

} else if (("客户").equals(supplierType)) {

type = "出库";

subType = "销售";

}

String [] organArray = depotHeadService.getOrganArray(subType, "");

beginTime = Tools.parseDayToTime(beginTime,BusinessConstants.DAY\_FIRST\_TIME);

endTime = Tools.parseDayToTime(endTime,BusinessConstants.DAY\_LAST\_TIME);

List<DepotHeadVo4StatementAccount> list = depotHeadService.getStatementAccount(beginTime, endTime, organId, organArray,

supplierType, type, subType, (currentPage-1)\*pageSize, pageSize);

int total = depotHeadService.getStatementAccountCount(beginTime, endTime, organId, organArray,

supplierType, type, subType);

for(DepotHeadVo4StatementAccount item: list) {

BigDecimal preNeed = item.getBeginNeed().add(item.getPreDebtMoney()).subtract(item.getPreBackMoney());

item.setPreNeed(preNeed);

BigDecimal allNeedGet = preNeed.add(item.getDebtMoney()).subtract(item.getBackMoney());

item.setAllNeed(allNeedGet);

}

map.put("rows", list);

map.put("total", total);

List<DepotHeadVo4StatementAccount> totalPayList = depotHeadService.getStatementAccountTotalPay(beginTime, endTime, organId, organArray, supplierType, type, subType);

if(totalPayList.size()>0) {

DepotHeadVo4StatementAccount totalPayItem = totalPayList.get(0);

BigDecimal firstMoney = BigDecimal.ZERO;

BigDecimal lastMoney = BigDecimal.ZERO;

if(totalPayItem!=null) {

firstMoney = totalPayItem.getBeginNeed().add(totalPayItem.getPreDebtMoney()).subtract(totalPayItem.getPreBackMoney());

lastMoney = firstMoney.add(totalPayItem.getDebtMoney()).subtract(totalPayItem.getBackMoney());

}

map.put("firstMoney", firstMoney); //期初

map.put("lastMoney", lastMoney); //期末

}

res.code = 200;

res.data = map;

} catch(Exception e){

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 根据编号查询单据信息

\* @param number

\* @param request

\* @return

\*/

@GetMapping(value = "/getDetailByNumber")

@ApiOperation(value = "根据编号查询单据信息")

public BaseResponseInfo getDetailByNumber(@RequestParam("number") String number,

HttpServletRequest request)throws Exception {

BaseResponseInfo res = new BaseResponseInfo();

DepotHeadVo4List dhl = new DepotHeadVo4List();

try {

List<DepotHeadVo4List> list = depotHeadService.getDetailByNumber(number);

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

dhl = list.get(0);

}

res.code = 200;

res.data = dhl;

} catch(Exception e){

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 新增单据主表及单据子表信息

\* @param body

\* @param request

\* @return

\* @throws Exception

\*/

@PostMapping(value = "/addDepotHeadAndDetail")

@ApiOperation(value = "新增单据主表及单据子表信息")

public Object addDepotHeadAndDetail(@RequestBody DepotHeadVo4Body body, HttpServletRequest request) throws Exception{

JSONObject result = ExceptionConstants.standardSuccess();

String beanJson = body.getInfo();

String rows = body.getRows();

depotHeadService.addDepotHeadAndDetail(beanJson, rows, request);

return result;

}

/\*\*

\* 更新单据主表及单据子表信息

\* @param body

\* @param request

\* @return

\* @throws Exception

\*/

@PutMapping(value = "/updateDepotHeadAndDetail")

@ApiOperation(value = "更新单据主表及单据子表信息")

public Object updateDepotHeadAndDetail(@RequestBody DepotHeadVo4Body body, HttpServletRequest request) throws Exception{

JSONObject result = ExceptionConstants.standardSuccess();

String beanJson = body.getInfo();

String rows = body.getRows();

depotHeadService.updateDepotHeadAndDetail(beanJson,rows,request);

return result;

}

/\*\*

\* 统计今日采购额、昨日采购额、本月采购额、今年采购额|销售额|零售额

\* @param request

\* @return

\*/

@GetMapping(value = "/getBuyAndSaleStatistics")

@ApiOperation(value = "统计今日采购额、昨日采购额、本月采购额、今年采购额|销售额|零售额")

public BaseResponseInfo getBuyAndSaleStatistics(@RequestParam(value = "roleType", required = false) String roleType,

HttpServletRequest request) {

BaseResponseInfo res = new BaseResponseInfo();

try {

String today = Tools.getNow() + BusinessConstants.DAY\_FIRST\_TIME;

String monthFirstDay = Tools.firstDayOfMonth(Tools.getCurrentMonth()) + BusinessConstants.DAY\_FIRST\_TIME;

String yesterdayBegin = Tools.getYesterday() + BusinessConstants.DAY\_FIRST\_TIME;

String yesterdayEnd = Tools.getYesterday() + BusinessConstants.DAY\_LAST\_TIME;

String yearBegin = Tools.getYearBegin() + BusinessConstants.DAY\_FIRST\_TIME;

String yearEnd = Tools.getYearEnd() + BusinessConstants.DAY\_LAST\_TIME;

Map<String, Object> map = depotHeadService.getBuyAndSaleStatistics(today, monthFirstDay,

yesterdayBegin, yesterdayEnd, yearBegin, yearEnd, roleType, request);

res.code = 200;

res.data = map;

} catch(Exception e){

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 根据当前用户获取操作员数组，用于控制当前用户的数据权限，限制可以看到的单据范围

\* 注意：该接口提供给部分插件使用，勿删

\* @param request

\* @return

\*/

@GetMapping(value = "/getCreatorByCurrentUser")

@ApiOperation(value = "根据当前用户获取操作员数组")

public BaseResponseInfo getCreatorByRoleType(HttpServletRequest request) {

BaseResponseInfo res = new BaseResponseInfo();

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

try {

String creator = "";

String roleType = redisService.getObjectFromSessionByKey(request,"roleType").toString();

if(StringUtil.isNotEmpty(roleType)) {

creator = depotHeadService.getCreatorByRoleType(roleType);

}

res.code = 200;

res.data = creator;

} catch (Exception e) {

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 查询存在欠款的单据

\* @param search

\* @param request

\* @return

\* @throws Exception

\*/

@GetMapping(value = "/debtList")

@ApiOperation(value = "查询存在欠款的单据")

public String debtList(@RequestParam(value = Constants.SEARCH, required = false) String search,

HttpServletRequest request)throws Exception {

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

String organIdStr = StringUtil.getInfo(search, "organId");

Long organId = Long.parseLong(organIdStr);

String materialParam = StringUtil.getInfo(search, "materialParam");

String number = StringUtil.getInfo(search, "number");

String beginTime = StringUtil.getInfo(search, "beginTime");

String endTime = StringUtil.getInfo(search, "endTime");

String type = StringUtil.getInfo(search, "type");

String subType = StringUtil.getInfo(search, "subType");

String roleType = StringUtil.getInfo(search, "roleType");

String status = StringUtil.getInfo(search, "status");

List<DepotHeadVo4List> list = depotHeadService.debtList(organId, materialParam, number, beginTime, endTime, type, subType, roleType, status);

if (list != null) {

objectMap.put("rows", list);

return returnJson(objectMap, ErpInfo.OK.name, ErpInfo.OK.code);

} else {

objectMap.put("rows", new ArrayList<>());

return returnJson(objectMap, "查找不到数据", ErpInfo.OK.code);

}

}

}

package com.erp.controller;

import com.alibaba.fastjson.JSONArray;

import com.alibaba.fastjson.JSONObject;

import com.erp.datasource.entities.MaterialInitialStockWithMaterial;

import com.erp.datasource.entities.MaterialVo4Unit;

import com.erp.datasource.entities.Unit;

import com.erp.service.depot.DepotService;

import com.erp.service.depotItem.DepotItemService;

import com.erp.service.material.MaterialService;

import com.erp.service.redis.RedisService;

import com.erp.service.unit.UnitService;

import com.erp.utils.\*;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.web.multipart.MultipartFile;

import javax.annotation.Resource;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.io.File;

import java.math.BigDecimal;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import static com.erp.utils.ResponseJsonUtil.returnJson;

@RestController

@RequestMapping(value = "/material")

@Api(tags = {"商品管理"})

public class MaterialController {

private Logger logger = LoggerFactory.getLogger(MaterialController.class);

@Resource

private MaterialService materialService;

@Resource

private DepotItemService depotItemService;

@Resource

private UnitService unitService;

@Resource

private DepotService depotService;

@Resource

private RedisService redisService;

/\*\*

\* 检查商品是否存在

\* @param id

\* @param name

\* @param model

\* @param color

\* @param standard

\* @param mfrs

\* @param otherField1

\* @param otherField2

\* @param otherField3

\* @param unit

\* @param unitId

\* @param request

\* @return

\* @throws Exception

\*/

@GetMapping(value = "/checkIsExist")

@ApiOperation(value = "检查商品是否存在")

public String checkIsExist(@RequestParam("id") Long id, @RequestParam("name") String name,

@RequestParam("model") String model, @RequestParam("color") String color,

@RequestParam("standard") String standard, @RequestParam("mfrs") String mfrs,

@RequestParam("otherField1") String otherField1, @RequestParam("otherField2") String otherField2,

@RequestParam("otherField3") String otherField3, @RequestParam("unit") String unit,@RequestParam("unitId") Long unitId,

HttpServletRequest request)throws Exception {

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

int exist = materialService.checkIsExist(id, name, StringUtil.toNull(model), StringUtil.toNull(color),

StringUtil.toNull(standard), StringUtil.toNull(mfrs), StringUtil.toNull(otherField1),

StringUtil.toNull(otherField2), StringUtil.toNull(otherField3), StringUtil.toNull(unit), unitId);

if(exist > 0) {

objectMap.put("status", true);

} else {

objectMap.put("status", false);

}

return returnJson(objectMap, ErpInfo.OK.name, ErpInfo.OK.code);

}

/\*\*

\* 批量设置状态-启用或者禁用

\* @param jsonObject

\* @param request

\* @return

\* @throws Exception

\*/

@PostMapping(value = "/batchSetStatus")

@ApiOperation(value = "批量设置状态-启用或者禁用")

public String batchSetStatus(@RequestBody JSONObject jsonObject,

HttpServletRequest request)throws Exception {

Boolean status = jsonObject.getBoolean("status");

String ids = jsonObject.getString("ids");

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

int res = materialService.batchSetStatus(status, ids);

if(res > 0) {

return returnJson(objectMap, ErpInfo.OK.name, ErpInfo.OK.code);

} else {

return returnJson(objectMap, ErpInfo.ERROR.name, ErpInfo.ERROR.code);

}

}

/\*\*

\* 根据id来查询商品名称

\* @param id

\* @param request

\* @return

\*/

@GetMapping(value = "/findById")

@ApiOperation(value = "根据id来查询商品名称")

public BaseResponseInfo findById(@RequestParam("id") Long id, HttpServletRequest request) throws Exception{

BaseResponseInfo res = new BaseResponseInfo();

try {

List<MaterialVo4Unit> list = materialService.findById(id);

res.code = 200;

res.data = list;

} catch(Exception e){

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 根据meId来查询商品名称

\* @param meId

\* @param request

\* @return

\*/

@GetMapping(value = "/findByIdWithBarCode")

@ApiOperation(value = "根据meId来查询商品名称")

public BaseResponseInfo findByIdWithBarCode(@RequestParam("meId") Long meId,

@RequestParam("mpList") String mpList,

HttpServletRequest request) throws Exception{

BaseResponseInfo res = new BaseResponseInfo();

try {

String[] mpArr = mpList.split(",");

MaterialVo4Unit mu = new MaterialVo4Unit();

List<MaterialVo4Unit> list = materialService.findByIdWithBarCode(meId);

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

mu = list.get(0);

mu.setMaterialOther(materialService.getMaterialOtherByParam(mpArr, mu));

}

res.code = 200;

res.data = mu;

} catch(Exception e){

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 查找商品信息-下拉框

\* @param mpList

\* @param request

\* @return

\*/

@GetMapping(value = "/findBySelect")

@ApiOperation(value = "查找商品信息")

public JSONObject findBySelect(@RequestParam(value = "categoryId", required = false) Long categoryId,

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

@RequestParam("mpList") String mpList,

@RequestParam(value = "depotId", required = false) Long depotId,

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

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

@RequestParam("page") Integer currentPage,

@RequestParam("rows") Integer pageSize,

HttpServletRequest request) throws Exception{

JSONObject object = new JSONObject();

try {

List<MaterialVo4Unit> dataList = materialService.findBySelectWithBarCode(categoryId, q, enableSerialNumber,

enableBatchNumber, (currentPage-1)\*pageSize, pageSize);

String[] mpArr = mpList.split(",");

int total = materialService.findBySelectWithBarCodeCount(categoryId, q, enableSerialNumber,

enableBatchNumber);

object.put("total", total);

JSONArray dataArray = new JSONArray();

//存放数据json数组

if (null != dataList) {

for (MaterialVo4Unit material : dataList) {

JSONObject item = new JSONObject();

item.put("id", material.getMeId()); //商品扩展表的id

String ratioStr = ""; //比例

Unit unit = new Unit();

if (material.getUnitId() == null) {

ratioStr = "";

} else {

unit = unitService.getUnit(material.getUnitId());

//拼接副单位的比例

String commodityUnit = material.getCommodityUnit();

if(commodityUnit.equals(unit.getBasicUnit())) {

ratioStr = "[基本]";

}

if(commodityUnit.equals(unit.getOtherUnit())) {

ratioStr = "[" + unit.getRatio() + unit.getBasicUnit() + "]";

}

if(commodityUnit.equals(unit.getOtherUnitTwo())) {

ratioStr = "[" + unit.getRatioTwo() + unit.getBasicUnit() + "]";

}

if(commodityUnit.equals(unit.getOtherUnitThree())) {

ratioStr = "[" + unit.getRatioThree() + unit.getBasicUnit() + "]";

}

}

item.put("mBarCode", material.getmBarCode());

item.put("name", material.getName());

item.put("categoryName", material.getCategoryName());

item.put("standard", material.getStandard());

item.put("model", material.getModel());

item.put("color", material.getColor());

item.put("unit", material.getCommodityUnit() + ratioStr);

item.put("sku", material.getSku());

item.put("enableSerialNumber", material.getEnableSerialNumber());

item.put("enableBatchNumber", material.getEnableBatchNumber());

BigDecimal stock;

if(StringUtil.isNotEmpty(material.getSku())){

stock = depotItemService.getSkuStockByParam(depotId,material.getMeId(),null,null);

} else {

stock = depotItemService.getStockByParam(depotId,material.getId(),null,null);

if (material.getUnitId()!=null){

String commodityUnit = material.getCommodityUnit();

stock = unitService.parseStockByUnit(stock, unit, commodityUnit);

}

}

item.put("stock", stock);

item.put("expand", materialService.getMaterialOtherByParam(mpArr, material));

item.put("imgName", material.getImgName());

dataArray.add(item);

}

}

object.put("rows", dataArray);

} catch (Exception e) {

e.printStackTrace();

}

return object;

}

/\*\*

\* 根据商品id查找商品信息

\* @param meId

\* @param request

\* @return

\* @throws Exception

\*/

@GetMapping(value = "/getMaterialByMeId")

@ApiOperation(value = "根据商品id查找商品信息")

public JSONObject getMaterialByMeId(@RequestParam(value = "meId", required = false) Long meId,

@RequestParam("mpList") String mpList,

HttpServletRequest request) throws Exception{

JSONObject item = new JSONObject();

try {

String[] mpArr = mpList.split(",");

List<MaterialVo4Unit> materialList = materialService.getMaterialByMeId(meId);

if(materialList!=null && materialList.size()!=1) {

return item;

} else if(materialList.size() == 1) {

MaterialVo4Unit material = materialList.get(0);

item.put("Id", material.getMeId()); //商品扩展表的id

String ratio; //比例

if (material.getUnitId() == null || material.getUnitId().equals("")) {

ratio = "";

} else {

ratio = material.getUnitName();

ratio = ratio.substring(ratio.indexOf("("));

}

//名称/型号/扩展信息/包装

String MaterialName = "";

MaterialName = MaterialName + material.getmBarCode() + "\_" + material.getName()

+ ((material.getStandard() == null || material.getStandard().equals("")) ? "" : "(" + material.getStandard() + ")");

String expand = materialService.getMaterialOtherByParam(mpArr, material); //扩展信息

MaterialName = MaterialName + expand + ((material.getUnit() == null || material.getUnit().equals("")) ? "" : "(" + material.getUnit() + ")") + ratio;

item.put("MaterialName", MaterialName);

item.put("name", material.getName());

item.put("expand", expand);

item.put("model", material.getModel());

item.put("standard", material.getStandard());

item.put("unit", material.getUnit() + ratio);

}

} catch (Exception e) {

e.printStackTrace();

}

return item;

}

/\*\*

\* 生成excel表格

\* @param categoryId

\* @param materialParam

\* @param color

\* @param weight

\* @param expiryNum

\* @param enabled

\* @param enableSerialNumber

\* @param enableBatchNumber

\* @param remark

\* @param mpList

\* @param request

\* @param response

\*/

@GetMapping(value = "/exportExcel")

@ApiOperation(value = "生成excel表格")

public void exportExcel(@RequestParam(value = "categoryId", required = false) String categoryId,

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

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

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

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

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

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

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

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

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

HttpServletRequest request, HttpServletResponse response) {

try {

String[] mpArr = new String[]{};

if(StringUtil.isNotEmpty(mpList)){

mpArr= mpList.split(",");

}

List<MaterialVo4Unit> dataList = materialService.exportExcel(StringUtil.toNull(materialParam), StringUtil.toNull(color),

StringUtil.toNull(weight), StringUtil.toNull(expiryNum), StringUtil.toNull(enabled), StringUtil.toNull(enableSerialNumber),

StringUtil.toNull(enableBatchNumber), StringUtil.toNull(remark), StringUtil.toNull(categoryId));

String[] names = {"条码", "名称", "规格", "型号", "颜色", "类别", "扩展信息", "单位", "基础重量", "保质期", "采购价", "零售价", "销售价", "最低售价", "备注", "状态", "序列号", "批号"};

String title = "商品信息";

List<String[]> objects = new ArrayList<>();

if (null != dataList) {

for (MaterialVo4Unit m : dataList) {

String[] objs = new String[100];

objs[0] = m.getmBarCode();

objs[1] = m.getName();

objs[2] = m.getStandard();

objs[3] = m.getModel();

objs[4] = m.getColor();

objs[5] = m.getCategoryName();

objs[6] = materialService.getMaterialOtherByParam(mpArr, m);

objs[7] = m.getCommodityUnit();

objs[8] = m.getWeight() == null? "" : m.getWeight().toString();

objs[9] = m.getExpiryNum() == null? "" : m.getExpiryNum().toString();

objs[10] = m.getPurchaseDecimal() == null? "" : m.getPurchaseDecimal().toString();

objs[11] = m.getCommodityDecimal() == null? "" : m.getCommodityDecimal().toString();

objs[12] = m.getWholesaleDecimal() == null? "" : m.getWholesaleDecimal().toString();

objs[13] = m.getLowDecimal() == null? "" : m.getLowDecimal().toString();

objs[14] = m.getRemark();

objs[15] = m.getEnabled() ? "启用" : "禁用";

objs[16] = "1".equals(m.getEnableSerialNumber()) ? "有" : "无";

objs[17] = "1".equals(m.getEnableBatchNumber()) ? "有" : "无";

objects.add(objs);

}

}

File file = ExcelUtils.exportObjectsWithoutTitle(title, names, title, objects);

ExportExecUtil.showExec(file, file.getName(), response);

} catch (Exception e) {

e.printStackTrace();

}

}

/\*\*

\* excel表格导入产品（含初始库存）

\* @param file

\* @param request

\* @param response

\* @return

\*/

@PostMapping(value = "/importExcel")

@ApiOperation(value = "excel表格导入产品")

public BaseResponseInfo importExcel(MultipartFile file,

HttpServletRequest request, HttpServletResponse response) throws Exception{

BaseResponseInfo res = new BaseResponseInfo();

try {

res = materialService.importExcel(file, request);

} catch (Exception e) {

e.printStackTrace();

}

return res;

}

public BigDecimal parseBigDecimalEx(String str)throws Exception{

if(!StringUtil.isEmpty(str)) {

return new BigDecimal(str);

} else {

return null;

}

}

/\*\*

\* 获取商品序列号

\* @param q

\* @param currentPage

\* @param pageSize

\* @param request

\* @param response

\* @return

\* @throws Exception

\*/

@GetMapping(value = "/getMaterialEnableSerialNumberList")

@ApiOperation(value = "获取商品序列号")

public JSONObject getMaterialEnableSerialNumberList(

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

@RequestParam("page") Integer currentPage,

@RequestParam("rows") Integer pageSize,

HttpServletRequest request,

HttpServletResponse response)throws Exception {

JSONObject object= new JSONObject();

try {

List<MaterialVo4Unit> list = materialService.getMaterialEnableSerialNumberList(q, (currentPage-1)\*pageSize, pageSize);

Long count = materialService.getMaterialEnableSerialNumberCount(q);

object.put("rows", list);

object.put("total", count);

} catch (Exception e) {

e.printStackTrace();

}

return object;

}

/\*\*

\* 获取最大条码

\* @return

\* @throws Exception

\*/

@GetMapping(value = "/getMaxBarCode")

@ApiOperation(value = "获取最大条码")

public BaseResponseInfo getMaxBarCode() throws Exception {

BaseResponseInfo res = new BaseResponseInfo();

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

String barCode = materialService.getMaxBarCode();

map.put("barCode", barCode);

res.code = 200;

res.data = map;

return res;

}

/\*\*

\* 商品名称模糊匹配

\* @return

\* @throws Exception

\*/

@GetMapping(value = "/getMaterialNameList")

@ApiOperation(value = "商品名称模糊匹配")

public JSONArray getMaterialNameList() throws Exception {

JSONArray arr = new JSONArray();

try {

List<String> list = materialService.getMaterialNameList();

for (String s : list) {

JSONObject item = new JSONObject();

item.put("value", s);

item.put("text", s);

arr.add(item);

}

} catch (Exception e) {

e.printStackTrace();

}

return arr;

}

/\*\*

\* 根据条码查询商品信息

\* @return

\* @throws Exception

\*/

@GetMapping(value = "/getMaterialByBarCode")

@ApiOperation(value = "根据条码查询商品信息")

public BaseResponseInfo getMaterialByBarCode(@RequestParam("barCode") String barCode,

@RequestParam(value = "depotId", required = false) Long depotId,

@RequestParam("mpList") String mpList,

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

HttpServletRequest request) throws Exception {

BaseResponseInfo res = new BaseResponseInfo();

try {

String[] mpArr = mpList.split(",");

List<MaterialVo4Unit> list = materialService.getMaterialByBarCode(barCode);

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

for(MaterialVo4Unit mvo: list) {

mvo.setMaterialOther(materialService.getMaterialOtherByParam(mpArr, mvo));

if ("LSCK".equals(prefixNo) || "LSTH".equals(prefixNo)) {

//零售价

mvo.setBillPrice(mvo.getCommodityDecimal());

} else if ("CGDD".equals(prefixNo) || "CGRK".equals(prefixNo) || "CGTH".equals(prefixNo)

|| "QTRK".equals(prefixNo) || "DBCK".equals(prefixNo) || "ZZD".equals(prefixNo) || "CXD".equals(prefixNo)

|| "PDLR".equals(prefixNo) || "PDFP".equals(prefixNo)) {

//采购价

mvo.setBillPrice(mvo.getPurchaseDecimal());

} else if ("XSDD".equals(prefixNo) || "XSCK".equals(prefixNo) || "XSTH".equals(prefixNo) || "QTCK".equals(prefixNo)) {

//销售价

mvo.setBillPrice(mvo.getWholesaleDecimal());

}

//仓库id

if (depotId == null) {

JSONArray depotArr = depotService.findDepotByCurrentUser();

for (Object obj : depotArr) {

JSONObject depotObj = JSONObject.parseObject(obj.toString());

if (depotObj.get("isDefault") != null) {

Boolean isDefault = depotObj.getBoolean("isDefault");

if (isDefault) {

Long id = depotObj.getLong("id");

if (!"CGDD".equals(prefixNo) && !"XSDD".equals(prefixNo)) {

//除订单之外的单据才有仓库

mvo.setDepotId(id);

}

getStockByMaterialInfo(mvo);

}

}

}

} else {

mvo.setDepotId(depotId);

getStockByMaterialInfo(mvo);

}

}

}

res.code = 200;

res.data = list;

} catch(Exception e){

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 根据商品信息获取库存，进行赋值

\* @param mvo

\* @throws Exception

\*/

private void getStockByMaterialInfo(MaterialVo4Unit mvo) throws Exception {

BigDecimal stock;

if (StringUtil.isNotEmpty(mvo.getSku())) {

stock = depotItemService.getSkuStockByParam(mvo.getDepotId(), mvo.getMeId(), null, null);

} else {

stock = depotItemService.getStockByParam(mvo.getDepotId(), mvo.getId(), null, null);

if (mvo.getUnitId() != null) {

Unit unit = unitService.getUnit(mvo.getUnitId());

String commodityUnit = mvo.getCommodityUnit();

stock = unitService.parseStockByUnit(stock, unit, commodityUnit);

}

}

mvo.setStock(stock);

}

/\*\*

\* 商品库存查询

\* @param currentPage

\* @param pageSize

\* @param depotIds

\* @param categoryId

\* @param materialParam

\* @param mpList

\* @param column

\* @param order

\* @param request

\* @return

\* @throws Exception

\*/

@GetMapping(value = "/getListWithStock")

@ApiOperation(value = "商品库存查询")

public BaseResponseInfo getListWithStock(@RequestParam("currentPage") Integer currentPage,

@RequestParam("pageSize") Integer pageSize,

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

@RequestParam(value = "categoryId", required = false) Long categoryId,

@RequestParam("materialParam") String materialParam,

@RequestParam("zeroStock") Integer zeroStock,

@RequestParam("mpList") String mpList,

@RequestParam("column") String column,

@RequestParam("order") String order,

HttpServletRequest request)throws Exception {

BaseResponseInfo res = new BaseResponseInfo();

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

try {

List<Long> idList = new ArrayList<>();

List<Long> depotList = new ArrayList<>();

if(categoryId != null){

idList = materialService.getListByParentId(categoryId);

}

if(StringUtil.isNotEmpty(depotIds)) {

depotList = StringUtil.strToLongList(depotIds);

} else {

//未选择仓库时默认为当前用户有权限的仓库

JSONArray depotArr = depotService.findDepotByCurrentUser();

for(Object obj: depotArr) {

JSONObject object = JSONObject.parseObject(obj.toString());

depotList.add(object.getLong("id"));

}

}

Map<Long, BigDecimal> initialStockMap = new HashMap<>();

List<MaterialInitialStockWithMaterial> initialStockList = materialService.getInitialStockWithMaterial(depotList);

for (MaterialInitialStockWithMaterial mism: initialStockList) {

initialStockMap.put(mism.getMaterialId(), mism.getNumber());

}

List<MaterialVo4Unit> dataList = materialService.getListWithStock(depotList, idList, StringUtil.toNull(materialParam), zeroStock,

StringUtil.safeSqlParse(column), StringUtil.safeSqlParse(order), (currentPage-1)\*pageSize, pageSize);

int total = materialService.getListWithStockCount(depotList, idList, StringUtil.toNull(materialParam), zeroStock);

MaterialVo4Unit materialVo4Unit= materialService.getTotalStockAndPrice(depotList, idList, StringUtil.toNull(materialParam));

map.put("total", total);

map.put("currentStock", materialVo4Unit.getCurrentStock());

map.put("currentStockPrice", materialVo4Unit.getCurrentStockPrice());

for(MaterialVo4Unit item: dataList) {

item.setInitialStock(initialStockMap.get(item.getId()));

}

map.put("rows", dataList);

res.code = 200;

res.data = map;

} catch(Exception e){

e.printStackTrace();

res.code = 500;

res.data = "获取数据失败";

}

return res;

}

/\*\*

\* 批量设置商品当前的实时库存（按每个仓库）

\* @param jsonObject

\* @param request

\* @return

\* @throws Exception

\*/

@PostMapping(value = "/batchSetMaterialCurrentStock")

@ApiOperation(value = "批量设置商品当前的实时库存（按每个仓库）")

public String batchSetMaterialCurrentStock(@RequestBody JSONObject jsonObject,

HttpServletRequest request)throws Exception {

String ids = jsonObject.getString("ids");

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

int res = materialService.batchSetMaterialCurrentStock(ids);

if(res > 0) {

return returnJson(objectMap, ErpInfo.OK.name, ErpInfo.OK.code);

} else {

return returnJson(objectMap, ErpInfo.ERROR.name, ErpInfo.ERROR.code);

}

}

/\*\*

\* 批量更新商品信息

\* @param jsonObject

\* @param request

\* @return

\* @throws Exception

\*/

@PostMapping(value = "/batchUpdate")

@ApiOperation(value = "批量更新商品信息")

public String batchUpdate(@RequestBody JSONObject jsonObject,

HttpServletRequest request)throws Exception {

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

int res = materialService.batchUpdate(jsonObject);

if(res > 0) {

return returnJson(objectMap, ErpInfo.OK.name, ErpInfo.OK.code);

} else {

return returnJson(objectMap, ErpInfo.ERROR.name, ErpInfo.ERROR.code);

}

}

}

<template>

<a-row :gutter="24">

<a-col :md="24">

<a-card :style="cardStyle" :bordered="false">

<!-- 查询区域 -->

<div class="table-page-search-wrapper">

<!-- 搜索区域 -->

<a-form layout="inline" @keyup.enter.native="searchQuery">

<a-row :gutter="24">

<a-col :md="6" :sm="24">

<a-form-item label="单据编号" :labelCol="labelCol" :wrapperCol="wrapperCol">

<a-input placeholder="请输入单据编号" v-model="queryParam.number"></a-input>

</a-form-item>

</a-col>

<a-col :md="6" :sm="24">

<a-form-item label="商品信息" :labelCol="labelCol" :wrapperCol="wrapperCol">

<a-input placeholder="请输入条码、名称、规格、型号" v-model="queryParam.materialParam"></a-input>

</a-form-item>

</a-col>

<a-col :md="6" :sm="24">

<a-form-item label="单据日期" :labelCol="labelCol" :wrapperCol="wrapperCol">

<a-range-picker

style="width:100%"

v-model="queryParam.createTimeRange"

format="YYYY-MM-DD"

:placeholder="['开始时间', '结束时间']"

@change="onDateChange"

@ok="onDateOk"

/>

</a-form-item>

</a-col>

<span style="float: left;overflow: hidden;" class="table-page-search-submitButtons">

<a-col :md="6" :sm="24">

<a-button type="primary" @click="searchQuery">查询</a-button>

<a-button style="margin-left: 8px" @click="searchReset">重置</a-button>

<a @click="handleToggleSearch" style="margin-left: 8px">

{{ toggleSearchStatus ? '收起' : '展开' }}

<a-icon :type="toggleSearchStatus ? 'up' : 'down'"/>

</a>

</a-col>

</span>

<template v-if="toggleSearchStatus">

<a-col :md="6" :sm="24">

<a-form-item label="仓库名称" :labelCol="labelCol" :wrapperCol="wrapperCol">

<a-select placeholder="请选择仓库" showSearch optionFilterProp="children" v-model="queryParam.depotId">

<a-select-option v-for="(depot,index) in depotList" :value="depot.id">

{{ depot.depotName }}

</a-select-option>

</a-select>

</a-form-item>

</a-col>

<a-col :md="6" :sm="24">

<a-form-item label="操作员" :labelCol="labelCol" :wrapperCol="wrapperCol">

<a-select placeholder="选择操作员" showSearch optionFilterProp="children" v-model="queryParam.creator">

<a-select-option v-for="(item,index) in userList" :key="index" :value="item.id">

{{ item.userName }}

</a-select-option>

</a-select>

</a-form-item>

</a-col>

<a-col :md="6" :sm="24">

<a-form-item label="单据状态" :labelCol="labelCol" :wrapperCol="wrapperCol">

<a-select placeholder="选择单据状态" v-model="queryParam.status">

<a-select-option value="0">未审核</a-select-option>

<a-select-option value="1">已审核</a-select-option>

</a-select>

</a-form-item>

</a-col>

<a-col :md="6" :sm="24">

<a-form-item label="单据备注" :labelCol="labelCol" :wrapperCol="wrapperCol">

<a-input placeholder="请输入单据备注" v-model="queryParam.remark"></a-input>

</a-form-item>

</a-col>

</template>

</a-row>

</a-form>

</div>

<!-- 操作按钮区域 -->

<div class="table-operator" style="margin-top: 5px">

<a-button v-if="btnEnableList.indexOf(1)>-1" @click="myHandleAdd" type="primary" icon="plus">新增</a-button>

<a-dropdown>

<a-menu slot="overlay">

<a-menu-item key="1" v-if="btnEnableList.indexOf(1)>-1" @click="batchDel"><a-icon type="delete"/>删除</a-menu-item>

<a-menu-item key="2" v-if="btnEnableList.indexOf(2)>-1" @click="batchSetStatus(1)"><a-icon type="check"/>审核</a-menu-item>

<a-menu-item key="3" v-if="btnEnableList.indexOf(7)>-1" @click="batchSetStatus(0)"><a-icon type="stop"/>反审核</a-menu-item>

</a-menu>

<a-button>

批量操作 <a-icon type="down" />

</a-button>

</a-dropdown>

<a-tooltip placement="left" title="用于两个仓库之间的商品调拨，调拨单会影响库存。" slot="action">

<a-icon v-if="btnEnableList.indexOf(1)>-1" type="question-circle" style="font-size:20px;float:right;" />

</a-tooltip>

</div>

<!-- table区域-begin -->

<div>

<a-table

ref="table"

size="middle"

bordered

rowKey="id"

:columns="columns"

:dataSource="dataSource"

:pagination="ipagination"

:scroll="scroll"

:loading="loading"

:rowSelection="{selectedRowKeys: selectedRowKeys, onChange: onSelectChange}"

@change="handleTableChange">

<span slot="action" slot-scope="text, record">

<a @click="myHandleDetail(record, '调拨出库')">查看</a>

<a-divider v-if="btnEnableList.indexOf(1)>-1" type="vertical" />

<a v-if="btnEnableList.indexOf(1)>-1" @click="myHandleEdit(record)">编辑</a>

<a-divider v-if="btnEnableList.indexOf(1)>-1" type="vertical" />

<a v-if="btnEnableList.indexOf(1)>-1" @click="myHandleCopyAdd(record)">复制</a>

<a-divider v-if="btnEnableList.indexOf(1)>-1" type="vertical" />

<a-popconfirm v-if="btnEnableList.indexOf(1)>-1" title="确定删除吗?" @confirm="() => myHandleDelete(record)">

<a>删除</a>

</a-popconfirm>

</span>

<template slot="customRenderStatus" slot-scope="status">

<a-tag v-if="status == '0'" color="red">未审核</a-tag>

<a-tag v-if="status == '1'" color="green">已审核</a-tag>

</template>

</a-table>

</div>

<!-- table区域-end -->

<!-- 表单区域 -->

<allocation-out-modal ref="modalForm" @ok="modalFormOk"></allocation-out-modal>

<bill-detail ref="modalDetail" @ok="modalFormOk"></bill-detail>

</a-card>

</a-col>

</a-row>

</template>

<script>

import AllocationOutModal from './modules/AllocationOutModal'

import BillDetail from './dialog/BillDetail'

import { JeecgListMixin } from '@/mixins/JeecgListMixin'

import { BillListMixin } from './mixins/BillListMixin'

import JDate from '@/components/jeecg/JDate'

import Vue from 'vue'

export default {

name: "AllocationOutList",

mixins:[JeecgListMixin,BillListMixin],

components: {

AllocationOutModal,

BillDetail,

JDate

},

data () {

return {

// 查询条件

queryParam: {

number: "",

materialParam: "",

type: "出库",

subType: "调拨",

roleType: Vue.ls.get('roleType'),

depotId: "",

creator: "",

status: "",

remark: ""

},

labelCol: {

span: 5

},

wrapperCol: {

span: 18,

offset: 1

},

// 表头

columns: [

{

title: '操作',

dataIndex: 'action',

align:"center", width: 150,

scopedSlots: { customRender: 'action' },

},

{ title: '单据编号', dataIndex: 'number',width:160,

customRender:function (text,record,index) {

if(record.linkNumber) {

return text + "[转]";

} else {

return text;

}

}

},

{ title: '商品信息', dataIndex: 'materialsList',width:220, ellipsis:true,

customRender:function (text,record,index) {

if(text) {

return text.replace(",","，");

}

}

},

{ title: '单据日期', dataIndex: 'operTimeStr',width:145},

{ title: '操作员', dataIndex: 'userName',width:80, ellipsis:true},

{ title: '数量', dataIndex: 'materialCount',width:60},

{ title: '金额合计', dataIndex: 'totalPrice',width:80},

{ title: '状态', dataIndex: 'status', width: 80, align: "center",

scopedSlots: { customRender: 'customRenderStatus' }

}

],

url: {

list: "/depotHead/list",

delete: "/depotHead/delete",

deleteBatch: "/depotHead/deleteBatch",

batchSetStatusUrl: "/depotHead/batchSetStatus"

}

}

},

computed: {

},

created() {

this.getDepotData()

this.initUser()

},

methods: {

}

}

</script>

<style scoped>

@import '~@assets/less/common.less'

</style>

<template>

<a-row :gutter="24">

<a-col :md="24">

<a-card :style="cardStyle" :bordered="false">

<!-- 查询区域 -->

<div class="table-page-search-wrapper">

<!-- 搜索区域 -->

<a-form layout="inline" @keyup.enter.native="searchQuery">

<a-row :gutter="24">

<a-col :md="6" :sm="24">

<a-form-item label="单据编号" :labelCol="labelCol" :wrapperCol="wrapperCol">

<a-input placeholder="请输入单据编号" v-model="queryParam.number"></a-input>

</a-form-item>

</a-col>

<a-col :md="6" :sm="24">

<a-form-item label="商品信息" :labelCol="labelCol" :wrapperCol="wrapperCol">

<a-input placeholder="请输入条码、名称、规格、型号" v-model="queryParam.materialParam"></a-input>

</a-form-item>

</a-col>

<a-col :md="6" :sm="24">

<a-form-item label="单据日期" :labelCol="labelCol" :wrapperCol="wrapperCol">

<a-range-picker

style="width:100%"

v-model="queryParam.createTimeRange"

format="YYYY-MM-DD"

:placeholder="['开始时间', '结束时间']"

@change="onDateChange"

@ok="onDateOk"

/>

</a-form-item>

</a-col>

<span style="float: left;overflow: hidden;" class="table-page-search-submitButtons">

<a-col :md="6" :sm="24">

<a-button type="primary" @click="searchQuery">查询</a-button>

<a-button style="margin-left: 8px" @click="searchReset">重置</a-button>

<a @click="handleToggleSearch" style="margin-left: 8px">

{{ toggleSearchStatus ? '收起' : '展开' }}

<a-icon :type="toggleSearchStatus ? 'up' : 'down'"/>

</a>

</a-col>

</span>

<template v-if="toggleSearchStatus">

<a-col :md="6" :sm="24">

<a-form-item label="仓库名称" :labelCol="labelCol" :wrapperCol="wrapperCol">

<a-select placeholder="请选择仓库" showSearch optionFilterProp="children" v-model="queryParam.depotId">

<a-select-option v-for="(depot,index) in depotList" :value="depot.id">

{{ depot.depotName }}

</a-select-option>

</a-select>

</a-form-item>

</a-col>

<a-col :md="6" :sm="24">

<a-form-item label="操作员" :labelCol="labelCol" :wrapperCol="wrapperCol">

<a-select placeholder="选择操作员" showSearch optionFilterProp="children" v-model="queryParam.creator">

<a-select-option v-for="(item,index) in userList" :key="index" :value="item.id">

{{ item.userName }}

</a-select-option>

</a-select>

</a-form-item>

</a-col>

<a-col :md="6" :sm="24">

<a-form-item label="单据状态" :labelCol="labelCol" :wrapperCol="wrapperCol">

<a-select placeholder="选择单据状态" v-model="queryParam.status">

<a-select-option value="0">未审核</a-select-option>

<a-select-option value="1">已审核</a-select-option>

</a-select>

</a-form-item>

</a-col>

<a-col :md="6" :sm="24">

<a-form-item label="单据备注" :labelCol="labelCol" :wrapperCol="wrapperCol">

<a-input placeholder="请输入单据备注" v-model="queryParam.remark"></a-input>

</a-form-item>

</a-col>

</template>

</a-row>

</a-form>

</div>

<!-- 操作按钮区域 -->

<div class="table-operator" style="margin-top: 5px">

<a-button v-if="btnEnableList.indexOf(1)>-1" @click="myHandleAdd" type="primary" icon="plus">新增</a-button>

<a-dropdown>

<a-menu slot="overlay">

<a-menu-item key="1" v-if="btnEnableList.indexOf(1)>-1" @click="batchDel"><a-icon type="delete"/>删除</a-menu-item>

<a-menu-item key="2" v-if="btnEnableList.indexOf(2)>-1" @click="batchSetStatus(1)"><a-icon type="check"/>审核</a-menu-item>

<a-menu-item key="3" v-if="btnEnableList.indexOf(7)>-1" @click="batchSetStatus(0)"><a-icon type="stop"/>反审核</a-menu-item>

</a-menu>

<a-button>

批量操作 <a-icon type="down" />

</a-button>

</a-dropdown>

<a-tooltip placement="left" title="用于将多种商品合并成一个商品，被合并的商品库存减少，合并后的商品库存增加。" slot="action">

<a-icon v-if="btnEnableList.indexOf(1)>-1" type="question-circle" style="font-size:20px;float:right;" />

</a-tooltip>

</div>

<!-- table区域-begin -->

<div>

<a-table

ref="table"

size="middle"

bordered

rowKey="id"

:columns="columns"

:dataSource="dataSource"

:pagination="ipagination"

:scroll="scroll"

:loading="loading"

:rowSelection="{selectedRowKeys: selectedRowKeys, onChange: onSelectChange}"

@change="handleTableChange">

<span slot="action" slot-scope="text, record">

<a @click="myHandleDetail(record, '组装单')">查看</a>

<a-divider v-if="btnEnableList.indexOf(1)>-1" type="vertical" />

<a v-if="btnEnableList.indexOf(1)>-1" @click="myHandleEdit(record)">编辑</a>

<a-divider v-if="btnEnableList.indexOf(1)>-1" type="vertical" />

<a v-if="btnEnableList.indexOf(1)>-1" @click="myHandleCopyAdd(record)">复制</a>

<a-divider v-if="btnEnableList.indexOf(1)>-1" type="vertical" />

<a-popconfirm v-if="btnEnableList.indexOf(1)>-1" title="确定删除吗?" @confirm="() => myHandleDelete(record)">

<a>删除</a>

</a-popconfirm>

</span>

<template slot="customRenderStatus" slot-scope="status">

<a-tag v-if="status == '0'" color="red">未审核</a-tag>

<a-tag v-if="status == '1'" color="green">已审核</a-tag>

</template>

</a-table>

</div>

<!-- table区域-end -->

<!-- 表单区域 -->

<assemble-modal ref="modalForm" @ok="modalFormOk"></assemble-modal>

<bill-detail ref="modalDetail" @ok="modalFormOk"></bill-detail>

</a-card>

</a-col>

</a-row>

</template>

<script>

import AssembleModal from './modules/AssembleModal'

import BillDetail from './dialog/BillDetail'

import { JeecgListMixin } from '@/mixins/JeecgListMixin'

import { BillListMixin } from './mixins/BillListMixin'

import JDate from '@/components/jeecg/JDate'

import Vue from 'vue'

export default {

name: "AssembleList",

mixins:[JeecgListMixin,BillListMixin],

components: {

AssembleModal,

BillDetail,

JDate

},

data () {

return {

// 查询条件

queryParam: {

number: "",

materialParam: "",

type: "其它",

subType: "组装单",

roleType: Vue.ls.get('roleType'),

depotId: "",

creator: "",

status: "",

remark: ""

},

labelCol: {

span: 5

},

wrapperCol: {

span: 18,

offset: 1

},

// 表头

columns: [

{

title: '操作',

dataIndex: 'action',

align:"center", width: 150,

scopedSlots: { customRender: 'action' },

},

{ title: '单据编号', dataIndex: 'number',width:160,

customRender:function (text,record,index) {

if(record.linkNumber) {

return text + "[转]";

} else {

return text;

}

}

},

{ title: '商品信息', dataIndex: 'materialsList',width:220, ellipsis:true,

customRender:function (text,record,index) {

if(text) {

return text.replace(",","，");

}

}

},

{ title: '单据日期', dataIndex: 'operTimeStr',width:145},

{ title: '操作员', dataIndex: 'userName',width:80, ellipsis:true},

{ title: '数量', dataIndex: 'materialCount',width:60},

{ title: '金额合计', dataIndex: 'totalPrice',width:80},

{ title: '状态', dataIndex: 'status', width: 80, align: "center",

scopedSlots: { customRender: 'customRenderStatus' }

}

],

url: {

list: "/depotHead/list",

delete: "/depotHead/delete",

deleteBatch: "/depotHead/deleteBatch",

batchSetStatusUrl: "/depotHead/batchSetStatus"

}

}

},

computed: {

},

created() {

this.getDepotData()

this.initUser()

},

methods: {

}

}

</script>

<style scoped>

@import '~@assets/less/common.less'

</style>

<template>

<a-row :gutter="24">

<a-col :md="24">

<a-card :style="cardStyle" :bordered="false">

<!-- 查询区域 -->

<div class="table-page-search-wrapper">

<!-- 搜索区域 -->

<a-form layout="inline" @keyup.enter.native="searchQuery">

<a-row :gutter="24">

<a-col :md="6" :sm="24">

<a-form-item label="单据编号" :labelCol="labelCol" :wrapperCol="wrapperCol">

<a-input placeholder="请输入单据编号" v-model="queryParam.number"></a-input>

</a-form-item>

</a-col>

<a-col :md="6" :sm="24">

<a-form-item label="商品信息" :labelCol="labelCol" :wrapperCol="wrapperCol">

<a-input placeholder="请输入条码、名称、规格、型号" v-model="queryParam.materialParam"></a-input>

</a-form-item>

</a-col>

<a-col :md="6" :sm="24">

<a-form-item label="单据日期" :labelCol="labelCol" :wrapperCol="wrapperCol">

<a-range-picker

style="width:100%"

v-model="queryParam.createTimeRange"

format="YYYY-MM-DD"

:placeholder="['开始时间', '结束时间']"

@change="onDateChange"

@ok="onDateOk"

/>

</a-form-item>

</a-col>

<span style="float: left;overflow: hidden;" class="table-page-search-submitButtons">

<a-col :md="6" :sm="24">

<a-button type="primary" @click="searchQuery">查询</a-button>

<a-button style="margin-left: 8px" @click="searchReset">重置</a-button>

<a @click="handleToggleSearch" style="margin-left: 8px">

{{ toggleSearchStatus ? '收起' : '展开' }}

<a-icon :type="toggleSearchStatus ? 'up' : 'down'"/>

</a>

</a-col>

</span>

<template v-if="toggleSearchStatus">

<a-col :md="6" :sm="24">

<a-form-item label="仓库名称" :labelCol="labelCol" :wrapperCol="wrapperCol">

<a-select placeholder="请选择仓库" showSearch optionFilterProp="children" v-model="queryParam.depotId">

<a-select-option v-for="(depot,index) in depotList" :value="depot.id">

{{ depot.depotName }}

</a-select-option>

</a-select>

</a-form-item>

</a-col>

<a-col :md="6" :sm="24">

<a-form-item label="操作员" :labelCol="labelCol" :wrapperCol="wrapperCol">

<a-select placeholder="选择操作员" showSearch optionFilterProp="children" v-model="queryParam.creator">

<a-select-option v-for="(item,index) in userList" :key="index" :value="item.id">

{{ item.userName }}

</a-select-option>

</a-select>

</a-form-item>

</a-col>

<a-col :md="6" :sm="24">

<a-form-item label="单据状态" :labelCol="labelCol" :wrapperCol="wrapperCol">

<a-select placeholder="选择单据状态" v-model="queryParam.status">

<a-select-option value="0">未审核</a-select-option>

<a-select-option value="1">已审核</a-select-option>

</a-select>

</a-form-item>

</a-col>

<a-col :md="6" :sm="24">

<a-form-item label="单据备注" :labelCol="labelCol" :wrapperCol="wrapperCol">

<a-input placeholder="请输入单据备注" v-model="queryParam.remark"></a-input>

</a-form-item>

</a-col>

</template>

</a-row>

</a-form>

</div>

<!-- 操作按钮区域 -->

<div class="table-operator" style="margin-top: 5px">

<a-button v-if="btnEnableList.indexOf(1)>-1" @click="myHandleAdd" type="primary" icon="plus">新增</a-button>

<a-dropdown>

<a-menu slot="overlay">

<a-menu-item key="1" v-if="btnEnableList.indexOf(1)>-1" @click="batchDel"><a-icon type="delete"/>删除</a-menu-item>

<a-menu-item key="2" v-if="btnEnableList.indexOf(2)>-1" @click="batchSetStatus(1)"><a-icon type="check"/>审核</a-menu-item>

<a-menu-item key="3" v-if="btnEnableList.indexOf(7)>-1" @click="batchSetStatus(0)"><a-icon type="stop"/>反审核</a-menu-item>

</a-menu>

<a-button>

批量操作 <a-icon type="down" />

</a-button>

</a-dropdown>

<a-tooltip placement="left" title="用于将一个商品拆分成多种商品，被拆分的商品库存增加，拆分后的商品库存减少。" slot="action">

<a-icon v-if="btnEnableList.indexOf(1)>-1" type="question-circle" style="font-size:20px;float:right;" />

</a-tooltip>

</div>

<!-- table区域-begin -->

<div>

<a-table

ref="table"

size="middle"

bordered

rowKey="id"

:columns="columns"

:dataSource="dataSource"

:pagination="ipagination"

:scroll="scroll"

:loading="loading"

:rowSelection="{selectedRowKeys: selectedRowKeys, onChange: onSelectChange}"

@change="handleTableChange">

<span slot="action" slot-scope="text, record">

<a @click="myHandleDetail(record, '拆卸单')">查看</a>

<a-divider v-if="btnEnableList.indexOf(1)>-1" type="vertical" />

<a v-if="btnEnableList.indexOf(1)>-1" @click="myHandleEdit(record)">编辑</a>

<a-divider v-if="btnEnableList.indexOf(1)>-1" type="vertical" />

<a v-if="btnEnableList.indexOf(1)>-1" @click="myHandleCopyAdd(record)">复制</a>

<a-divider v-if="btnEnableList.indexOf(1)>-1" type="vertical" />

<a-popconfirm v-if="btnEnableList.indexOf(1)>-1" title="确定删除吗?" @confirm="() => myHandleDelete(record)">

<a>删除</a>

</a-popconfirm>

</span>

<template slot="customRenderStatus" slot-scope="status">

<a-tag v-if="status == '0'" color="red">未审核</a-tag>

<a-tag v-if="status == '1'" color="green">已审核</a-tag>

</template>

</a-table>

</div>

<!-- table区域-end -->

<!-- 表单区域 -->

<disassemble-modal ref="modalForm" @ok="modalFormOk"></disassemble-modal>

<bill-detail ref="modalDetail" @ok="modalFormOk"></bill-detail>

</a-card>

</a-col>

</a-row>

</template>

<script>

import DisassembleModal from './modules/DisassembleModal'

import BillDetail from './dialog/BillDetail'

import { JeecgListMixin } from '@/mixins/JeecgListMixin'

import { BillListMixin } from './mixins/BillListMixin'

import JDate from '@/components/jeecg/JDate'

import Vue from 'vue'

export default {

name: "DisassembleList",

mixins:[JeecgListMixin,BillListMixin],

components: {

DisassembleModal,

BillDetail,

JDate

},

data () {

return {

// 查询条件

queryParam: {

number: "",

materialParam: "",

type: "其它",

subType: "拆卸单",

roleType: Vue.ls.get('roleType'),

depotId: "",

creator: "",

status: "",

remark: ""

},

labelCol: {

span: 5

},

wrapperCol: {

span: 18,

offset: 1

},

// 表头

columns: [

{

title: '操作',

dataIndex: 'action',

align:"center", width: 150,

scopedSlots: { customRender: 'action' },

},

{ title: '单据编号', dataIndex: 'number',width:160,

customRender:function (text,record,index) {

if(record.linkNumber) {

return text + "[转]";

} else {

return text;

}

}

},

{ title: '商品信息', dataIndex: 'materialsList',width:220, ellipsis:true,

customRender:function (text,record,index) {

if(text) {

return text.replace(",","，");

}

}

},

{ title: '单据日期', dataIndex: 'operTimeStr',width:145},

{ title: '操作员', dataIndex: 'userName',width:80, ellipsis:true},

{ title: '数量', dataIndex: 'materialCount',width:60},

{ title: '金额合计', dataIndex: 'totalPrice',width:80},

{ title: '状态', dataIndex: 'status', width: 80, align: "center",

scopedSlots: { customRender: 'customRenderStatus' }

}

],

url: {

list: "/depotHead/list",

delete: "/depotHead/delete",

deleteBatch: "/depotHead/deleteBatch",

batchSetStatusUrl: "/depotHead/batchSetStatus"

}

}

},

computed: {

},

created() {

this.getDepotData()

this.initUser()

},

methods: {

}

}

</script>

<style scoped>

@import '~@assets/less/common.less'

</style>