package com.nork.intelligence.controller;

import java.util.ArrayList;

import java.util.List;

import java.util.Map;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import com.nork.onekeydesign.model.vo.DimMatchInfoVO;

import com.nork.onekeydesign.service.\*;

import com.nork.intelligence.constant.IntelligenceDecorationConstant;

import net.sf.json.JSONArray;

import org.apache.commons.lang3.StringUtils;

import org.apache.logging.log4j.LogManager;

import org.apache.logging.log4j.Logger;

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

import org.springframework.stereotype.Controller;

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

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

import com.nork.common.constant.util.SystemCommonUtil;

import com.nork.common.model.LoginUser;

import com.nork.common.model.ResponseEnvelope;

import com.nork.common.util.Constants;

import com.nork.common.util.collections.Lists;

import com.nork.onekeydesign.common.DesignPlanConstants;

import com.nork.onekeydesign.exception.IntelligenceDecorationException;

import com.nork.onekeydesign.model.BedroomProductMatchDTO;

import com.nork.onekeydesign.model.BedroomProductMatchDTO.GroupMatchInfoDTO;

import com.nork.onekeydesign.model.BedroomProductMatchDTO.ProductMatchInfoDTO;

import com.nork.onekeydesign.model.BedroomProductMatchDTO.StructureMatchInfoDTO;

import com.nork.onekeydesign.model.DesignPlan;

import com.nork.onekeydesign.model.DesignPlanRecommended;

import com.nork.onekeydesign.model.DesignPlanRecommendedProduct;

import com.nork.onekeydesign.model.DesignTemplet;

import com.nork.onekeydesign.model.DesignTempletProduct;

import com.nork.onekeydesign.model.PosNameInfo;

import com.nork.onekeydesign.model.ProductListByTypeInfo;

import com.nork.onekeydesign.model.UnityDesignPlan;

import com.nork.system.model.ResDesign;

import com.nork.system.model.ResFile;

import com.nork.system.model.bo.SysDictionaryBo;

import com.nork.system.service.ResDesignService;

import com.nork.system.service.ResFileService;

import com.sandu.search.exception.ElasticSearchException;

/\*\*

\* 一件装修controller copy from 通用版本

\*

\* @author huangsongbo

\* 2017.11.28

\*/

@Controller

@RequestMapping("/{style}/web/design/intelligenceDecoration")

public class IntelligenceDecorationController {

private static Logger logger = LogManager.getLogger(IntelligenceDecorationController.class);

private String logPrefixClass = "function:IntelligenceDecorationController.";

@Autowired

private IntelligenceDecorationService intelligenceDecorationService;

@Autowired

private DesignTempletService designTempletService;

@Autowired

private DesignPlanRecommendedService designPlanRecommendedService;

@Autowired

private OptimizePlanService optimizePlanService;

@Autowired

private DesignTempletProductService designTempletProductService;

@Autowired

Private DesignPlanRecommendedProductServiceV2 designPlanRecommendedProductServiceV2;

@Autowired

private DesignPlanService designPlanService;

@Autowired

private ResFileService resFileService;

@Autowired

private ResDesignService resDesignService;

@Autowired

private DesignPlanTemplateService designPlanTemplateService;

/\*\*

\* 卧室一建装修产品匹配

\* 单品:单品/天花/背景墙

\* 结构

\* 组合

\*

\* @author huangsongbo

\* @param designTempletId 样板房id

\* @param planRecommendedId 推荐方案id

\* @param msgId

\* @param matchType 0:全屋替换 ;1:硬装替换

\* @param opType 是否是自动渲染

\* @param spaceCode 空间编码(用于识别房间类型,来决定启用哪种房间类型的一件装修逻辑)

\* @return

\*/

@SuppressWarnings("unchecked")

@RequestMapping("/bedroomProductMatch")

@ResponseBody

public Object bedroomProductMatch(

Integer designTempletId,

Integer planRecommendedId,

String msgId,

Integer matchType,

Integer opType,

String spaceCode,

HttpServletRequest request,

String planSource,

Integer returnParamType

,String posNameDebug

) {

// \*参数验证 ->start

if(designTempletId == null) {

return new ResponseEnvelope<>(false, "参数designTempletId不能为空", msgId);

}

if(planRecommendedId == null) {

return new ResponseEnvelope<>(false, "参数planRecommendedId不能为空", msgId);

}

DesignTemplet designTemplet = designTempletService.getForOneKey(designTempletId);

if(designTemplet == null) {

return new ResponseEnvelope<>(false, "样板房数据不存在!(id = " + designTempletId + ")", msgId);

}

DesignPlanRecommended designPlanRecommended = designPlanRecommendedService.get(planRecommendedId);

if(designPlanRecommended == null) {

return new ResponseEnvelope<>(false, "推荐方案数据不存在!(id = " + planRecommendedId + ")", msgId);

}

//如果是主方案，匹配适用于该样板空间的子方案 add by : xiaoxc20180911

if (designPlanRecommended.getGroupPrimaryId().equals(planRecommendedId)) {

planRecommendedId = designPlanRecommendedService.getBestMatchInPlanGroup(designTempletId, planRecommendedId);

designPlanRecommended = designPlanRecommendedService.get(planRecommendedId);

if(designPlanRecommended == null) {

return new ResponseEnvelope<>(false, "推荐方案数据不存在!(bestMatchPlanId = " + planRecommendedId + ")", msgId);

}

}

if(matchType == null) {

matchType = 0;

}

if (opType == null) {

opType = DesignPlanConstants.USER\_RENDER;

}

LoginUser loginUser = SystemCommonUtil.getLoginUserFromRequest(request);

if(loginUser == null){

return new ResponseEnvelope<UnityDesignPlan>(false, "未检测到登录信息,请登录", msgId);

}

// 防止没有userName的账号 update by huangsongbo 2018.6.12

if(StringUtils.isEmpty(loginUser.getLoginName())) {

loginUser.setLoginName("noUsername");

}

// \*参数验证 ->end

// 返回结果

BedroomProductMatchDTO bedroomProductMatchDTO = new BedroomProductMatchDTO();

// \*整理单品/结构/组合成bean ->start

// 整理样板房产品表

List<DesignTempletProduct> designTempletProductList = designTempletProductService.getListByTempletId(designTempletId);

if(Lists.isEmpty(designTempletProductList)) {

return new ResponseEnvelope<>(false, "检测到样板房的产品列表为空,designCode:" + designTemplet.getDesignCode(), msgId);

}

// 组装样板房中单品/结构/组合数据

ProductListByTypeInfo productListByTypeInfo = designTempletProductService.getProductListByTypeInfo(designTempletProductList, matchType, planRecommendedId);

// 整理推荐方案产品表

List<DesignPlanRecommendedProduct> designPlanRecommendedProductList = designPlanRecommendedProductServiceV2.getListByRecommendedId(planRecommendedId);

if(Lists.isEmpty(designPlanRecommendedProductList)) {

return new ResponseEnvelope<>(false, "检测到一建装修方案的产品列表为空", msgId);

}

// 组装推荐方案中单品/结构/组合数据

ProductListByTypeInfo productListByTypeInfoRecommended;

try {

productListByTypeInfoRecommended = designPlanRecommendedProductServiceV2.getProductListByTypeInfo(designPlanRecommendedProductList);

} catch (IntelligenceDecorationException e) {

return new ResponseEnvelope<BedroomProductMatchDTO>(false, e.getMessage());

}

// \*整理单品/结构/组合成bean ->end

// \*新建设计方案(作为一件装修后生成的设计方案) ->start

String mediaType = null!=loginUser?loginUser.getMediaType():null;

DesignPlan designPlan = new DesignPlan();

designPlan.setPlanSource(planSource);

designPlan = designPlanService.createDesignPlanForOneKey(designPlan, null,designTemplet, mediaType, loginUser, 1, opType);

// 设置新设计方案拼花配置文件 ->start

long resFileId = resDesignService.copySpellingFlowerFile(designPlanRecommended.getSpellingFlowerFileId(), designPlan.getId(), designPlan.getPlanCode());

designPlan.setSpellingFlowerFileId(Integer.valueOf(resFileId + ""));

designPlan.setSpellingFlowerProduct(designPlanRecommended.getSpellingFlowerProduct());

designPlanService.update(designPlan);

// 设置新设计方案拼花配置文件 ->end

// \*新建设计方案(作为一件装修后生成的设计方案) ->end

// 匹配结构

Map<String, Object> structureListMatchResultMap = null;

try {

structureListMatchResultMap = intelligenceDecorationService.structureListMatch(

productListByTypeInfo,

productListByTypeInfoRecommended.getStructureInfo(),

designPlan.getId(),

loginUser.getLoginName(),

opType,

productListByTypeInfoRecommended.getProductListMap(),

spaceCode,

designTemplet

);

} catch (IntelligenceDecorationException e) {

return new ResponseEnvelope<BedroomProductMatchDTO>(false, e.getMessage());

}

List<StructureMatchInfoDTO> structureMatchInfo = null;

if(structureListMatchResultMap != null) {

structureMatchInfo = (List<StructureMatchInfoDTO>) structureListMatchResultMap.get(Constants.STRUCTURELISTMATCHRESULTMAPENUM\_STRUCTUREMATCHINFO);

productListByTypeInfo = (ProductListByTypeInfo) structureListMatchResultMap.get(Constants.STRUCTURELISTMATCHRESULTMAPENUM\_PRODUCTLISTBYTYPEINFO);

}

// 匹配组合

Map<String, Object> groupMatchResultMap = intelligenceDecorationService.groupListMatch(

productListByTypeInfo,

productListByTypeInfoRecommended.getGroupInfo(),

designPlan.getId(),

loginUser.getLoginName(),

opType);

List<GroupMatchInfoDTO> groupMatchInfo = null;

if(groupMatchResultMap != null) {

groupMatchInfo = (List<GroupMatchInfoDTO>) groupMatchResultMap.get(IntelligenceDecorationConstant.GROUPMATCHRESULTMAP\_GROUPMATCHINFO);

productListByTypeInfo = (ProductListByTypeInfo) groupMatchResultMap.get(IntelligenceDecorationConstant.GROUPMATCHRESULTMAP\_PRODUCTLISTBYTYPEINFO);

}

// 匹配单品(单品/背景墙/天花)

Map<String, Object> productListMatchResultMap = null;

try {

productListMatchResultMap = intelligenceDecorationService.productListMatch(

productListByTypeInfo.getProductList(),

productListByTypeInfoRecommended.getProductListMap(),

// 户型绘制需求,导致地面/天花逻辑完全变化 update by huangsongbo 2018.3.16 ->start

productListByTypeInfoRecommended.getStructureInfo(),

// 户型绘制需求,导致地面/天花逻辑完全变化 update by huangsongbo 2018.3.16 ->end

designPlan.getId(),

loginUser.getLoginName(),

matchType,

opType,

designTemplet,

planRecommendedId);

} catch (IntelligenceDecorationException e) {

return new ResponseEnvelope<BedroomProductMatchDTO>(false, e.getMessage());

} catch (ElasticSearchException e) {

return new ResponseEnvelope<BedroomProductMatchDTO>(false, e.getMessage());

}

List<ProductMatchInfoDTO> productMatchInfo = null;

List<DimMatchInfoVO> dimMatchInfo = null;

if(productListMatchResultMap != null) {

productMatchInfo = (List<ProductMatchInfoDTO>) productListMatchResultMap.get(IntelligenceDecorationConstant.PRODUCTLISTMATCHRESULTMAP\_PRODUCTMATCHINFO);

dimMatchInfo = (List<DimMatchInfoVO>) productListMatchResultMap.get(IntelligenceDecorationConstant.PRODUCTLISTMATCHRESULTMAP\_DIMMATCHINFOVO);

}

// \*样板房/推荐方案配置文件 ->start

Integer designTempletConfigFileId = designTemplet.getConfigFileId();

String designTempletConfigFileUrl = null;

if(designTempletConfigFileId != null) {

ResFile resFile = resFileService.get(designTempletConfigFileId);

if(resFile != null) {

designTempletConfigFileUrl = resFile.getFilePath();

}

}

Integer designPlanRecommendedConfigFileId = designPlanRecommended.getConfigFileId();

String designPlanRecommendedConfigFileUrl = null;

if(designPlanRecommendedConfigFileId != null) {

ResDesign resDesign = resDesignService.get(designPlanRecommendedConfigFileId);

if(resDesign != null) {

designPlanRecommendedConfigFileUrl = resDesign.getFilePath();

}

}

// \*样板房/推荐方案配置文件 ->start

// \*组装返回结果 ->start

bedroomProductMatchDTO.setDesignTempletConfigUrl(designTempletConfigFileUrl);

bedroomProductMatchDTO.setGroupMatchInfo(groupMatchInfo);

bedroomProductMatchDTO.setProductMatchInfo(productMatchInfo);

bedroomProductMatchDTO.setRecommendedPlanConfigUrl(designPlanRecommendedConfigFileUrl);

bedroomProductMatchDTO.setStructureMatchInfo(structureMatchInfo);

bedroomProductMatchDTO.setDimMatchInfo(dimMatchInfo);

Integer dataType = 1;

if(designTemplet.getOrigin() != null && 1 == designTemplet.getOrigin().intValue()) {

dataType = 2;

}else {

}

bedroomProductMatchDTO.setDataType(dataType);

// \*组装返回结果 ->end

// 返回小面积推荐方案的id ->start

// add by huangsongbo 2018.7.17

if(returnParamType != null && 1 == returnParamType.intValue()) {

Integer standbyRecommendedId = null;

try {

standbyRecommendedId = designPlanRecommendedService.getStandbyRecommendedId(planRecommendedId);

}catch (IntelligenceDecorationException e) {

return new ResponseEnvelope<BedroomProductMatchDTO>(false, e.getMessage());

}

if(standbyRecommendedId != null) {

bedroomProductMatchDTO.setStandbyRecommendedId(standbyRecommendedId);

}

}

// 返回小面积推荐方案的id ->end

return new ResponseEnvelope<BedroomProductMatchDTO>(true, bedroomProductMatchDTO, msgId);

}

/\*\*

\* 根据配置文件,生成设计方案,并返回设计方案信息(进入房间)

\*

\* @author huangsongbo

\* @param designTempletId

\* @param recommendedPlanId

\* @param configEncrypt

\* @param msgId

\* @param posNameInfoJson json格式,posName对应designPlanProductId的格式

\* @return

\* @throws Exception

\*/

@RequestMapping("/createPlanByConfig")

@ResponseBody

public Object createPlanByConfig(

Integer designTempletId,

Integer recommendedPlanId,

String configEncrypt,

String msgId,

String posNameInfoJson,

Integer opType,

HttpServletRequest request,

HttpServletResponse response

) throws Exception {

String logPrefixFunction = logPrefixClass + "createPlanByConfig -> ";

// \*参数验证/参数处理 ->start

if(StringUtils.isEmpty(msgId)){

return new ResponseEnvelope<>(false, "参数msgId不能为空", msgId);

}

if(StringUtils.isEmpty(posNameInfoJson)) {

return new ResponseEnvelope<>(false, "参数posNameInfo不能为空", msgId);

}

List<PosNameInfo> posNameInfoList = new ArrayList<PosNameInfo>();

try {

JSONArray jsonArray = JSONArray.fromObject(posNameInfoJson);

for (int index = 0; index < jsonArray.size(); index++) {

PosNameInfo posNameInfo = new PosNameInfo(); posNameInfo.setPosName(jsonArray.getJSONObject(index).getString("posName")); posNameInfo.setDeignPlanProductId(jsonArray.getJSONObject(index).getInt("deignPlanProductId"));

posNameInfoList.add(posNameInfo);

}

}catch (Exception e) {

return new ResponseEnvelope<>(false, "参数posNameInfo格式不正确", msgId);

}

if (opType == null) {

opType = DesignPlanConstants.USER\_RENDER;

}

LoginUser loginUser = SystemCommonUtil.getLoginUserFromRequest(request);

if(loginUser == null){

return new ResponseEnvelope<UnityDesignPlan>(false, "未检测到登录信息,请登录", msgId);

}

// 防止没有userName的账号 update by huangsongbo 2018.6.12

if(StringUtils.isEmpty(loginUser.getLoginName())) {

loginUser.setLoginName("noUsername");

}

logger.error("------createPlanByConfig -> loginUser:" + loginUser);

DesignTemplet designTemplet = designTempletService.getForOneKey(designTempletId);

if(designTemplet == null) {

logger.error(logPrefixFunction + "designTemplet = null;designTempletId = " + designTempletId);

return new ResponseEnvelope<>(false, "样板房未找到,一件装修失败", msgId);

}

// \*参数验证/参数处理 ->end

// \*附加信息 ->start

String mediaType = null!=loginUser?loginUser.getMediaType():null;

logger.error("------createPlanByConfig -> mediaType:" + mediaType);

// \*附加信息 ->end

Integer planId = designPlanService.createPlanByConfig(designTempletId, recommendedPlanId, configEncrypt, mediaType, loginUser, posNameInfoList, opType);

if(planId == null || planId < 1) {

return new ResponseEnvelope<>(false, "设计方案创建失败", msgId);

}

UnityDesignPlan unityDesignPlan = new UnityDesignPlan();

if(DesignPlanConstants.AUTO\_RENDER != opType) {

// \*取得设计方案信息,进入房间 ->start

@SuppressWarnings("unchecked")

ResponseEnvelope<UnityDesignPlan> responseEnvelopeInfo = (ResponseEnvelope<UnityDesignPlan>)

designPlanService.getDesignPlanInfoForOneKey(planId, null, null, null, null, null, loginUser, mediaType, Constants.NO\_CACHE);

if(responseEnvelopeInfo.isSuccess()) {

unityDesignPlan = (UnityDesignPlan)responseEnvelopeInfo.getObj();

}else {

return new ResponseEnvelope<>(false, "进入设计方案失败", msgId);

}

unityDesignPlan = designPlanService.wrapperData(planId, unityDesignPlan);

unityDesignPlan.setDesignTempletId(designTempletId);

// \*取得设计方案信息,进入房间 ->end

}else {

// 自动渲染

//获取设计方案信息

@SuppressWarnings("unchecked")

ResponseEnvelope<UnityDesignPlan> responseEnvelopeInfo = (ResponseEnvelope<UnityDesignPlan>) optimizePlanService

.getDesignPlanInfoForOneKey(planId, null, null, null, null, false, loginUser, mediaType);

if (responseEnvelopeInfo.isSuccess()) {

unityDesignPlan = (UnityDesignPlan) responseEnvelopeInfo.getObj();

} else {

logger.error(responseEnvelopeInfo.getMessage());

return new ResponseEnvelope<>(false, "进入设计方案失败！", msgId);

}

unityDesignPlan = optimizePlanService.wrapperData(planId, unityDesignPlan);

unityDesignPlan.setDesignTempletId(designTempletId);

}

Integer dataType = 1;

if(designTemplet.getOrigin() != null && 1 == designTemplet.getOrigin().intValue()) {

dataType = 2;

}else {

}

unityDesignPlan.setDataType(dataType);

return new ResponseEnvelope<UnityDesignPlan>(unityDesignPlan, msgId, true);

}

/\*\*

\* 查询样板房在推荐方案组中最匹配的方案ID

\* @param designTemplateId 样板房ID

\* @param recommendedPlanId 打组推荐方案的主ID

\* @return

\*/

@RequestMapping("/getBestMatchInPlanGroup")

@ResponseBody

public Object getBestMatchInPlanGroup(Integer designTemplateId, Integer recommendedPlanId){

if( designTemplateId == null || designTemplateId.intValue() < 1 ){

return new ResponseEnvelope<>(false, "样板房错误", null);

}

if( recommendedPlanId == null || recommendedPlanId.intValue() < 1 ){

return new ResponseEnvelope<>(false, "推荐方案错误", null);

}

DesignPlanRecommended designPlanRecommended = designPlanRecommendedService.get(recommendedPlanId);

if( designPlanRecommended == null

|| (designPlanRecommended.getGroupPrimaryId().intValue() == 0 || designPlanRecommended.getGroupPrimaryId().intValue() != designPlanRecommended.getId()) ){

return new ResponseEnvelope<>(false, "推荐方案错误", null);

}

Integer bestMatchPlanId = designPlanRecommendedService.getBestMatchInPlanGroup(designTemplateId, recommendedPlanId);

return new ResponseEnvelope<>(true, bestMatchPlanId, null);

}

/\*\*

\* 保存一键装修生成方案副本

\*

\* @author huangsongbo

\* @param recommendedPlanId 一键装修选择的推荐方案id

\* @param templetId 一键装修选择的样板房id

\* @param planId 一键装修生成的设计方案id

\* @param matchType 0:全屋替换 ;1:硬装替换

\* @param msgId

\* @return

\*/

@RequestMapping("/saveOnekeytemplate")

@ResponseBody

public Object saveOnekeytemplate(

Integer recommendedPlanId,

Integer templetId,

Integer planId,

Integer matchType,

String msgId

) {

// 参数验证 ->start

if(recommendedPlanId == null) {

return new ResponseEnvelope<>(false, "参数recommendedPlanId不能为空", msgId);

}

if(templetId == null) {

return new ResponseEnvelope<>(false, "参数templetId不能为空", msgId);

}

if(planId == null) {

return new ResponseEnvelope<>(false, "参数planId不能为空", msgId);

}

if(matchType == null) {

return new ResponseEnvelope<>(false, "参数matchType不能为空", msgId);

}

// 参数验证 ->end

// 检测副本开关,并做对应处理 add by huangsongbo 2018.9.5 ->start

boolean button = intelligenceDecorationService.getTemplateButton();

if(!button) {

return new ResponseEnvelope<>(true, "副本功能处于停用状态", msgId);

}

// 检测副本开关,并做对应处理 add by huangsongbo 2018.9.5 ->end

// 如果已存在副本,则不保存 ->start

// 不能避免并发现象

try {

boolean isStock = designPlanTemplateService.checkIsStock(recommendedPlanId, templetId, matchType);

if(isStock) {

return new ResponseEnvelope<>(true, "已存在对应副本,无需重复创建", msgId);

}

} catch (IntelligenceDecorationException e) {

logger.error(logPrefixClass + e.getMessage());

return new ResponseEnvelope<>(false, e.getMessage(), msgId);

}

// 如果已存在副本,则不保存 ->end

new Thread() {

@Override

public void run() {

try {

designPlanTemplateService.saveTemplate(recommendedPlanId, templetId, planId, matchType);

} catch (IntelligenceDecorationException e) {

e.printStackTrace();

}

super.run();

}

}.start();

return new ResponseEnvelope<>(true, "success", msgId);

}

/\*\*

\* 检测是否有对应的一键装修生成方案副本,如果有,则直接copy该副本转为designPlan,并进入房间

\* update by huangsongbo 2018.7.28

\* 如果是自动渲染,则copy到onekey\_decoration\_design\_plan

\*

\* @author huangsongbo

\* @param recommendedPlanId 一键装修选择的推荐方案id

\* @param templetId 一键装修选择的样板房id

\* @param matchType 0:全屋替换 ;1:硬装替换

\* @param opType 渲染机:0/通用版:1

\* @param msgId

\* @return

\*/

@RequestMapping("/checkTemplate")

@ResponseBody

public Object checkTemplate(

Integer recommendedPlanId,

Integer templetId,

Integer matchType,

Integer opType,

String msgId,

HttpServletRequest request

) {

// 参数验证 ->start

if(recommendedPlanId == null) {

return new ResponseEnvelope<>(false, "参数recommendedPlanId不能为空", msgId);

}

if(templetId == null) {

return new ResponseEnvelope<>(false, "参数templetId不能为空", msgId);

}

if(matchType == null) {

return new ResponseEnvelope<>(false, "参数matchType不能为空", msgId);

}

LoginUser loginUser = SystemCommonUtil.getLoginUserFromRequest(request);

if(loginUser == null){

return new ResponseEnvelope<UnityDesignPlan>(false, "未检测到登录信息,请登录", msgId);

}

// 参数验证 ->end

// 检测副本开关,并做对应处理 add by huangsongbo 2018.9.5 ->start

boolean button = intelligenceDecorationService.getTemplateButton();

if(!button) {

return new ResponseEnvelope<>(true, "副本功能处于停用状态", msgId, false);

}

// 检测副本开关,并做对应处理 add by huangsongbo 2018.9.5 ->end

try {

// check 有没有对应的副本

boolean isStock = designPlanTemplateService.checkIsStock(recommendedPlanId, templetId, matchType);

if(isStock) {

UnityDesignPlan unityDesignPlan = designPlanTemplateService.getUnityDesignPlan(recommendedPlanId, templetId, matchType, opType, loginUser);

ResponseEnvelope<UnityDesignPlan> result = new ResponseEnvelope<>(true, unityDesignPlan, msgId);

result.setFlag(true);

return result;

}else {

// update by huangsongbo 2018.8.1 ->start

// 由于前端,success = false, 会强制弹框,所以改为用flag来判断是否存在对应副本

return new ResponseEnvelope<>(true, "没有找到对应/可正常使用的副本", msgId, false);

// update by huangsongbo 2018.8.1 ->end

}

}catch (Exception e) {

logger.error(logPrefixClass + e.getMessage());

return new ResponseEnvelope<>(false, e.getMessage(), msgId);

}

}

}

package com.nork.onekeydesign.model;

import java.io.Serializable;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

import com.nork.common.model.Mapper;

/\*\*

\* 方案推荐

\* @author Administrator

\*

\*/

public class DesignPlanRecommended extends Mapper implements Serializable{

private static final long serialVersionUID = 1L;

private Integer id;

/\*方案推荐类型 1分享 2一键装修\*/

private Integer recommendedType;

private Integer planId;

/\*\* 方案编码 \*\*/

private String planCode;

/\*\* 方案名称 \*\*/

private String planName;

/\*\* 设计者id \*\*/

private Integer userId;

/\*\* 方案来源类型 \*\*/

private Integer designSourceType;

/\*\* 来源id \*\*/

private Integer designId;

/\*\* 设计风格 \*\*/

private Integer designStyleId;

/\*\* 方案图片id \*\*/

private Integer picId;

/\*\* 模型文件id \*\*/

private Integer model3dId;

/\*\* 模型文件id \*\*/

private Integer modelU3dId;

/\*\* 模型文件id \*\*/

private Integer modelId;

/\*\* 配置文件id \*\*/

private Integer configFileId;

/\*\* 空间id \*\*/

private Integer spaceCommonId;

/\*\* 方案描述 \*\*/

private String planDesc;

/\*\* 系统编码 \*\*/

private String sysCode;

/\*\* 创建者 \*\*/

private String creator;

/\*\* 创建时间 \*\*/

private Date gmtCreate;

/\*\* 修改人 \*\*/

private String modifier;

/\*\* 修改时间 \*\*/

private Date gmtModified;

/\*\* 是否删除 \*\*/

private Integer isDeleted;

/\*\* 备注 \*\*/

private String remark;

/\*\* 设计效果图 \*\*/

private String designPlanRenderPicIds;

/\*\* 样板房ID \*\*/

private Integer designTemplateId;

private Integer webModelU3dId;

private Integer iosModelU3dId;

private Integer androidModelU3dId;

private Integer pcModelU3dId;

private Integer ipadModelU3dId;

private Integer macBookpcModelU3dId;

/\*\* 媒介类型 \*\*/

private Integer mediaType;

/\*\* 分享数量 \*\*/

private Integer shareTotal;

/\*公开\*/

private Integer isOpen;

//草稿状态

private Integer draftState;

//白模状态

private Integer baiMoState;

//硬装完成状态

private Integer stuffFinishState;

//装修完成状态

private Integer decorateFinishState;

//方案是否变更

private Integer isChange;

//设计方案是否装修过 （1 装修过，0 没装修过）

private Integer isDecorated;

/\*\*方案来源\*/

private String planSource;

/\*\*小区户型名称\*/

private String residentialUnitsName;

private Integer houseId;

private Integer livingId;

/\*\*设计方案封面\*/

private Integer coverPicId;

/\*方案推荐风格ID\*/

private Integer designRecommendedStyleId;

/\*如果这个有值，默认用这个风格将数据置顶，只用于一件装修页面\*/

private Integer designRecommendedStyleIdTop;

/\*设计方案是否发布 0 否 1是\*/

private Integer isRelease;

private List<Integer>IsReleases;

/\*设计方案发布时间 \*/

private Date releaseTime;

/\*特效配置\*/

private String effectsConfig;

/\*发布权限 0无 1有\*/

private Integer releasePermissions;

/\*复制权限 0无 1有\*/

private Integer copyPermissions;

/\*多功能查询， 如果 前后带@号，搜索品牌，把推荐给该品牌的方案筛选出来，否则只是模糊查询方案名\*/

private String multifunctionQuery;

private String brandName;

/\*720渲染图二维码路径\*/

private String qrCode720Path;

/\*是否默认推荐 1 默认 0 非默认\*/

private Integer isRecommended;

/\*是否支持一键装修 1 支持 0不支持\*/

private Integer isDefaultDecorate;

/\*方案编号\*/

private String planNumber;

/\*多点渲染图\*/

private Integer render720MultipointPicId;

/\*全景渲染图id\*/

private Integer render720PanoramaPicId;

/\*渲染视频id\*/

private Integer render720VideoId;

private String displayType;

/\*\*用于列表查询 begin>>\*/

/\*\*品牌id \*/

private List<Integer> BrandIds;

/\*\*空间类型value\*/

private Integer SpaceFunctionId;

/\*\*多个空间类型value\*/

List<Integer>spaceFunctionIds ;

/\*\* 面积类型value\*/

private String areaValue;

/\*\* 小区名\*/

private String livingName;

/\*\* 是否为内部用户\*/

private String IsInternalUser;

/\*\*\* 是否为方案审核管理员\*/

private String checkAdministrator;

/\*\*end<<\*/

private Integer picType;

private List<RenderPicInfo> picList = new ArrayList<RenderPicInfo>();

private String planRecommendedUserName;

private String spaceCode;

private String spaceName;

private String spaceAreas;

/\*该方案推荐的产品数量\*/

private Integer planRecommendedProductCount;

// '是否装修完成状态 (1.未装修完成 2.已装修完成)'

private Integer recommendedDecorateState;

private Integer designPlanId;

private String favoriteBid;

//适用于空间面积（一键装修）

private String applySpaceAreas;

/\*\*\* 拼花 json 传\*/

private Integer spellingFlowerFileId;

/\*\* \* 拼花 产品ids \*/

private String spellingFlowerProduct;

/\*\* 方案包主方案ID \*\*/

private Integer groupPrimaryId;

public Integer getGroupPrimaryId() {

return groupPrimaryId;

}

public void setGroupPrimaryId(Integer groupPrimaryId) {

this.groupPrimaryId = groupPrimaryId;

}

public String getSpellingFlowerProduct() {

return spellingFlowerProduct;

}

public void setSpellingFlowerProduct(String spellingFlowerProduct) {

this.spellingFlowerProduct = spellingFlowerProduct;

}

public Integer getSpellingFlowerFileId() {

return spellingFlowerFileId;

}

public void setSpellingFlowerFileId(Integer spellingFlowerFileId) {

this.spellingFlowerFileId = spellingFlowerFileId;

}

public Integer getDesignRecommendedStyleIdTop() {

return designRecommendedStyleIdTop;

}

public void setDesignRecommendedStyleIdTop(Integer designRecommendedStyleIdTop) {

this.designRecommendedStyleIdTop = designRecommendedStyleIdTop;

}

public String getApplySpaceAreas() {

return applySpaceAreas;

}

public void setApplySpaceAreas(String applySpaceAreas) {

this.applySpaceAreas = applySpaceAreas;

}

public String getFavoriteBid() {

return favoriteBid;

}

public void setFavoriteBid(String favoriteBid) {

this.favoriteBid = favoriteBid;

}

public Integer getRecommendedDecorateState() {

return recommendedDecorateState;

}

public void setRecommendedDecorateState(Integer recommendedDecorateState) {

this.recommendedDecorateState = recommendedDecorateState;

}

public Integer getPlanRecommendedProductCount() {

return planRecommendedProductCount;

}

public void setPlanRecommendedProductCount(Integer planRecommendedProductCount) {

this.planRecommendedProductCount = planRecommendedProductCount;

}

public String getSpaceCode() {

return spaceCode;

}

public void setSpaceCode(String spaceCode) {

this.spaceCode = spaceCode;

}

public String getSpaceName() {

return spaceName;

}

public void setSpaceName(String spaceName) {

this.spaceName = spaceName;

}

public String getSpaceAreas() {

return spaceAreas;

}

public void setSpaceAreas(String spaceAreas) {

this.spaceAreas = spaceAreas;

}

public String getPlanRecommendedUserName() {

return planRecommendedUserName;

}

public void setPlanRecommendedUserName(String planRecommendedUserName) {

this.planRecommendedUserName = planRecommendedUserName;

}

public List<RenderPicInfo> getPicList() {

return picList;

}

public void setPicList(List<RenderPicInfo> picList) {

this.picList = picList;

}

public Integer getPicType() {

return picType;

}

public void setPicType(Integer picType) {

this.picType = picType;

}

public String getDisplayType() {

return displayType;

}

public void setDisplayType(String displayType) {

this.displayType = displayType;

}

public List<Integer> getSpaceFunctionIds() {

return spaceFunctionIds;

}

public String getCheckAdministrator() {

return checkAdministrator;

}

public void setCheckAdministrator(String checkAdministrator) {

this.checkAdministrator = checkAdministrator;

}

public void setSpaceFunctionIds(List<Integer> spaceFunctionIds) {

this.spaceFunctionIds = spaceFunctionIds;

}

public List<Integer> getIsReleases() {

return IsReleases;

}

public void setIsReleases(List<Integer> isReleases) {

IsReleases = isReleases;

}

public String getLivingName() {

return livingName;

}

public void setLivingName(String livingName) {

this.livingName = livingName;

}

public String getIsInternalUser() {

return IsInternalUser;

}

public void setIsInternalUser(String isInternalUser) {

IsInternalUser = isInternalUser;

}

public String getAreaValue() {

return areaValue;

}

public void setAreaValue(String areaValue) {

this.areaValue = areaValue;

}

public Integer getSpaceFunctionId() {

return SpaceFunctionId;

}

public void setSpaceFunctionId(Integer spaceFunctionId) {

SpaceFunctionId = spaceFunctionId;

}

public List<Integer> getBrandIds() {

return BrandIds;

}

public void setBrandIds(List<Integer> brandIds) {

BrandIds = brandIds;

}

public Integer getRecommendedType() {

return recommendedType;

}

public void setRecommendedType(Integer recommendedType) {

this.recommendedType = recommendedType;

}

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public Integer getPlanId() {

return planId;

}

public void setPlanId(Integer planId) {

this.planId = planId;

}

public String getPlanCode() {

return planCode;

}

public void setPlanCode(String planCode) {

this.planCode = planCode;

}

public String getPlanName() {

return planName;

}

public void setPlanName(String planName) {

this.planName = planName;

}

public Integer getUserId() {

return userId;

}

public void setUserId(Integer userId) {

this.userId = userId;

}

public Integer getDesignSourceType() {

return designSourceType;

}

public void setDesignSourceType(Integer designSourceType) {

this.designSourceType = designSourceType;

}

public Integer getDesignId() {

return designId;

}

public void setDesignId(Integer designId) {

this.designId = designId;

}

public Integer getDesignStyleId() {

return designStyleId;

}

public void setDesignStyleId(Integer designStyleId) {

this.designStyleId = designStyleId;

}

public Integer getPicId() {

return picId;

}

public void setPicId(Integer picId) {

this.picId = picId;

}

public Integer getModel3dId() {

return model3dId;

}

public void setModel3dId(Integer model3dId) {

this.model3dId = model3dId;

}

public Integer getModelU3dId() {

return modelU3dId;

}

public void setModelU3dId(Integer modelU3dId) {

this.modelU3dId = modelU3dId;

}

public Integer getModelId() {

return modelId;

}

public void setModelId(Integer modelId) {

this.modelId = modelId;

}

public Integer getConfigFileId() {

return configFileId;

}

public void setConfigFileId(Integer configFileId) {

this.configFileId = configFileId;

}

public Integer getSpaceCommonId() {

return spaceCommonId;

}

public void setSpaceCommonId(Integer spaceCommonId) {

this.spaceCommonId = spaceCommonId;

}

public String getPlanDesc() {

return planDesc;

}

public void setPlanDesc(String planDesc) {

this.planDesc = planDesc;

}

public String getSysCode() {

return sysCode;

}

public void setSysCode(String sysCode) {

this.sysCode = sysCode;

}

public String getCreator() {

return creator;

}

public void setCreator(String creator) {

this.creator = creator;

}

public Date getGmtCreate() {

return gmtCreate;

}

public void setGmtCreate(Date gmtCreate) {

this.gmtCreate = gmtCreate;

}

public String getModifier() {

return modifier;

}

public void setModifier(String modifier) {

this.modifier = modifier;

}

public Date getGmtModified() {

return gmtModified;

}

public void setGmtModified(Date gmtModified) {

this.gmtModified = gmtModified;

}

public Integer getIsDeleted() {

return isDeleted;

}

public void setIsDeleted(Integer isDeleted) {

this.isDeleted = isDeleted;

}

public String getRemark() {

return remark;

}

public void setRemark(String remark) {

this.remark = remark;

}

public String getDesignPlanRenderPicIds() {

return designPlanRenderPicIds;

}

public void setDesignPlanRenderPicIds(String designPlanRenderPicIds) {

this.designPlanRenderPicIds = designPlanRenderPicIds;

}

public Integer getDesignTemplateId() {

return designTemplateId;

}

public void setDesignTemplateId(Integer designTemplateId) {

this.designTemplateId = designTemplateId;

}

public Integer getWebModelU3dId() {

return webModelU3dId;

}

public void setWebModelU3dId(Integer webModelU3dId) {

this.webModelU3dId = webModelU3dId;

}

public Integer getIosModelU3dId() {

return iosModelU3dId;

}

public void setIosModelU3dId(Integer iosModelU3dId) {

this.iosModelU3dId = iosModelU3dId;

}

public Integer getAndroidModelU3dId() {

return androidModelU3dId;

}

public void setAndroidModelU3dId(Integer androidModelU3dId) {

this.androidModelU3dId = androidModelU3dId;

}

public Integer getPcModelU3dId() {

return pcModelU3dId;

}

public void setPcModelU3dId(Integer pcModelU3dId) {

this.pcModelU3dId = pcModelU3dId;

}

public Integer getIpadModelU3dId() {

return ipadModelU3dId;

}

public void setIpadModelU3dId(Integer ipadModelU3dId) {

this.ipadModelU3dId = ipadModelU3dId;

}

public Integer getMacBookpcModelU3dId() {

return macBookpcModelU3dId;

}

public void setMacBookpcModelU3dId(Integer macBookpcModelU3dId) {

this.macBookpcModelU3dId = macBookpcModelU3dId;

}

public Integer getMediaType() {

return mediaType;

}

public void setMediaType(Integer mediaType) {

this.mediaType = mediaType;

}

public Integer getShareTotal() {

return shareTotal;

}

public void setShareTotal(Integer shareTotal) {

this.shareTotal = shareTotal;

}

public Integer getIsOpen() {

return isOpen;

}

public void setIsOpen(Integer isOpen) {

this.isOpen = isOpen;

}

public Integer getDraftState() {

return draftState;

}

public void setDraftState(Integer draftState) {

this.draftState = draftState;

}

public Integer getBaiMoState() {

return baiMoState;

}

public void setBaiMoState(Integer baiMoState) {

this.baiMoState = baiMoState;

}

public Integer getStuffFinishState() {

return stuffFinishState;

}

public void setStuffFinishState(Integer stuffFinishState) {

this.stuffFinishState = stuffFinishState;

}

public Integer getDecorateFinishState() {

return decorateFinishState;

}

public void setDecorateFinishState(Integer decorateFinishState) {

this.decorateFinishState = decorateFinishState;

}

public Integer getIsChange() {

return isChange;

}

public void setIsChange(Integer isChange) {

this.isChange = isChange;

}

public Integer getIsDecorated() {

return isDecorated;

}

public void setIsDecorated(Integer isDecorated) {

this.isDecorated = isDecorated;

}

public String getPlanSource() {

return planSource;

}

public void setPlanSource(String planSource) {

this.planSource = planSource;

}

public String getResidentialUnitsName() {

return residentialUnitsName;

}

public void setResidentialUnitsName(String residentialUnitsName) {

this.residentialUnitsName = residentialUnitsName;

}

public Integer getHouseId() {

return houseId;

}

public void setHouseId(Integer houseId) {

this.houseId = houseId;

}

public Integer getLivingId() {

return livingId;

}

public void setLivingId(Integer livingId) {

this.livingId = livingId;

}

public Integer getCoverPicId() {

return coverPicId;

}

public void setCoverPicId(Integer coverPicId) {

this.coverPicId = coverPicId;

}

public Integer getDesignRecommendedStyleId() {

return designRecommendedStyleId;

}

public void setDesignRecommendedStyleId(Integer designRecommendedStyleId) {

this.designRecommendedStyleId = designRecommendedStyleId;

}

public Integer getIsRelease() {

return isRelease;

}

public void setIsRelease(Integer isRelease) {

this.isRelease = isRelease;

}

public Date getReleaseTime() {

return releaseTime;

}

public void setReleaseTime(Date releaseTime) {

this.releaseTime = releaseTime;

}

public String getEffectsConfig() {

return effectsConfig;

}

public void setEffectsConfig(String effectsConfig) {

this.effectsConfig = effectsConfig;

}

public Integer getReleasePermissions() {

return releasePermissions;

}

public void setReleasePermissions(Integer releasePermissions) {

this.releasePermissions = releasePermissions;

}

public Integer getCopyPermissions() {

return copyPermissions;

}

public void setCopyPermissions(Integer copyPermissions) {

this.copyPermissions = copyPermissions;

}

public String getMultifunctionQuery() {

return multifunctionQuery;

}

public void setMultifunctionQuery(String multifunctionQuery) {

this.multifunctionQuery = multifunctionQuery;

}

public String getBrandName() {

return brandName;

}

public void setBrandName(String brandName) {

this.brandName = brandName;

}

public String getQrCode720Path() {

return qrCode720Path;

}

public void setQrCode720Path(String qrCode720Path) {

this.qrCode720Path = qrCode720Path;

}

public Integer getIsRecommended() {

return isRecommended;

}

public void setIsRecommended(Integer isRecommended) {

this.isRecommended = isRecommended;

}

public Integer getIsDefaultDecorate() {

return isDefaultDecorate;

}

public void setIsDefaultDecorate(Integer isDefaultDecorate) {

this.isDefaultDecorate = isDefaultDecorate;

}

public String getPlanNumber() {

return planNumber;

}

public void setPlanNumber(String planNumber) {

this.planNumber = planNumber;

}

public Integer getRender720MultipointPicId() {

return render720MultipointPicId;

}

public void setRender720MultipointPicId(Integer render720MultipointPicId) {

this.render720MultipointPicId = render720MultipointPicId;

}

public Integer getRender720PanoramaPicId() {

return render720PanoramaPicId;

}

public void setRender720PanoramaPicId(Integer render720PanoramaPicId) {

this.render720PanoramaPicId = render720PanoramaPicId;

}

public Integer getRender720VideoId() {

return render720VideoId;

}

public void setRender720VideoId(Integer render720VideoId) {

this.render720VideoId = render720VideoId;

}

public Integer getDesignPlanId() {

return designPlanId;

}

public void setDesignPlanId(Integer designPlanId) {

this.designPlanId = designPlanId;

}

public DesignPlan recommendedCopy(){

DesignPlan obj = new DesignPlan();

obj.setPlanCode(this.planCode);

obj.setPlanName(this.planName);

obj.setPlanSource(this.planSource);

obj.setHouseId(this.houseId);

obj.setLivingId(this.livingId);

obj.setResidentialUnitsName(residentialUnitsName);

obj.setUserId(this.userId);

obj.setDesignSourceType(this.designSourceType);

obj.setDesignId(this.designId);

obj.setDesignStyleId(this.designStyleId);

obj.setPicId(this.picId);

obj.setModelId(this.modelId);

obj.setModel3dId(this.model3dId);

obj.setConfigFileId(this.configFileId);

obj.setSpaceCommonId(this.spaceCommonId);

obj.setPlanDesc(this.planDesc);

obj.setSysCode(this.sysCode);

obj.setCreator(this.creator);

obj.setGmtCreate(this.gmtCreate);

obj.setModifier(this.modifier);

obj.setGmtModified(this.gmtModified);

obj.setIsDeleted(this.isDeleted);

obj.setRemark(this.remark);

obj.setDesignTemplateId(this.designTemplateId);

obj.setIpadModelU3dId(this.ipadModelU3dId);

obj.setIosModelU3dId(this.iosModelU3dId);

obj.setAndroidModelU3dId(this.androidModelU3dId);

obj.setMacBookpcModelU3dId(this.macBookpcModelU3dId);

obj.setPcModelU3dId(this.pcModelU3dId);

obj.setWebModelU3dId(this.webModelU3dId);

obj.setMediaType(this.mediaType);

/\*obj.setEveningFileId(this.eveningFileId);

obj.setDawnFileId(this.dawnFileId );

obj.setNightFileId(this.nightFileId );\*/

obj.setShareTotal(this.shareTotal);

obj.setIsOpen(this.isOpen );

obj.setDraftState(this.draftState );

obj.setBaiMoState(this.baiMoState );

obj.setStuffFinishState(this.stuffFinishState);

obj.setDecorateFinishState(this.decorateFinishState);

obj.setIsChange(this.isChange);

obj.setIsDecorated(this.isDecorated);

obj.setPlanSource(this.planSource);

obj.setResidentialUnitsName(this.residentialUnitsName );

obj.setHouseId(this.houseId);

obj.setLivingId(this.livingId);

obj.setEffectsConfig(this.effectsConfig);

obj.setSpellingFlowerProduct(this.spellingFlowerProduct);

return obj;

}

}

package com.nork.intelligence.constant;

/\*\*

\* 一件装修相关常量

\*

\* @author huangsongbo

\*

\*/

public class IntelligenceDecorationConstant {

public static String PRODUCTLISTMATCHRESULTMAP\_PRODUCTMATCHINFO = "productMatchInfo";

public static String PRODUCTLISTMATCHRESULTMAP\_DIMMATCHINFOVO = "dimMatchInfo";

public static String GROUPMATCHRESULTMAP\_GROUPMATCHINFO = "groupMatchInfo";

public static String GROUPMATCHRESULTMAP\_PRODUCTLISTBYTYPEINFO = "productListByTypeInfo";

package com.nork.onekeydesign.service.impl;

import java.io.File;

import java.util.ArrayList;

import java.util.Arrays;

import java.util.Collections;

import java.util.Comparator;

import java.util.Date;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.Map.Entry;

import java.util.Objects;

import java.util.regex.Pattern;

import com.aliyun.openservices.shade.com.alibaba.rocketmq.shade.com.alibaba.fastjson.JSON;

import com.nork.common.constant.SystemCommonConstant;

import com.nork.onekeydesign.cache.DesignPlanCacher;

import com.nork.onekeydesign.model.\*;

import com.nork.onekeydesign.model.constant.PlanVisibleCode;

import com.nork.onekeydesign.model.vo.DimMatchInfoVO;

import com.nork.onekeydesign.model.vo.MatchItenCodeVO;

import com.nork.onekeydesign.model.vo.PosNameInfoVO;

import com.nork.onekeydesign.model.vo.StructureDetailInfoVO;

import com.nork.onekeydesign.service.\*;

import com.nork.intelligence.constant.IntelligenceDecorationConstant;

import com.nork.pay.metadata.ProductType;

import com.nork.product.model.\*;

import com.nork.system.cache.ResourceCacher;

import net.sf.json.groovy.GJson;

import org.apache.commons.lang3.StringUtils;

import org.apache.log4j.LogManager;

import org.apache.log4j.Logger;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.google.gson.Gson;

import com.google.gson.reflect.TypeToken;

import com.nork.aes.model.AESFileConstant;

import com.nork.common.model.LoginUser;

import com.nork.common.model.ResponseEnvelope;

import com.nork.common.properties.AppProperties;

import com.nork.common.util.Constants;

import com.nork.common.util.FileUploadUtils;

import com.nork.common.util.FtpUploadUtils;

import com.nork.common.util.Utils;

import com.nork.common.util.collections.CustomerListUtils;

import com.nork.common.util.collections.Lists;

import com.nork.onekeydesign.cache.DesignCacher;

import com.nork.onekeydesign.common.DesignPlanConstants;

import com.nork.onekeydesign.dao.DesignTempletProductMapper;

import com.nork.onekeydesign.exception.IntelligenceDecorationException;

import com.nork.onekeydesign.exception.IntelligenceDecorationExceptionEnum;

import com.nork.onekeydesign.model.BedroomProductMatchDTO.DetailInfo;

import com.nork.onekeydesign.model.BedroomProductMatchDTO.GroupMatchInfoDTO;

import com.nork.onekeydesign.model.BedroomProductMatchDTO.ProductMatchInfoDTO;

import com.nork.onekeydesign.model.BedroomProductMatchDTO.StructureMatchInfoDTO;

import com.nork.onekeydesign.model.BedroomProductMatchDTO.WallTypeConstant;

import com.nork.onekeydesign.model.ProductListByTypeInfo.PlanGroupInfo;

import com.nork.onekeydesign.model.ProductListByTypeInfo.PlanProductInfo;

import com.nork.onekeydesign.model.ProductListByTypeInfo.PlanStructureInfo;

import com.nork.onekeydesign.model.unity.ProductDataEx;

import com.nork.onekeydesign.model.unity.ProductIndexConfig;

import com.nork.product.dao.BaseProductMapper;

import com.nork.product.model.result.DesignProductResult;

import com.nork.product.model.result.SearchStructureProductResult;

import com.nork.product.model.search.BaseProductSearch;

import com.nork.product.model.search.StructureProductSearch;

import com.nork.product.service.BaseProductService;

import com.nork.product.service.GroupProductService;

import com.nork.product.service.StructureProductService;

import com.nork.product.service.impl.BaseProductServiceImpl.getSmallTypeValueListBySmallTypeValueKeyMapKeyEnum;

import com.nork.product.service.impl.ProductCategoryRelServiceImpl;

import com.nork.productprops.constant.ProductAttributePropValue;

import com.nork.productprops.constant.ProductPropsConstant;

import com.nork.productprops.model.ProductPropsSimple;

import com.nork.productprops.service.ProductPropsService;

import com.nork.system.model.ResDesign;

import com.nork.system.model.ResFile;

import com.nork.system.model.ResTexture;

import com.nork.system.service.ResDesignService;

import com.nork.system.service.ResFileService;

import com.nork.system.service.ResTextureService;

import com.nork.system.service.SysDictionaryService;

import com.nork.user.model.UserTypeCode;

import com.sandu.search.entity.elasticsearch.search.product.ProductSearchForOneKeyDTO;

import com.sandu.search.exception.ElasticSearchException;

import net.sf.json.JSONObject;

@Service("intelligenceDecorationService")

@Transactional

public class IntelligenceDecorationServiceImpl implements IntelligenceDecorationService {

private static Logger logger = LogManager.getLogger(IntelligenceDecorationServiceImpl.class);

private final static String CLASS\_LOG\_PREFIX = "[一键生成服务]:";

// 白膜背景墙valukeyList

private static List<String> beijingValuekeyList = Utils.getListFromStr(

Utils.getValueByFileKey(AppProperties.APP, AppProperties.SMALLPRODUCTTYPE\_BEIJINGWALL\_FILEKEY, "basic\_dians,basic\_shaf,basic\_cant,basic\_chuangt,basic\_xingx,basic\_beij")

);

private static List<Integer> beijingValueList = null;

private static int beijingRange = Integer.parseInt(Utils.getValue("app.filter.stretch.length", "10"));

private static final String CLASSNAME = "[IntelligenceDecorationServiceImpl]";

@Autowired

private DesignPlanService designPlanService;

@Autowired

private DesignTempletService designTempletService;

@Autowired

private ResDesignService resDesignService;

@Autowired

private ResFileService resFileService;

@Autowired

private BaseProductService baseProductService;

@Autowired

private BaseProductMapper baseProductMapper;

@Autowired

private SysDictionaryService sysDictionaryService;

@Autowired

private DesignPlanProductService designPlanProductService;

@Autowired

private StructureProductService structureProductService;

@Autowired

private DesignTempletProductService designTempletProductService;

@Autowired

private DesignTempletProductMapper designTempletProductMapper;

@Autowired

private TempletProductIndexInfoService templetProductIndexInfoService;

@Autowired

private DesignPlanRecommendedService designPlanRecommendedServiceV2;

@Autowired

private DesignPlanRecommendedProductServiceV2 designPlanRecommendedProductServiceV2;

@Autowired

private GroupProductService groupProductService;

@Autowired

private ProductPropsService productPropsService;

@Autowired

private OptimizePlanService optimizePlanService;

@Autowired

private DesignPlanOperationLogService designPlanOperationLogService;

@Autowired

private ResTextureService resTextureService;

private String logPrefixClass = "function:IntelligenceDecorationServiceImpl.";

/\*\*

\* @param recommendedPlanId 推荐方案ID

\* @param designTempletId 样板房ID

\* @param msgId

\* @return

\*/

@Override

public ResponseEnvelope<Map<String,String>> findDesignConfig(Integer recommendedPlanId, Integer designTempletId, String msgId) {

logger.info(CLASS\_LOG\_PREFIX + "进入findDesignConfig(),Parma ：recommendedPlanId = " + recommendedPlanId);

ResponseEnvelope <Map<String,String>>result = null;

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

DesignPlanRecommended designPlanRecommended = designPlanRecommendedServiceV2.get(recommendedPlanId);

DesignTemplet designTemplet = designTempletService.get(designTempletId);

if (designPlanRecommended == null || designTemplet == null) {

result = new ResponseEnvelope<>(false, "找不到推荐的方案或样板房！", msgId);

return result;

}

if (designPlanRecommended.getDesignTemplateId() == designTempletId) {

result = new ResponseEnvelope<Map<String,String>>(true, map, msgId);

return result;

}

if (designPlanRecommended.getConfigFileId() != null) {

ResDesign resDesign = resDesignService.get(designPlanRecommended.getConfigFileId());

map.put("designPlanRecommendConfigPath", resDesign == null ? "" : resDesign.getFilePath());

}

if (designTemplet.getConfigFileId() != null) {

ResFile resFile = resFileService.get(designTemplet.getConfigFileId());

map.put("designTempletConfigPath", resFile == null ? "" : resFile.getFilePath());

}

result = new ResponseEnvelope<Map<String,String>>(true, map, msgId);//success, obj, msgId

return result;

}

@Override

public ResponseEnvelope<BaseProduct> disposeOneKeyDecorateData(String templetProductCode, String designTempletProductCode,

String msgId, LoginUser loginUser) {

ResponseEnvelope<BaseProduct> result = null;

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

BaseProduct aseProduct = baseProductService.findDataByCode(templetProductCode);

BaseProduct bmBaseProduct = baseProductService.findDataByCode(designTempletProductCode);

if(baseProduct == null || bmBaseProduct == null) {

result = new ResponseEnvelope<BaseProduct>(false, "找不到产品productCode="+templetProductCode+" 或 "+designTempletProductCode, msgId);

logger.error("找不到产品productCode="+templetProductCode+" 或 "+designTempletProductCode);

return result;

}

BaseProductSearch searchProduct = new BaseProductSearch();

if("1".equals(baseProduct.getProductTypeValue())||3".equals(baseProduct.getProductTypeValue())) {

searchProduct.setBmIds(String.valueOf(bmBaseProduct.getId()));

} else {

searchProduct.setProductTypeValue(bmBaseProduct.getProductTypeValue());

}

searchProduct.setProductSmallTypeValue(baseProduct.getProductSmallTypeValue());

searchProduct.setProductTypeCode(baseProduct.getProductTypeMark());

searchProduct.setPutawayStateList(baseProductService.getPutawayList(loginUser));

searchProduct.setStyleId(baseProduct.getStyleId());

searchProduct.setProductStyleIdInfo(baseProduct.getProductStyleIdInfo());

searchProduct.setColorId(baseProduct.getColorId());

searchProduct.setStart(0);

searchProduct.setLimit(1);

List<BaseProduct> productList = baseProductService.getProductData(searchProduct);

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

BaseProduct productObj = productList.get(0);

map.put("productCode",productObj.getProductCode());

map.put("bmProductCode",designTempletProductCode);

}

return new ResponseEnvelope<BaseProduct>(true, map, msgId);

}

/\*\*

\* 设置返回的map值

\* @author xiaoxc

\* @param templetProduct

\* @param planProduct

\* @return

\*/

Public osNameInfoVO setPosNameInfoVO(DesignProductResult templetProduct,DesignProductResult planProduct){

//区分墙结构 0、实体墙 1、背景墙 2、门框

String bgWall = "0";

if (ProductTypeConstant.PRODUCT\_BIG\_TYPE\_QIANGM.equals(templetProduct.getBigTypeKey())) {

bgWall = Utils.getIsBgWall(templetProduct.getSmallTypeKey())+"";

}

PosNameInfoVO posNameInfoVO = new PosNameInfoVO();

posNameInfoVO.setBmPosName(templetProduct.getPosName());

posNameInfoVO.setPlanPosName(planProduct.getPosName());

posNameInfoVO.setProductCode(planProduct.getProductCode());

posNameInfoVO.setIsBgWall(bgWall);

posNameInfoVO.setMatchLogInfo(templetProduct.getMatchErrorInfo());

return posNameInfoVO;

}

/\*\*

\* 设置返回匹配不到的产品map值

\* @author xiaoxc

\* @param templetProduct

\* @return

\*/

public PosNameInfoVO setPosNameInfoVO(DesignProductResult templetProduct,List<DesignProductResult> unMatchProductList){

unMatchProductList.add(templetProduct);

PosNameInfoVO posNameInfoVO = new PosNameInfoVO();

posNameInfoVO.setBmPosName(templetProduct.getPosName());

posNameInfoVO.setIsBgWall("0");

posNameInfoVO.setMatchLogInfo(templetProduct.getMatchErrorInfo());

return posNameInfoVO;

}

public boolean isMatchingPlanProduct(DesignProductResult templetProduct, DesignProductResult planProduct, StringBuffer matchErrorInfo){

int planProductLength = Utils.getIntValue(planProduct.getProductLength());

int planProductHeight = Utils.getIntValue(planProduct.getProductHeight());

int planProductWidth = Utils.getIntValue(planProduct.getProductWidth());

int templetProductLength = Utils.getIntValue(templetProduct.getProductLength());

int templetProductHeight = Utils.getIntValue(templetProduct.getProductHeight());

int templetProductWidth = Utils.getIntValue(templetProduct.getProductWidth());

Map<String,String> stretchZoomMap = baseProductService.getStretchZoomLength(templetProduct.getSmallTypeKey());

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

int stretchZoomLength = Utils.getIntValue(stretchZoomMap.get(ProductModelConstant.STRETCH\_LENGTH));

String isHeightFilter = stretchZoomMap.get(ProductModelConstant.IS\_HEIGHT\_FILTER);

int fullPaveLength = 0;

int startLength = 0;

int endLength = 0;

if (StringUtils.isNotEmpty(templetProduct.getFullPaveLength())) {

fullPaveLength = Utils.getIntValue(templetProduct.getFullPaveLength()) ;

}else{}

if (fullPaveLength < 1) {

matchErrorInfo.append("白膜产品编码"+templetProduct.getProductCode()+"无全铺长度;");

templetProduct.setMatchErrorInfo(matchErrorInfo.toString());

return false;

}else {

startLength = fullPaveLength - stretchZoomLength;

endLength = fullPaveLength + stretchZoomLength;

}

if (startLength < planProductLength && planProductLength <= endLength){

if ("3".equals(templetProduct.getBigTypeValue())) { //背景墙逻辑

if ("30".equals(templetProduct.getProductHeight()) || "50".equals(templetProduct.getProductHeight())) {

if (planProductHeight == templetProductHeight) {

return true;

}else {

return false;

}

}else{

if (planProductHeight == templetProductHeight || (planProductHeight > 50 && planProductHeight < 230)) {

return true;

}else{

return false;

}

}

}else {//窗帘、淋浴屏。。。

if (ProductModelConstant.HEIGHT\_FILTER\_VALUE\_YES.equals(isHeightFilter)) {

if (planProductHeight == templetProductHeight) {

return true;

}else {

return false;

}

}else{

return true;

}

}

}else{

return false;

}

}

//过滤产品长、 高信息

String productSmallTypes\_LH = Utils.getValue("filter.productLH.productSmallType","");

if (Utils.isMateProductType(templetProduct.getSmallTypeKey(),productSmallTypes\_LH)) {

if (planProductHeight == templetProductHeight && planProductLength == templetProductLength) {

return true;

}else{

return false;

}

}

//过滤产品长、宽信息

String productSmallTypes\_LW = Utils.getValue("filter.productLW.productSmallType","");

if (Utils.isMateProductType(templetProduct.getSmallTypeKey(),productSmallTypes\_LW)) {

if (planProductLength == templetProductLength && planProductWidth <= templetProductWidth) {

return true;

}else{

return false;

}

}

return true;

}

/\*\*

\* 按条件匹配系统单品

\* @author xiaoxc

\* @param templetProduct

\* @param planProduct

\* @param loginUser

\* @return

\*/

private BaseProduct getMatchSystemProduct(DesignProductResult templetProduct,DesignProductResult planProduct,LoginUser loginUser){

BaseProductSearch searchProduct = new BaseProductSearch();

searchProduct.setProductTypeValue(templetProduct.getBigTypeValue());

baseProductService.getProductInfoFilter(searchProduct,templetProduct);

searchProduct.setProductTypeCode(templetProduct.getBigTypeKey());

searchProduct.setPutawayStateList(baseProductService.getPutawayList(loginUser));

searchProduct.setStart(0);

searchProduct.setLimit(1);

searchProduct.setProductStyleIdInfo(planProduct.getProductStyleIdInfo());

searchProduct.setColorId(planProduct.getColorId());

searchProduct.setProductSmallTypeValue(planProduct.getSmallTypeValue());

searchProduct.setStyleId(planProduct.getProStyleId());

searchProduct.setProductModelNumber(planProduct.getProductModelNumber());

if ("tianh".equals(templetProduct.getBigTypeKey())) {

searchProduct.setMeasureCode(templetProduct.getMeasureCode());

searchProduct.setStyleId(planProduct.getStyleId());

}

// 如果是背景墙

if(beijingValuekeyList.indexOf(templetProduct.getSmallTypeKey()) != -1) {

// 可以搜索多小类,但是本小类排前面

searchProduct.setProductSmallTypeValue(null);

searchProduct.setSmallTypeValueListForShowAll(this.getBeijingValueList());

searchProduct.setOrderSmallTypeValue(planProduct.getSmallTypeValue());

}

return this.getMatchProductDate(searchProduct);

}

/\*\*

\* 由于sql查询排序较慢，则由程序处理

\* @param searchProduct

\* @return

\*/

private BaseProduct getMatchProductDate(BaseProductSearch searchProduct){

BaseProduct matchProductData = null;

Integer orderSmallTypeValue = searchProduct.getOrderSmallTypeValue();

/\*Integer styleId = searchProduct.getStyleId();\*/

String productModelNumber = searchProduct.getProductModelNumber();

String productStyleIdInfo = searchProduct.getProductStyleIdInfo();

List<BaseProduct> productList = baseProductService.getProductData(searchProduct);

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

matchProductData = productList.get(0);

return matchProductData;

}

//先根据款式或小分类搜索

searchProduct.setProductModelNumber(null);

searchProduct.setProductStyleIdInfo(null);

productList = baseProductService.getProductData(searchProduct);

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

matchProductData = productList.get(0);

return matchProductData;

}

//在根据款式搜索

searchProduct.setOrderSmallTypeValue(null);

productList = baseProductService.getProductData(searchProduct);

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

matchProductData = productList.get(0);

return matchProductData;

}

//再根据型号和小分类搜索

searchProduct.setProductModelNumber(productModelNumber);

searchProduct.setOrderSmallTypeValue(orderSmallTypeValue);

searchProduct.setStyleId(null);

productList = baseProductService.getProductData(searchProduct);

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

matchProductData = productList.get(0);

return matchProductData;

}

//再根据型号搜索

searchProduct.setProductModelNumber(productModelNumber);

searchProduct.setOrderSmallTypeValue(null);

productList = baseProductService.getProductData(searchProduct);

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

matchProductData = productList.get(0);

return matchProductData;

}

//再根据风格和小分类搜索

searchProduct.setProductStyleIdInfo(productStyleIdInfo);

searchProduct.setOrderSmallTypeValue(orderSmallTypeValue);

searchProduct.setProductModelNumber(null);

productList = baseProductService.getProductData(searchProduct);

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

matchProductData = productList.get(0);

return matchProductData;

}

//再根据风格查询

searchProduct.setOrderSmallTypeValue(null);

productList = baseProductService.getProductData(searchProduct);

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

matchProductData = productList.get(0);

return matchProductData;

}

//再根据小分类搜索

searchProduct.setOrderSmallTypeValue(orderSmallTypeValue);

searchProduct.setProductStyleIdInfo(null);

productList = baseProductService.getProductData(searchProduct);

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

matchProductData = productList.get(0);

return matchProductData;

} else {

return null;

}

}

@Override

public ResponseEnvelope<UnityDesignPlan> findGeneratePlanInfo(Integer designPlanId, Integer designTempletId,

DesignPlanRecommended designPlanRecommended, String context, String msgId , LoginUser loginUser, String mediaType) {

DesignTemplet designTemplet = designTempletService.get(designTempletId);

if (designTemplet == null) {

return new ResponseEnvelope<>(false, "找不到样板房！designTempletId = " + designTemplet, msgId);

}

DesignPlan designPlan = designPlanService.get(designPlanId);

//方案在匹配单品时创建的，删除状态为1，这里需更新为0

DesignPlan newDesignPlan = new DesignPlan();

newDesignPlan.setId(designPlanId);

newDesignPlan.setIsDeleted(Constants.IS\_DELETED\_STATE\_ZERO);

designPlanService.update(newDesignPlan);

if( designPlan == null ){

return new ResponseEnvelope<>(false, "创建设计方案失败！", msgId);

}

//配置内容存储路径

String configKey = Utils.getPropertyName("config/res", "design.designPlan.u3dconfig.upload.path", "/AA/d\_userdesign/[yyyy]/[MM]/[dd]/[HH]/onekeydesign/designPlan/u3dconfig/").trim();

configKey = Utils.replaceDate(configKey);

//文件名称

String fileName = System.currentTimeMillis()+"\_" + Utils.generateRandomDigitString(6)+".txt";

//先把文件保存到本地

String filePath = Utils.getAbsolutePath(configKey + fileName, null);

if( "linux".equals(FileUploadUtils.SYSTEM\_FORMAT) ){

filePath = filePath.replace("\\", "/");

}

/\*\*写入文件存储到指定目录\*\*/

boolean uploadFtpFlag = FileUploadUtils.writeTxtFile(filePath,context);

if( uploadFtpFlag ){

Map <String, String>map = FileUploadUtils.getMap(new File(filePath), "design.designPlan.u3dconfig.upload.path", false);

//生成资源数据

boolean flag = resDesignService.saveResDesign(designPlan,map);

if( !flag ){

return new ResponseEnvelope<>(false, "保存资源数据库失败！", msgId);

}

}else{

return new ResponseEnvelope<>(false, "保存设计方案配置文件失败！", msgId);

}

/\*\* 解析配置文件生成设计方案列表 \*\*/

JSONObject resultJSON = null;

try {

resultJSON = designPlanProductService.analysisJson(designTempletId,designPlanRecommended.getId(),designPlan,context, loginUser);

if( resultJSON != null && !resultJSON.getBoolean("success") ){

return new ResponseEnvelope<>(false, resultJSON.getBoolean("message"), msgId);

}

} catch (Exception e) {

logger.error(e.getMessage());

return new ResponseEnvelope<>(false, "保存设计方案产品列表异常！", msgId);

}

//获取设计方案信息

UnityDesignPlan unityDesignPlan = new UnityDesignPlan();

ResponseEnvelope<UnityDesignPlan> responseEnvelopeInfo = (ResponseEnvelope)designPlanService.getDesignPlanInfoForOneKey(designPlan.getId(),null,null,null,null,false,loginUser,mediaType, null);

if( responseEnvelopeInfo.isSuccess() ){

unityDesignPlan = (UnityDesignPlan)responseEnvelopeInfo.getObj();

}else{

logger.error(responseEnvelopeInfo.getMessage());

return new ResponseEnvelope<>(false, "进入设计方案失败！", msgId);

}

unityDesignPlan = designPlanService.wrapperData(designPlan.getId(), unityDesignPlan);

return new ResponseEnvelope<UnityDesignPlan>(unityDesignPlan, msgId, true);

}

/\*\*

\* 处理副本配置文件

\* @param resId

\* @param resType

\* @param fileKey

\* @param bussniess

\* @param request

\* @param code

\* @return

\*/

public Integer planCopyFileFromResDesignScene(String resId,String fileKey,String defaultPath, Integer bussniess,String code,LoginUser loginUser) {

String resFilePath = "";

Integer newResId = -1;

ResDesign resDesign = new ResDesign();

if (!StringUtils.isEmpty(resId)) {

String targetName = null;

ResDesign file = resDesignService.get(Integer.parseInt(resId));

if (file != null && !StringUtils.isEmpty(file.getFilePath())) {

resFilePath = file.getFilePath();

resDesign = file.resDesignCopy();

}

if (!StringUtils.isEmpty(resFilePath)) {

String srcPath = Utils.dealWithPath(Utils.getAbsolutePath(resFilePath, null), null);

File srcresourcesFile = new File(srcPath);

if (!srcresourcesFile.getParentFile().exists()) {

srcresourcesFile.getParentFile().mkdirs();

}

String resourcesName = resFilePath.substring(resFilePath.replace("/", "\\").lastIndexOf("\\") + 1,

resFilePath.length());

if ("linux".equals(FileUploadUtils.SYSTEM\_FORMAT)) {

resourcesName = resFilePath.substring(resFilePath.lastIndexOf("/") + 1, resFilePath.length());

}

String newPath = Utils.getPropertyName("config/res", fileKey + ".upload.path", defaultPath).trim();

newPath = Utils.replaceDate(newPath);

String tarName = resourcesName.substring(resourcesName.lastIndexOf("/") + 1,

resourcesName.lastIndexOf("\_")) + "\_" + Utils.getCurrentDateTime(Utils.DATETIMESSS)

+ resourcesName.substring(resourcesName.indexOf("."));

targetName = newPath + tarName;

targetName = targetName.replace(".txt", AESFileConstant.AES\_FIRST+".txt");//复制的配置文件因为源文件已经加密，要在文件后面加个标识

String local\_targetPath = Utils.dealWithPath(Utils.getAbsolutePath(targetName, null), null);

File local\_targetFile = new File(local\_targetPath);

if (!local\_targetFile.getParentFile().exists()) {

local\_targetFile.getParentFile().mkdirs();

}

boolean flag = false;

String resPath = resFilePath.substring(0, resFilePath.lastIndexOf("/") + 1);

String dbFilePath = Utils.getAbsolutePath(newPath + tarName, null);

if (Utils.getIntValue(FileUploadUtils.FTP\_UPLOAD\_METHOD) == 1) {

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

flag = FileUploadUtils.fileCopy(srcresourcesFile, local\_targetFile);

} else {

logger.error("srcresourcesFile is not exists !srcresourcesFile="+srcresourcesFile);

}

} else if (Utils.getIntValue(FileUploadUtils.FTP\_UPLOAD\_METHOD) == 2) {

flag = FtpUploadUtils.downFile(resPath, resourcesName);// 下载到本地

if (FileUploadUtils.fileCopy(srcresourcesFile, local\_targetFile)) {

if (flag) {

flag = FtpUploadUtils.uploadFile(tarName, dbFilePath, newPath);// 上传ftp服务器

if (flag) {

FileUploadUtils.deleteFile(newPath + tarName); // 删除本地

} else {

return newResId;

}

} else {

return newResId;

}

} else {

logger.error("copy file is error");

return -1;

}

} else {

// 3 本地和ftp同时上传(默认是本地上传)resPath：FTP服务器上的相对路径，resourcesName：要下载的文件名，newPath+tarName：下载到本地文件路径+文件名称

flag = FtpUploadUtils.downFile(resPath, resourcesName);// 下载到本地

if (!flag || FileUploadUtils.fileCopy(srcresourcesFile, local\_targetFile)) {

logger.error("copy file is error");

return -1;

}

if (flag) {// tarName:文件名称，dbFilePath:本地文件路径，newPath:ftp服务器存放文件路径

flag = FtpUploadUtils.uploadFile(tarName, dbFilePath, newPath);// 上传ftp服务器

if (!flag) {

return newResId;

}

} else {

return newResId;

}

}

}

resDesign.setSysCode(code);

resDesign.setFilePath(targetName);

resDesign.setFileKey(fileKey);

resDesign.setBusinessId(bussniess);

resDesign.setFileCode(code);

sysSave(resDesign, loginUser);

newResId = resDesignService.add(resDesign);

}

return newResId;

}

/\*\*

\* 自动存储系统字段

\*/

private void sysSave(ResDesign model, LoginUser loginUser) {

if (model != null) {

if (model.getId() == null) {

model.setGmtCreate(new Date());

model.setCreator(loginUser.getLoginName());

model.setIsDeleted(0);

if (model.getSysCode() == null || "".equals(model.getSysCode())) {

model.setSysCode(

Utils.getCurrentDateTime(Utils.DATETIMESSS) + "\_" + Utils.generateRandomDigitString(6));

}

}

model.setGmtModified(new Date());

model.setModifier(loginUser.getLoginName());

}

}

@Override

public ResponseEnvelope<DesignTempletProductResult> findDesignTempletProductList(Integer designTempletId,

Integer groupType, String msgId) {

DesignTempletProduct designTempletproduct = new DesignTempletProduct();

designTempletproduct.setDesignTempletId(designTempletId);

designTempletproduct.setGroupType(groupType);

List<DesignTempletProductResult> list = designTempletProductService.getTempletIdProductList(designTempletproduct);

int total = Utils.getListTotal(list);

return new ResponseEnvelope<DesignTempletProductResult>(total,list, msgId);

}

@Override

public ResponseEnvelope<SearchStructureProductResult> matchingStructureData(String structureCodes,

Integer designTempletId, Integer designPlanId, String msgId, LoginUser loginUser) {

DesignTemplet designTemplet = null;

if(Utils.enableRedisCache()){

designTemplet = DesignCacher.getTemplet(designTempletId);

}else{

designTemplet = designTempletService.get(designTempletId);

}

if (designTemplet == null) {

return new ResponseEnvelope<SearchStructureProductResult>(false, "样板房不存在", msgId);

}

List<SearchStructureProductResult> list = new ArrayList<SearchStructureProductResult>();

List<StructureProduct> structureList = structureProductService.getStructuresByDesignPlanId(designPlanId);

Map<String,StructureProduct> structure\_map = new HashMap<>();

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

for (StructureProduct structureProduct : structureList){

structure\_map.put(structureProduct.getStructureCode(),structureProduct);

}

}

List<DesignTempletProduct> templetStructureProducts = designTempletProductService.getStructureProductByDesignId(designTempletId);

for (DesignTempletProduct templetProduct : templetStructureProducts) {

SearchStructureProductResult searchStructureProductResult = null;

int djTemp = 0;

StructureProduct djStructureProduct = null;

for (StructureProduct planStructureProduct : structureList) {

//匹配方案结构区域、尺寸代码

if ("DJ".equals(templetProduct.getStructureType()) && "DJ".equals(planStructureProduct.getStructureType())) {

//如果是主地面则匹配方案主地面，没有当单品处理

if ("1".equals(templetProduct.getRegionMark())) {

if ("1".equals(planStructureProduct.getPlanStructureRegionMark())) {

djTemp++;

if (templetProduct.getMeasureCode().equals(planStructureProduct.getMeasureCode())) {

//结构详情列表json

searchStructureProductResult = structureProductService.getSearchStructureProductResult(planStructureProduct, templetProduct.getStructureCode());

break;

} else {

StructureProduct structure = matcheSystemStructure(templetProduct, planStructureProduct, loginUser);

if (structure != null) {

//结构详情列表json

searchStructureProductResult = structureProductService.getSearchStructureProductResult(structure, templetProduct.getStructureCode());

break;

}

}

}

}else {

if (templetProduct.getRegionMark().equals(planStructureProduct.getPlanStructureRegionMark())) {

djTemp++;

if (templetProduct.getMeasureCode().equals(planStructureProduct.getMeasureCode())) {

//结构详情列表json

searchStructureProductResult = structureProductService.getSearchStructureProductResult(planStructureProduct,templetProduct.getStructureCode());

break;

} else {

StructureProduct structure = matcheSystemStructure(templetProduct, planStructureProduct, loginUser);

if (structure != null) {

//结构详情列表json

searchStructureProductResult = structureProductService.getSearchStructureProductResult(structure,templetProduct.getStructureCode());

break;

}

}

} else {

djStructureProduct = planStructureProduct;//记录该地面结构需要用它的款式去匹配（structureList有按区域正序排序，取最后一条去匹配）

continue;

}

}

} else {

StructureProduct structure = matcheSystemStructure(templetProduct, planStructureProduct, loginUser);

if (structure != null) {

//结构详情列表json

searchStructureProductResult = structureProductService.getSearchStructureProductResult(structure,templetProduct.getStructureCode());

break;

}

}

}

if (!"1".equals(templetProduct.getRegionMark()) && djTemp == 0) {

StructureProduct structure = matcheSystemStructure(templetProduct, djStructureProduct, loginUser);

if (structure != null) {

//结构详情列表json

searchStructureProductResult = structureProductService.getSearchStructureProductResult(structure,templetProduct.getStructureCode());

}

}

if (searchStructureProductResult != null) {

list.add(searchStructureProductResult);

}

}

int total = Utils.getListTotal(list) ;

return new ResponseEnvelope<SearchStructureProductResult>(total,list, msgId);

}

/\*\*

\* 根据方案结构类型序号和样板房款式和尺寸代码匹配结构

\* @author xiaoxc

\* @param designTempletProduct

\* @param planStructure

\* @param loginUser

\* @return

\*/

private StructureProduct matcheSystemStructure(DesignTempletProduct designTempletProduct,StructureProduct planStructure,LoginUser loginUser){

// 参数验证 ->start

if(planStructure == null){

return null;

}

// 参数验证 ->start

//匹配结构条件搜索

StructureProductSearch templetStructureSearch = new StructureProductSearch();

templetStructureSearch.setStructureType(planStructure.getStructureType());//方案结构类型

templetStructureSearch.setStructureNumber(planStructure.getStructureNumber());//方案结构序号

templetStructureSearch.setStyleId(planStructure.getStyleId());//结构款式

//通用结构按尺寸代码匹配，定制按样板房ID匹配

if (designTempletProduct.getIsStandard() == 1) {

templetStructureSearch.setMeasureCode(designTempletProduct.getMeasureCode());//尺寸代码

}else{

templetStructureSearch.setTempletId(designTempletProduct.getDesignTempletId());

}

List<Integer> statusList = new ArrayList<>();

if (loginUser.getUserType().intValue() == UserTypeCode.USER\_TYPE\_INNER.intValue()) {

statusList.add(StructureProductStatusCode.HAS\_BEEN\_PUTAWAY);

statusList.add(StructureProductStatusCode.TESTING);

statusList.add(StructureProductStatusCode.HAS\_BEEN\_RELEASE);

} else {

statusList.add(StructureProductStatusCode.HAS\_BEEN\_RELEASE);

}

templetStructureSearch.setStatusList(statusList);

templetStructureSearch.setIsDeleted(0);

templetStructureSearch.setStart(0);

templetStructureSearch.setLimit(1);

List<StructureProduct> templetStructureList = structureProductService.getStructureObject(templetStructureSearch);

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

StructureProduct stp = templetStructureList.get(0);

stp.setPlanStructureRegionMark(planStructure.getPlanStructureRegionMark());

return stp;

}else{

return null;

}

}

private void getGroupMatcheDataV2(

DesignPlanRecommendedProduct designPlanRecommendedProduct,

MatchGroupProductResult matchGroupProductResult,

Integer planRecommendedId

){

matchGroupProductResult.setTempletMianProductCode(designPlanRecommendedProduct.getProductCode());

matchGroupProductResult.setTempletPosName(designPlanRecommendedProduct.getPosName());

matchGroupProductResult.setPlanGroupId(designPlanRecommendedProduct.getPlanGroupId());

// 目前数据库 location字段， 分为 txt 文件路径、json 串。

String filePath = designPlanRecommendedProduct.getFilePath();

logger.error("getGroupMatcheDataV2().filePath = " + filePath);

if ( filePath != null && !filePath.trim().isEmpty() ) {

String url = Utils.getAbsolutePath(filePath, Utils.getAbsolutePathType.encrypt);

logger.error("getGroupMatcheDataV2().url = " + url);

String text = FileUploadUtils.getFileContext(url);

logger.error("getGroupMatcheDataV2().text = " + text);

matchGroupProductResult.setTempletGroupConfig(text);

}else{

matchGroupProductResult.setTempletGroupConfig(designPlanRecommendedProduct.getLocation());

logger.error("getGroupMatcheDataV2().Location = " + designPlanRecommendedProduct.getLocation());

}

//推荐方案组合产品 （方案相同产品客户端无法匹配）

List<DesignPlanRecommendedProduct> designPlanRecommendedProductList =

designPlanRecommendedProductServiceV2.getListByPlanIdAndGroupId(planRecommendedId, designPlanRecommendedProduct.getProductGroupId());

Map<String,String> map = null;

List<Map<String,String>> groupProductList = new ArrayList<>();

if (Lists.isNotEmpty(designPlanRecommendedProductList)) {

for (DesignPlanRecommendedProduct designPlanRecommendedProductListItem : designPlanRecommendedProductList) {

map = new HashMap<>();

map.put("productCode",designPlanRecommendedProductListItem.getProductCode());

map.put("posIndexPath",designPlanRecommendedProductListItem.getPosIndexPath());

map.put("planPosName",designPlanRecommendedProductListItem.getPosName());

map.put("groupProductUniqueId",designPlanRecommendedProductListItem.getGroupProductUniqueId());

groupProductList.add(map);

}

}

matchGroupProductResult.setTempletGroupProductList(groupProductList);

}

@Override

public ResponseEnvelope<Object> getTempletProductIndexConfig(String msgId,String templetCode ){

List<TempletProductIndexInfo> list = templetProductIndexInfoService.findIndexInfoByCode(templetCode);

String productIndexInfo = "";

if( Lists.isNotEmpty(list) ){

productIndexInfo = list.get(0).getProductIndexInfo();

}

return new ResponseEnvelope(true,productIndexInfo,msgId);

}

@Override

public ResponseEnvelope byConfigSaveProductIndex(String msgId, String templetCode, String context,

LoginUser loginUser){

DesignTemplet designTemplet = designTempletService.getDesignTempletByCode(templetCode);

if( designTemplet == null || designTemplet.getId() < 1){

return new ResponseEnvelope(false,"找不到样板房信息！templetCode="+templetCode,msgId);

}

Integer designTempletId = designTemplet.getId();

//存储context，如果之前样板房有存储先逻辑删除

TempletProductIndexInfo templetProductIndexInfo = new TempletProductIndexInfo();

templetProductIndexInfo.setDesignTempletId(designTempletId);

templetProductIndexInfo.setIsDeleted(0);

List<TempletProductIndexInfo> list = templetProductIndexInfoService.getList(templetProductIndexInfo);

List<Integer> indexInfoIds = new ArrayList<>();

for( TempletProductIndexInfo templetProductIndexInfo1 : list ){

indexInfoIds.add(templetProductIndexInfo1.getId());

}

//批量更新

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

TempletProductIndexInfo productIndexInfo = new TempletProductIndexInfo();

productIndexInfo.setIsDeleted(1);

productIndexInfo.setIndexInfoIds(indexInfoIds);

templetProductIndexInfoService.updateIndexInfoByIds(productIndexInfo);

}

templetProductIndexInfo.setProductIndexInfo(context);

templetProductIndexInfoService.sysSave(templetProductIndexInfo,loginUser);

templetProductIndexInfoService.add(templetProductIndexInfo);

//存储样板房产品序号

Gson gson2 = new Gson();

List<ProductIndexConfig> productIndexList = gson2.fromJson(context, new TypeToken<List<ProductIndexConfig>>() {}.getType());

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

logger.error("样板房code = "+templetCode+" 无更新的样板房产品属性。");

}

StringBuffer sb = new StringBuffer();

for (ProductIndexConfig productIndexConfig : productIndexList) {

List<ProductDataEx> productList = productIndexConfig.getProductDataEx();

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

logger.error("分类productSmallType = "+productIndexConfig.getProductSmallType()+" 无更新的产品属性。");

}

for (ProductDataEx product : productList ) {

DesignTempletProduct designTempletProduct = new DesignTempletProduct();

designTempletProduct.setPosName(product.getPosName());

designTempletProduct.setDesignTempletId(designTempletId);

designTempletProduct.setSameProductTypeIndex(product.getIndex());

designTempletProduct.setIsStandard(product.getIsStandard());

designTempletProduct.setRegionMark(product.getRegionMark());

designTempletProduct.setStyleId(product.getStyleId());

designTempletProduct.setMeasureCode(product.getMeasureCode());

designTempletProduct.setProductIndex(product.getProductIndex());

//如果是背景墙结构序号为10的默认为主产品

if (product.getProductIndex() != null && product.getProductIndex() == 10) {

designTempletProduct.setIsMainStructureProduct(1);

designTempletProduct.setIsGroupReplaceWay(1);

}else{

designTempletProduct.setIsMainStructureProduct(product.getIsMainStructureProduct());

designTempletProduct.setIsGroupReplaceWay(product.getIsGroupReplaceWay());

}

designTempletProduct.setWallOrientation(product.getWallOrientation());

designTempletProduct.setWallType(product.getWallType());

int i = designTempletProductService.updateSameProductTypeIndex(designTempletProduct);

if (i == 0) {

logger.error("分类："+productIndexConfig.getProductSmallType()+";posName="+product.getPosName()+";designTempletId="+designTempletId+"更新失败");

sb.append("productCode="+product.getItemCode()+",");

}

}

}

if (sb != null && sb.length() > 0) {

return new ResponseEnvelope(true,sb.toString()+"保存失败！",msgId);

}

return new ResponseEnvelope(true,"保存成功！",msgId);

}

@Override

public List<SearchStructureProductResult> matchingStructureDataV2(

DesignTemplet designTemplet,

DesignPlanRecommended designPlanRecommended,

LoginUser loginUser,

List<String> regionMarkList

) {

List<SearchStructureProductResult> list = new ArrayList<SearchStructureProductResult>();

// 参数验证 ->start

if(designTemplet == null){

return list;

}

if(designPlanRecommended == null){

return list;

}

if(loginUser == null){

return list;

}

// 参数验证 ->end

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

return list;

}

List<StructureProduct> structureList = structureProductService.getStructuresByRecommendedPlanId(designPlanRecommended.getId());

Map<String,StructureProduct> structure\_map = new HashMap<>();

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

for (StructureProduct structureProduct : structureList){

structure\_map.put(structureProduct.getStructureCode(),structureProduct);

}

}

List<DesignTempletProduct> templetStructureProducts = designTempletProductService.getStructureProductByDesignId(designTemplet.getId());

for (DesignTempletProduct templetProduct : templetStructureProducts) {

SearchStructureProductResult searchStructureProductResult = null;

int djTemp = 0;

StructureProduct djStructureProduct = null;

for (StructureProduct planStructureProduct : structureList) {

//匹配方案结构区域、尺寸代码

if ("DJ".equals(templetProduct.getStructureType()) && "DJ".equals(planStructureProduct.getStructureType())) {

if (templetProduct.getRegionMark().equals(planStructureProduct.getPlanStructureRegionMark())) {

djTemp++;

if (templetProduct.getMeasureCode().equals(planStructureProduct.getMeasureCode())) {

//结构详情列表json

searchStructureProductResult = structureProductService.getSearchStructureProductResult(planStructureProduct,templetProduct.getStructureCode());

break;

} else {

StructureProduct structure = null;

if (planStructureProduct.getStructureNumber().intValue() == 1) { //如果是整块结构则查本身结构数据

structure = structureProductService.get(templetProduct.getProductGroupId());

}else{

structure = matcheSystemStructure(templetProduct, planStructureProduct, loginUser);

}

if (structure != null) {

//结构详情列表json

searchStructureProductResult = structureProductService.getSearchStructureProductResult(structure,templetProduct.getStructureCode());

break;

}

}

} else {

djStructureProduct = planStructureProduct;//记录该地面结构需要用它的款式去匹配（structureList有按区域正序排序，取最后一条去匹配）

continue;

}

} else {

StructureProduct structure = matcheSystemStructure(templetProduct, planStructureProduct, loginUser);

if (structure != null) {

//结构详情列表json

searchStructureProductResult = structureProductService.getSearchStructureProductResult(structure,templetProduct.getStructureCode());

break;

}

}

}

if (!"1".equals(templetProduct.getRegionMark()) && djTemp == 0) {

StructureProduct structure = matcheSystemStructure(templetProduct, djStructureProduct, loginUser);

if (structure != null) {

//结构详情列表json

searchStructureProductResult = structureProductService.getSearchStructureProductResult(structure,templetProduct.getStructureCode());

}

}

if (searchStructureProductResult != null) {

list.add(searchStructureProductResult);

}

}

// 检测该设计方案结构有没有换成别的结构(是不是初始化结构,如果是初始化结构,则continue) ->start add by huangsongbo

// 该逻辑适用于U3D前端

if(list != null){

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

if(regionMarkList.indexOf(list.get(index).getPlanStructureRegionMark()) == -1){

list.remove(index);

index --;

}

}

}

// 检测该设计方案结构有没有换成别的结构(是不是初始化结构,如果是初始化结构,则continue) ->end add by huangsongbo

return list;

}

@Override

public MatchItenCodeVO findByItemCodeInfoV2(

String posNames, DesignTemplet designTemplet,

DesignPlanRecommended designPlanRecommended,

LoginUser loginUser,Integer opType,

// add by huangsongbo 2018.1.9, 前端要这个参数,决定是从空间入口装修(选择空间类型+面积)的方案还是从户型入口(选择小区->户型)装修的方案

String planSource

) throws IntelligenceDecorationException {

MatchItenCodeVO matchItenCodeVO = new MatchItenCodeVO();

List<PosNameInfoVO> posNameInfoVOList = new ArrayList<PosNameInfoVO>();

// 参数验证 ->start

if(StringUtils.isEmpty(posNames)){

return matchItenCodeVO;

}

if(designTemplet == null){

return matchItenCodeVO;

}

if(designPlanRecommended == null){

return matchItenCodeVO;

}

if (opType == null) {

opType = DesignPlanConstants.USER\_RENDER;

}

// 参数验证 ->end

//创建一个逻辑删除的新方案,等更新配置文件在更新删除状态为0 ---start

DesignPlan designPlan = new DesignPlan();

designPlan.setDesignTemplateId(designTemplet.getId());

designPlan.setRecommendedPlanId(designPlanRecommended.getId());

designPlan.setPlanSource(planSource);

/\*\* 通过样板房创建设计方案信息 \*\*/

if (opType.intValue() == DesignPlanConstants.AUTO\_RENDER.intValue()) {

optimizePlanService.createDesignPlan(designPlan,designPlanRecommended,designTemplet,loginUser.getMediaType(),loginUser, Constants.IS\_DELETED\_STATE\_ONE);

} else {

designPlanService.createDesignPlanForOneKey(designPlan,designPlanRecommended,designTemplet,loginUser.getMediaType(),loginUser, Constants.IS\_DELETED\_STATE\_ONE, null);

}

matchItenCodeVO.setDesignPlanId(designPlan.getId());

//创建一个逻辑删除的新方案------end

String arrayPosName[] = posNames.split(",");

PosNameInfoVO posNameInfoVO = null;

//未匹配到的样板房产品

List<DesignProductResult> unmatchProduct = new ArrayList<>();

Map<DesignProductResult, DesignProductResult> againMatchMap = new HashMap<DesignProductResult, DesignProductResult>();

List<String> posNameList = Arrays.asList(arrayPosName);//白膜的产品Code列表

//白膜的产品列表

List<DesignProductResult> templetProductList = baseProductMapper.getTempletProductList(posNameList, designTemplet.getId());

//设计方案产品列表

List<DesignProductResult> planProductList = baseProductMapper.getPlanProductRecommendedList(designPlanRecommended.getId());

if (Lists.isEmpty(planProductList)) {

return matchItenCodeVO;

}

HashMap<String, DesignProductResult> code\_pro\_map = new HashMap <String, DesignProductResult>();

for (DesignProductResult pro : templetProductList) {

code\_pro\_map.put(pro.getPosName(), pro);

}

int mapSize = 0;

List<DesignProductResult> list = null;

// 用来储存样板房背景墙数据,放在后面匹配(普通背景墙+结构背景墙)

List<DesignProductResult> designProductResultListBeijing = new ArrayList<DesignProductResult>();

for (String posName : arrayPosName) {

StringBuffer matchErrorInfo = new StringBuffer();

posNameInfoVO = new PosNameInfoVO();

DesignProductResult templetProduct = code\_pro\_map.get(posName);

if (templetProduct == null) {

continue;

}

// 背景墙单独匹配,先放到designProductResultListBeijing中,匹配完其他产品再匹配背景墙

if(beijingValuekeyList.indexOf(templetProduct.getSmallTypeKey()) != -1) {

designProductResultListBeijing.add(templetProduct);

continue;

}

matchErrorInfo = getLogInfo(templetProduct,null,matchErrorInfo);//获取错误数据日志

templetProduct.setMatchErrorInfo(matchErrorInfo==null?"":matchErrorInfo.toString());

//包含有组合产品的不予匹配

if (templetProduct.getProductGroupId() != null && templetProduct.getProductGroupId() > 0 && 0 == templetProduct.getGroupType().intValue()) {

posNameInfoVOList.add(setPosNameInfoVO(templetProduct,new DesignProductResult()));

continue;

}

list = new ArrayList<>();

int temp = 0;

DesignProductResult noRegionTianh = null;

DesignProductResult noRegionDim = null;

DesignProductResult sameTypeDim = null;

for (DesignProductResult planProduct : planProductList) {

// 背景墙放在最后匹配 ->start

if(beijingValuekeyList.indexOf(planProduct.getSmallTypeKey()) != -1) {

continue;

}

// 背景墙放在最后匹配 ->end

if (ProductTypeConstant.PRODUCT\_BIG\_TYPE\_TIANH.equals(templetProduct.getBigTypeKey())

&& ProductTypeConstant.PRODUCT\_BIG\_TYPE\_TIANH.equals(planProduct.getBigTypeKey())) {

matchErrorInfo = getLogInfo(null,planProduct,matchErrorInfo);//获取错误数据日志

templetProduct.setMatchErrorInfo(matchErrorInfo==null?"":matchErrorInfo.toString());

//天花通过区域标识匹配/尺寸代码、，如果匹配不上通过方案天花款式、白膜尺寸代码、区域标识匹配系统数据

if (templetProduct.getRegionMark().equals(planProduct.getRegionMark())) {

temp ++ ;

if (templetProduct.getMeasureCode().equals(planProduct.getMeasureCode())) {

posNameInfoVO = setPosNameInfoVO(templetProduct,planProduct);

break;

}else{

againMatchMap.put(templetProduct,planProduct);

break;

}

}else{

noRegionTianh = planProduct;//记录该天花需要用它的款式去匹配（planProductList我有按区域正序排序，去最后一条去匹配）

continue;

}

}else if (ProductTypeConstant.PRODUCT\_BIG\_TYPE\_DIM.equals(templetProduct.getBigTypeKey())

&& ProductTypeConstant.PRODUCT\_BIG\_TYPE\_DIM.equals(planProduct.getBigTypeKey())){

if (StringUtils.isNotEmpty(templetProduct.getRegionMark()) && templetProduct.getRegionMark().equals(planProduct.getRegionMark())) {

temp ++ ;

posNameInfoVO = setPosNameInfoVO(templetProduct,planProduct);

break;

}else{

//同分类地面取同小分类地面产品（阳台、门槛石没录区域）

if (templetProduct.getSmallTypeValue().equals(planProduct.getBmSmallTypeValue())) {

sameTypeDim = planProduct;

}else{

//跳过门槛石匹配

if (!"17".equals(planProduct.getBmSmallTypeValue())){

noRegionDim = planProduct;

}

}

}

}else if (ProductTypeConstant.PRODUCT\_BIG\_TYPE\_QIANGM.equals(templetProduct.getBigTypeKey())

&& ProductTypeConstant.PRODUCT\_BIG\_TYPE\_QIANGM.equals(planProduct.getBigTypeKey())) {

matchErrorInfo = getLogInfo(null,planProduct,matchErrorInfo);//获取错误数据日志

templetProduct.setMatchErrorInfo(matchErrorInfo==null?"":matchErrorInfo.toString());

//实体墙匹配

if (templetProduct.getSmallTypeValue() != null && templetProduct.getSmallTypeValue() == 8) {

if (planProduct.getBmSmallTypeValue() == 8) {

temp ++ ;

posNameInfoVO = setPosNameInfoVO(templetProduct, planProduct);

break;

}else{

continue;

}

}

//门框匹配

if (templetProduct.getSmallTypeValue() != null && templetProduct.getSmallTypeValue() == 19) {

if (planProduct.getBmSmallTypeValue() == 19) {

if (isMatchingPlanProduct(templetProduct, planProduct, matchErrorInfo) && planProduct.getMatchSign()==null) {

temp ++ ;

posNameInfoVO = setPosNameInfoVO(templetProduct, planProduct);

planProduct.setMatchSign(1);//已匹配标记

break;

} else {

list.add(planProduct); //匹配不到的数据存起来，匹配系统产品用

continue;

}

}else{

continue;

}

}

//窗框匹配

if (templetProduct.getSmallTypeValue() != null && templetProduct.getSmallTypeValue() == 18) {

if (planProduct.getBmSmallTypeValue() == 18) {

temp ++ ;

if (isMatchingPlanProduct(templetProduct, planProduct, matchErrorInfo)) {

posNameInfoVO = setPosNameInfoVO(templetProduct, planProduct);

break;

} else {

againMatchMap.put(templetProduct,planProduct);

break;

}

}else{

continue;

}

}

}else if (ProductTypeConstant.PRODUCT\_BIG\_TYPE\_MENG.equals(templetProduct.getBigTypeKey())

&& ProductTypeConstant.PRODUCT\_BIG\_TYPE\_MENG.equals(planProduct.getBigTypeKey())) {

//门按白膜小分类匹配，相同房间门用一个去替换（强子定的）

if (templetProduct.getBigTypeValue().equals(planProduct.getBigTypeValue())

&& templetProduct.getSmallTypeValue() == planProduct.getBmSmallTypeValue()) {

temp ++;

if (isMatchingPlanProduct(templetProduct, planProduct, matchErrorInfo)) {

posNameInfoVO = setPosNameInfoVO(templetProduct, planProduct);

break;

}else{

againMatchMap.put(templetProduct,planProduct);

break;

}

}else {

continue;

}

}else if (ProductTypeConstant.PRODUCT\_BIG\_TYPE\_LI.equals(templetProduct.getBigTypeKey())

&& ProductTypeConstant.PRODUCT\_BIG\_TYPE\_LI.equals(planProduct.getBigTypeKey())) {

matchErrorInfo = getLogInfo(null,planProduct,matchErrorInfo);//获取错误数据日志

templetProduct.setMatchErrorInfo(matchErrorInfo==null?"":matchErrorInfo.toString());

//灯具按白膜小分类、墙体分类匹配,

if (templetProduct.getSmallTypeValue() == planProduct.getBmSmallTypeValue()) {

if (templetProduct.getWallType().trim().equals(planProduct.getWallType().trim())) {

posNameInfoVO = setPosNameInfoVO(templetProduct, planProduct);

break;

}else{

continue;

}

}

}else if (ProductTypeConstant.PRODUCT\_BIG\_TYPE\_PE.equals(templetProduct.getBigTypeKey())

&& ProductTypeConstant.PRODUCT\_BIG\_TYPE\_PE.equals(planProduct.getBigTypeKey())) {

// 并且要满足互搜逻辑

@SuppressWarnings("unchecked")

Map<String, List<String>> showMoreSmallTypeInfoMap = (Map<String, List<String>>) ProductCategoryRelServiceImpl.showMoreSmallTypeMap.get("showMoreSmallTypeInfoMap");

if(showMoreSmallTypeInfoMap.containsKey(templetProduct.getSmallTypeKey().replace("basic\_", ""))){

List<String> smallTypeList = showMoreSmallTypeInfoMap.get(templetProduct.getSmallTypeKey().replace("basic\_", ""));

if(smallTypeList.indexOf(planProduct.getSmallTypeKey().replace("basic\_", "")) != -1){

//挂画按墙体方位匹配或墙体分类

if (!"0".equals(templetProduct.getWallOrientation())

&& templetProduct.getWallOrientation().equals(planProduct.getWallOrientation())) {

temp ++;

posNameInfoVO = setPosNameInfoVO(templetProduct, planProduct);

break;

}else if (templetProduct.getWallType().equals(planProduct.getWallType())) {

list.add(planProduct);

continue;

}else{

continue;

}

}

}

}else{

if (templetProduct.getBigTypeValue().equals(planProduct.getBigTypeValue()) && templetProduct.getSmallTypeValue() == planProduct.getBmSmallTypeValue()) {

if (isMatchingPlanProduct(templetProduct, planProduct, matchErrorInfo)) {

posNameInfoVO = setPosNameInfoVO(templetProduct, planProduct);

break;

}else{

againMatchMap.put(templetProduct,planProduct);

break;

}

}else {

continue;

}

}

}

if (ProductTypeConstant.PRODUCT\_BIG\_TYPE\_TIANH.equals(templetProduct.getBigTypeKey()) && temp == 0 && noRegionTianh != null) {

againMatchMap.put(templetProduct,noRegionTianh);

}

if (ProductTypeConstant.PRODUCT\_BIG\_TYPE\_TIANH.equals(templetProduct.getBigTypeKey()) && temp == 0 && noRegionTianh == null) {

posNameInfoVO = setPosNameInfoVO(templetProduct,unmatchProduct);

unmatchProduct.add(templetProduct);

}

if (ProductTypeConstant.PRODUCT\_BIG\_TYPE\_DIM.equals(templetProduct.getBigTypeKey()) && temp == 0 && sameTypeDim != null) {

posNameInfoVO = setPosNameInfoVO(templetProduct,sameTypeDim);

}else if (ProductTypeConstant.PRODUCT\_BIG\_TYPE\_DIM.equals(templetProduct.getBigTypeKey()) && temp == 0 && noRegionDim != null) {

posNameInfoVO = setPosNameInfoVO(templetProduct,noRegionDim);

}

//list

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

for (DesignProductResult matchResult : list) {

if (ProductTypeConstant.PRODUCT\_BIG\_TYPE\_PE.equals(matchResult.getBigTypeKey())) {

posNameInfoVO = setPosNameInfoVO(templetProduct, matchResult);

break;

} else {

againMatchMap.put(templetProduct,matchResult);

break;

}

}

}

//匹配不到方案里的同小分类门，则匹配系统产品

if (ProductTypeConstant.PRODUCT\_BIG\_TYPE\_MENG.equals(templetProduct.getBigTypeKey()) && temp == 0) {

againMatchMap.put(templetProduct,new DesignProductResult());

}

if (againMatchMap != null && againMatchMap.size() == mapSize) {

if (posNameInfoVO == null || posNameInfoVO.getBmPosName() == null) {

posNameInfoVO = setPosNameInfoVO(templetProduct,unmatchProduct);

}

posNameInfoVOList.add(posNameInfoVO);

}else{

mapSize = againMatchMap.size();

}

}

//方案匹配不到，则匹配系统产品

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

for (Map.Entry<DesignProductResult, DesignProductResult> entry : againMatchMap.entrySet()) {

DesignProductResult templetProductResult = entry.getKey();

logger.debug("posname:" + templetProductResult.getPosName());

DesignProductResult planProductResult = entry.getValue();

BaseProduct matchProductData = getMatchSystemProduct(templetProductResult,planProductResult,loginUser);

if (matchProductData != null) {

planProductResult.setProductCode(matchProductData.getProductCode());

posNameInfoVO = setPosNameInfoVO(templetProductResult,planProductResult);

}else{

posNameInfoVO = setPosNameInfoVO(templetProductResult,unmatchProduct);

}

posNameInfoVOList.add(posNameInfoVO);

}

}

// 背景墙的删除/匹配 ->start

// 取出推荐方案中所有的背景墙,并且剔除非主墙 ->start

List<DesignProductResult> designProductRecommendedResultList = this.getDesignProductRecommendedResultList(planProductList);

// 取出推荐方案中所有的背景墙,并且剔除非主墙 ->end

// 整理出一份墙体方位:推荐方案产品信息的Map:墙体方位:List<墙体类型>

Map<String, List<String>> recommendedProductMapByWallOrientation = this.getRecommendedProductMapByWallOrientation(designProductRecommendedResultList);

// 取出样板房中附墙+非主墙/主墙的list

Map<getDesignProductResultListBeijingMapEnum, List<DesignProductResult>> designProductResultListBeijingMap = this.getDesignProductResultListBeijingMap(designProductResultListBeijing);

List<DesignProductResult> designTempletMainBeijingInfoList = designProductResultListBeijingMap.get(getDesignProductResultListBeijingMapEnum.mainBeijing);

List<DesignProductResult> designTempletDeputyBeijingInfoList = designProductResultListBeijingMap.get(getDesignProductResultListBeijingMapEnum.deputyBeijing);

// 整理推荐方案中的背景墙:整理成wallType:背景墙数据的格式 ->start

Map<String, List<DesignProductResult>> designProductRecommendedResultListByWallType = this.getDesignProductResultListByWallType(designProductRecommendedResultList);

// 整理推荐方案中的背景墙:整理成wallType:背景墙数据的格式 ->end

// 整理样板房中的背景墙:整理成wallType:背景墙数据的格式 ->start

Map<String, List<DesignProductResult>> designProductTempletResultListByWallType = this.getDesignProductResultListBeijingByWallType(designTempletMainBeijingInfoList);

// 整理样板房中的背景墙:整理成wallType:背景墙数据的格式 ->end

// 删除背景墙逻辑 ->start

// 获取被删除的背景墙的数据(主墙) ->start

List<Map<deletedBeijingProductInfoMapKeyEnum, String>> deletedBeijingProductInfoList = this.getDeletedBeijingProductInfo(designPlanRecommended.getId());

// 获取被删除的背景墙的数据(主墙) ->end

// 获取样板房中不同类别背景墙要删除的个数 ->start

// wallType:数量,eg:A:2,标识墙体分类为A的墙还要删除2面

// {E=1, H=2}

Map<String, Integer> deleteNumInfoMap = this.getDeleteNumInfoMap(designProductTempletResultListByWallType, designProductRecommendedResultListByWallType, deletedBeijingProductInfoList);

// 获取样板房中不同类别背景墙要删除的个数 ->end

// 删除背景墙 ->start

this.deleteTempletBeijingProduct(posNameInfoVOList, deleteNumInfoMap, designTempletMainBeijingInfoList, recommendedProductMapByWallOrientation,unmatchProduct);

// 删除背景墙 ->end

// 删除背景墙逻辑 ->end

// 背景墙匹配(主墙) ->start

// 得到一个墙体方位:产品信息的map,为了附墙根据主墙去搜索

Map<String, DesignProductResult> matchResponseInfoMap = this.matchMainBeijingProduct(posNameInfoVOList, designTempletMainBeijingInfoList, designProductRecommendedResultList, loginUser, unmatchProduct);

// 背景墙匹配(主墙) ->end

// 背景墙附墙/结构背景墙非主墙匹配 ->start

// 背景墙附墙/结构背景墙非主墙匹配 ->end

this.matchDeputyBeijingProduct(posNameInfoVOList, designTempletDeputyBeijingInfoList, matchResponseInfoMap, loginUser, unmatchProduct);

// 背景墙的删除/匹配 ->end

//批量新增方案产品（未匹配到样板房产品）

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

designPlanService.batchAddPlanProduct(designPlan.getId(),unmatchProduct,loginUser,Constants.IS\_DELETED\_STATE\_ONE,opType);

}

matchItenCodeVO.setPosNameInfoVOList(posNameInfoVOList);

return matchItenCodeVO;

}

/\*\*

\* 背景墙附墙/结构背景墙非主墙匹配(根据主墙搜索)

\*

\* @author huangsongbo

\* @param dataList

\* @param designTempletDeputyBeijingInfoList

\* @param matchResponseInfoMap

\* @throws IntelligenceDecorationException

\*/

private void matchDeputyBeijingProduct(List<PosNameInfoVO> dataList,

List<DesignProductResult> designTempletDeputyBeijingInfoList,

Map<String, DesignProductResult> matchResponseInfoMap,

LoginUser loginUser,List<DesignProductResult> unmatchProductList

) throws IntelligenceDecorationException {

// 参数验证/处理 ->start

if(dataList == null) {

return;

}

if(Lists.isEmpty(designTempletDeputyBeijingInfoList)) {

return;

}

if(matchResponseInfoMap == null) {

return;

}

// 参数验证/处理 ->end

for(DesignProductResult designProductResult : designTempletDeputyBeijingInfoList) {

logger.debug("productCode:" + designProductResult.getProductCode() + ";posName:" + designProductResult.getPosName());

PosNameInfoVO posNameInfoVO = new PosNameInfoVO();

// 识别对应的墙体方位

String wallOrientation = designProductResult.getWallOrientation().trim();

if(StringUtils.isBlank(wallOrientation)) {

throw new IntelligenceDecorationException("样板房中某个背景墙的墙体方位为空,productCode:" + designProductResult.getProductCode());

}

if(matchResponseInfoMap.containsKey(wallOrientation.substring(0, 1))) {

DesignProductResult designProductResultMatchedMainProduct = matchResponseInfoMap.get(wallOrientation.substring(0, 1));

BaseProduct baseProduct = getMatchSystemProduct(designProductResult, designProductResultMatchedMainProduct, loginUser);

if(baseProduct != null) {

designProductResultMatchedMainProduct.setProductCode(baseProduct.getProductCode());

posNameInfoVO = setPosNameInfoVO(designProductResult, designProductResultMatchedMainProduct);

}else {

posNameInfoVO = setPosNameInfoVO(designProductResult,unmatchProductList);

}

}else {

posNameInfoVO = setPosNameInfoVO(designProductResult,unmatchProductList);

}

dataList.add(posNameInfoVO);

}

}

/\*\*

\* 匹配样板房背景墙主墙,返回一个匹配结构供附墙搜索用

\*

\* @author huangsongbo

\* @param dataList

\* @param designTempletMainBeijingInfoList

\* @param designProductRecommendedResultList

\* @return

\* @throws IntelligenceDecorationException

\*/

private Map<String, DesignProductResult> matchMainBeijingProduct(List<PosNameInfoVO> dataList,

List<DesignProductResult> designTempletMainBeijingInfoList,

List<DesignProductResult> designProductRecommendedResultList,

LoginUser loginUser, List<DesignProductResult> unmatchProductList) throws IntelligenceDecorationException {

Map<String, DesignProductResult> returnMap = new HashMap<String, DesignProductResult>();

// 参数验证/参数处理 ->start

if(dataList == null) {

return returnMap;

}

if(Lists.isEmpty(designTempletMainBeijingInfoList)) {

return returnMap;

}

if(designProductRecommendedResultList == null) {

return returnMap;

}

if(loginUser == null) {

return returnMap;

}

// 参数验证/参数处理 ->end

for(DesignProductResult templetProductInfo : designTempletMainBeijingInfoList) {

logger.debug("templetProductInfo.posName:" + templetProductInfo.getPosName() + ";wallType:" + templetProductInfo.getWallType() + ";wallOrientation:" + templetProductInfo.getWallOrientation());

PosNameInfoVO posNameInfoVO = null;

DesignProductResult designProductResult = null;

DesignProductResult designProductResultMatched = this.matchBeijingProduct(templetProductInfo, designProductRecommendedResultList);

if(designProductResultMatched == null) {

}else {

// 验证长宽高

if(this.isMatchingPlanProduct(templetProductInfo, designProductResultMatched, new StringBuffer(""))) {

// 背景墙长高匹配成功

posNameInfoVO = this.setPosNameInfoVO(templetProductInfo, designProductResultMatched);

designProductResult = designProductResultMatched;

}else {

// 自行搜索

BaseProduct baseProduct = getMatchSystemProduct(templetProductInfo ,designProductResultMatched, loginUser);

if(baseProduct != null) {

designProductResultMatched.setProductCode(baseProduct.getProductCode());

posNameInfoVO = this.setPosNameInfoVO(templetProductInfo, designProductResultMatched);

designProductResult = this.getDesignProductResultByBaseProduct(baseProduct,designProductResultMatched);

}

}

}

if(posNameInfoVO != null) {

}else {

posNameInfoVO = setPosNameInfoVO(templetProductInfo,unmatchProductList);

}

dataList.add(posNameInfoVO);

// 将匹配结果加入Map<String, DesignProductResult>中 ->start

if(designProductResult != null) {

Integer wallOrientationStart = null;

if(StringUtils.isNotEmpty(templetProductInfo.getWallOrientation())) {

Integer wallOrientationInteger = null;

try {

wallOrientationInteger = Integer.valueOf(templetProductInfo.getWallOrientation().trim());

}catch (Exception e) {

}

if(wallOrientationInteger != null) {

wallOrientationStart = wallOrientationInteger / 100;

}

}

returnMap.put(wallOrientationStart + "", designProductResult);

}

// 将匹配结果加入Map<String, DesignProductResult>中 ->end

}

return returnMap;

}

/\*\*

\* BaseProduct转化为DesignProductResult

\*

\* @author huangsongbo

\* @param baseProduct

\* @return

\*/

private DesignProductResult getDesignProductResultByBaseProduct(BaseProduct baseProduct,DesignProductResult designProductResult) {

// 参数验证 ->start

if(baseProduct == null || designProductResult == null) {

return null;

}

// 参数验证 ->end

designProductResult.setBigTypeValue(baseProduct.getProductTypeValue());

designProductResult.setProductCode(baseProduct.getProductCode());

designProductResult.setProductHeight(baseProduct.getProductHeight());

designProductResult.setProductId(baseProduct.getId());

designProductResult.setProductLength(baseProduct.getProductLength());

designProductResult.setProductModelNumber(baseProduct.getProductModelNumber());

designProductResult.setProductWidth(baseProduct.getProductWidth());

designProductResult.setSmallTypeValue(baseProduct.getProductSmallTypeValue());

designProductResult.setStyleId(baseProduct.getStyleId());

designProductResult.setProductStyleIdInfo(baseProduct.getProductStyleIdInfo());

return designProductResult;

}

/\*\*

\* 墙体方位+墙体类型匹配背景墙

\*

\* @author huangsongbo

\* @param templetProductInfo

\* @param designProductRecommendedResultList

\* @return

\* @throws IntelligenceDecorationException

\*/

private DesignProductResult matchBeijingProduct(DesignProductResult templetProductInfo,

List<DesignProductResult> designProductRecommendedResultList) throws IntelligenceDecorationException {

// 参数验证/处理 ->start

if(templetProductInfo == null) {

return null;

}

if(Lists.isEmpty(designProductRecommendedResultList)) {

return null;

}

// 参数验证/处理 ->end

// 得到匹配分数的一个list

List<Integer> scoreList = this.getScoreListBeijing(templetProductInfo, designProductRecommendedResultList);

// 取分数最高的index

int index = Utils.getMaxNumIndexFromList(scoreList);

Integer scoreMax = scoreList.get(index);

if(-1 == scoreMax.intValue()) {

// 没有一个背景墙匹配上

return null;

}

return designProductRecommendedResultList.get(index);

}

/\*\*

\* 匹配背景墙,得到一个分数的结果集(墙体分类对上,加2分,墙体方位对上,加1分)

\*

\* @author huangsongbo

\* @param templetProductInfo

\* @param designProductRecommendedResultList

\* @return

\* @throws IntelligenceDecorationException

\*/

private List<Integer> getScoreListBeijing(DesignProductResult templetProductInfo,

List<DesignProductResult> designProductRecommendedResultList) throws IntelligenceDecorationException {

// 参数验证/处理 ->start

if(templetProductInfo == null) {

return null;

}

if(Lists.isEmpty(designProductRecommendedResultList)) {

return null;

}

// 参数验证/处理 ->end

List<Integer> scoreList = new ArrayList<Integer>();

String wallType = templetProductInfo.getWallType().trim();

if(StringUtils.isBlank(wallType)) {

throw new IntelligenceDecorationException("样板房中某个背景墙墙体分类为空;productCode:" + templetProductInfo.getProductCode());

}

String wallOrientation = templetProductInfo.getWallOrientation().trim();

if(StringUtils.isBlank(wallOrientation)) {

throw new IntelligenceDecorationException("样板房中某个背景墙墙体方位为空;productCode:" + templetProductInfo.getProductCode());

}

for(DesignProductResult recommendedProductResult : designProductRecommendedResultList) {

Integer score = -1;

// 墙体方位匹配上,加2分

if(StringUtils.equals(wallType, recommendedProductResult.getWallType().trim())) {

score += 2;

}

// 墙体方位匹配上,加1分

if(StringUtils.equals(wallOrientation, recommendedProductResult.getWallOrientation().trim())) {

score += 1;

}

scoreList.add(score);

}

return scoreList;

}

/\*\*

\* 整理出一份墙体方位:推荐方案产品信息的Map:墙体方位:List<墙体类型>

\*

\* @author huangsongbo

\* @param designProductRecommendedResultList

\* @return

\*/

private Map<String, List<String>> getRecommendedProductMapByWallOrientation(

List<DesignProductResult> designProductRecommendedResultList) {

Map<String, List<String>> returnMap = new HashMap<String, List<String>>();

// 参数验证/处理 ->start

if(Lists.isEmpty(designProductRecommendedResultList)) {

return returnMap;

}

// 参数验证/处理 ->end

for(DesignProductResult designProductResult : designProductRecommendedResultList) {

String wallOrientation = designProductResult.getWallOrientation();

if(StringUtils.isBlank(wallOrientation)) {

continue;

}

if(returnMap.containsKey(wallOrientation)) {

List<String> wallTypeList = returnMap.get(wallOrientation);

wallTypeList.add(designProductResult.getWallType());

}else {

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

wallTypeList.add(designProductResult.getWallType());

returnMap.put(wallOrientation, wallTypeList);

}

}

return returnMap;

}

/\*\*

\* 根据deleteNumInfoMap(根据推荐方案数据和推荐方案对应的样板房数据,得出的?类型墙面删除?面的一个数据信息)删除样房中的背景墙

\*

\* @author huangsongbo

\* @param dataList 匹配结果集

\* @param deleteNumInfoMap 根据推荐方案数据和推荐方案对应的样板房数据,得出的?类型墙面删除?面的一个数据信息

\* @param designTempletMainBeijingInfoList 样板房中主墙信息

\* @param recommendedProductMapByWallOrientation 推荐方案中主墙信息(墙体方位:List<墙体类型>)

\*/

private void deleteTempletBeijingProduct(List<PosNameInfoVO> dataList, Map<String, Integer> deleteNumInfoMap,

List<DesignProductResult> designTempletMainBeijingInfoList, Map<String, List<String>> recommendedProductMapByWallOrientation,

List<DesignProductResult> unmatchProductList) {

// 参数验证/处理 ->start

if(deleteNumInfoMap == null) {

return;

}

if(Lists.isEmpty(designTempletMainBeijingInfoList)) {

return;

}

if(recommendedProductMapByWallOrientation == null) {

return;

}

// 参数验证/处理 ->end

for (int index = 0; index < designTempletMainBeijingInfoList.size(); index++) {

DesignProductResult designTempletProductResult = designTempletMainBeijingInfoList.get(index);

// 根据墙体分类判断该背景墙要不要被删除

if(designTempletProductResult == null) {

continue;

}

String wallType = designTempletProductResult.getWallType();

if(StringUtils.isBlank(wallType)) {

continue;

}

if(deleteNumInfoMap.containsKey(wallType)) {

Integer num = deleteNumInfoMap.get(wallType);

if(num != null && num.intValue() > 0) {

// 校验要不要删除这面背景墙(检验相同墙体方位的推荐方案中的背景墙,墙体类型是否一样,如果一样,则不删除) ->start

String wallOrientation = designTempletProductResult.getWallOrientation();

if(StringUtils.isBlank(wallOrientation)) {

continue;

}

if(recommendedProductMapByWallOrientation.containsKey(wallOrientation)) {

List<String> wallTypelist = recommendedProductMapByWallOrientation.get(wallOrientation);

if(Lists.isNotEmpty(wallTypelist)) {

if(wallTypelist.indexOf(wallType.trim()) == -1) {

designTempletMainBeijingInfoList.remove(index);

index --;

num --;

deleteNumInfoMap.put(wallType, num);

// 加入结果集

dataList.add(setPosNameInfoVO(designTempletProductResult,unmatchProductList));

}

}

}else {

designTempletMainBeijingInfoList.remove(index);

index --;

num --;

deleteNumInfoMap.put(wallType, num);

// 加入结果集

dataList.add(setPosNameInfoVO(designTempletProductResult,unmatchProductList));

}

// 校验要不要删除这面背景墙(检验相同墙体方位的推荐方案中的背景墙,墙体类型是否一样,如果一样,则不删除) ->end

}

}

}

}

/\*\*

\* 得到一个map数据,标识?类型的背景墙,要删除?面

\* Map<String, Integer> : 墙体分类:样板房中要删除的面数

\*

\* @author huangsongbo

\* @param designProductTempletResultListByWallType 样板房中现有主墙信息

\* @param designProductRecommendedResultListByWallType 推荐方案中现有主墙信息

\* @param deletedBeijingProductInfoList 推荐方案中已经被删除的背景墙信息

\* @return

\* @throws IntelligenceDecorationException

\*/

private Map<String, Integer> getDeleteNumInfoMap(

Map<String, List<DesignProductResult>> designProductTempletResultListByWallType,

Map<String, List<DesignProductResult>> designProductRecommendedResultListByWallType,

List<Map<deletedBeijingProductInfoMapKeyEnum, String>> deletedBeijingProductInfoList

) throws IntelligenceDecorationException {

Map<String, Integer> returnMap = new HashMap<String, Integer>();

// 参数验证/处理 ->start

if(designProductTempletResultListByWallType == null) {

return returnMap;

}

if(designProductRecommendedResultListByWallType == null) {

return returnMap;

}

if(deletedBeijingProductInfoList == null) {

return returnMap;

}

// 参数验证/处理 ->end

for(Map<deletedBeijingProductInfoMapKeyEnum, String> map : deletedBeijingProductInfoList) {

String wallType = map.get(deletedBeijingProductInfoMapKeyEnum.wallType);

if(StringUtils.isBlank(wallType)) {

throw new IntelligenceDecorationException("推荐方案对应的样板房某个背景墙的墙体分类为空,一键装修失败");

}

// 检测推荐方案中该墙体类型的背景墙还剩几个

int num = 0;

if(designProductRecommendedResultListByWallType.containsKey(wallType)) {

List<DesignProductResult> designProductResultList = designProductRecommendedResultListByWallType.get(wallType);

if(Lists.isNotEmpty(designProductResultList)) {

num = designProductResultList.size();

}

}

// 计算出样板房要删除几个该墙体类型的背景墙

int num2 = 0;

if(designProductTempletResultListByWallType.containsKey(wallType)) {

List<DesignProductResult> designProductResultList = designProductTempletResultListByWallType.get(wallType);

if(Lists.isNotEmpty(designProductResultList)) {

num2 = designProductResultList.size();

}

}

returnMap.put(wallType, num2 - num);

}

return returnMap;

}

private Map<String, List<DesignProductResult>> getDesignProductResultListBeijingByWallType(

List<DesignProductResult> designProductResultListBeijing) throws IntelligenceDecorationException {

Map<String, List<DesignProductResult>> returnMap = new HashMap<String, List<DesignProductResult>>();

// 参数验证/处理 ->start

if(Lists.isEmpty(designProductResultListBeijing)) {

return returnMap;

}

// 参数验证/处理 ->end

// 墙体分类归类 ->start

for(DesignProductResult designProductResult : designProductResultListBeijing) {

String wallType = designProductResult.getWallType();

if(StringUtils.isBlank(wallType)) {

throw new IntelligenceDecorationException("样板房中有背景墙没有录墙体类型,productCode:" + designProductResult.getProductCode());

}

if(returnMap.containsKey(wallType)) {

List<DesignProductResult> designProductResultList = returnMap.get(wallType);

designProductResultList.add(designProductResult);

}else {

List<DesignProductResult> designProductResultList = new ArrayList<DesignProductResult>();

designProductResultList.add(designProductResult);

returnMap.put(wallType, designProductResultList);

}

}

// 墙体分类归类 ->end

return returnMap;

}

/\*\*

\* 取出样板房中附墙+非主墙/主墙的list

\*

\* @author huangsongbo

\* @param designProductResultListBeijing

\* @return

\* @throws IntelligenceDecorationException bedroomProductMatch

\*/

private Map<getDesignProductResultListBeijingMapEnum, List<DesignProductResult>> getDesignProductResultListBeijingMap(

List<DesignProductResult> designProductResultListBeijing) throws IntelligenceDecorationException {

Map<getDesignProductResultListBeijingMapEnum, List<DesignProductResult>> returnMap = new HashMap<IntelligenceDecorationServiceImpl.getDesignProductResultListBeijingMapEnum, List<DesignProductResult>>();

List<DesignProductResult> designTempletMainBeijingInfoList = new ArrayList<DesignProductResult>();

List<DesignProductResult> designTempletDeputyBeijingInfoList = new ArrayList<DesignProductResult>();

// 分类 ->start

if(Lists.isNotEmpty(designProductResultListBeijing)) {

for(DesignProductResult designProductResult : designProductResultListBeijing) {

if(designProductResult.getGroupType() != null && 1 == designProductResult.getGroupType().intValue()

&& designProductResult.getIsMainStructureProduct() != null && 1 != designProductResult.getIsMainStructureProduct().intValue()) {

// 结构背景墙中的非主墙

designTempletDeputyBeijingInfoList.add(designProductResult);

continue;

}

Integer wallOrientationInteger = null;

try {

wallOrientationInteger = Integer.valueOf(designProductResult.getWallOrientation().trim());

}catch (Exception e) {

throw new IntelligenceDecorationException("样板房中某个背景墙的墙体方位数据异常:wallOrientation:" + designProductResult.getWallOrientation());

}

if(wallOrientationInteger != null) {

if(wallOrientationInteger % 100 != 0) {

// 附墙

designTempletDeputyBeijingInfoList.add(designProductResult);

continue;

}

}

// 剩下的为主墙

designTempletMainBeijingInfoList.add(designProductResult);

}

}

// 分类 ->end

returnMap.put(getDesignProductResultListBeijingMapEnum.mainBeijing, designTempletMainBeijingInfoList);

returnMap.put(getDesignProductResultListBeijingMapEnum.deputyBeijing, designTempletDeputyBeijingInfoList);

return returnMap;

}

/\*\*

\* mainBeijing 主墙

\* deputyBeijing 附墙+结构背景墙非主墙

\*

\* @author huangsongbo

\*/

private enum getDesignProductResultListBeijingMapEnum{

mainBeijing, deputyBeijing

}

/\*\*

\* 剔除部分不需要匹配的背景墙(结构背景墙非主墙/附墙)

\*

\* @author huangsongbo

\* @param planProductList

\* @return

\* @throws IntelligenceDecorationException

\*/

private List<DesignProductResult> getDesignProductRecommendedResultList(List<DesignProductResult> planProductList)

throws IntelligenceDecorationException {

List<DesignProductResult> designProductRecommendedResultList = new ArrayList<DesignProductResult>();

if(planProductList == null) {

return designProductRecommendedResultList;

}

for(DesignProductResult designProductResult : planProductList) {

logger.debug("posName:" + designProductResult.getPosName());

// 去白膜背景墙,迷之原因发现推荐方案中有白膜背景墙但是改造发现不显示 huangsongbo update 2017.12.20->start

logger.debug("------productCode:" + designProductResult.getProductCode());

if(StringUtils.isNotEmpty(designProductResult.getProductCode()) && designProductResult.getProductCode().startsWith("baimo\_")){

continue;

}

// 去白膜背景墙,迷之原因发现推荐方案中有白膜背景墙但是改造发现不显示 huangsongbo update 2017.12.20->end

if(beijingValuekeyList.indexOf(designProductResult.getSmallTypeKey()) != -1) {

// 结构背景墙非主墙不参与匹配 ->start

if(designProductResult.getGroupType() != null

&& 1 == designProductResult.getGroupType().intValue()

&& (designProductResult.getIsMainStructureProduct() == null || 1 != designProductResult.getIsMainStructureProduct().intValue())) {

// 是结构背景墙但是非主墙

continue;

}

// 结构背景墙非主墙不参与匹配 ->end

// 附墙不参与匹配 ->start

if(StringUtils.isNotBlank(designProductResult.getWallOrientation())) {

Integer wallOrientationInteger = null;

try {

wallOrientationInteger = Integer.valueOf(designProductResult.getWallOrientation().trim());

}catch (Exception e) {

throw new IntelligenceDecorationException("样板房中某个背景墙的墙体方位数据异常:wallOrientation:" + designProductResult.getWallOrientation());

}

if(wallOrientationInteger != null) {

if(wallOrientationInteger % 100 != 0) {

// 附墙

continue;

}

}

}

// 附墙不参与匹配 ->end

designProductRecommendedResultList.add(designProductResult);

}

}

return designProductRecommendedResultList;

}

/\*\*

\* 背景墙数据list分类(按照墙体分类)

\*

\* @author huangsongbo

\* @param designProductRecommendedResultList

\* @return

\* @throws IntelligenceDecorationException

\*/

private Map<String, List<DesignProductResult>> getDesignProductResultListByWallType(

List<DesignProductResult> designProductRecommendedResultList) throws IntelligenceDecorationException {

Map<String, List<DesignProductResult>> returnMap = new HashMap<String, List<DesignProductResult>>();

// 参数验证/处理 ->start

if(Lists.isEmpty(designProductRecommendedResultList)) {

return returnMap;

}

// 参数验证/处理 ->end

// 墙体分类归类 ->start

for(DesignProductResult designProductResult : designProductRecommendedResultList) {

String wallType = designProductResult.getWallType();

if(StringUtils.isBlank(wallType)) {

throw new IntelligenceDecorationException("样板房中有背景墙没有录墙体类型,productCode:" + designProductResult.getProductCode());

}

if(returnMap.containsKey(wallType)) {

List<DesignProductResult> designProductResultList = returnMap.get(wallType);

designProductResultList.add(designProductResult);

}else {

List<DesignProductResult> designProductResultList = new ArrayList<DesignProductResult>();

designProductResultList.add(designProductResult);

returnMap.put(wallType, designProductResultList);

}

}

// 墙体分类归类 ->end

return returnMap;

}

/\*\*

\* 获取被删除的背景墙信息

\*

\* @author huangsongbo

\* @param recommendedPlanId

\* @return

\* @throws IntelligenceDecorationException

\*/

private List<Map<deletedBeijingProductInfoMapKeyEnum, String>> getDeletedBeijingProductInfo(Integer recommendedPlanId) throws IntelligenceDecorationException {

if(recommendedPlanId == null) {

throw new IntelligenceDecorationException("推荐方案id不能为空");

}

DesignPlanRecommended designPlanRecommended = designPlanRecommendedServiceV2.get(recommendedPlanId);

if(designPlanRecommended == null) {

throw new IntelligenceDecorationException("推荐方案没有找到,推荐方案id:" + recommendedPlanId);

}

// 返回结果

Integer designTempletId = designPlanRecommended.getDesignTemplateId();

// 获取样板房中所有背景墙信息

List<DesignTempletProduct> beijingDesignTempletProductList = designTempletProductService.getBeijingProductInfo(designTempletId);

List<Map<deletedBeijingProductInfoMapKeyEnum, String>> beijingDesignTempletProductMapList= new ArrayList<Map<deletedBeijingProductInfoMapKeyEnum,String>>();

if(Lists.isNotEmpty(beijingDesignTempletProductList)) {

for(DesignTempletProduct designTempletProduct : beijingDesignTempletProductList) {

Map<deletedBeijingProductInfoMapKeyEnum, String> designTempletProductMap = new HashMap<IntelligenceDecorationServiceImpl.deletedBeijingProductInfoMapKeyEnum, String>();

designTempletProductMap.put(deletedBeijingProductInfoMapKeyEnum.groupType, designTempletProduct.getGroupType() + "");

designTempletProductMap.put(deletedBeijingProductInfoMapKeyEnum.isMainStructureProduct, designTempletProduct.getIsMainStructureProduct() + "");

designTempletProductMap.put(deletedBeijingProductInfoMapKeyEnum.wallOrientation, designTempletProduct.getWallOrientation());

designTempletProductMap.put(deletedBeijingProductInfoMapKeyEnum.wallType, designTempletProduct.getWallType());

beijingDesignTempletProductMapList.add(designTempletProductMap);

}

}

// 获取推荐方案中所有背景墙信息

List<DesignPlanRecommendedProduct> beijingDesignPlanRecommendedProductList = designPlanRecommendedProductServiceV2.getBeijingProductInfo(recommendedPlanId);

List<Map<deletedBeijingProductInfoMapKeyEnum, String>> beijingDesignPlanRecommendedProductMapList= new ArrayList<Map<deletedBeijingProductInfoMapKeyEnum,String>>();

if(Lists.isNotEmpty(beijingDesignPlanRecommendedProductList)) {

for(DesignPlanRecommendedProduct designPlanRecommendedProduct : beijingDesignPlanRecommendedProductList) {

Map<deletedBeijingProductInfoMapKeyEnum, String> designPlanRecommendedProductMap = new HashMap<IntelligenceDecorationServiceImpl.deletedBeijingProductInfoMapKeyEnum, String>();

designPlanRecommendedProductMap.put(deletedBeijingProductInfoMapKeyEnum.groupType, designPlanRecommendedProduct.getGroupType() + "");

designPlanRecommendedProductMap.put(deletedBeijingProductInfoMapKeyEnum.isMainStructureProduct, designPlanRecommendedProduct.getIsMainStructureProduct() + "");

designPlanRecommendedProductMap.put(deletedBeijingProductInfoMapKeyEnum.wallOrientation, designPlanRecommendedProduct.getWallOrientation());

designPlanRecommendedProductMap.put(deletedBeijingProductInfoMapKeyEnum.wallType, designPlanRecommendedProduct.getWallType());

beijingDesignPlanRecommendedProductMapList.add(designPlanRecommendedProductMap);

}

}

// 处理出被删除的样板房信息

beijingDesignTempletProductMapList.removeAll(beijingDesignPlanRecommendedProductMapList);

// 剔除非主墙背景墙(结构背景墙非主墙+附墙) ->start

for (int index = 0; index < beijingDesignTempletProductMapList.size(); index++) {

Map<deletedBeijingProductInfoMapKeyEnum, String> deletedBeijingProductInfoMap = beijingDesignTempletProductMapList.get(index);

if(deletedBeijingProductInfoMap == null) {

beijingDesignTempletProductMapList.remove(index);

index --;

continue;

}

// 踢除附墙 ->start

String wallOrientation = deletedBeijingProductInfoMap.get(deletedBeijingProductInfoMapKeyEnum.wallOrientation);

if(StringUtils.isNotEmpty(wallOrientation)) {

Integer wallOrientationInteger = null;

try {

wallOrientationInteger = Integer.valueOf(wallOrientation.trim());

}catch (Exception e) {

throw new IntelligenceDecorationException("样板房中某个背景墙的墙体方位数据异常:wallOrientation:" + wallOrientation);

}

if(wallOrientationInteger == null) {

throw new IntelligenceDecorationException("样板房中某个背景墙的墙体方位数据异常:wallOrientation:" + wallOrientation);

}

if(wallOrientationInteger % 100 != 0) {

// 附墙

beijingDesignTempletProductMapList.remove(index);

index --;

continue;

}

}

// 踢除附墙 ->end

// 提出结构背景墙非主墙 ->start

String groupType = deletedBeijingProductInfoMap.get(deletedBeijingProductInfoMapKeyEnum.groupType);

String isMainStructureProduct = deletedBeijingProductInfoMap.get(deletedBeijingProductInfoMapKeyEnum.isMainStructureProduct);

if(StringUtils.equals("1", groupType) && !StringUtils.equals("1", isMainStructureProduct)) {

beijingDesignTempletProductMapList.remove(index);

index --;

continue;

}

// 提出结构背景墙非主墙 ->end

}

// 剔除非主墙背景墙(结构背景墙非主墙+附墙) ->end

return beijingDesignTempletProductMapList;

}

/\*\*

\* wallType 墙体分类

\* wallOrientation 墙体方位

\* groupType 是否是结构背景墙 0:否;1:是

\* isMainStructureProduct 是否是主墙

\* @author huangsongbo

\*

\*/

public enum deletedBeijingProductInfoMapKeyEnum {

wallType, wallOrientation, groupType, isMainStructureProduct

}

/\*\*

\* @author xiaoxc

\* 错误数据日志打印

\* @param templetProduct

\* @param planProduct

\* @param logInfo

\* @return

\*/

private StringBuffer getLogInfo(DesignProductResult templetProduct,DesignProductResult planProduct,StringBuffer logInfo){

String bjType = Utils.getValue("app.smallProductType.beiJingWall", "");

if (templetProduct != null) {

String templetProductCode = templetProduct.getProductCode();

String templetWallType = templetProduct.getWallType();

String templetWallOrientation = templetProduct.getWallOrientation();

Integer templetProductIndex = templetProduct.getProductIndex();

String templetPosName = templetProduct.getPosName();

String templetRegionMark = templetProduct.getRegionMark();

if (ProductTypeConstant.PRODUCT\_BIG\_TYPE\_TIANH.equals(templetProduct.getBigTypeKey())) {

if (StringUtils.isEmpty(templetRegionMark) || "0".equals(templetRegionMark)) {

logInfo.append("[样板房:posName="+templetPosName+";productCode="+templetProductCode+";regionMark="+templetRegionMark+";尺寸代码为空];");

}

}

if (ProductTypeConstant.PRODUCT\_BIG\_TYPE\_QIANGM.equals(templetProduct.getBigTypeKey())) {

if (templetProduct.getProductGroupId() != null && templetProduct.getProductGroupId() > 0) {

if (StringUtils.isEmpty(templetWallOrientation) || "0".equals(templetWallOrientation)

|| StringUtils.isEmpty(templetWallType) || "0".equals(templetWallType)

|| templetProductIndex == null || templetProductIndex == 0) {

logInfo.append("[样板房:posName="+templetPosName+";productCode="+templetProductCode+";wallType="+templetWallType+";wallOrientation="+templetWallOrientation+";productIndex"+templetProductIndex+"];");

}

}else{

if(bjType.indexOf(templetProduct.getSmallTypeKey()) != -1){

if (StringUtils.isEmpty(templetWallOrientation) || "0".equals(templetWallOrientation)

|| StringUtils.isEmpty(templetWallType) || "0".equals(templetWallType)) {

logInfo.append("[样板房:posName="+templetPosName+";productCode="+templetProductCode+";wallType="+templetWallType+";wallOrientation="+templetWallOrientation+"];");

}

}

}

}

if(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_LI.equals(templetProduct.getBigTypeKey())){

if (StringUtils.isEmpty(templetWallType) || "0".equals(templetWallType)) {

logInfo.append("[样板房:posName="+templetPosName+";productCode="+templetProductCode+";wallType="+templetWallType+"];");

}

}

}

if (planProduct != null) {

String planProductCode = planProduct.getProductCode();

String planWallType = planProduct.getWallType();

String planWallOrientation = planProduct.getWallOrientation();

Integer planProductIndex = planProduct.getProductIndex();

String planPosName = planProduct.getPosName();

String planRegionMark = planProduct.getRegionMark();

if (ProductTypeConstant.PRODUCT\_BIG\_TYPE\_TIANH.equals(planProduct.getBigTypeKey())) {

if (StringUtils.isEmpty(planRegionMark) || "0".equals(planRegionMark)) {

logInfo.append("[方案：posName="+planPosName+";productCode="+planProductCode+";regionMark="+planRegionMark+";尺寸代码为空];");

}

}

if (ProductTypeConstant.PRODUCT\_BIG\_TYPE\_QIANGM.equals(planProduct.getBigTypeKey())) {

if (planProduct.getProductGroupId() != null && planProduct.getProductGroupId() > 0) {

if (StringUtils.isEmpty(planWallType) || "0".equals(planWallType)

|| StringUtils.isEmpty(planWallOrientation) || "0".equals(planWallOrientation)

|| planProductIndex == null || planProductIndex == 0) {

logInfo.append("方案：posName="+planPosName+";productCode="+planProductCode+";wallType="+planWallType+";wallOrientation="+planWallOrientation+";planProductIndex"+planProductIndex+"];");

}

}else{

if(bjType.indexOf(planProduct.getSmallTypeKey()) != -1){

if (StringUtils.isEmpty(planWallType) || "0".equals(planWallType)

|| StringUtils.isEmpty(planWallOrientation) || "0".equals(planWallOrientation)) {

logInfo.append("方案：posName="+planPosName+";productCode="+planProductCode+";wallType="+planWallType+";wallOrientation="+planWallOrientation+"];");

}

}

}

}

if(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_LI.equals(planProduct.getBigTypeKey())){

if (StringUtils.isEmpty(planWallType) || "0".equals(planWallType)){

logInfo.append("方案：posName="+planPosName+";productCode="+planProductCode+";wallType="+planWallType+"];");

}

}

}

return logInfo;

}

/\*\*

\* 判断是否需要删除背景墙

\* 场景:

\* 主墙被删除,一件装修后副墙也应该被删除

\*

\* @author huangsongbo

\* @param wallOrientation

\* @param deleteWallOrientationList

\* @return

\*/

private boolean judgeWhetherToDelete(String wallOrientation, List<String> deleteWallOrientationList) {

// 参数验证 ->start

if(StringUtils.isBlank(wallOrientation)) {

return false;

}

if(Lists.isEmpty(deleteWallOrientationList)) {

return false;

}

// 参数验证 ->end

// wallOrientation:200; deleteWallOrientationList:[200]

// 前缀是2,且wallOrientation>deleteWallOrientationList(主墙一般是200,副墙是210,220..)

// 找出deleteWallOrientationList中是否有前缀2的墙体方位

for(String deleteWallOrientation : deleteWallOrientationList) {

if(wallOrientation.substring(0, 1).equals(deleteWallOrientation.substring(0, 1))) {

if(Pattern.matches(wallOrientation.substring(0, 1) + "0+", wallOrientation)) {

// 如果wallOrientation是200,那么deleteWallOrientation也要是200

if(StringUtils.equals(wallOrientation, deleteWallOrientation)) {

return true;

}

}else {

// 必须相等,或者deleteWallOrientation是200

if(StringUtils.equals(wallOrientation, deleteWallOrientation) || Pattern.matches(deleteWallOrientation.substring(0, 1) + "0+", deleteWallOrientation)) {

return true;

}

}

}else {

}

}

return false;

}

@Override

public List<MatchGroupProductResult> getGroupProductData(DesignPlanRecommended designPlanRecommended,

DesignTemplet designTemplet) {

// 返回结果

List<MatchGroupProductResult> returnList = new ArrayList<MatchGroupProductResult>();

// 参数验证 ->start

if(designPlanRecommended == null){

return returnList;

}

if(designTemplet == null){

return returnList;

}

// 参数验证 ->end

// 查询样板房白膜组合的主产品列表

List<DesignTempletProduct> designTempletProductList = designTempletProductService.getMainProductListByTempletId(designTemplet.getId());

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

return returnList;

}

//查询方案推荐中存在的组合的主产品列表

List<DesignPlanRecommendedProduct> designPlanRecommendedProductList =

designPlanRecommendedProductServiceV2.getMainProductListByPlanRecommendedId(designPlanRecommended.getId());

if (Lists.isEmpty(designPlanRecommendedProductList)) {

return returnList;

}

if (designPlanRecommendedProductList == null || designPlanRecommendedProductList.size() < 0) {

logger.error("designTempletProductList size为==>"+designTempletProductList.size());

}

// 匹配组合 ->start

// 记录已经推荐方案中已经匹配过的组合

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

for(DesignTempletProduct designTempletProduct : designTempletProductList){

MatchGroupProductResult matchGroupProductResult = new MatchGroupProductResult();

// 得到匹配分数

List<Integer> scoreList = this.getScoreList(designTempletProduct, designPlanRecommendedProductList);

// 如果scoreList的元素都是0,意思是没有匹配上组合,需要把样板房该白膜组合删除

boolean noMatch = this.getNoMatch(scoreList);

// 设置主产品对应的组合的所有产品信息(bmGroupProductList)

List<Map<String,String>> bmGroupProductList = this.getBmGroupProductListByTempletIdAndGroupId(designTemplet.getId(), designTempletProduct.getProductGroupId());

matchGroupProductResult.setBmMianProductCode(designTempletProduct.getProductCode());

matchGroupProductResult.setBmGroupProductList(bmGroupProductList);

if(noMatch){

}else{

// 找出匹配分数最高的组合主产品

int indexMaxScore = this.getIndexMaxScore(scoreList);

// 识别该组合是否被匹配过,及对应的逻辑处理 ->start

String planGroupId = designPlanRecommendedProductList.get(indexMaxScore).getPlanGroupId();

if(StringUtils.isNotEmpty(planGroupId)){

if(planGroupIdList.indexOf(planGroupId) == -1){

planGroupIdList.add(planGroupId);

}else{

matchGroupProductResult.setIsReName(ProductConstant.MATCHGROUPPRODUCTRESULT\_ISRENAME\_ISMATCHED);

}

}

// 识别该组合是否被匹配过,及对应的逻辑处理 ->end

this.getGroupMatcheDataV2(designPlanRecommendedProductList.get(indexMaxScore), matchGroupProductResult, designPlanRecommended.getId());

if(matchGroupProductResult.getIsReName().intValue() == ProductConstant.MATCHGROUPPRODUCTRESULT\_ISRENAME\_ISMATCHED.intValue()){

String planGroupIdOld = matchGroupProductResult.getPlanGroupId();

String planGroupIdNew = planGroupIdOld.substring(0, planGroupIdOld.indexOf("\_")) + "\_" + new Date().getTime();

matchGroupProductResult.setPlanGroupId(planGroupIdNew);

}

}

if(StringUtils.isNotEmpty(matchGroupProductResult.getTempletMianProductCode())){

returnList.add(matchGroupProductResult);

}else{}

}

return returnList;

}

/\*\*

\* scoreList得到最大元素的index

\* @param scoreList

\* @return

\*/

private int getIndexMaxScore(List<Integer> scoreList) {

int MaxScoreIndex = 0;

Integer item = 0;

for (int index = 0; index < scoreList.size(); index ++) {

Integer score = scoreList.get(index);

if(score > item){

item = score;

MaxScoreIndex = index;

}

}

return MaxScoreIndex;

}

/\*\*

\* 检测scoreList的元素是不是都是0,如果都是0,返回true

\*

\* @author huangsongbo

\* @param scoreList

\* @return

\*/

private boolean getNoMatch(List<Integer> scoreList) {

boolean returnBoolean = true;

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

return false;

}

for(Integer score : scoreList){

if(0 != score.intValue()){

return false;

}

}

return returnBoolean;

}

/\*\*

\* 通过样板房id,组合id,获取该样板房的组合明细信息

\* copy from xiaoxiangcheng's function

\*

\* @author huangsongbo

\* @param designTempletId

\* @param productGroupId

\* @return

\*/

private List<Map<String, String>> getBmGroupProductListByTempletIdAndGroupId(Integer designTempletId, Integer productGroupId) {

// 参数验证 ->start

if(designTempletId == null){

return null;

}

if(productGroupId == null){

return null;

}

// 参数验证 ->end

DesignTempletProduct templetGroupProduct = new DesignTempletProduct();

templetGroupProduct.setDesignTempletId(designTempletId);

templetGroupProduct.setProductGroupId(productGroupId);

templetGroupProduct.setPlanGroupId(templetGroupProduct.getPlanGroupId());

templetGroupProduct.setIsDeleted(0);

templetGroupProduct.setGroupType(0);

List<DesignTempletProduct> groupProductList = designTempletProductMapper.byTempletIdMainProduct(templetGroupProduct);

List<Map<String,String>> bmGroupProductList = new ArrayList<>();

Map<String,String> map = null;

if (Lists.isNotEmpty(groupProductList)) {

for (DesignTempletProduct groupProduct : groupProductList) {

map = new HashMap<>();

map.put("productCode",groupProduct.getProductCode());

map.put("posIndexPath",groupProduct.getPosIndexPath());

bmGroupProductList.add(map);

}

}

return bmGroupProductList;

}

/\*\*

\* 匹配样板房白膜组合主产品和对应的推荐方案的所有组合(主产品),得到一个分数的list

\* (该list的index为推荐方案组合主产品List的index,分数越高则匹配度越高,0为未匹配上,未匹配上的白膜组合要删除)

\*

\* @author huangsongbo

\* @param designTempletProduct

\* @param designPlanRecommendedProductList

\* @return

\*/

private List<Integer> getScoreList(DesignTempletProduct designTempletProduct,

List<DesignPlanRecommendedProduct> designPlanRecommendedProductList) {

List<Integer> scoreList = new ArrayList<Integer>();

// 参数验证 ->start

if(designTempletProduct == null){

return scoreList;

}

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

return scoreList;

}

Integer templetProductBigTypeValue = designTempletProduct.getProductTypeValue();

Integer templetProductSmallTypeValue = designTempletProduct.getProductSmallTypeValue();

// 需要转化为白膜对应的小分类

templetProductSmallTypeValue = baseProductService.dealWithBaimoSmallTypeValue(templetProductBigTypeValue, templetProductSmallTypeValue);

if(templetProductBigTypeValue == null){

return scoreList;

}

// 参数验证 ->end

// 初始化scoreList ->start

for (int index = 0; index < designPlanRecommendedProductList.size(); index++) {

scoreList.add(0);

}

// 初始化scoreList ->end

// 匹配组合 ->start

for (int index = 0; index < designPlanRecommendedProductList.size(); index++) {

DesignPlanRecommendedProduct designPlanRecommendedProduct = designPlanRecommendedProductList.get(index);

// 匹配组合类型

if (designTempletProduct.getCompositeType() != null && designTempletProduct.getCompositeType().equals(designPlanRecommendedProduct.getCompositeType())) {

// 大类匹配成功,score ++,并且匹配区域编码

scoreList.set(index, scoreList.get(index) + 1);

if(StringUtils.isEmpty(designTempletProduct.getRegionMark()) && StringUtils.isEmpty(designPlanRecommendedProduct.getRegionMark())){

continue;

}

if(StringUtils.equals(designTempletProduct.getRegionMark(), designPlanRecommendedProduct.getRegionMark())){

// 区域编码匹配成功,score += 1

scoreList.set(index, scoreList.get(index) + 1);

}

}

}

// 匹配组合 ->end

return scoreList;

}

public List<ProductMatchInfoDTO> productListMatch\_old(

List<PlanProductInfo> templetProductList,

Map<String, List<PlanProductInfo>> productListmap,

Integer planId,

String username,

Integer matchType,

Integer opType,

DesignTemplet designTemplet,

Integer planRecommendedId

) throws IntelligenceDecorationException, ElasticSearchException {

List<ProductMatchInfoDTO> productMatchInfoDTOList = new ArrayList<BedroomProductMatchDTO.ProductMatchInfoDTO>();

// \*参数验证 ->start

if(Lists.isEmpty(templetProductList)) {

return productMatchInfoDTOList;

}

if(productListmap == null) {

return productMatchInfoDTOList;

}

if (opType == null) {

opType = DesignPlanConstants.USER\_RENDER;

}

// \*参数验证 ->end

// 记录下所有背景墙信息,后面再特殊匹配

List<PlanProductInfo> beijingPlanProductInfoList = new ArrayList<ProductListByTypeInfo.PlanProductInfo>();

// 记录下所有挂画信息,后面再特殊匹配

List<PlanProductInfo> pipePlanProductInfoList = new ArrayList<ProductListByTypeInfo.PlanProductInfo>();

// 存系列产品,用来后面单独匹配(系列标识:样板房产品信息)(系列标识:该系列标识对应的样板房产品list)

Map<String, List<PlanProductInfo>> seriesProductInfoMap = new HashMap<String, List<PlanProductInfo>>();

for(PlanProductInfo planProductInfo : templetProductList) {

logger.debug("posName:" + planProductInfo.getPosName());

PlanProductInfo planProductInfoMatched = null;

// 组装系列产品,放到后面去匹配 ->start

String seriesSign = planProductInfo.getSeriesSign();

if(StringUtils.isNotEmpty(seriesSign)){

// 硬装替换,不需要进行系列匹配(默认系列都是软装)

if(seriesProductInfoMap.containsKey(seriesSign)) {

List<PlanProductInfo> planProductInfoList = seriesProductInfoMap.get(seriesSign);

if(Lists.isNotEmpty(planProductInfoList)) {

planProductInfoList.add(planProductInfo);

}else {

planProductInfoList = new ArrayList<ProductListByTypeInfo.PlanProductInfo>();

planProductInfoList.add(planProductInfo);

seriesProductInfoMap.put(seriesSign, planProductInfoList);

}

}else {

List<PlanProductInfo> planProductInfoList = new ArrayList<ProductListByTypeInfo.PlanProductInfo>();

planProductInfoList.add(planProductInfo);

seriesProductInfoMap.put(seriesSign, planProductInfoList);

}

continue;

}

// 组装系列产品,放到后面去匹配 ->end

StringBuffer matchInfo = new StringBuffer("");

matchInfo.append("正在匹配:posName:" + planProductInfo.getPosName() + ";baseProductCode:" + planProductInfo.getProductCode() + "\n");

logger.debug("posName:" + planProductInfo.getPosName());

// \*参数验证 ->start

if(StringUtils.isEmpty(planProductInfo.getBigTypeValuekey())) {

continue;

}

// \*参数验证 ->end

ProductMatchInfoDTO productMatchInfoDTO = new BedroomProductMatchDTO().new ProductMatchInfoDTO();

productMatchInfoDTO.setDelPosName(planProductInfo.getPosName());

String bigTypeValuekey = planProductInfo.getBigTypeValuekey();

String smallTypeValuekey = planProductInfo.getSmallTypeValuekey();

List<PlanProductInfo> theSameBigTypePlanProductInfoList = null;

if(productListmap.containsKey(bigTypeValuekey)) {

theSameBigTypePlanProductInfoList = productListmap.get(bigTypeValuekey);

}

if(0 == matchType) {

// 水龙头的话,可能推荐方案中没有,也要匹配

if(Lists.isEmpty(theSameBigTypePlanProductInfoList) && StringUtils.equals(ProductTypeConstant.PRODUCT\_SMALL\_TYPE\_LYBP, smallTypeValuekey)) {

// 未检测到推荐方案中有同大类的产品

matchInfo.append("匹配失败(未检测到推荐方案有该白膜对应的大类)(posName:" + planProductInfo.getPosName() + ";productCode:" + planProductInfo.getProductCode() + ")\n");

productMatchInfoDTO.setMatchInfo(matchInfo.toString());

productMatchInfoDTOList.add(productMatchInfoDTO);

continue;

}

}

// \*匹配地面->start

if(StringUtils.equals(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_DIM, bigTypeValuekey)) {

if(0 != matchType && (StringUtils.equals("basic\_dit", smallTypeValuekey) || StringUtils.equals("dit", smallTypeValuekey))) {

// 硬装替换的话,地毯不算硬装,特殊处理 ->start

planProductInfoMatched = planProductInfo.clone();

// 硬装替换的话,地毯不算硬装,特殊处理 ->end

}else {

planProductInfoMatched = this.productListMatchDim(planProductInfo, theSameBigTypePlanProductInfoList, matchInfo);

}

}

// \*匹配地面->end

// \*匹配门->start

else if(StringUtils.equals(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_MENG, bigTypeValuekey)) {

planProductInfoMatched = this.productListMatchMeng(planProductInfo, theSameBigTypePlanProductInfoList, matchInfo);

}

// \*匹配门->start

// \*匹配墙面 ->start

else if(StringUtils.equals(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_QIANGM, bigTypeValuekey)) {

if(beijingValuekeyList.indexOf(smallTypeValuekey) != -1) {

// \*背景墙特殊处理(处理完其他的单品,再单独处理背景墙list) ->start

beijingPlanProductInfoList.add(planProductInfo);

continue;

// \*背景墙特殊处理(处理完其他的单品,再单独处理背景墙list) ->end

}else {

// 其他qiangm

planProductInfoMatched = productListMatchQiangm(planProductInfo, theSameBigTypePlanProductInfoList, matchInfo);

}

}

// \*匹配墙面 ->end

// \*匹配天花 ->start

else if(StringUtils.equals(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_TIANH, bigTypeValuekey)) {

planProductInfoMatched = productListMatchTianhForDrawHouse(planProductInfo, theSameBigTypePlanProductInfoList, matchInfo, designTemplet.getSmallpoxIdentify());

}

// \*匹配天花 ->end

// \*匹配家纺 ->start

else if(StringUtils.equals(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_HO, bigTypeValuekey)) {

if(0 == matchType) {

planProductInfoMatched = productListMatchHo(planProductInfo, theSameBigTypePlanProductInfoList, matchInfo);

}else {

planProductInfoMatched = planProductInfo.clone();

}

}

// \*匹配家纺 ->end

// 匹配柜子 ->start

else if(StringUtils.equals(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_CA, bigTypeValuekey)) {

if(0 == matchType) {

planProductInfoMatched = productListMatchCa(planProductInfo, theSameBigTypePlanProductInfoList, matchInfo);

}else {

planProductInfoMatched = planProductInfo.clone();

}

}

// 匹配柜子 ->end

// 匹配卫浴(淋浴房) ->start

else if(StringUtils.equals(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_BA, bigTypeValuekey)) {

if(0 == matchType) {

planProductInfoMatched = productListMatchBa(planProductInfo, theSameBigTypePlanProductInfoList, matchInfo);

}else {

planProductInfoMatched = planProductInfo.clone();

}

}

// 匹配卫浴(淋浴房) ->end

// 厨房配件匹配(水槽) ->start

else if(StringUtils.equals(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_KP, bigTypeValuekey)) {

if(0 == matchType) {

planProductInfoMatched = productListMatchKp(planProductInfo, theSameBigTypePlanProductInfoList, matchInfo);

}else {

planProductInfoMatched = planProductInfo.clone();

}

}

// 厨房配件匹配(水槽) ->end

// 浴室配件匹配(主要是水龙头) ->start

else if(StringUtils.equals(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_BP, bigTypeValuekey)) {

if(0 == matchType) {

planProductInfoMatched = productListMatchBp(planProductInfo, productListmap, matchInfo);

}else {

planProductInfoMatched = planProductInfo.clone();

}

}

// 浴室配件匹配(主要是水龙头) ->end

// 饰品(主要是挂画)匹配逻辑 ->start

else if(StringUtils.equals(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_PE, bigTypeValuekey)) {

if(0 == matchType) {

// 挂画放到后面单独匹配 ->start

if(StringUtils.equals(ProductTypeConstant.PRODUCT\_SMALL\_TYPE\_BASIC\_PIPE, smallTypeValuekey)){

pipePlanProductInfoList.add(planProductInfo);

continue;

}

// 挂画放到后面单独匹配 ->end

planProductInfoMatched = productListMatchOther(planProductInfo, theSameBigTypePlanProductInfoList, matchInfo);

}else {

planProductInfoMatched = planProductInfo.clone();

}

}

// 饰品(主要是挂画)匹配逻辑 ->end

// \*其他大类 ->start

else {

if(0 == matchType) {

planProductInfoMatched = productListMatchOther(planProductInfo, theSameBigTypePlanProductInfoList, matchInfo);

}else {

planProductInfoMatched = planProductInfo.clone();

}

}

// \*其他大类 ->end

/\*if(flag) {\*/

// 添加设计方案产品数据

this.getProductMatchInfoDTO(planId, username, opType, planProductInfo, matchInfo, productMatchInfoDTO, planProductInfoMatched);

productMatchInfoDTOList.add(productMatchInfoDTO);

/\*}\*/

}

// 再特殊处理背景墙 ->start

List<PlanProductInfo> theSameBigTypePlanProductInfoList = null;

if(productListmap.containsKey(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_QIANGM)) {

theSameBigTypePlanProductInfoList = productListmap.get(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_QIANGM);

}

// 只保留背景墙

for (int index = 0; index < theSameBigTypePlanProductInfoList.size(); index++) {

if(beijingValuekeyList.indexOf(theSameBigTypePlanProductInfoList.get(index).getSmallTypeValuekey()) != -1) {

}else {

theSameBigTypePlanProductInfoList.remove(index);

index --;

}

}

this.productListMatchBeijingQiangm(beijingPlanProductInfoList, theSameBigTypePlanProductInfoList, productMatchInfoDTOList, planId, username, opType, planRecommendedId);

// 再特殊处理背景墙 ->end

// 特殊处理挂画 ->start

this.productListMatchPipeV2(pipePlanProductInfoList, productListmap.get(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_PE), productMatchInfoDTOList, planId, username, opType);

// 特殊处理挂画 ->end

// 特殊处理系列产品 ->start

this.productListMatchSeries(seriesProductInfoMap, productListmap, productMatchInfoDTOList, planId, username, opType, matchType);

// 特殊处理系列产品 ->end

return productMatchInfoDTOList;

}

@Override

public Map<String, Object> productListMatch(

List<PlanProductInfo> templetProductList,

Map<String, List<PlanProductInfo>> productListmap,

PlanStructureInfo recommendedPlanStructureInfo,

Integer planId,

String username,

Integer matchType,

Integer opType,

DesignTemplet designTemplet,

Integer planRecommendedId

) throws IntelligenceDecorationException, ElasticSearchException {

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

List<ProductMatchInfoDTO> productMatchInfoDTOList = new ArrayList<BedroomProductMatchDTO.ProductMatchInfoDTO>();

List<DimMatchInfoVO> dimMatchInfoVOList = new ArrayList<DimMatchInfoVO>();

// \*参数验证 ->start

if(Lists.isEmpty(templetProductList)) {

return null;

}

if(productListmap == null) {

return null;

}

if (opType == null) {

opType = DesignPlanConstants.USER\_RENDER;

}

// \*参数验证 ->end

// 记录下所有背景墙信息,后面再特殊匹配

List<PlanProductInfo> beijingPlanProductInfoList = new ArrayList<ProductListByTypeInfo.PlanProductInfo>();

// 记录下所有挂画信息,后面再特殊匹配

List<PlanProductInfo> pipePlanProductInfoList = new ArrayList<ProductListByTypeInfo.PlanProductInfo>();

// 存系列产品,用来后面单独匹配(系列标识:样板房产品信息)(系列标识:该系列标识对应的样板房产品list)

Map<String, List<PlanProductInfo>> seriesProductInfoMap = new HashMap<String, List<PlanProductInfo>>();

for(PlanProductInfo planProductInfo : templetProductList) {

logger.debug("posName:" + planProductInfo.getPosName());

PlanProductInfo planProductInfoMatched = null;

ProductMatchInfoDTO productMatchInfoDTO = new BedroomProductMatchDTO().new ProductMatchInfoDTO();

productMatchInfoDTO.setDelPosName(planProductInfo.getPosName());

if(!planProductInfo.isNeedBeMatched()) {

this.getProductMatchInfoDTO(planId, username, opType, planProductInfo, new StringBuffer(planProductInfo.getMatchInfo()), productMatchInfoDTO, planProductInfoMatched);

productMatchInfoDTOList.add(productMatchInfoDTO);

continue;

}

// 组装系列产品,放到后面去匹配 ->start

String seriesSign = planProductInfo.getSeriesSign();

if(StringUtils.isNotEmpty(seriesSign)){

// 硬装替换,不需要进行系列匹配(默认系列都是软装)

if(seriesProductInfoMap.containsKey(seriesSign)) {

List<PlanProductInfo> planProductInfoList = seriesProductInfoMap.get(seriesSign);

if(Lists.isNotEmpty(planProductInfoList)) {

planProductInfoList.add(planProductInfo);

}else {

planProductInfoList = new ArrayList<ProductListByTypeInfo.PlanProductInfo>();

planProductInfoList.add(planProductInfo);

seriesProductInfoMap.put(seriesSign, planProductInfoList);

}

}else {

List<PlanProductInfo> planProductInfoList = new ArrayList<ProductListByTypeInfo.PlanProductInfo>();

planProductInfoList.add(planProductInfo);

seriesProductInfoMap.put(seriesSign, planProductInfoList);

}

continue;

}

// 组装系列产品,放到后面去匹配 ->end

StringBuffer matchInfo = new StringBuffer("");

matchInfo.append("正在匹配:posName:" + planProductInfo.getPosName() + ";baseProductCode:" + planProductInfo.getProductCode() + "\n");

logger.debug("posName:" + planProductInfo.getPosName());

// \*参数验证 ->start

if(StringUtils.isEmpty(planProductInfo.getBigTypeValuekey())) {

continue;

}

// \*参数验证 ->end

String bigTypeValuekey = planProductInfo.getBigTypeValuekey();

String smallTypeValuekey = planProductInfo.getSmallTypeValuekey();

List<PlanProductInfo> theSameBigTypePlanProductInfoList = null;

if(productListmap.containsKey(bigTypeValuekey)) {

theSameBigTypePlanProductInfoList = productListmap.get(bigTypeValuekey);

}

if(0 == matchType) {

// 水龙头的话,可能推荐方案中没有,也要匹配

if(Lists.isEmpty(theSameBigTypePlanProductInfoList) && StringUtils.equals(ProductTypeConstant.PRODUCT\_SMALL\_TYPE\_LYBP, smallTypeValuekey)) {

// 未检测到推荐方案中有同大类的产品

matchInfo.append("匹配失败(未检测到推荐方案有该白膜对应的大类)(posName:" + planProductInfo.getPosName() + ";productCode:" + planProductInfo.getProductCode() + ")\n");

productMatchInfoDTO.setMatchInfo(matchInfo.toString());

productMatchInfoDTOList.add(productMatchInfoDTO);

continue;

}

}

// \*匹配地面->start

if(StringUtils.equals(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_DIM, bigTypeValuekey)) {

if(0 != matchType && (StringUtils.equals("basic\_dit", smallTypeValuekey) || StringUtils.equals("dit", smallTypeValuekey))) {

// 硬装替换的话,地毯不算硬装,特殊处理 ->start

planProductInfoMatched = planProductInfo.clone();

// 硬装替换的话,地毯不算硬装,特殊处理 ->end

}else {

// 以下小类的dim,可能是结构

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

dimTypeList.add(ProductTypeConstant.PRODUCT\_SMALL\_TYPE\_BASIC\_DIMZ);

dimTypeList.add(ProductTypeConstant.PRODUCT\_SMALL\_TYPE\_BASIC\_DIM);

dimTypeList.add(ProductTypeConstant.PRODUCT\_SMALL\_TYPE\_BASIC\_DIMGD);

dimTypeList.add(ProductTypeConstant.PRODUCT\_SMALL\_TYPE\_BASIC\_DIMXG);

if(dimTypeList.indexOf(planProductInfo.getSmallTypeValuekey()) != -1) {

Map<String, Object> productListMatchDimResultMap= this.productListMatchDimV2(planProductInfo, theSameBigTypePlanProductInfoList, recommendedPlanStructureInfo, matchInfo, dimTypeList);

if(productListMatchDimResultMap != null) {

planProductInfoMatched = (PlanProductInfo) productListMatchDimResultMap.get(IntelligenceDecorationConstant.PRODUCTLISTMATCHDIMRESULTMAP\_PLANPRODUCTINFO);

DimMatchInfoVO dimMatchInfoVO = (DimMatchInfoVO) productListMatchDimResultMap.get(IntelligenceDecorationConstant.PRODUCTLISTMATCHDIMRESULTMAP\_DIMMATCHINFOVO);

if(dimMatchInfoVO != null) {

dimMatchInfoVOList.add(dimMatchInfoVO);

}

}

}else {

planProductInfoMatched = this.productListMatchDim(planProductInfo, theSameBigTypePlanProductInfoList, matchInfo);

}

// 户型绘制新逻辑 add by huangsongbo 2018.3.16 ->end

}

}

// \*匹配地面->end

// \*匹配门->start

else if(StringUtils.equals(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_MENG, bigTypeValuekey)) {

planProductInfoMatched = this.productListMatchMeng(planProductInfo, theSameBigTypePlanProductInfoList, matchInfo);

}

// \*匹配门->start

// \*匹配墙面 ->start

else if(StringUtils.equals(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_QIANGM, bigTypeValuekey)) {

if(beijingValuekeyList.indexOf(smallTypeValuekey) != -1) {

// \*背景墙特殊处理(处理完其他的单品,再单独处理背景墙list) ->start

beijingPlanProductInfoList.add(planProductInfo);

continue;

// \*背景墙特殊处理(处理完其他的单品,再单独处理背景墙list) ->end

}else {

// 其他qiangm

planProductInfoMatched = productListMatchQiangm(planProductInfo, theSameBigTypePlanProductInfoList, matchInfo);

}

}

// \*匹配墙面 ->end

// \*匹配天花 ->start

else if(StringUtils.equals(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_TIANH, bigTypeValuekey)) {

planProductInfoMatched = productListMatchTianhForDrawHouse(planProductInfo, theSameBigTypePlanProductInfoList, matchInfo, designTemplet.getSmallpoxIdentify());

}

// \*匹配天花 ->end

// \*匹配家纺 ->start

else if(StringUtils.equals(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_HO, bigTypeValuekey)) {

if(0 == matchType) {

planProductInfoMatched = productListMatchHo(planProductInfo, theSameBigTypePlanProductInfoList, matchInfo);

}else {

planProductInfoMatched = planProductInfo.clone();

}

}

// \*匹配家纺 ->end

// 匹配柜子 ->start

else if(StringUtils.equals(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_CA, bigTypeValuekey)) {

if(0 == matchType) {

planProductInfoMatched = productListMatchCa(planProductInfo, theSameBigTypePlanProductInfoList, matchInfo);

}else {

planProductInfoMatched = planProductInfo.clone();

}

}

// 匹配柜子 ->end

// 匹配卫浴(淋浴房) ->start

else if(StringUtils.equals(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_BA, bigTypeValuekey)) {

if(0 == matchType) {

planProductInfoMatched = productListMatchBa(planProductInfo, theSameBigTypePlanProductInfoList, matchInfo);

}else {

planProductInfoMatched = planProductInfo.clone();

}

}

// 匹配卫浴(淋浴房) ->end

// 厨房配件匹配(水槽) ->start

else if(StringUtils.equals(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_KP, bigTypeValuekey)) {

if(0 == matchType) {

planProductInfoMatched = productListMatchKp(planProductInfo, theSameBigTypePlanProductInfoList, matchInfo);

}else {

planProductInfoMatched = planProductInfo.clone();

}

}

// 厨房配件匹配(水槽) ->end

// 浴室配件匹配(主要是水龙头) ->start

else if(StringUtils.equals(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_BP, bigTypeValuekey)) {

if(0 == matchType) {

planProductInfoMatched = productListMatchBp(planProductInfo, productListmap, matchInfo);

}else {

planProductInfoMatched = planProductInfo.clone();

}

}

// 浴室配件匹配(主要是水龙头) ->end

// 饰品(主要是挂画)匹配逻辑 ->start

else if(StringUtils.equals(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_PE, bigTypeValuekey)) {

if(0 == matchType) {

// 挂画放到后面单独匹配 ->start

if(StringUtils.equals(ProductTypeConstant.PRODUCT\_SMALL\_TYPE\_BASIC\_PIPE, smallTypeValuekey)){

pipePlanProductInfoList.add(planProductInfo);

continue;

}

// 挂画放到后面单独匹配 ->end

planProductInfoMatched = productListMatchOther(planProductInfo, theSameBigTypePlanProductInfoList, matchInfo);

}else {

planProductInfoMatched = planProductInfo.clone();

}

}

// 饰品(主要是挂画)匹配逻辑 ->end

// 大家电 add by huangsongbo 2018.5.3 ->start

else if(StringUtils.equals(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_EL, bigTypeValuekey)) {

if(0 == matchType) {

planProductInfoMatched = productListMatchEL(planProductInfo, theSameBigTypePlanProductInfoList, matchInfo);

}else {

planProductInfoMatched = planProductInfo.clone();

}

}

// 大家电 add by huangsongbo 2018.5.3 ->end

// \*其他大类 ->start

else {

if(0 == matchType) {

planProductInfoMatched = productListMatchOther(planProductInfo, theSameBigTypePlanProductInfoList, matchInfo);

}else {

planProductInfoMatched = planProductInfo.clone();

}

}

// \*其他大类 ->end

// 添加设计方案产品数据

this.getProductMatchInfoDTO(planId, username, opType, planProductInfo, matchInfo, productMatchInfoDTO, planProductInfoMatched);

productMatchInfoDTOList.add(productMatchInfoDTO);

}

// 再特殊处理背景墙 ->start

List<PlanProductInfo> theSameBigTypePlanProductInfoList = null;

if(productListmap.containsKey(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_QIANGM)) {

theSameBigTypePlanProductInfoList = productListmap.get(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_QIANGM);

}

// 只保留背景墙

for (int index = 0; index < theSameBigTypePlanProductInfoList.size(); index++) {

if(beijingValuekeyList.indexOf(theSameBigTypePlanProductInfoList.get(index).getSmallTypeValuekey()) != -1) {

}else {

theSameBigTypePlanProductInfoList.remove(index);

index --;

}

}

this.productListMatchBeijingQiangm(beijingPlanProductInfoList, theSameBigTypePlanProductInfoList, productMatchInfoDTOList, planId, username, opType, planRecommendedId);

// 再特殊处理背景墙 ->end

// 特殊处理挂画 ->start

this.productListMatchPipeV2(pipePlanProductInfoList, productListmap.get(ProductTypeConstant.PRODUCT\_BIG\_TYPE\_PE), productMatchInfoDTOList, planId, username, opType);

// 特殊处理挂画 ->end

// 特殊处理系列产品 ->start

this.productListMatchSeries(seriesProductInfoMap, productListmap, productMatchInfoDTOList, planId, username, opType, matchType);

// 特殊处理系列产品 ->end

returnMap.put(IntelligenceDecorationConstant.PRODUCTLISTMATCHRESULTMAP\_PRODUCTMATCHINFO, productMatchInfoDTOList);

returnMap.put(IntelligenceDecorationConstant.PRODUCTLISTMATCHRESULTMAP\_DIMMATCHINFOVO, dimMatchInfoVOList);

return returnMap;

}

}