package com.sandu.api.decoratecustomer.model;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

import java.io.Serializable;

import java.util.Date;

/\*\*

\* CopyRight (c) 2018 Sandu Technology Inc.

\* <p>

\* demo

\*

\* @author sandu <dev@sanduspace.cn>

\* @datetime 2018-10-20 14:27

\*/

@Data

@Builder

@NoArgsConstructor

@AllArgsConstructor

public class DecorateCustomer implements Serializable {

/\*\*

\* 客户ID

\*/

private Integer id;

/\*\*

\* 客户类型(0-客户店铺预约;1-抢单;2-平台派单)

\*/

private Integer customerType;

/\*\*

\* 装修业主Id(proprietor\_info的主键ID)

\*/

private Integer proprietorInfoId;

/\*\*

\* 店铺预约装企ID

\*/

private Integer companyId;

/\*\*

\* 内部指派装企ID

\*/

private Integer innerCompanyId;

/\*\*

\* 第一家抢单企业ID

\*/

private Integer firstSeckillCompany;

/\*\*

\* 第二家抢单企业ID

\*/

private Integer secondSeckillCompany;

/\*\*

\* 方案ID

\*/

private Integer designPlanId;

/\*\*

\* 回访时间

\*/

private Date revisitTime;

/\*\*

\* 装修类型

\*/

private Integer decorateType;

/\*\*

\* 签约装企

\*/

private Integer contractCompany;

/\*\*

\* 签约价格

\*/

private String contractPrice;

/\*\*

\* 备注

\*/

private String remark;

/\*\*

\* 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;

/\*\*

\* 装修预算

\*/

private Integer decorateBudget;

/\*\*

\* 报价截止时间

\*/

private Date bidEndTime;

/\*\*

\* 平台派单.报价状态 0-报价中; 1-完成报价; 2-完成了报价但是订单未指派出去

\*/

private Integer bidStatus;

}

package com.sandu.api.decoratecustomer.model;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok. NoArgsConstructor;

import java.io.Serializable;

import java.util.Date;

/\*\*

\* CopyRight (c) 2018 Sandu Technology Inc.

\* <p>

\* decorate

\*

\* @author sandu <dev@sanduspace.cn>

\* @datetime 2018-10-23 14:24

\*/

@Data

@Builder

@NoArgsConstructor

@AllArgsConstructor

public class DecoratePrice implements Serializable {

/\*\*

\* 自增id

\*/

private Integer id;

/\*\*

\* 报价企业ID

\*/

private Integer companyId;

/\*\*

\* 客户ID(关联decorate\_customer的主键ID)

\*/

private Integer customerId;

/\*\*

\* 装修业主Id(proprietor\_info的主键ID)

\*/

private Integer proprietorInfoId;

/\*\*

\* 状态(0-待报价;1-已报价;2-未选中;3-超时取消报价;4-已选中)

\*/

private Integer status;

private String price;

/\*\*

\* 报价用户ID

\*/

private Integer bidUserId;

/\*\*

\* 材料费(元)

\*/

private Double materialFee;

/\*\*

\* 质检费(元)

\*/

private Double checkFee;

/\*\*

\* 人工费(元)

\*/

private Double labourFee;

/\*\*

\* 设计费(元)

\*/

private Double designFee;

/\*\*

\* 报价提交时间

\*/

private Date submitTime;

/\*\*

\* 报价开始时间

\*/

private Date startTime;

/\*\*

\* 截止时间

\*/

private Date endTime;

/\*\*

\* 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;

}

package com.sandu.api.decoratecustomer.model;

import lombok.AllArgsConstructor;

import lombok.Builder;

import lombok.Data;

import lombok.NoArgsConstructor;

import java.io.Serializable;

import java.util.Date;

/\*\*

\* 描述:decorate\_seckill表的实体类

\*

\* @author: Sandu

\* @创建时间: 2018-10-25

\*/

@Builder

@NoArgsConstructor

@AllArgsConstructor

@Data

public class DecorateSeckill implements Serializable {

/\*\*

\* 自增id

\*/

private Integer id;

/\*\*

\* 客户ID(decorate\_customer主键ID)

\*/

private Integer customerId;

/\*\*

\* 第一家抢单企业(客服提交抢单时写入)

\*/

private Integer firstSeckillCompany;

/\*\*

\* 抢购用户ID

\*/

private Integer userId;

/\*\*

\* 装修业主Id(proprietor\_info的主键ID)

\*/

private Integer proprietorInfoId;

/\*\*

\* 状态(0-待抢购;1-已抢购)

\*/

private Byte status;

/\*\*

\* 客单价(元)

\*/

private Double price;

/\*\*

\* 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;

}

package com.sandu.api.decoratecustomer.output;

import com.fasterxml.jackson.annotation.JsonFormat;

import com.sandu.api.decoratecustomer.model.bo.DecorateCustomerBO;

import com.sandu.api.decoratecustomer.model.bo.DecorateCustomerListBO;

import io.swagger.annotations.ApiModelProperty;

import lombok.Data;

import java.io.Serializable;

import java.util.Date;

import java.util.List;

/\*\*

\* CopyRight (c) 2018 Sandu Technology Inc.

\*

\* <p>demo

\*

\* @author sandu <dev@sanduspace.cn>

\* @datetime 2018-Oct-20 14:27

\*/

@Data

public class DecorateCustomerListVO implements Serializable {

private Integer id;

@ApiModelProperty("客户名称")

private String userName;

@ApiModelProperty("客户手机")

private String mobile;

@ApiModelProperty("城市编码")

private String cityCode;

@ApiModelProperty("城市名称")

private String cityName;

@ApiModelProperty(name = "客户预约企业ID", hidden = true)

private Integer companyId;

@ApiModelProperty("客户预约企业")

private String companyName;

@ApiModelProperty(name = "第一家抢单企业ID", hidden = true)

private Integer firstSecKillCompany;

@ApiModelProperty("第一家抢单企业名称")

private String firstSecKillCompanyName;

@ApiModelProperty(name = "第2家抢单企业ID", hidden = true)

private Integer secondSecKillCompany;

@ApiModelProperty("第2家抢单企业名称")

private String secondSecKillCompanyName;

@ApiModelProperty("所有十家装企报价")

private List<DecorateCustomerListBO.SimpleDecoratePriceInfo> allPriceList;

@ApiModelProperty("选定装企报价")

private List<DecorateCustomerListBO.SimpleDecoratePriceInfo> choosePriceList;

@ApiModelProperty(value = "内部推荐企业", hidden = true)

private Integer innerCompanyId;

@ApiModelProperty("内部推荐企业名称")

private String innerCompanyName;

@ApiModelProperty("装修类型")

private Integer decorateType;

@ApiModelProperty("装修类型名称")

private String decorateTypeName;

@ApiModelProperty("回访时间")

@JsonFormat(pattern = "yyyy-MM-dd")

private Date revisitTime;

@ApiModelProperty("备注")

private String remark;

@ApiModelProperty("客户列表提交订单按钮权限")

private boolean commitOrderFunc;

@ApiModelProperty("签约装企")

private Integer contractCompany;

@ApiModelProperty("签约价格")

private String contractPrice;

@ApiModelProperty("业主信息")

private DecorateCustomerBO proprietorInfo;

}

package com.sandu.api.decoratecustomer.output;

import io.swagger.annotations.ApiModelProperty;

import lombok.Data;

import java.io.Serializable;

import java.util.Date;

/\*\*

\* CopyRight (c) 2018 Sandu Technology Inc.

\* @author sandu <dev@sanduspace.cn>

\* @datetime 2018-Oct-20 14:27

\*/

@Data

public class DecorateCustomerVO implements Serializable {

@ApiModelProperty("id")

private Integer id;

@ApiModelProperty("客户名称")

private String userName;

@ApiModelProperty("城市名称")

private String cityName;

@ApiModelProperty("回访时间")

private Date revisitTime;

@ApiModelProperty("备注")

private String remark;

@ApiModelProperty("小区")

private String areaName;

private String areaCode;

@ApiModelProperty("详细地址")

private String address;

@ApiModelProperty("面积")

private String houseAcreage;

@ApiModelProperty("户型")

private String houseType;

@ApiModelProperty("预算")

private Integer decorateBudget;

private String decorateBudgetStr;

@ApiModelProperty("装修类型")

private Integer decorateHouseType;

private String decorateHouseTypeStr;

@ApiModelProperty("倾向风格")

private Integer decorateStyle;

private String decorateStyleStr;

@ApiModelProperty("装修方式")

private Integer decorateType;

private String decorateTypeStr;

}

package com.sandu.api.decoratecustomer.service.biz;

import com.github.pagehelper.PageInfo;

import com.sandu.api.decoratecustomer.input.DecorateCustomerAdd;

import com.sandu.api.decoratecustomer.input.DecorateCustomerQuery;

import com.sandu.api.decoratecustomer.input.DecorateCustomerUpdate;

import com.sandu.api.decoratecustomer.model.bo.DecorateCustomerBO;

import com.sandu.api.decoratecustomer.model.bo.DecorateCustomerListBO;

import java.util.List;

/\*\*

\* CopyRight (c) 2018 Sandu Technology Inc.

\* @author sandu <dev@sanduspace.cn>

\* @datetime 2018-Oct-20 14:27

\*/

public interface DecorateCustomerBizService {

/\*\*

\* 创建

\*

\* @param input

\* @return

\*/

int create(DecorateCustomerAdd input);

/\*\*

\* 更新

\*

\* @param input

\* @return

\*/

int update(DecorateCustomerUpdate input);

/\*\*

\* 删除

\*/

int delete(String decorateCustomerId);

/\*\*

\* 通过ID获取详情

\*

\* @param decorateCustomerId

\* @return

\*/

DecorateCustomerBO getById(int decorateCustomerId);

/\*\*

\* 查询客户列表

\*

\* @param query

\* @return

\*/

PageInfo<DecorateCustomerListBO> query(DecorateCustomerQuery query);

/\*\*

\* 自动拍单/抢单

\*/

void dispatchDecoratePrice();

/\*\*

\* @return 预约装企生成订单

\*/

List<Integer> initOrderForCompanyCustomer();

/\*\*

\* 提交抢单

\*

\* @param customerId

\* @return

\*/

Boolean submitToSedKill(Integer customerId);

}

package com.sandu.api.decoratecustomer.service;

import com.sandu.api.decoratecustomer.model.DecoratePrice;

import java.util.List;

import java.util.Set;

/\*\*

\* CopyRight (c) 2018 Sandu Technology Inc.

\* <p>

\* decorate

\*

\* @author sandu <dev@sanduspace.cn>

\* @datetime 2018-Oct-23 14:24

\*/

public interface DecoratePriceService {

/\*\*

\* 插入

\*

\* @param decoratePrice

\* @return

\*/

int insert(DecoratePrice decoratePrice);

/\*\*

\* 更新

\*

\* @param decoratePrice

\* @return

\*/

int update(DecoratePrice decoratePrice);

/\*\*

\* 删除

\*

\* @param decoratePriceIds

\* @return

\*/

int delete(Set<Integer> decoratePriceIds);

/\*\*

\* 通过ID获取详情

\*

\* @param decoratePriceId

\* @return

\*/

DecoratePrice getById(int decoratePriceId);

List<DecoratePrice> listByCustomerIds(Set<Integer> ids);

/\*\*

\* 根据业主获取同城装企

\*

\* @param proprietorInfoId

\* @param limit

\* @return

\*/

List<Integer> listCompanyIdForDispatchPrice(Integer proprietorInfoId, int limit);

List<Integer> findNoneDispatchCustomerIds(List<Integer> ids);

void insertList(List<DecoratePrice> installList);

}

package com.sandu.api.decoratecustomer.service;

import com.sandu.api.decoratecustomer.model.DecorateSeckill;

import java.util.List;

public interface DecorateSecKillService {

void insertDecorateSedKill(DecorateSeckill decorateSeckill);

/\*\*

\* 根据客户ID list 过滤出 未参加过抢单的 客户ID

\*

\* @param collect

\* @return

\*/

List<Integer> findNoneSedKillCustomerIds(List<Integer> collect);

void insertDecorateSeckillList(List<DecorateSeckill> installList);

/\*\*

\* 根据客户ID查询抢单记录

\* @param asList

\* @return

\*/

List<DecorateSeckill> findSedKillByCustomerIds(List<Integer> asList);

Integer update(DecorateSeckill it);

}

package com.sandu.service.decoratecustomer.dao;

import com.sandu.api.decoratecustomer.input.DecorateCustomerQuery;

import com.sandu.api.decoratecustomer.model.DecorateCustomer;

import com.sandu.api.decoratecustomer.model.bo.DecorateCustomerBO;

import com.sandu.api.decoratecustomer.model.bo.DecorateCustomerListBO;

import org.apache.ibatis.annotations.Param;

import org.springframework.stereotype.Repository;

import java.util.List;

import java.util.Set;

/\*\*

\* CopyRight (c) 2018 Sandu Technology Inc.

\* @author sandu <dev@sanduspace.cn>

\* @datetime 2018-Oct-20 14:27

\*/

@Repository

public interface DecorateCustomerMapper {

int insert(DecorateCustomer decorateCustomer);

int updateByPrimaryKey(DecorateCustomer decorateCustomer);

int deleteByPrimaryKey(@Param("decorateCustomerIds") Set<Integer> decorateCustomerIds);

DecorateCustomer selectByPrimaryKey(@Param("decorateCustomerId") int decorateCustomerId);

List<DecorateCustomerListBO> findAll(DecorateCustomerQuery query);

List<DecorateCustomer> findByOption(@Param("query") DecorateCustomer query);

DecorateCustomerBO getByIdWithProprietorInfo(@Param("decorateCustomerId") int decorateCustomerId);

List<DecorateCustomer> listForDispatchDecoratePrice();

List<DecorateCustomer> findWithNullCompanyId();

DecorateCustomer getByProprietorInfoId(@Param("proprietorInfoId") Integer id);

List<DecorateCustomer> listCustomerWithInnerCompanyAndNoneOrder();

List<DecorateCustomer> getCustomerByProprietorIds(@Param("propIds") List<Integer> propId);

}

package com.sandu.service.decoratecustomer.dao;

import com.sandu.api.decoratecustomer.model.DecoratePrice;

import org.apache.ibatis.annotations.Param;

import org.springframework.stereotype.Repository;

import java.util.List;

import java.util.Set;

/\*\*

\* CopyRight (c) 2018 Sandu Technology Inc.

\* @author sandu <dev@sanduspace.cn>

\* @datetime 2018-Oct-23 14:24

\*/

@Repository

public interface DecoratePriceMapper {

int insert(DecoratePrice decoratePrice);

int updateByPrimaryKey(DecoratePrice decoratePrice);

int deleteByPrimaryKey(@Param("decoratePriceIds") Set<Integer> decoratePriceIds);

DecoratePrice selectByPrimaryKey(@Param("decoratePriceId") int decoratePriceId);

List<DecoratePrice> listByCustomerIds(@Param("ids") Set<Integer> ids);

List<Integer> listCompanyIdForDispatchPrice(@Param("proprietorInfoId") Integer proprietorInfoId, @Param("limit") int limit);

List<Integer> findDispatchCustomerIds(@Param("ids") List<Integer> ids);

}

package com.sandu.service.decoratecustomer.impl.biz;

import com.github.pagehelper.PageInfo;

import com.google.common.base.Strings;

import com.google.gson.Gson;

import com.sandu.api.area.output.BaseAreaListVO;

import com.sandu.api.area.service.BaseAreaService;

import com.sandu.api.company.service.CompanyService;

import com.sandu.api.company.service.CompanyShopService;

import com.sandu.api.decoratecustomer.input.DecorateCustomerAdd;

import com.sandu.api.decoratecustomer.input.DecorateCustomerQuery;

import com.sandu.api.decoratecustomer.input.DecorateCustomerUpdate;

import com.sandu.api.decoratecustomer.model.DecorateCustomer;

import com.sandu.api.decoratecustomer.model.DecoratePrice;

import com.sandu.api.decoratecustomer.model.DecorateSeckill;

import com.sandu.api.decoratecustomer.model.bo.DecorateCustomerBO;

import com.sandu.api.decoratecustomer.model.bo.DecorateCustomerListBO;

import com.sandu.api.decoratecustomer.service.DecorateCustomerService;

import com.sandu.api.decoratecustomer.service.DecoratePriceService;

import com.sandu.api.decoratecustomer.service.DecorateSecKillService;

import com.sandu.api.decoratecustomer.service.biz.DecorateCustomerBizService;

import com.sandu.api.decorateorder.input.DecorateOrderRefundUpdate;

import com.sandu.api.decorateorder.model.DecorateOrder;

import com.sandu.api.decorateorder.service.DecorateOrderService;

import com.sandu.api.decorateorder.service.biz.DecorateOrderBizService;

import com.sandu.api.dictionary.model.Dictionary;

import com.sandu.api.dictionary.service.DictionaryService;

import com.sandu.api.proprietorinfo.model.ProprietorInfo;

import com.sandu.api.proprietorinfo.service.ProprietorInfoService;

import com.sandu.api.shop.model.CompanyShop;

import com.sandu.api.sysusermessage.model.SysUserMessage;

import com.sandu.api.user.model.User;

import com.sandu.api.user.service.SysUserService;

import com.sandu.api.user.service.UserService;

import com.sandu.util.SocketIOUtil;

import com.sandu.util.Stringer;

import lombok.extern.slf4j.Slf4j;

import org.apache.commons.httpclient.util.DateUtil;

import org.apache.commons.lang3.StringUtils;

import org.apache.commons.lang3.time.DateUtils;

import org.springframework.beans.BeanUtils;

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

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

import org.springframework.stereotype.Service;

import java.util.\*;

import java.util.stream.Collectors;

/\*\*

\* CopyRight (c) 2018 Sandu Technology Inc.

\*

\* <p>demo

\*

\* @author sandu <dev@sanduspace.cn>

\* @datetime 2018-Oct-20 14:27

\*/

@Slf4j

@Service("decorateCustomerBizService")

public class DecorateCustomerBizServiceImpl implements DecorateCustomerBizService {

private final DecorateCustomerService decorateCustomerService;

private final CompanyService companyService;

private final DictionaryService dictionaryService;

private final DecoratePriceService decoratePriceService;

private final BaseAreaService baseAreaService;

private final ProprietorInfoService proprietorInfoService;

private final DecorateSecKillService decorateSecKillService;

private final DecorateOrderService decorateOrderService;

private final DecorateOrderBizService decorateOrderBizService;

private final UserService userService;

public static final Gson gson = new Gson();

@Value("${decorate.message.system.dispatchPrice}")

private String dispatchMessage;

@Value("${decorate.message.system.innerSet}")

private String innerSetMessage;

@Autowired private CompanyShopService companyShopService;

@Autowired private SysUserService sysUserService;

@Autowired

public DecorateCustomerBizServiceImpl(

DecorateCustomerService decorateCustomerService,

CompanyService companyService,

DictionaryService dictionaryService,

DecoratePriceService decoratePriceService,

BaseAreaService baseAreaService,

ProprietorInfoService proprietorInfoService,

DecorateSecKillService decorateSecKillService,

DecorateOrderService decorateOrderService,

DecorateOrderBizService decorateOrderBizService,

UserService userService) {

this.decorateCustomerService = decorateCustomerService;

this.companyService = companyService;

this.dictionaryService = dictionaryService;

this.decoratePriceService = decoratePriceService;

this.baseAreaService = baseAreaService;

this.proprietorInfoService = proprietorInfoService;

this.decorateSecKillService = decorateSecKillService;

this.decorateOrderService = decorateOrderService;

this.decorateOrderBizService = decorateOrderBizService;

this.userService = userService;

}

@Override

public int create(DecorateCustomerAdd input) {

DecorateCustomer.DecorateCustomerBuilder builder = DecorateCustomer.builder();

DecorateCustomer decorateCustomer = builder.build();

BeanUtils.copyProperties(input, decorateCustomer);

return decorateCustomerService.insert(decorateCustomer);

}

@Override

public int update(DecorateCustomerUpdate input) {

DecorateCustomer.DecorateCustomerBuilder builder = DecorateCustomer.builder();

DecorateCustomer decorateCustomer = builder.build();

BeanUtils.copyProperties(input, decorateCustomer);

decorateCustomer.setId(input.getCustomerId());

int result = decorateCustomerService.update(decorateCustomer);

if (input.getDecorateType().equals(4)) {

decoratePriceService

.listByCustomerIds(Collections.singleton(input.getCustomerId()))

.stream()

.filter(it -> it.getStatus().equals(4))

.limit(3)

.forEach(

price -> {

DecorateOrder order = decorateOrderService.findOrderByPriceId(price.getId());

if (price.getCompanyId().equals(input.getContractCompany())) {

// 签约成功企业

order.setOrderStatus(5);

order.setContractStatus(0);

log.debug("start update order status, order id :{}", order.getId());

decorateOrderService.update(order);

} else {

// 退款企业

DecorateOrderRefundUpdate update = new DecorateOrderRefundUpdate();

update.setId(order.getId());

update.setOperateType(0);

update.setRefundRejectReason("系统自动退款");

decorateOrderBizService.updateRefund(update, input.getLoginUser().getId(), order);

order.setOrderStatus(6);

order.setContractFee(0 + "");

log.debug("start update order status, order id :{}", order.getId());

decorateOrderService.update(order);

}

});

}

return result;

}

@Override

public int delete(String decorateCustomerId) {

if (Strings.isNullOrEmpty(decorateCustomerId)) {

return 0;

}

Set<Integer> decorateCustomerIds = new HashSet<>();

List<String> list = Stringer.split(decorateCustomerId, ",");

list.forEach(id -> decorateCustomerIds.add(Integer.valueOf(id)));

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

return 0;

}

return decorateCustomerService.delete(decorateCustomerIds);

}

@Override

public DecorateCustomerBO getById(int decorateCustomerId) {

DecorateCustomerBO customer =

decorateCustomerService.getByIdWithProprietorInfo(decorateCustomerId);

if (customer == null) {

return null;

}

if (StringUtils.isNotBlank(customer.getAreaCode())) {

BaseAreaListVO area = baseAreaService.queryAreaByCode(customer.getAreaCode());

if (area != null) {

customer.setAreaName(area.getAreaName());

}

}

if (StringUtils.isNotBlank(customer.getCityCode())) {

BaseAreaListVO city = baseAreaService.queryAreaByCode(customer.getCityCode());

if (city != null) {

customer.setCityName(city.getAreaName());

}

}

// 设置字典数据

setCustomerDictionaryValues(customer);

return customer;

}

private void setCustomerDictionaryValues(DecorateCustomerBO customer) {

if (customer.getDecorateBudget() != null) {

Dictionary dictionary =

dictionaryService.getByTypeAndValue("decorateBudget", customer.getDecorateBudget());

if (dictionary != null) {

customer.setDecorateBudgetStr(dictionary.getName());

}

}

if (customer.getDecorateType() != null) {

Dictionary dictionary =

dictionaryService.getByTypeAndValue("decorateType", customer.getDecorateType());

if (dictionary != null) {

customer.setDecorateTypeStr(dictionary.getName());

}

}

// 装修类型

switch (customer.getDecorateHouseType()) {

case 0:

customer.setDecorateHouseTypeStr("新房装修");

break;

case 1:

customer.setDecorateHouseTypeStr("旧房改造");

break;

default:

break;

}

if (customer.getDecorateStyle() != null) {

Dictionary dictionary =

dictionaryService.getByTypeAndValue("goodStyle", customer.getDecorateStyle());

if (dictionary != null) {

customer.setDecorateStyleStr(dictionary.getName());

}

}

}

@Override

public PageInfo<DecorateCustomerListBO> query(DecorateCustomerQuery query) {

PageInfo<DecorateCustomerListBO> customers = decorateCustomerService.findAllWithBiz(query);

Set<Integer> companyIds = new HashSet<>();

customers

.getList()

.forEach(

it -> {

companyIds.add(it.getCompanyId());

companyIds.add(it.getInnerCompanyId());

companyIds.add(it.getFirstSecKillCompany());

companyIds.add(it.getSecondSecKillCompany());

});

Map<Long, String> companyId2Name = companyService.idAndNameMap(new LinkedList<>(companyIds));

// 设置客户相关信息

setCustomerShowInfo(customers, companyId2Name);

return customers;

}

private void setCustomerShowInfo(

PageInfo<DecorateCustomerListBO> customers, Map<Long, String> companyId2Name) {

// 装修进度

Map<Integer, String> typeMap = dictionaryService.valueAndName2Map("customerDecorateType");

// 获取列表客户报价

List<DecoratePrice> decoratePrices =

decoratePriceService.listByCustomerIds(

customers

.getList()

.stream()

.map(DecorateCustomerListBO::getId)

.collect(Collectors.toSet()));

// 获取所有报价公司信息

Map<Long, String> priceCompanyName =

companyService.idAndNameMap(

decoratePrices

.stream()

.map(DecoratePrice::getCompanyId)

.distinct()

.collect(Collectors.toList()));

for (DecorateCustomerListBO it : customers.getList()) {

//设置业主信息

it.setProprietorInfo((getById(it.getId())));

if (it.getCompanyId() != null) {

it.setCompanyName(companyId2Name.get(it.getCompanyId().longValue()));

}

if (it.getInnerCompanyId() != null) {

it.setInnerCompanyName(companyId2Name.get(it.getInnerCompanyId().longValue()));

}

if (it.getFirstSecKillCompany() != null) {

it.setCommitOrderFunc(true);

it.setFirstSecKillCompanyName(companyId2Name.get(it.getFirstSecKillCompany().longValue()));

}

if (it.getSecondSecKillCompany() != null) {

it.setSecondSecKillCompanyName(

companyId2Name.get(it.getSecondSecKillCompany().longValue()));

}

// 城市

BaseAreaListVO baseArea = baseAreaService.queryAreaByCode(it.getCityCode());

if (baseArea != null) {

it.setCityName(baseArea.getAreaName());

}

// 装修进度

it.setDecorateTypeName(typeMap.get(it.getDecorateType()));

it.setAllPriceList(new LinkedList<>());

it.setChoosePriceList(new LinkedList<>());

// 设置报价企业

decoratePrices

.stream()

.filter(price -> price.getCustomerId().equals(it.getId()))

.forEach(

price -> {

double v = price.getCheckFee() == null ? 0 : price.getCheckFee();

double v1 = price.getDesignFee() == null ? 0 : price.getDesignFee();

double v2 = price.getLabourFee() == null ? 0 : price.getLabourFee();

double v3 = price.getMaterialFee() == null ? 0 : price.getMaterialFee();

double totalFee = v1 + v2 + v3 + v;

DecorateCustomerListBO.SimpleDecoratePriceInfo simplePrice =

DecorateCustomerListBO.SimpleDecoratePriceInfo.builder()

.name(priceCompanyName.get(price.getCompanyId().longValue()))

.price(totalFee / 10000)

.build();

it.getAllPriceList().add(simplePrice);

if (price.getStatus() != null && price.getStatus().equals(4)) {

it.getChoosePriceList().add(simplePrice);

}

});

}

}

@Override

public void dispatchDecoratePrice() {

// 获取未指定

List<DecorateCustomer> dCustomers = decorateCustomerService.findWithNullCompanyId();

// 处理派单

List<Integer> dispatchIds =

dCustomers

.stream()

.filter(

customer -> customer.getDesignPlanId() != null && customer.getDesignPlanId() > 0)

.collect(Collectors.collectingAndThen(Collectors.toList(), this::fixWithDispatchPrice));

log.info("decoratePrice Ids :{}", dispatchIds);

// 处理抢单

List<Integer> sedKillIds =

dCustomers

.stream()

.filter(customer -> customer.getDesignPlanId() == null)

.collect(

Collectors.collectingAndThen(Collectors.toList(), this::fixWithDecorateSedKill));

log.info("sedKill Ids :{}", sedKillIds);

}

// 预约装企客户生成订单

@Override

public List<Integer> initOrderForCompanyCustomer() {

// 获取指定装企的业主

List<DecorateCustomer> oCustomers =

decorateCustomerService.listCustomerWithInnerCompanyAndNoneOrder();

Map<Integer, Double> priceMap =

dictionaryService

.listByType("decorateBudget")

.stream()

.collect(

Collectors.toMap(

Dictionary::getValue, it -> it.getNumAtt3() == null ? 100 : it.getNumAtt3()));

List<Integer> orderIds = new LinkedList<>();

for (DecorateCustomer customer : oCustomers) {

// 生成订单

ProprietorInfo info = proprietorInfoService.getById(customer.getProprietorInfoId());

CompanyShop shop = companyShopService.getById(info.getShopId().intValue());

User user = userService.selectById(shop.getUserId().longValue());

if (user == null) {

log.error("根据店铺获取用户失败,shopId:{},ProprietorInfo id:{}", info.getShopId(), info.getId());

continue;

}

Date date = new Date();

DecorateOrder order =

DecorateOrder.builder()

.customerId(customer.getId())

.proprietorInfoId(customer.getProprietorInfoId())

.companyId(customer.getCompanyId())

.orderType(DecorateOrder.ORDER\_TYPE\_COMPANY)

.orderStatus(2)

.price(0 + "")

.userId(user.getId())

.creator("sanduAdmin")

.modifier("sanduAdmin")

.gmtCreate(date)

.gmtModified(date)

.isDeleted(0)

.build();

int id = decorateOrderService.insert(order);

if (id > 0) {

orderIds.add(order.getId());

// 推送消息

sendMessage("推荐业主", innerSetMessage, user, id);

} else {

log.error("生成订单失败,customerId:{}", customer.getId());

}

}

log.info("auto init order Ids :{}", orderIds);

return orderIds;

}

@Override

public Boolean submitToSedKill(Integer customerId) {

// 验证当前用户是否拥有提交抢单资格

DecorateCustomer customer = decorateCustomerService.getById(customerId);

if (customer.getFirstSeckillCompany() != null && customer.getSecondSeckillCompany() == null) {

List<DecorateSeckill> list =

decorateSecKillService.findSedKillByCustomerIds(Collections.singletonList(customerId));

list.stream()

.filter(

it -> it.getCustomerId().equals(customerId) && it.getFirstSeckillCompany() == null)

.findFirst()

.ifPresent(

it -> {

Date date = new Date();

// update it

it.setFirstSeckillCompany(customer.getFirstSeckillCompany());

// it.setGmtModified(date);

// decorateSecKillService.update(it);

// insert second

DecorateSeckill second = new DecorateSeckill();

BeanUtils.copyProperties(it, second);

second.setId(null);

second.setStatus((byte) 0);

second.setGmtCreate(date);

second.setGmtModified(date);

decorateSecKillService.insertDecorateSedKill(second);

});

return true;

}

return false;

}

private List<Integer> fixWithDispatchPrice(List<DecorateCustomer> list) {

// 过滤出未派单的客户

List<Integer> noneDispatchCustomerIds =

decoratePriceService.findNoneDispatchCustomerIds(

list.stream().map(DecorateCustomer::getId).collect(Collectors.toList()));

List<Integer> resultIds = new LinkedList<>();

list.stream()

.filter(it -> noneDispatchCustomerIds.contains(it.getId()))

.forEach(

customer -> {

List<Integer> companyIds =

decoratePriceService.listCompanyIdForDispatchPrice(

customer.getProprietorInfoId(), 10);

log.info("init decorate price companyIds is ----{}", companyIds);

List<DecoratePrice> insertList = new LinkedList<>();

Date date = new Date();

// Date endDate = DateUtils.addMinutes(date, 4);

Date endDate = DateUtils.addDays(date, 1);

companyIds

.stream()

.distinct()

.forEach(

companyId -> {

insertList.add(

DecoratePrice.builder()

.customerId(customer.getId())

.companyId(companyId)

.proprietorInfoId(customer.getProprietorInfoId())

.price(899 + "")

.status(0)

.startTime(date)

.endTime(endDate)

.creator("sanduAdmin")

.modifier("sanduAdmin")

.gmtCreate(date)

.gmtModified(date)

.build());

});

decoratePriceService.insertList(insertList);

customer.setBidStatus(0);

customer.setBidEndTime(endDate);

decorateCustomerService.update(customer);

insertList.forEach(insert -> resultIds.add(insert.getId()));

// 推送消息

for (DecoratePrice price : insertList) {

HashSet<Integer> companyIdsSet = new HashSet<>();

companyIdsSet.add(price.getCompanyId());

List<User> users = userService.listUserByCompanyIds(companyIdsSet);

users

.stream()

.distinct()

.forEach(

it -> {

System.out.println(it);

sendMessage("系统派单", dispatchMessage, it, price.getId());

});

}

});

return resultIds;

}

private List<Integer> fixWithDecorateSedKill(List<DecorateCustomer> list) {

// 获取未抢单的客户

List<Integer> noneSedKillCustomerIds =

decorateSecKillService.findNoneSedKillCustomerIds(

list.stream().map(DecorateCustomer::getId).collect(Collectors.toList()));

Map<Integer, Double> priceMap =

dictionaryService

.listByType("decorateBudget")

.stream()

.collect(

Collectors.toMap(

Dictionary::getValue, it -> it.getNumAtt3() == null ? 100 : it.getNumAtt3()));

priceMap.put(null, 0D);

Date date = new Date();

List<DecorateSeckill> insertList = new LinkedList<>();

list.stream()

.filter(it -> noneSedKillCustomerIds.contains(it.getId()))

.forEach(

customer -> {

insertList.add(

DecorateSeckill.builder()

.customerId(customer.getId())

.proprietorInfoId(customer.getProprietorInfoId())

.price(priceMap.get(customer.getDecorateBudget()))

.status((byte) 0)

.gmtCreate(date)

.creator("sanduAdmin")

.gmtModified(date)

.modifier("sanduAdmin")

.isDeleted(0)

.build());

});

decorateSecKillService.insertDecorateSeckillList(insertList);

return insertList.stream().map(DecorateSeckill::getId).collect(Collectors.toList());

}

private void sendMessage(String title, String content, User user, Integer bussnessId) {

// save message to db

SysUserMessage message =

decorateOrderBizService.insertSysUserMessage(title, content, -1, bussnessId, user.getId());

// push message

String messageFinal = gson.toJson(message);

log.info("updateRefund开始推送消息,开始时间{}", DateUtil.formatDate(new Date()));

log.info("updateRefund消息体{}", messageFinal);

SocketIOUtil.handlerSendMessage(

SocketIOUtil.IM\_PUSH\_MSG\_EVENT,

SocketIOUtil.IM\_PUSH\_SYSTEM\_MSG\_CODE,

user.getUuid(),

messageFinal);

}

}

package com.sandu.service.decoratecustomer.impl;

import com.github.pagehelper.PageHelper;

import com.github.pagehelper.PageInfo;

import com.sandu.api.decoratecustomer.input.DecorateCustomerQuery;

import com.sandu.api.decoratecustomer.model.DecorateCustomer;

import com.sandu.api.decoratecustomer.model.bo.DecorateCustomerBO;

import com.sandu.api.decoratecustomer.model.bo.DecorateCustomerListBO;

import com.sandu.api.decoratecustomer.service.DecorateCustomerService;

import com.sandu.service.decoratecustomer.dao.DecorateCustomerMapper;

import lombok.extern.slf4j.Slf4j;

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

import org.springframework.stereotype.Service;

import java.util.Collections;

import java.util.List;

import java.util.Objects;

import java.util.Set;

/\*\*

\* CopyRight (c) 2018 Sandu Technology Inc.

\* <p>

\* demo

\*

\* @author sandu <dev@sanduspace.cn>

\* @datetime 2018-Oct-20 14:27

\*/

@Slf4j

@Service("decorateCustomerService")

public class DecorateCustomerServiceImpl implements DecorateCustomerService {

@Autowired

private DecorateCustomerMapper decorateCustomerMapper;

@Override

public int insert(DecorateCustomer decorateCustomer) {

int result = decorateCustomerMapper.insert(decorateCustomer);

if (result > 0) {

return decorateCustomer.getId();

}

return 0;

}

@Override

public int update(DecorateCustomer decorateCustomer) {

return decorateCustomerMapper.updateByPrimaryKey(decorateCustomer);

}

@Override

public int delete(Set<Integer> decorateCustomerIds) {

return decorateCustomerMapper.deleteByPrimaryKey(decorateCustomerIds);

}

@Override

public DecorateCustomer getById(int decorateCustomerId) {

return decorateCustomerMapper.selectByPrimaryKey(decorateCustomerId);

}

@Override

public PageInfo<DecorateCustomerListBO> findAllWithBiz(DecorateCustomerQuery query) {

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

return new PageInfo<>(decorateCustomerMapper.findAll(query));

}

@Override

public List<DecorateCustomer> findByOption(DecorateCustomer query) {

if (Objects.isNull(query)) {

return Collections.emptyList();

}

return decorateCustomerMapper.findByOption(query);

}

@Override

public DecorateCustomerBO getByIdWithProprietorInfo(int decorateCustomerId) {

return decorateCustomerMapper.getByIdWithProprietorInfo(decorateCustomerId);

}

@Override

public void insertList(List<DecorateCustomer> decorateCustomers) {

for (DecorateCustomer decorateCustomer : decorateCustomers) {

//

insert(decorateCustomer);

}

}

@Override

public List<DecorateCustomer> listForDispatchDecoratePrice() {

return decorateCustomerMapper.listForDispatchDecoratePrice();

}

@Override

public List<DecorateCustomer> findWithNullCompanyId() {

return decorateCustomerMapper.findWithNullCompanyId();

}

@Override

public DecorateCustomer getByProprietorInfoId(Integer id) {

return decorateCustomerMapper.getByProprietorInfoId(id);

}

@Override

public List<DecorateCustomer> listCustomerWithInnerCompanyAndNoneOrder() {

return decorateCustomerMapper.listCustomerWithInnerCompanyAndNoneOrder();

}

@Override

public List<DecorateCustomer> getCustomerByProprietorIds(List<Integer> propId) {

if (propId == null || propId.isEmpty()) {

return Collections.emptyList();

}

return decorateCustomerMapper.getCustomerByProprietorIds(propId);

}

}

package com.sandu.service.decoratecustomer.impl;

import com.sandu.api.decoratecustomer.model.DecoratePrice;

import com.sandu.api.decoratecustomer.service.DecoratePriceService;

import com.sandu.service.decoratecustomer.dao.DecoratePriceMapper;

import com.sandu.service.decoratecustomer.dao.DecorateSeckillMapper;

import lombok.extern.slf4j.Slf4j;

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

import org.springframework.stereotype.Service;

import java.util.Collections;

import java.util.List;

import java.util.Set;

import java.util.stream.Collectors;

/\*\*

\* CopyRight (c) 2018 Sandu Technology Inc.

\* <p>

\* decorate

\*

\* @author sandu <dev@sanduspace.cn>

\* @datetime 2018-Oct-23 14:24

\*/

@Slf4j

@Service("decoratePriceService")

public class DecoratePriceServiceImpl implements DecoratePriceService {

@Autowired

private DecoratePriceMapper decoratePriceMapper;

@Autowired

private DecorateSeckillMapper decorateSeckillMapper;

@Override

public int insert(DecoratePrice decoratePrice) {

int result = decoratePriceMapper.insert(decoratePrice);

if (result > 0) {

return decoratePrice.getId();

}

return 0;

}

@Override

public int update(DecoratePrice decoratePrice) {

return decoratePriceMapper.updateByPrimaryKey(decoratePrice);

}

@Override

public int delete(Set<Integer> decoratePriceIds) {

return decoratePriceMapper.deleteByPrimaryKey(decoratePriceIds);

}

@Override

public DecoratePrice getById(int decoratePriceId) {

return decoratePriceMapper.selectByPrimaryKey(decoratePriceId);

}

@Override

public List<DecoratePrice> listByCustomerIds(Set<Integer> ids) {

if (ids == null || ids.isEmpty()) {

return Collections.emptyList();

}

return decoratePriceMapper.listByCustomerIds(ids);

}

@Override

public List<Integer> listCompanyIdForDispatchPrice(Integer proprietorInfoId, int limit) {

if (limit < 1) {

return Collections.emptyList();

}

return decoratePriceMapper.listCompanyIdForDispatchPrice(proprietorInfoId, limit);

}

@Override

public List<Integer> findNoneDispatchCustomerIds(List<Integer> ids) {

if (ids == null || ids.isEmpty()) {

return Collections.emptyList();

}

List<Integer> beDispatchIds = decoratePriceMapper.findDispatchCustomerIds(ids);

return ids.stream().filter(it -> !beDispatchIds.contains(it)).distinct().collect(Collectors.toList());

}

@Override

public void insertList(List<DecoratePrice> installList) {

installList.forEach(this::insert);

}

}

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >

<mapper namespace="com.sandu.service.decoratecustomer.dao.DecorateCustomerMapper">

<resultMap id="BaseResultMap" type="com.sandu.api.decoratecustomer.model.DecorateCustomer">

<id column="id" jdbcType="INTEGER" property="id"/>

<result column="id" jdbcType="INTEGER" property="id"/>

<result column="customer\_type" jdbcType="TINYINT" property="customerType"/>

<result column="proprietor\_info\_id" jdbcType="INTEGER" property="proprietorInfoId"/>

<result column="company\_id" jdbcType="INTEGER" property="companyId"/>

<result column="inner\_company\_id" jdbcType="INTEGER" property="innerCompanyId"/>

<result column="first\_seckill\_company" jdbcType="INTEGER" property="firstSeckillCompany"/>

<result column="second\_seckill\_company" jdbcType="INTEGER" property="secondSeckillCompany"/>

<result column="design\_plan\_id" jdbcType="INTEGER" property="designPlanId"/>

<result column="revisit\_time" jdbcType="VARCHAR" property="revisitTime"/>

<result column="decorate\_type" jdbcType="TINYINT" property="decorateType"/>

<result column="contract\_company" jdbcType="INTEGER" property="contractCompany"/>

<result column="contract\_price" jdbcType="VARCHAR" property="contractPrice"/>

<result column="remark" jdbcType="VARCHAR" property="remark"/>

<result column="creator" jdbcType="VARCHAR" property="creator"/>

<result column="gmt\_create" jdbcType="TIMESTAMP" property="gmtCreate"/>

<result column="modifier" jdbcType="VARCHAR" property="modifier"/>

<result column="gmt\_modified" jdbcType="TIMESTAMP" property="gmtModified"/>

<result column="is\_deleted" jdbcType="INTEGER" property="isDeleted"/>

<result column="decorate\_budget" jdbcType="INTEGER" property="decorateBudget"/>

</resultMap>

<sql id="Base\_Column\_List">

id,customer\_type,proprietor\_info\_id,company\_id,inner\_company\_id,first\_seckill\_company,second\_seckill\_company,design\_plan\_id,revisit\_time,decorate\_type,contract\_company,contract\_price,remark,creator,gmt\_create,modifier,gmt\_modified,is\_deleted

</sql>

<insert id="insert" keyProperty="id" useGeneratedKeys="true"

parameterType="com.sandu.api.decoratecustomer.model.DecorateCustomer">

insert into decorate\_customer

<trim prefix="(" suffix=")" suffixOverrides=",">

<if test="id != null">

id,

</if>

<if test="customerType != null">

customer\_type,

</if>

<if test="proprietorInfoId != null">

proprietor\_info\_id,

</if>

<if test="companyId != null">

company\_id,

</if>

<if test="innerCompanyId != null">

inner\_company\_id,

</if>

<if test="firstSeckillCompany != null">

first\_seckill\_company,

</if>

<if test="secondSeckillCompany != null">

second\_seckill\_company,

</if>

<if test="designPlanId != null">

design\_plan\_id,

</if>

<if test="revisitTime != null">

revisit\_time,

</if>

<if test="decorateType != null">

decorate\_type,

</if>

<if test="contractCompany != null">

contract\_company,

</if>

<if test="contractPrice != null">

contract\_price,

</if>

<if test="remark != null">

remark,

</if>

<if test="creator != null">

creator,

</if>

<if test="gmtCreate != null">

gmt\_create,

</if>

<if test="modifier != null">

modifier,

</if>

<if test="gmtModified != null">

gmt\_modified,

</if>

<if test="isDeleted != null">

is\_deleted,

</if>

<if test="decorateBudget != null">

decorate\_budget,

</if>

</trim>

<trim prefix="values (" suffix=")" suffixOverrides=",">

<if test="customerType != null">

#{customerType,jdbcType=TINYINT},

</if>

<if test="proprietorInfoId != null">

#{proprietorInfoId,jdbcType=INTEGER},

</if>

<if test="companyId != null">

#{companyId,jdbcType=INTEGER},

</if>

<if test="innerCompanyId != null">

#{innerCompanyId,jdbcType=INTEGER},

</if>

<if test="firstSeckillCompany != null">

#{firstSeckillCompany,jdbcType=INTEGER},

</if>

<if test="secondSeckillCompany != null">

#{secondSeckillCompany,jdbcType=INTEGER},

</if>

<if test="designPlanId != null">

#{designPlanId,jdbcType=INTEGER},

</if>

<if test="revisitTime != null">

#{revisitTime,jdbcType=VARCHAR},

</if>

<if test="decorateType != null">

#{decorateType,jdbcType=TINYINT},

</if>

<if test="contractCompany != null">

#{contractCompany,jdbcType=INTEGER},

</if>

<if test="contractPrice != null">

#{contractPrice,jdbcType=VARCHAR},

</if>

<if test="remark != null">

#{remark,jdbcType=VARCHAR},

</if>

<if test="creator != null">

#{creator,jdbcType=VARCHAR},

</if>

<if test="gmtCreate != null">

#{gmtCreate,jdbcType=TIMESTAMP},

</if>

<if test="modifier != null">

#{modifier,jdbcType=VARCHAR},

</if>

<if test="gmtModified != null">

#{gmtModified,jdbcType=TIMESTAMP},

</if>

<if test="isDeleted != null">

#{isDeleted,jdbcType=INTEGER},

</if>

<if test="decorateBudget != null">

#{decorateBudget},

</if>

</trim>

</insert>

<update id="updateByPrimaryKey" parameterType="com.sandu.api.decoratecustomer.model.DecorateCustomer">

update decorate\_customer

<set>

<if test="id != null">

id = #{id,jdbcType=INTEGER},

</if>

<if test="customerType != null">

customer\_type = #{customerType,jdbcType=TINYINT},

</if>

<if test="proprietorInfoId != null">

proprietor\_info\_id = #{proprietorInfoId,jdbcType=INTEGER},

</if>

<if test="companyId != null">

company\_id = #{companyId,jdbcType=INTEGER},

</if>

<if test="innerCompanyId != null">

inner\_company\_id = #{innerCompanyId,jdbcType=INTEGER},

</if>

<if test="firstSeckillCompany != null">

first\_seckill\_company = #{firstSeckillCompany,jdbcType=INTEGER},

</if>

<if test="secondSeckillCompany != null">

second\_seckill\_company = #{secondSeckillCompany,jdbcType=INTEGER},

</if>

<if test="designPlanId != null">

design\_plan\_id = #{designPlanId,jdbcType=INTEGER},

</if>

<if test="revisitTime != null">

revisit\_time = #{revisitTime},

</if>

<if test="decorateType != null">

decorate\_type = #{decorateType,jdbcType=TINYINT},

</if>

<if test="contractCompany != null">

contract\_company = #{contractCompany,jdbcType=INTEGER},

</if>

<if test="contractPrice != null">

contract\_price = #{contractPrice,jdbcType=VARCHAR},

</if>

<if test="remark != null">

remark = #{remark,jdbcType=VARCHAR},

</if>

<if test="creator != null">

creator = #{creator,jdbcType=VARCHAR},

</if>

<if test="gmtCreate != null">

gmt\_create = #{gmtCreate,jdbcType=TIMESTAMP},

</if>

<if test="modifier != null">

modifier = #{modifier,jdbcType=VARCHAR},

</if>

<if test="gmtModified != null">

gmt\_modified = #{gmtModified,jdbcType=TIMESTAMP},

</if>

<if test="isDeleted != null">

is\_deleted = #{isDeleted,jdbcType=INTEGER},

</if>

<if test="decorateBudget != null">

decorate\_budget = #{decorateBudget},

</if>

<if test="bidEndTime != null">

bid\_end\_time = #{bidEndTime},

</if>

<if test="bidStatus != null">

bid\_status = #{bidStatus},

</if>

</set>

where id = #{id,jdbcType=INTEGER}

</update>

<update id="deleteByPrimaryKey" parameterType="java.lang.Integer">

update decorate\_customer set is\_deleted = 1

where id IN

<foreach separator="," open="(" close=")" collection="decorateCustomerIds" item="decorateCustomerId">

#{decorateCustomerId}

</foreach>

</update>

<select id="selectByPrimaryKey" parameterType="java.lang.Integer" resultMap="BaseResultMap">

select

\*

from decorate\_customer

where id = #{decorateCustomerId,jdbcType=INTEGER}

</select>

<select id="findAll" resultType="com.sandu.api.decoratecustomer.model.bo.DecorateCustomerListBO"

parameterType="com.sandu.api.decoratecustomer.input.DecorateCustomerQuery">

select dc.id,dc.proprietor\_info\_id,

dc.company\_id ,dc.first\_seckill\_company,dc.second\_seckill\_company,

dc.inner\_company\_id,dc.decorate\_type,dc.revisit\_time,dc.remark,dc.contract\_company,dc.contract\_price,

pi.user\_name,pi.mobile,pi.city\_code

from decorate\_customer dc,

proprietor\_info pi

where dc.proprietor\_info\_id = pi.id

and dc.is\_deleted = 0

and pi.is\_deleted = 0

<if test="cityCode != null">

and pi.city\_code = #{cityCode}

</if>

<if test="decorateType != null">

and dc.decorate\_Type = #{decorateType}

</if>

<if test="revisitTime != null">

and dc.revisit\_Time = #{revisitTime}

</if>

<if test="userName != null">

and pi.user\_name like concat('%',#{userName},'%')

</if>

<if test="mobile != null">

and pi.mobile like concat('%',#{mobile},'%')

</if>

order by dc.id desc

</select>

<select id="findByOption" resultType="com.sandu.api.decoratecustomer.model.DecorateCustomer">

select \*

from decorate\_customer

where is\_deleted = 0

<if test="id != null">

and id = #{id,jdbcType=INTEGER}

</if>

<if test="customerType != null">

and customer\_type = #{customerType,jdbcType=TINYINT}

</if>

<if test="proprietorInfoId != null">

and proprietor\_info\_id = #{proprietorInfoId,jdbcType=INTEGER}

</if>

<if test="companyId != null">

and company\_id = #{companyId,jdbcType=INTEGER}

</if>

<if test="innerCompanyId != null">

and inner\_company\_id = #{innerCompanyId,jdbcType=INTEGER}

</if>

<if test="firstSeckillCompany != null">

and first\_seckill\_company = #{firstSeckillCompany,jdbcType=INTEGER}

</if>

<if test="secondSeckillCompany != null">

and second\_seckill\_company = #{secondSeckillCompany,jdbcType=INTEGER}

</if>

<if test="designPlanId != null">

and design\_plan\_id = #{designPlanId,jdbcType=INTEGER}

</if>

<if test="revisitTime != null">

and revisit\_time = #{revisitTime,jdbcType=VARCHAR}

</if>

<if test="decorateType != null">

and decorate\_type = #{decorateType,jdbcType=TINYINT}

</if>

<if test="contractCompany != null">

and contract\_company = #{contractCompany,jdbcType=INTEGER}

</if>

<if test="contractPrice != null">

and contract\_price = #{contractPrice,jdbcType=VARCHAR}

</if>

<if test="remark != null">

and remark = #{remark,jdbcType=VARCHAR}

</if>

<if test="creator != null">

and creator = #{creator,jdbcType=VARCHAR}

</if>

<if test="gmtCreate != null">

and gmt\_create = #{gmtCreate,jdbcType=TIMESTAMP}

</if>

<if test="modifier != null">

and modifier = #{modifier,jdbcType=VARCHAR}

</if>

<if test="gmtModified != null">

and gmt\_modified = #{gmtModified,jdbcType=TIMESTAMP}

</if>

</select>

<select id="getByIdWithProprietorInfo"

resultType="com.sandu.api.decoratecustomer.model.bo.DecorateCustomerBO">

select dc.id,dc.proprietor\_info\_id,

pi.user\_name,

pi.area\_name,

pi.address,

pi.house\_acreage,

concat(pi.bedroom\_num,'室',pi.living\_room\_num,'厅',pi.toilet\_num,'卫') as houseType,

pi.decorate\_budget,

pi.decorate\_house\_type,

pi.decorate\_type ,

pi.decorate\_style,

pi.city\_code

from decorate\_customer dc inner join proprietor\_info pi on dc.proprietor\_info\_id = pi.id

where

dc.is\_deleted = 0

and pi.is\_deleted = 0

and dc.id = #{decorateCustomerId}

</select>

<select id="listForDispatchDecoratePrice"

resultType="com.sandu.api.decoratecustomer.model.DecorateCustomer">

</select>

<select id="findWithNullCompanyId" resultType="com.sandu.api.decoratecustomer.model.DecorateCustomer">

select \* from decorate\_customer where company\_id is null ;

</select>

<select id="getByProprietorInfoId" parameterType="java.lang.Integer" resultMap="BaseResultMap">

select

<include refid="Base\_Column\_List"/>

from decorate\_customer

where proprietor\_info\_id = #{proprietorInfoId,jdbcType=INTEGER}

</select>

<select id="listCustomerWithInnerCompanyAndNoneOrder"

resultType="com.sandu.api.decoratecustomer.model.DecorateCustomer">

select dc.\*

from decorate\_customer dc

where is\_deleted = 0

and dc.company\_id > 0

and not exists(select 1 from decorate\_order do where dc.id = do.customer\_id and dc.company\_id = do.company\_id and do.order\_type = 0 );

</select>

<select id="getCustomerByProprietorIds" resultType="com.sandu.api.decoratecustomer.model.DecorateCustomer">

select \* from decorate\_customer

where is\_deleted = 0

and proprietor\_info\_id in

<foreach collection="propIds" item="propId" separator="," close=")" open="(">

#{propId}

</foreach>

</select>

</mapper>

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >

<mapper namespace="com.sandu.service.decoratecustomer.dao.DecoratePriceMapper">

<resultMap id="BaseResultMap" type="com.sandu.api.decoratecustomer.model.DecoratePrice">

<id column="id" jdbcType="INTEGER" property="id"/>

<result column="id" jdbcType="INTEGER" property="id"/>

<result column="company\_id" jdbcType="INTEGER" property="companyId"/>

<result column="customer\_id" jdbcType="INTEGER" property="customerId"/>

<result column="proprietor\_info\_id" jdbcType="INTEGER" property="proprietorInfoId"/>

<result column="status" jdbcType="TINYINT" property="status"/>

<result column="bid\_user\_id" jdbcType="INTEGER" property="bidUserId"/>

<result column="material\_fee" jdbcType="VARCHAR" property="materialFee"/>

<result column="check\_fee" jdbcType="VARCHAR" property="checkFee"/>

<result column="labour\_fee" jdbcType="VARCHAR" property="labourFee"/>

<result column="design\_fee" jdbcType="VARCHAR" property="designFee"/>

<result column="submit\_time" jdbcType="TIMESTAMP" property="submitTime"/>

<result column="start\_time" jdbcType="TIMESTAMP" property="startTime"/>

<result column="end\_time" jdbcType="TIMESTAMP" property="endTime"/>

<result column="creator" jdbcType="VARCHAR" property="creator"/>

<result column="gmt\_create" jdbcType="TIMESTAMP" property="gmtCreate"/>

<result column="modifier" jdbcType="VARCHAR" property="modifier"/>

<result column="gmt\_modified" jdbcType="TIMESTAMP" property="gmtModified"/>

<result column="is\_deleted" jdbcType="INTEGER" property="isDeleted"/>

<result column="price" jdbcType="INTEGER" property="price"/>

</resultMap>

<sql id="Base\_Column\_List">

id,company\_id,customer\_id,proprietor\_info\_id,status,bid\_user\_id,material\_fee,check\_fee,labour\_fee,

design\_fee,submit\_time,start\_time,end\_time,creator,gmt\_create,modifier,gmt\_modified,is\_deleted,price

</sql>

<insert id="insert" keyProperty="id" useGeneratedKeys="true"

parameterType="com.sandu.api.decoratecustomer.model.DecoratePrice">

insert into decorate\_price

<trim prefix="(" suffix=")" suffixOverrides=",">

<if test="id != null">

id,

</if>

<if test="companyId != null">

company\_id,

</if>

<if test="customerId != null">

customer\_id,

</if>

<if test="proprietorInfoId != null">

proprietor\_info\_id,

</if>

<if test="status != null">

status,

</if>

<if test="bidUserId != null">

bid\_user\_id,

</if>

<if test="materialFee != null">

material\_fee,

</if>

<if test="checkFee != null">

check\_fee,

</if>

<if test="labourFee != null">

labour\_fee,

</if>

<if test="designFee != null">

design\_fee,

</if>

<if test="submitTime != null">

submit\_time,

</if>

<if test="startTime != null">

start\_time,

</if>

<if test="endTime != null">

end\_time,

</if>

<if test="creator != null">

creator,

</if>

<if test="gmtCreate != null">

gmt\_create,

</if>

<if test="modifier != null">

modifier,

</if>

<if test="gmtModified != null">

gmt\_modified,

</if>

<if test="isDeleted != null">

is\_deleted,

</if>

<if test="price != null">

price,

</if>

</trim>

<trim prefix="values (" suffix=")" suffixOverrides=",">

<if test="id != null">

#{id,jdbcType=INTEGER},

</if>

<if test="companyId != null">

#{companyId,jdbcType=INTEGER},

</if>

<if test="customerId != null">

#{customerId,jdbcType=INTEGER},

</if>

<if test="proprietorInfoId != null">

#{proprietorInfoId,jdbcType=INTEGER},

</if>

<if test="status != null">

#{status,jdbcType=TINYINT},

</if>

<if test="bidUserId != null">

#{bidUserId,jdbcType=INTEGER},

</if>

<if test="materialFee != null">

#{materialFee,jdbcType=VARCHAR},

</if>

<if test="checkFee != null">

#{checkFee,jdbcType=VARCHAR},

</if>

<if test="labourFee != null">

#{labourFee,jdbcType=VARCHAR},

</if>

<if test="designFee != null">

#{designFee,jdbcType=VARCHAR},

</if>

<if test="submitTime != null">

#{submitTime,jdbcType=TIMESTAMP},

</if>

<if test="startTime != null">

#{startTime,jdbcType=TIMESTAMP},

</if>

<if test="endTime != null">

#{endTime,jdbcType=TIMESTAMP},

</if>

<if test="creator != null">

#{creator,jdbcType=VARCHAR},

</if>

<if test="gmtCreate != null">

#{gmtCreate,jdbcType=TIMESTAMP},

</if>

<if test="modifier != null">

#{modifier,jdbcType=VARCHAR},

</if>

<if test="gmtModified != null">

#{gmtModified,jdbcType=TIMESTAMP},

</if>

<if test="isDeleted != null">

#{isDeleted,jdbcType=INTEGER},

</if>

<if test="price != null">

#{price},

</if>

</trim>

</insert>

<update id="updateByPrimaryKey" parameterType="com.sandu.api.decoratecustomer.model.DecoratePrice">

update decorate\_price

<set>

<if test="id != null">

id = #{id,jdbcType=INTEGER},

</if>

<if test="companyId != null">

company\_id = #{companyId,jdbcType=INTEGER},

</if>

<if test="customerId != null">

customer\_id = #{customerId,jdbcType=INTEGER},

</if>

<if test="proprietorInfoId != null">

proprietor\_info\_id = #{proprietorInfoId,jdbcType=INTEGER},

</if>

<if test="status != null">

status = #{status,jdbcType=TINYINT},

</if>

<if test="bidUserId != null">

bid\_user\_id = #{bidUserId,jdbcType=INTEGER},

</if>

<if test="materialFee != null">

material\_fee = #{materialFee,jdbcType=VARCHAR},

</if>

<if test="checkFee != null">

check\_fee = #{checkFee,jdbcType=VARCHAR},

</if>

<if test="labourFee != null">

labour\_fee = #{labourFee,jdbcType=VARCHAR},

</if>

<if test="designFee != null">

design\_fee = #{designFee,jdbcType=VARCHAR},

</if>

<if test="submitTime != null">

submit\_time = #{submitTime,jdbcType=TIMESTAMP},

</if>

<if test="startTime != null">

start\_time = #{startTime,jdbcType=TIMESTAMP},

</if>

<if test="endTime != null">

end\_time = #{endTime,jdbcType=TIMESTAMP},

</if>

<if test="creator != null">

creator = #{creator,jdbcType=VARCHAR},

</if>

<if test="gmtCreate != null">

gmt\_create = #{gmtCreate,jdbcType=TIMESTAMP},

</if>

<if test="modifier != null">

modifier = #{modifier,jdbcType=VARCHAR},

</if>

<if test="gmtModified != null">

gmt\_modified = #{gmtModified,jdbcType=TIMESTAMP},

</if>

<if test="isDeleted != null">

is\_deleted = #{isDeleted,jdbcType=INTEGER},

</if>

</set>

where id = #{id,jdbcType=INTEGER}

</update>

<update id="deleteByPrimaryKey" parameterType="java.lang.Integer">

update decorate\_price set is\_deleted = 1

where id IN

<foreach separator="," open="(" close=")" collection="decoratePriceIds" item="decoratePriceId">

#{decoratePriceId}

</foreach>

</update>

<select id="selectByPrimaryKey" parameterType="java.lang.Integer" resultMap="BaseResultMap">

select

<include refid="Base\_Column\_List"/>

from decorate\_price

where id = #{decoratePriceId,jdbcType=INTEGER}

</select>

<select id="listByCustomerIds" resultType="com.sandu.api.decoratecustomer.model.DecoratePrice">

select

<include refid="Base\_Column\_List"/>

from decorate\_price

where is\_deleted = 0 and customer\_id in

<foreach collection="ids" item="id" open="(" close=")" separator=",">

#{id}

</foreach>

</select>

<select id="listCompanyIdForDispatchPrice" resultType="java.lang.Integer">

select bc.id from base\_company bc

inner join proprietor\_info pi on bc.city\_code =pi.city\_code

left join decorate\_price dp on bc.id = dp.company\_id

where bc.business\_type = 5 and bc.is\_deleted = 0

and bc.is\_receive\_platform\_dispatch = 1

and pi.id = #{proprietorInfoId}

group by bc.id order by id,count(1)

limit #{limit}

</select>

<select id="findDispatchCustomerIds" resultType="java.lang.Integer">

select distinct customer\_id from decorate\_price

where is\_deleted = 0 and

customer\_id in

<foreach collection="ids" item="id" separator="," open="(" close=")">

#{id}

</foreach>

</select>

</mapper>

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >

<mapper namespace="com.sandu.service.decoratecustomer.dao.DecorateSeckillMapper">

<resultMap id="BaseResultMap" type="com.sandu.api.decoratecustomer.model.DecorateSeckill">

<id column="id" property="id" jdbcType="BIGINT"/>

<result column="customer\_id" property="customerId" jdbcType="BIGINT"/>

<result column="first\_seckill\_company" property="firstSeckillCompany" jdbcType="BIGINT"/>

<result column="user\_id" property="userId" jdbcType="BIGINT"/>

<result column="proprietor\_info\_id" property="proprietorInfoId" jdbcType="INTEGER"/>

<result column="status" property="status" jdbcType="TINYINT"/>

<result column="price" property="price" jdbcType="DECIMAL"/>

<result column="creator" property="creator" jdbcType="VARCHAR"/>

<result column="gmt\_create" property="gmtCreate" jdbcType="TIMESTAMP"/>

<result column="modifier" property="modifier" jdbcType="VARCHAR"/>

<result column="gmt\_modified" property="gmtModified" jdbcType="TIMESTAMP"/>

<result column="is\_deleted" property="isDeleted" jdbcType="INTEGER"/>

</resultMap>

<sql id="Base\_Column\_List">

id, customer\_id, first\_seckill\_company, user\_id, proprietor\_info\_id, status, price,

creator, gmt\_create, modifier, gmt\_modified, is\_deleted

</sql>

<select id="selectByPrimaryKey" resultMap="BaseResultMap" parameterType="java.lang.Long">

select

<include refid="Base\_Column\_List"/>

from decorate\_seckill

where id = #{id,jdbcType=BIGINT}

</select>

<delete id="deleteByPrimaryKey" parameterType="java.lang.Long">

delete from decorate\_seckill

where id = #{id,jdbcType=BIGINT}

</delete>

<insert id="insert" parameterType="com.sandu.api.decoratecustomer.model.DecorateSeckill">

insert into decorate\_seckill (id, customer\_id, first\_seckill\_company,

user\_id, proprietor\_info\_id, status,

price, creator, gmt\_create,

modifier, gmt\_modified, is\_deleted

)

values (#{id,jdbcType=BIGINT}, #{customerId,jdbcType=BIGINT}, #{firstSeckillCompany,jdbcType=BIGINT},

#{userId,jdbcType=BIGINT}, #{proprietorInfoId,jdbcType=INTEGER}, #{status,jdbcType=TINYINT},

#{price,jdbcType=DECIMAL}, #{creator,jdbcType=VARCHAR}, #{gmtCreate,jdbcType=TIMESTAMP},

#{modifier,jdbcType=VARCHAR}, #{gmtModified,jdbcType=TIMESTAMP}, #{isDeleted,jdbcType=INTEGER}

)

</insert>

<insert id="insertSelective" parameterType="com.sandu.api.decoratecustomer.model.DecorateSeckill">

insert into decorate\_seckill

<trim prefix="(" suffix=")" suffixOverrides=",">

<if test="id != null">

id,

</if>

<if test="customerId != null">

customer\_id,

</if>

<if test="firstSeckillCompany != null">

first\_seckill\_company,

</if>

<if test="userId != null">

user\_id,

</if>

<if test="proprietorInfoId != null">

proprietor\_info\_id,

</if>

<if test="status != null">

status,

</if>

<if test="price != null">

price,

</if>

<if test="creator != null">

creator,

</if>

<if test="gmtCreate != null">

gmt\_create,

</if>

<if test="modifier != null">

modifier,

</if>

<if test="gmtModified != null">

gmt\_modified,

</if>

<if test="isDeleted != null">

is\_deleted,

</if>

</trim>

<trim prefix="values (" suffix=")" suffixOverrides=",">

<if test="id != null">

#{id,jdbcType=BIGINT},

</if>

<if test="customerId != null">

#{customerId,jdbcType=BIGINT},

</if>

<if test="firstSeckillCompany != null">

#{firstSeckillCompany,jdbcType=BIGINT},

</if>

<if test="userId != null">

#{userId,jdbcType=BIGINT},

</if>

<if test="proprietorInfoId != null">

#{proprietorInfoId,jdbcType=INTEGER},

</if>

<if test="status != null">

#{status,jdbcType=TINYINT},

</if>

<if test="price != null">

#{price,jdbcType=DECIMAL},

</if>

<if test="creator != null">

#{creator,jdbcType=VARCHAR},

</if>

<if test="gmtCreate != null">

#{gmtCreate,jdbcType=TIMESTAMP},

</if>

<if test="modifier != null">

#{modifier,jdbcType=VARCHAR},

</if>

<if test="gmtModified != null">

#{gmtModified,jdbcType=TIMESTAMP},

</if>

<if test="isDeleted != null">

#{isDeleted,jdbcType=INTEGER},

</if>

</trim>

</insert>

<update id="updateByPrimaryKeySelective" parameterType="com.sandu.api.decoratecustomer.model.DecorateSeckill">

update decorate\_seckill

<set>

<if test="customerId != null">

customer\_id = #{customerId,jdbcType=BIGINT},

</if>

<if test="firstSeckillCompany != null">

first\_seckill\_company = #{firstSeckillCompany,jdbcType=BIGINT},

</if>

<if test="userId != null">

user\_id = #{userId,jdbcType=BIGINT},

</if>

<if test="proprietorInfoId != null">

proprietor\_info\_id = #{proprietorInfoId,jdbcType=INTEGER},

</if>

<if test="status != null">

status = #{status,jdbcType=TINYINT},

</if>

<if test="price != null">

price = #{price,jdbcType=DECIMAL},

</if>

<if test="creator != null">

creator = #{creator,jdbcType=VARCHAR},

</if>

<if test="gmtCreate != null">

gmt\_create = #{gmtCreate,jdbcType=TIMESTAMP},

</if>

<if test="modifier != null">

modifier = #{modifier,jdbcType=VARCHAR},

</if>

<if test="gmtModified != null">

gmt\_modified = #{gmtModified,jdbcType=TIMESTAMP},

</if>

<if test="isDeleted != null">

is\_deleted = #{isDeleted,jdbcType=INTEGER},

</if>

</set>

where id = #{id,jdbcType=BIGINT}

</update>

<update id="updateByPrimaryKey" parameterType="com.sandu.api.decoratecustomer.model.DecorateSeckill">

update decorate\_seckill

set customer\_id = #{customerId,jdbcType=BIGINT},

first\_seckill\_company = #{firstSeckillCompany,jdbcType=BIGINT},

user\_id = #{userId,jdbcType=BIGINT},

proprietor\_info\_id = #{proprietorInfoId,jdbcType=INTEGER},

status = #{status,jdbcType=TINYINT},

price = #{price,jdbcType=DECIMAL},

creator = #{creator,jdbcType=VARCHAR},

gmt\_create = #{gmtCreate,jdbcType=TIMESTAMP},

modifier = #{modifier,jdbcType=VARCHAR},

gmt\_modified = #{gmtModified,jdbcType=TIMESTAMP},

is\_deleted = #{isDeleted,jdbcType=INTEGER}

where id = #{id,jdbcType=BIGINT}

</update>

<select id="findSedKillCustomerIds" resultType="java.lang.Integer">

select distinct customer\_id from decorate\_seckill

where is\_deleted = 0 and

customer\_id in

<foreach collection="ids" item="id" separator="," open="(" close=")">

#{id}

</foreach>

</select>

<select id="findSedKillByCustomerIds" resultType="com.sandu.api.decoratecustomer.model.DecorateSeckill">

select \* from decorate\_seckill

where is\_deleted = 0

and customer\_id in

<foreach collection="ids" item="id" separator="," open="(" close=")">

#{id}

</foreach>

</select>

</mapper>

package com.sandu.web.decoratecustomer.controller;

import com.github.pagehelper.PageInfo;

import com.google.common.collect.Lists;

import com.sandu.api.decoratecustomer.input.DecorateCustomerAdd;

import com.sandu.api.decoratecustomer.input.DecorateCustomerQuery;

import com.sandu.api.decoratecustomer.input.DecorateCustomerUpdate;

import com.sandu.api.decoratecustomer.model.bo.DecorateCustomerBO;

import com.sandu.api.decoratecustomer.model.bo.DecorateCustomerListBO;

import com.sandu.api.decoratecustomer.output.DecorateCustomerListVO;

import com.sandu.api.decoratecustomer.output.DecorateCustomerVO;

import com.sandu.api.decoratecustomer.service.biz.DecorateCustomerBizService;

import com.sandu.api.decorateorder.input.DecorateOrderAdd;

import com.sandu.api.decorateorder.service.biz.DecorateOrderBizService;

import com.sandu.authz.annotation.RequiresPermissions;

import com.sandu.common.BaseController;

import com.sandu.common.LoginContext;

import com.sandu.common.ReturnData;

import com.sandu.commons.LoginUser;

import com.sandu.constant.ResponseEnum;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiImplicitParam;

import io.swagger.annotations.ApiImplicitParams;

import io.swagger.annotations.ApiOperation;

import lombok.extern.slf4j.Slf4j;

import org.apache.commons.lang3.time.DateUtils;

import org.springframework.beans.BeanUtils;

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

import org.springframework.validation.BindingResult;

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

import javax.validation.Valid;

import java.util.List;

/\*\*

\* CopyRight (c) 2018 Sandu Technology Inc.

\*

\* <p>demo

\*

\* @author sandu <dev@sanduspace.cn>

\* @datetime 2018-Oct-20 14:27

\*/

@Api(value = "DecorateCustomer", tags = "decorateCustomer", description = "抢单派单客户管理")

@RestController

@RequestMapping(value = "/v1/decorateCustomer")

@Slf4j

public class DecorateCustomerController extends BaseController {

@Autowired

private DecorateCustomerBizService decorateCustomerBizService;

@Autowired

private DecorateOrderBizService decorateOrderBizService;

@ApiOperation(value = "新建decorateCustomer", response = ReturnData.class)

// @PostMapping

public ReturnData createDecorateCustomer(

@Valid @RequestBody DecorateCustomerAdd input, BindingResult validResult) {

ReturnData data = ReturnData.builder();

if (validResult.hasErrors()) {

return processValidError(validResult, data);

}

int decorateCustomerId = decorateCustomerBizService.create(input);

if (decorateCustomerId > 0) {

return data.code(ResponseEnum.CREATED).data(decorateCustomerId);

}

return data.code(ResponseEnum.ERROR).message("输入数据有误");

}

@RequiresPermissions({"decorate:decorateCustomer:edit"})

@ApiOperation(value = "编辑decorateCustomer", response = ReturnData.class)

@PutMapping

public ReturnData updateDecorateCustomer(

@Valid @RequestBody DecorateCustomerUpdate input, BindingResult validResult) {

ReturnData data = ReturnData.builder();

if (validResult.hasErrors()) {

return processValidError(validResult, data);

}

input.setLoginUser(LoginContext.getLoginUser(LoginUser.class));

if (input.getDecorateType() == 4

&& (input.getContractCompany() == null || input.getContractPrice() == null)) {

log.info("input param :{}", input);

return data.code(ResponseEnum.ERROR).message("输入数据有误");

}

int result = decorateCustomerBizService.update(input);

if (result > 0) {

return data.code(ResponseEnum.SUCCESS).success(true).message("操作成功");

}

return data.code(ResponseEnum.ERROR).message("输入数据有误");

}

@ApiOperation(value = "删除decorateCustomer", response = ReturnData.class)

// @DeleteMapping

public ReturnData deleteDecorateCustomer(String decorateCustomerId) {

ReturnData data = ReturnData.builder();

int result = decorateCustomerBizService.delete(decorateCustomerId);

if (result > 0) {

return data.code(ResponseEnum.SUCCESS).data(result);

}

return data.code(ResponseEnum.ERROR).message("不存在");

}

@RequiresPermissions({"decorate:decorateCustomer:view"})

@ApiOperation(value = "获取decorateCustomer详情", response = DecorateCustomerVO.class)

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

public ReturnData getByDecorateCustomerId(@PathVariable int decorateCustomerId) {

ReturnData data = ReturnData.builder();

if (decorateCustomerId <= 0) {

return data.code(ResponseEnum.ERROR).message("ID无效");

}

DecorateCustomerBO decorateCustomer = decorateCustomerBizService.getById(decorateCustomerId);

if (decorateCustomer == null) {

return data.code(ResponseEnum.ERROR).message("decorateCustomer不存在");

}

DecorateCustomerVO output = new DecorateCustomerVO();

BeanUtils.copyProperties(decorateCustomer, output);

// 原字段ID转模块ID

// output.setDecorateCustomerId(decorateCustomer.getId());

return data.code(ResponseEnum.SUCCESS).data(output);

}

@RequiresPermissions({"decorate:decorateCustomer:view"})

@ApiOperation(value = "查询decorateCustomer列表", response = DecorateCustomerListVO.class)

@ApiImplicitParams({

@ApiImplicitParam(name = "page", value = "页码", paramType = "query", dataType = "Integer"),

@ApiImplicitParam(name = "limit", value = "每页数量", paramType = "query", dataType = "Integer")

})

@GetMapping(value = "/list")

public ReturnData queryDecorateCustomerList(

@Valid DecorateCustomerQuery query, BindingResult validResult) {

ReturnData data = ReturnData.builder();

if (validResult.hasErrors()) {

return processValidError(validResult, data);

}

final PageInfo<DecorateCustomerListBO> results = decorateCustomerBizService.query(query);

final List<DecorateCustomerListVO> decorateCustomers = Lists.newArrayList();

if (results != null && results.getTotal() > 0) {

results

.getList()

.forEach(

decorateCustomer -> {

DecorateCustomerListVO output = new DecorateCustomerListVO();

BeanUtils.copyProperties(decorateCustomer, output);

if (output.getRevisitTime() != null) {

output.setRevisitTime(DateUtils.addDays(output.getRevisitTime(), 1));

}

decorateCustomers.add(output);

});

return data.code(ResponseEnum.SUCCESS).list(decorateCustomers).total(results.getTotal());

}

return data.code(ResponseEnum.SUCCESS).message("暂无数据");

}

@RequiresPermissions({"decorate:decorateCustomer:manage"})

@ApiOperation("提交抢单")

@GetMapping("submit/sedkill/{customerId}")

public ReturnData submitSedKill(@PathVariable("customerId") Integer customerId) {

Boolean ret = decorateCustomerBizService.submitToSedKill(customerId);

String message = ret ? "提交成功" : "提交失败";

return ReturnData.builder().success(ret).code(ResponseEnum.SUCCESS).message(message);

}

@RequiresPermissions({"decorate:decorateCustomer:manage"})

@ApiOperation("内部推荐")

@GetMapping("inner/recommend/{customerId}/{companyId}")

public ReturnData innerRecommend(

@PathVariable("customerId") Integer customerId,

@PathVariable("companyId") Integer companyId) {

LoginUser loginUser = LoginContext.getLoginUser(LoginUser.class);

// LoginUser loginUser = null;

// loginUser = new LoginUser();

// loginUser.setId(123);

DecorateOrderAdd add = new DecorateOrderAdd();

add.setCompanyId(companyId);

add.setCustomerId(customerId);

add.setUser(loginUser);

try {

decorateOrderBizService.create(add);

} catch (Exception e) {

log.error(e.getMessage());

return ReturnData.builder().code(ResponseEnum.ERROR).success(false).message("指派失败");

}

return ReturnData.builder().code(ResponseEnum.SUCCESS).success(true).message("指派成功");

}

@GetMapping("test")

public void testDispatch() {

decorateCustomerBizService.dispatchDecoratePrice();

}

}