package com.chinamcom.framework.products.services.impl;

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.chinamcom.framework.common.response.RespCode;

import com.chinamcom.framework.common.service.BaseService;

import com.chinamcom.framework.common.util.StringUtil;

import com.chinamcom.framework.products.mapper.TbActivityMapper;

import com.chinamcom.framework.products.mapper.TbProductColorMapper;

import com.chinamcom.framework.products.mapper.TbProductDevicesMapper;

import com.chinamcom.framework.products.mapper.TbProductImageboxMapper;

import com.chinamcom.framework.products.mapper.TbProductInfoMapper;

import com.chinamcom.framework.products.mapper.TbProductModelMapper;

import com.chinamcom.framework.products.mapper.TbProductStockMapper;

import com.chinamcom.framework.products.mapper.TbProductTypeMapper;

import com.chinamcom.framework.products.model.TbActivity;

import com.chinamcom.framework.products.model.TbActivityExample;

import com.chinamcom.framework.products.model.TbProductColor;

import com.chinamcom.framework.products.model.TbProductColorExample;

import com.chinamcom.framework.products.model.TbProductDevices;

import com.chinamcom.framework.products.model.TbProductDevicesExample;

import com.chinamcom.framework.products.model.TbProductImagebox;

import com.chinamcom.framework.products.model.TbProductImageboxExample;

import com.chinamcom.framework.products.model.TbProductInfo;

import com.chinamcom.framework.products.model.TbProductInfoExample;

import com.chinamcom.framework.products.model.TbProductModel;

import com.chinamcom.framework.products.model.TbProductModelExample;

import com.chinamcom.framework.products.model.TbProductStock;

import com.chinamcom.framework.products.model.TbProductStockExample;

import com.chinamcom.framework.products.model.TbProductType;

import com.chinamcom.framework.products.model.TbProductTypeExample;

import com.chinamcom.framework.products.services.ITbProductInfoService;

import io.vertx.core.json.JsonArray;

import io.vertx.core.json.JsonObject;

@Service("tbProductInfoService")

public class TbProductInfoServiceImpl extends BaseService implements ITbProductInfoService {

@Autowired

private TbActivityMapper activityMapper;

@Autowired

private TbProductInfoMapper productInfoMapper;

@Autowired

private TbProductDevicesMapper productDevicesMapper;

@Autowired

private TbProductImageboxMapper productImageboxMapper;

@Autowired

private TbProductStockMapper productStockMapper;

@Autowired

private TbProductTypeMapper productTypeMapper;

@Autowired

private TbProductModelMapper productModelMapper;

@Autowired

private TbProductColorMapper productColorMapper;

@Override

public String selectTbProductTypeList(String sn, JsonObject param) throws Exception {

String message = respWriter.toError(sn);

JsonArray result = new JsonArray();

try {

Optional<Integer> prodtype = Optional.ofNullable(param.getInteger("prodtype"));// 产品类别40耳机41手表

TbProductTypeExample typeExample = new TbProductTypeExample();

TbProductTypeExample.Criteria typeCriteria = typeExample.createCriteria();

if(prodtype.isPresent()){

typeCriteria.andTypeCodeEqualTo(prodtype.get().toString());

}

List<TbProductType> lTbProductType = productTypeMapper.selectByExample(typeExample);

if (!lTbProductType.isEmpty()) {

lTbProductType.forEach(tpt->{

JsonObject type = new JsonObject();

type.put("code", tpt.getTypeCode());

type.put("name", tpt.getTypeName());

TbProductModelExample modelExample = new TbProductModelExample();

modelExample.createCriteria().andTypeCodeEqualTo(tpt.getTypeCode());

List<TbProductModel> lTbProductModel = productModelMapper.selectByExample(modelExample);

JsonArray model = new JsonArray();

if (!lTbProductModel.isEmpty()) {

lTbProductModel.forEach(tpm -> {

JsonObject \_model = new JsonObject();

\_model.put("code", tpm.getModelCode());

\_model.put("name", tpm.getModelName());

TbProductColorExample colorExample = new TbProductColorExample();

colorExample.createCriteria().andModelCodeEqualTo(tpm.getModelCode()).andTypeCodeEqualTo(tpt.getTypeCode());

List<TbProductColor> lTbProductColor = productColorMapper.selectByExample(colorExample);

JsonArray color = new JsonArray();

if(!lTbProductColor.isEmpty()){

lTbProductColor.forEach(tpc->{

JsonObject \_color = new JsonObject();

\_color.put("code", tpc.getColorCode());

\_color.put("name", tpc.getColorName());

color.add(\_color);

});

}

\_model.put("color", color);

model.add(\_model);

});

}

type.put("model", model);

result.add(type);

});

}

message = respWriter.toSuccess(result, sn);

} catch (Exception e) {

throw e;

}

return message;

}

@Override

public String selectTbProductInfoList(String sn, JsonObject param) {

String message = respWriter.toError(sn);

try {

Optional<String> prodcode = Optional.ofNullable(param.getString("prodcode"));// 产品编号

Optional<Integer> prodid = Optional.ofNullable(param.getInteger("prodid"));// 产品ID

Optional<Integer> prodtype = Optional.ofNullable(param.getInteger("prodtype"));// 产品类别40耳机41手表

TbProductInfoExample piExample = new TbProductInfoExample();

TbProductInfoExample.Criteria criteria = piExample.createCriteria();

criteria.andStatusEqualTo(1);

if (prodcode.isPresent() || prodid.isPresent()) {// 查询某个产品信息

if (prodcode.isPresent()) {

criteria.andProdcodeEqualTo(prodcode.get());

}

if (prodid.isPresent()) {

criteria.andIdEqualTo(prodid.get());

}

}

if(prodtype.isPresent()){

criteria.andProdtypeEqualTo(prodtype.get());

}

List<TbProductInfo> pinfos = productInfoMapper.selectByExample(piExample);

if (!pinfos.isEmpty()) {

pinfos.forEach(prod -> {

// 判断产品是否有活动信息

if (prod.getActivity() == 1) {

TbActivityExample activity = new TbActivityExample();

activity.createCriteria().andProdcodeEqualTo(prod.getProdcode()).andProdidEqualTo(prod.getId());

List<TbActivity> activityinfo = activityMapper.selectByExample(activity);

if (!activityinfo.isEmpty()) {

prod.setActivityPrice(activityinfo.get(0).getPrice());

prod.setBeginTime(activityinfo.get(0).getBeginTime());

prod.setEndTime(activityinfo.get(0).getEndTime());

prod.setActivityDesc(activityinfo.get(0).getDesc());

}

}

prod.setModel(StringUtil.subModelFromCode(prod.getProdcode()));

prod.setColor(StringUtil.subColorFromCode(prod.getProdcode()));

prod.setColorname(StringUtil.getColorName(prod.getColor()));

// 检索产品的库存信息

TbProductStockExample stockExample = new TbProductStockExample();

stockExample.createCriteria().andProdidEqualTo(prod.getId()).andProdcodeEqualTo(prod.getProdcode());

List<TbProductStock> stockinfo = productStockMapper.selectByExample(stockExample);

if (!stockinfo.isEmpty()) {

prod.setStock(stockinfo.get(0).getStock());

prod.setWarning(stockinfo.get(0).getWarning());

}

// 检索产品展示图片信息

TbProductImageboxExample imageExample = new TbProductImageboxExample();

imageExample.createCriteria().andProdidEqualTo(prod.getId()).andProdcodeEqualTo(prod.getProdcode()).andTypeEqualTo(2);

List<TbProductImagebox> imageinfo = productImageboxMapper.selectByExample(imageExample);

if (!imageinfo.isEmpty()) {

JsonObject showImages = new JsonObject();

imageinfo.forEach(image -> {

showImages.put(image.getTypenum().toString(), image.getLocation());

});

prod.setShowImages(showImages);

}

// 检索产品购买时图片信息

TbProductImageboxExample imagBExample = new TbProductImageboxExample();

imagBExample.createCriteria().andProdidEqualTo(prod.getId()).andProdcodeEqualTo(prod.getProdcode()).andTypeEqualTo(3);

List<TbProductImagebox> imagebinfo = productImageboxMapper.selectByExample(imagBExample);

if (!imagebinfo.isEmpty()) {

JsonObject buyImages = new JsonObject();

imagebinfo.forEach(image -> {

buyImages.put(image.getTypenum().toString(), image.getLocation());

});

prod.setBuyImages(buyImages);

;

}

});

message = respWriter.toSuccess(pinfos, sn);

}else{

message = respWriter.toError(sn, RespCode.CODE\_504);

}

} catch (Exception e) {

throw e;

}

return message;

}

@Override

public String selectTbProductDevicesList(String sn, JsonObject param) throws Exception {

String message = respWriter.toError(sn);

try {

JsonObject result = new JsonObject();

Optional<Integer> prodid = Optional.ofNullable(param.getInteger("prodid"));// 产品ID

Optional<Integer> prodtype = Optional.ofNullable(param.getInteger("prodtype"));// 产品类别40：耳机41：手表

Optional<String> prodcode = Optional.ofNullable(param.getString("prodcode"));// 产品编码

Optional<String> devicescode = Optional.ofNullable(param.getString("devicescode"));// 设备编号

TbProductDevicesExample example = new TbProductDevicesExample();

TbProductDevicesExample.Criteria criteria = example.createCriteria();

if (prodid.isPresent()) {

criteria.andProdidEqualTo(prodid.get());

}

if (prodtype.isPresent()) {

criteria.andProdtypeEqualTo(prodtype.get());

}

if (prodcode.isPresent()) {

criteria.andProdcodeEqualTo(prodcode.get());

}

if (devicescode.isPresent()) {

criteria.andDevicescodeEqualTo(devicescode.get());

}

List<TbProductDevices> devices = productDevicesMapper.selectByExample(example);

if (!devices.isEmpty()) {

//result.put("devices", devices);

TbProductDevices d = devices.get(0);

TbProductInfo tbProductInfo = productInfoMapper.selectByPrimaryKey(d.getProdid());

if (tbProductInfo != null) {

result.put("decives", d.getDevicescode());

result.put("bdate", param.getLong("date"));

result.put("name", tbProductInfo.getProdname());

result.put("code",tbProductInfo.getProdcode());

result.put("desc", tbProductInfo.getProddesc());

result.put("price", tbProductInfo.getProdprice().floatValue());

result.put("total", 1);

//检索产品购买时图片信息

TbProductImageboxExample imagBExample = new TbProductImageboxExample();

imagBExample.createCriteria().andProdidEqualTo(d.getProdid()).andTypeEqualTo(3);

Optional<List<TbProductImagebox>> imagebinfo = Optional.ofNullable(productImageboxMapper.selectByExample(imagBExample));

if(imagebinfo.isPresent()){

JsonObject buyImages = new JsonObject();

imagebinfo.get().forEach(image->{

buyImages.put(image.getTypenum().toString(), image.getLocation());

});

result.put("image", buyImages);

}

}

message = respWriter.toSuccess(result, sn);

} else {

message = respWriter.toError(sn, RespCode.CODE\_504);

}

} catch (Exception e) {

throw e;

}

return message;

}

}

package com.chinamcom.framework.products.services.impl;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.chinamcom.framework.common.service.BaseService;

import com.chinamcom.framework.products.mapper.TbProductStockMapper;

import com.chinamcom.framework.products.model.TbProductStock;

import com.chinamcom.framework.products.model.TbProductStockExample;

import com.chinamcom.framework.products.services.ITbProductStockService;

import io.vertx.core.json.JsonArray;

import io.vertx.core.json.JsonObject;

@Service("tbProductStockService")

public class TbProductStockServiceImpl extends BaseService implements ITbProductStockService {

@Autowired

private TbProductStockMapper productStockMapper;

@Override

public String selectTbProductStockList(String sn, JsonObject param) throws Exception {

String message = respWriter.toError(sn);

try {

JsonObject result = new JsonObject();

List<TbProductStock> product = new ArrayList<TbProductStock>();

JsonArray products = param.getJsonArray("product");

products.forEach(prod -> {

JsonObject p = (JsonObject) prod;

Optional<Integer> prodid = Optional.ofNullable(p.getInteger("prodid"));

Optional<String> prodcode = Optional.ofNullable(p.getString("prodcode"));

TbProductStockExample example = new TbProductStockExample();

TbProductStockExample.Criteria criteria = example.createCriteria();

if (prodid.isPresent()) {

criteria.andProdidEqualTo(prodid.get());

}

if (prodcode.isPresent()) {

criteria.andProdcodeEqualTo(prodcode.get());

}

List<TbProductStock> rdata = productStockMapper.selectByExample(example);

if (!rdata.isEmpty()) {

product.add(rdata.get(0));

}

});

result.put("stock", product);

message = respWriter.toSuccess(result, sn);

} catch (Exception e) {

throw e;

}

return message;

}

@Override

public String updateTbProductStockList(String sn, JsonObject param) throws Exception {

String message = respWriter.toError(sn);

try {

Integer type = param.getInteger("type");// 1增加2减少

JsonArray products = param.getJsonArray("product");

products.forEach(product -> {

JsonObject p = (JsonObject) product;

Optional<Integer> prodid = Optional.ofNullable(p.getInteger("prodid"));

Optional<String> prodcode = Optional.ofNullable(p.getString("prodcode"));

Optional<Integer> number = Optional.ofNullable(p.getInteger("number"));// 数量

if (number.isPresent()) {

TbProductStockExample example = new TbProductStockExample();

TbProductStockExample.Criteria criteria = example.createCriteria();

if (prodid.isPresent()) {

criteria.andProdidEqualTo(prodid.get());

}

if (prodcode.isPresent()) {

criteria.andProdcodeEqualTo(prodcode.get());

}

List<TbProductStock> rdata = productStockMapper.selectByExample(example);

if (!rdata.isEmpty()) {

TbProductStock prod = rdata.get(0);

Integer stock = prod.getStock();// 库存

Integer sales = prod.getSales();// 销量

// 库存预警green:库存充足（STOCK>100)red:库存紧张(STOCK<100)gray:无库存(STOCK=0)

String warning = prod.getWarning();

if (type == 1) {

stock = stock + number.get();

sales = sales - number.get();

warning = stock > 100 ? "green" : stock == 0 ? "gray" : "red";

}

if (type == 2) {

stock = stock - number.get();

sales = sales + number.get();

warning = stock > 100 ? "green" : stock == 0 ? "gray" : "red";

}

TbProductStock record = new TbProductStock();

record.setStock(stock);

record.setSales(sales);

record.setWarning(warning);

record.setUpdateTime(new Date());

TbProductStockExample tpse = new TbProductStockExample();

TbProductStockExample.Criteria c = tpse.createCriteria();

if (prodid.isPresent()) {

c.andProdidEqualTo(prodid.get());

}

if (prodcode.isPresent()) {

c.andProdcodeEqualTo(prodcode.get());

}

productStockMapper.updateByExampleSelective(record, tpse);

}

}

});

message = respWriter.toSuccess(sn);

} catch (Exception e) {

throw e;

}

return message;

}

}

package com.chinamcom.framework.order.services.impl;

import java.math.BigDecimal;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.chinamcom.framework.common.constant.Constant;

import com.chinamcom.framework.common.model.PageModel;

import com.chinamcom.framework.common.service.BaseService;

import com.chinamcom.framework.common.util.StringUtil;

import com.chinamcom.framework.order.mapper.TbOrderInstanceMapper;

import com.chinamcom.framework.order.mapper.TbOrderMapper;

import com.chinamcom.framework.order.model.TbOrder;

import com.chinamcom.framework.order.model.TbOrderExample;

import com.chinamcom.framework.order.model.TbOrderExample.Criteria;

import com.chinamcom.framework.order.model.TbOrderInstance;

import com.chinamcom.framework.order.model.TbOrderInstanceExample;

import com.chinamcom.framework.order.services.ITbOrderService;

import com.github.pagehelper.Page;

import com.github.pagehelper.PageHelper;

import io.vertx.core.json.JsonArray;

import io.vertx.core.json.JsonObject;

@Service("tbOrderService")

public class TbOrderServiceImpl extends BaseService implements ITbOrderService {

@Autowired

private TbOrderMapper tbOrderMapper;

@Autowired

private TbOrderInstanceMapper tbOrderInstanceMapper;

//@Autowired

//private TbOrderLogisticsMapper tbOrderLogisticsMapper;

@Override

public String selectTbOrderList(String sn, JsonObject param) {

String message = respWriter.toError(sn);

List<JsonObject> result = new ArrayList<JsonObject>();

try {

Optional<Integer> uid = Optional.ofNullable(param.getInteger("uid"));// 用户ID

Integer pageNum = Optional.ofNullable(param.getInteger("pageNum")).orElse(1);// 页码

Integer pageSize = Optional.ofNullable(param.getInteger("pageSize")).orElse(Integer.MAX\_VALUE);// 每页显示条数

TbOrderExample example = new TbOrderExample();

Criteria criteria = example.createCriteria();

if (uid.isPresent()) {

criteria.andUserIdEqualTo(uid.get());

}

Page<Object> page = PageHelper.startPage(pageNum, pageSize);

List<TbOrder> orders = tbOrderMapper.selectByExample(example);

if (!orders.isEmpty()) {

orders.forEach(item -> {

JsonObject reply\_order = new JsonObject();

reply\_order.put("oid", item.getId());// 订单ID

reply\_order.put("code", item.getSerialNumber());// 订单编号

reply\_order.put("ctime", item.getCreateTime().getTime());// 下单时间

reply\_order.put("status", item.getStatus());// 订单状态

reply\_order.put("logistics\_name", item.getLogisticsName());// 物流公司

reply\_order.put("logistics\_code", "");// 物流单号

JsonArray reply\_instance = new JsonArray();

TbOrderInstanceExample instance = new TbOrderInstanceExample();

instance.createCriteria().andOrderIdEqualTo(item.getId());

List<TbOrderInstance> instances = tbOrderInstanceMapper.selectByExample(instance);

if (!instances.isEmpty()) {

instances.forEach(inst->{

JsonObject i = new JsonObject();

i.put("name", inst.getProductName());

i.put("code", inst.getProductCode());

i.put("total", inst.getProductTotal());

i.put("price", inst.getProductPrice().intValue());

i.put("fee", inst.getPayFee().intValue());

reply\_instance.add(i);

});

reply\_order.put("prods", reply\_instance);

}

result.add(reply\_order);

});

}

message = respWriter.toSuccess(new PageModel(page, result), sn);

} catch (Exception e) {

throw e;

}

return message;

}

@Override

public String submitTbOrderInfoByCart(String sn, JsonObject param) {

String message = respWriter.toError(sn);

try {

Date date = new Date();

Integer uid = param.getInteger("uid");// 用户ID

String channel = Optional.ofNullable(param.getString("channel")).orElse("pc");

Integer expressid = Optional.ofNullable(param.getInteger("expressid")).orElse(-1);

Optional<Integer> expresspay = Optional.ofNullable(param.getInteger("expresspay"));

String delivers = Optional.ofNullable(param.getString("delivers")).orElse("随时");

String logistics = Optional.ofNullable(param.getString("logistics")).orElse("宅急送(免费)");

//查询购物车中待结算的产品

TbOrderInstanceExample tbOrderInstanceExample = new TbOrderInstanceExample();

tbOrderInstanceExample.createCriteria().andUserIdEqualTo(uid).andStatusEqualTo(2).andOrderIdIsNull();

List<TbOrderInstance> listTbOrderInstance = tbOrderInstanceMapper.selectByExample(tbOrderInstanceExample);

Integer order\_type = 99;

BigDecimal fee = new BigDecimal("0.00");// 订单原价

BigDecimal pay\_fee = new BigDecimal("0.00");// 实付金额

BigDecimal reduce\_fee = new BigDecimal("0.00");// 减免金额

if (!listTbOrderInstance.isEmpty()) {

for (int i = 0; i < listTbOrderInstance.size(); i++) {

TbOrderInstance instance = listTbOrderInstance.get(i);

BigDecimal price = instance.getProductPrice();

BigDecimal reducefee = instance.getReduceFee();

BigDecimal total = new BigDecimal(instance.getProductTotal());

BigDecimal i\_all\_payfee = price.multiply(total);// 商品总额

BigDecimal i\_reducefee = reducefee.multiply(total);// 商品减免总额

BigDecimal i\_payfee = i\_all\_payfee.subtract(i\_reducefee);// 商品实付金额

fee = fee.add(i\_payfee);

reduce\_fee = reduce\_fee.add(i\_reducefee);

}

//快递费用

BigDecimal expay = null;

if (expresspay.isPresent()) {

expay = new BigDecimal(expresspay.get());

} else {

expay = StringUtil.getExpressPay(param.getString("logistics"));

}

pay\_fee = fee.subtract(reduce\_fee).add(expay);

String serial\_number = StringUtil.getOrderCode();// 序列号

// 生成订单

TbOrder order = new TbOrder();

LOG.info("\*\*\*\*\*\*处理订单\*\*\*\*\*\*");

order.setSerialNumber(serial\_number);

order.setFee(fee);

order.setPayFee(pay\_fee);

order.setReduceFee(reduce\_fee);

order.setOrderType(order\_type);

order.setChannel(channel);

order.setStatus(Constant.ORDER\_STATUS\_NOPAY);

order.setUserId(uid);

order.setCreateTime(date);

order.setExpressId(expressid);

order.setLogisticsName(logistics);

order.setExpressPay(expay);

order.setDeliversTime(delivers);

if(param.containsKey("invoce")){

order.setInvoceHead(param.getString("invoce"));

}

tbOrderMapper.insertSelective(order);

LOG.info("order builder succeed.");

Integer orderid = order.getId();

LOG.info("orderid:" + orderid);

// 更新购物车中待结算的产品

TbOrderInstance record = new TbOrderInstance();

record.setUpdateTime(new Date());

record.setStatus(3);

record.setOrderId(orderid);

tbOrderInstanceMapper.updateByExampleSelective(record, tbOrderInstanceExample);

LOG.info("order\_instance update succeed. ");

message = respWriter.toSuccess(new JsonObject().put("ordercode", order.getSerialNumber()).put("payfee", order.getPayFee().intValue()), sn);

}else{

message = respWriter.toError(sn, "订单生成失败.");

}

} catch (Exception e) {

throw e;

}

return message;

}

@Override

public String submitTbOrderInfo(String sn, JsonObject param) {

String message = respWriter.toError(sn);

try {

Date date = new Date();

Integer uid = param.getInteger("uid");// 用户ID

String channel = Optional.ofNullable(param.getString("channel")).orElse("pc");

Integer expressid = Optional.ofNullable(param.getInteger("expressid")).orElse(-1);

Optional<Integer> expresspay = Optional.ofNullable(param.getInteger("expresspay"));

String delivers = Optional.ofNullable(param.getString("delivers")).orElse("随时");

String logistics = Optional.ofNullable(param.getString("logistics")).orElse("宅急送(免费)");

JsonObject product = param.getJsonObject("product");// 产品信息

Integer order\_type = 99;

BigDecimal price = new BigDecimal(product.getInteger("price"));

BigDecimal reducefee = new BigDecimal(product.containsKey("reducefee") ? product.getString("reducefee") : "0.00");

BigDecimal total = new BigDecimal(product.containsKey("total") ? product.getInteger("total") : 1);

BigDecimal fee = price.multiply(total);// 商品总额

BigDecimal reduce\_fee = reducefee.multiply(total);// 商品减免总额

BigDecimal payfee = fee.subtract(reduce\_fee);// 商品实付金额

//快递费用

BigDecimal expay = null;

if (expresspay.isPresent()) {

expay = new BigDecimal(expresspay.get());

} else {

expay = StringUtil.getExpressPay(param.getString("logistics"));

}

//payfee = fee.add(expay);

String serial\_number = StringUtil.getOrderCode();// 序列号

// 生成订单

TbOrder order = new TbOrder();

LOG.info("\*\*\*\*\*\*处理订单\*\*\*\*\*\*");

order.setSerialNumber(serial\_number);

order.setFee(fee);

order.setPayFee(fee.add(expay));

order.setReduceFee(reduce\_fee);

order.setOrderType(order\_type);

order.setChannel(channel);

order.setStatus(Constant.ORDER\_STATUS\_NOPAY);

order.setUserId(uid);

order.setCreateTime(date);

order.setExpressId(expressid);

order.setLogisticsName(logistics);

order.setExpressPay(expay);

order.setDeliversTime(delivers);

if(param.containsKey("invoce")){

order.setInvoceHead(param.getString("invoce"));

}

tbOrderMapper.insertSelective(order);

LOG.info("order builder succeed.");

Integer orderid = order.getId();

LOG.info("orderid:" + orderid);

// 订单明细

String product\_code = product.getString("code");

String product\_name = product.getString("name");

TbOrderInstance instance = new TbOrderInstance();

instance.setOrderId(orderid);

instance.setProductCode(product\_code);

instance.setProductName(product\_name);

instance.setProductPrice(price);

instance.setProductTotal(total.intValue());

instance.setPayFee(payfee);

instance.setReduceFee(reduce\_fee);

instance.setCreateTime(date);

instance.setStatus(Constant.INSTANCE\_STATUS\_YESORDER);

instance.setUserId(uid);

tbOrderInstanceMapper.insertSelective(instance);

LOG.info("build order\_instance succeed.");

message = respWriter.toSuccess(new JsonObject().put("ordercode", order.getSerialNumber()).put("payfee", order.getPayFee().intValue()), sn);

} catch (Exception e) {

LOG.error("订单提交失败：" + e);

message = respWriter.toError(sn);

}

return message;

}

@Override

public void selectTbOrderInfo(JsonObject param,JsonObject result) {

try {

Optional<Integer> orderid = Optional.ofNullable(param.getInteger("orderid"));// 订单ID

Optional<String> ordercode = Optional.ofNullable(param.getString("ordercode"));

Optional<String> prodcode = Optional.ofNullable(param.getString("prodcode"));

TbOrderExample example = new TbOrderExample();

Criteria criteria = example.createCriteria();

if(orderid.isPresent()){

criteria.andIdEqualTo(orderid.get());

}

if(ordercode.isPresent()){

criteria.andSerialNumberEqualTo(ordercode.get());

}

List<TbOrder> lTbOrder = tbOrderMapper.selectByExample(example);

if (!lTbOrder.isEmpty()) {

TbOrder order = lTbOrder.get(0);

result.put("oid", order.getId());// 订单ID

result.put("code", order.getSerialNumber());// 订单编号

result.put("ctime", order.getCreateTime().getTime());// 下单时间

result.put("status", order.getStatus());// 订单状态

result.put("ptime", order.getPayTime());// 支付时间

result.put("stime", order.getSendTime());// 发货时间

result.put("otime", order.getCompateTime());// 完成时间

result.put("invoce", order.getInvoceHead());// 发票抬头

result.put("devivers", order.getDeliversTime());//送货时间

result.put("logistics\_name", order.getLogisticsName());//物流公司

result.put("logistics\_code", "1212121212");//物流单号

result.put("express", order.getExpressId());

JsonArray reply\_instance = new JsonArray();

TbOrderInstanceExample instance = new TbOrderInstanceExample();

TbOrderInstanceExample.Criteria cri = instance.createCriteria();

cri.andOrderIdEqualTo(order.getId());

if(prodcode.isPresent()){

cri.andProductCodeEqualTo(prodcode.get());

}

List<TbOrderInstance> instances = tbOrderInstanceMapper.selectByExample(instance);

if (!instances.isEmpty()) {

instances.forEach(inst -> {

JsonObject i = new JsonObject();

i.put("name", inst.getProductName());

i.put("code", inst.getProductCode());

i.put("total", inst.getProductTotal());

i.put("price", inst.getProductPrice().intValue());

i.put("fee", inst.getPayFee().intValue());

reply\_instance.add(i);

});

result.put("prods", reply\_instance);

}

}

} catch (Exception e) {

LOG.error("检索订单详情异常：" + e);

}

}

@Override

public String updateTbOrderInfo(String sn, JsonObject param) {

String message = respWriter.toError(sn);

List<TbOrder> result = new ArrayList<TbOrder>();

try {

Date date = new Date();

Integer orderid = param.getInteger("orderid");// 订单ID

Integer flag = param.containsKey("flag") ? param.getInteger("flag") : 0;

TbOrder record = new TbOrder();

record.setId(orderid);

switch (flag) {

case 2://支付成功，状态=2

record.setStatus(Constant.ORDER\_STATUS\_NOSEND);

record.setPayTime(date);

break;

case 3://已发货，状态=3

record.setStatus(Constant.ORDER\_STATUS\_SENDED);

record.setSendTime(date);

break;

case 4://交易完成，状态=4

record.setStatus(Constant.ORDER\_STATUS\_COMPLATE);

record.setCompateTime(date);

break;

case 5://订单取消，状态=5

record.setStatus(Constant.ORDER\_STATUS\_CANCEL);

record.setUpdateTime(date);

break;

default:

record.setUpdateTime(date);

break;

}

tbOrderMapper.updateByPrimaryKeySelective(record);

message = respWriter.toSuccess(result, sn);

} catch (Exception e) {

throw e;

}

return message;

}

}

package com.chinamcom.framework.order.services.impl;

import java.math.BigDecimal;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.chinamcom.framework.common.model.PageModel;

import com.chinamcom.framework.common.service.BaseService;

import com.chinamcom.framework.order.mapper.TbOrderInstanceMapper;

import com.chinamcom.framework.order.model.TbOrderInstance;

import com.chinamcom.framework.order.model.TbOrderInstanceExample;

import com.chinamcom.framework.order.services.ICartService;

import com.github.pagehelper.Page;

import com.github.pagehelper.PageHelper;

import io.vertx.core.json.JsonArray;

import io.vertx.core.json.JsonObject;

@Service("cartServiceImpl")

public class CartServiceImpl extends BaseService implements ICartService {

@Autowired

private TbOrderInstanceMapper tbOrderInstanceMapper;

@Override

public String addCart(String sn, JsonObject param) {

// TODO addCart

String message = null;

try {

Integer uid = param.getInteger("uid");

Integer prodtotal = Optional.ofNullable(param.getInteger("total")).orElse(1);// 数量

String prodcode = param.getString("code");

String prodname = param.getString("name");

Integer id = param.getInteger("id");

BigDecimal prodprice = new BigDecimal(param.getInteger("price"));

BigDecimal payfee = prodprice.multiply(new BigDecimal(prodtotal));

TbOrderInstance instance = new TbOrderInstance();

// 先查询购物车是否存在

TbOrderInstanceExample tbOrderInstanceExample = new TbOrderInstanceExample();

TbOrderInstanceExample.Criteria criteria = tbOrderInstanceExample.createCriteria();

criteria.andUserIdEqualTo(uid).andProductCodeEqualTo(prodcode);

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

status.add(1);

status.add(2);

criteria.andStatusIn(status);

List<TbOrderInstance> tbOrderInstanceList = tbOrderInstanceMapper.selectByExample(tbOrderInstanceExample);

if (!tbOrderInstanceList.isEmpty()) {//更新

Integer \_prodtotal = tbOrderInstanceList.get(0).getProductTotal();

BigDecimal \_payfee = tbOrderInstanceList.get(0).getPayFee();

instance.setProductTotal(prodtotal.intValue() + \_prodtotal);

instance.setPayFee(payfee.add(\_payfee));

instance.setUpdateTime(new Date());

tbOrderInstanceMapper.updateByExampleSelective(instance, tbOrderInstanceExample);

}else{//新增

instance.setUserId(uid);

instance.setProductId(id);

instance.setProductCode(prodcode);

instance.setProductName(prodname);

instance.setProductPrice(prodprice);

instance.setProductTotal(prodtotal);

instance.setPayFee(payfee);

instance.setCreateTime(new Date());

instance.setStatus(1);

tbOrderInstanceMapper.insertSelective(instance);

}

message = respWriter.toSuccess(sn);

} catch (Exception e) {

throw e;

}

return message;

}

@Override

public String incrCart(String sn, JsonObject param) {

// TODO incrCart

String message = null;

try {

Integer instanceid = param.getInteger("instanceid");// 购物车中产品的ID

TbOrderInstance instance = tbOrderInstanceMapper.selectByPrimaryKey(instanceid);

Integer total = instance.getProductTotal();

BigDecimal price = instance.getProductPrice();

BigDecimal payfee = instance.getPayFee();

TbOrderInstance record = new TbOrderInstance();

record.setId(instanceid);

record.setUpdateTime(new Date());

record.setProductTotal(total + 1);

record.setPayFee(payfee.add(price));

tbOrderInstanceMapper.updateByPrimaryKeySelective(record);

message = respWriter.toSuccess(new JsonObject().put("total", total + 1).put("instanceid", instanceid), sn);

} catch (Exception e) {

throw e;

}

return message;

}

@Override

public String decrCart(String sn, JsonObject param) {

// TODO decrCart

String message = null;

try {

Integer instanceid = param.getInteger("instanceid");// 购物车中产品的ID

TbOrderInstance instance = tbOrderInstanceMapper.selectByPrimaryKey(instanceid);

Integer total = instance.getProductTotal();

BigDecimal price = instance.getProductPrice();

BigDecimal payfee = instance.getPayFee();

TbOrderInstance record = new TbOrderInstance();

record.setId(instanceid);

record.setUpdateTime(new Date());

record.setProductTotal(total - 1);

record.setPayFee(payfee.subtract(price));

tbOrderInstanceMapper.updateByPrimaryKeySelective(record);

message = respWriter.toSuccess(new JsonObject().put("total", total - 1).put("instanceid", instanceid), sn);

} catch (Exception e) {

throw e;

}

return message;

}

@Override

public void account(JsonObject buy, JsonObject param) {

// TODO account

try {

JsonArray instances = new JsonArray();

Integer uid = param.getInteger("uid");// 用户ID

Integer status = Optional.ofNullable(param.getInteger("status")).orElse(2);

TbOrderInstanceExample example = new TbOrderInstanceExample();

example.createCriteria().andUserIdEqualTo(uid).andStatusEqualTo(status);

List<TbOrderInstance> result = tbOrderInstanceMapper.selectByExample(example);

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

result.forEach(item -> {

JsonObject i = new JsonObject();

if (item.getId() != null) {

i.put("instanceid", item.getId());

}

if (item.getProductCode() != null) {

i.put("prodcode", item.getProductCode());

}

if (item.getProductName() != null) {

i.put("prodname", item.getProductName());

}

if (item.getProductPrice() != null) {

i.put("prodprice", item.getProductPrice().toString());

}

if (item.getProductTotal() != null) {

i.put("prodtotal", item.getProductTotal());

}

if (item.getPayFee() != null) {

i.put("payfee", item.getPayFee().toString());

}

instances.add(i);

});

buy.put("product", instances);

}

} catch (Exception e) {

throw e;

}

}

@Override

public String checked(String sn, JsonObject param) {

// TODO buyCart

String message = null;

try {

Integer instanceid = param.getInteger("instanceid");// 购物车中产品的ID

Integer uid = param.getInteger("uid");// 用户ID

Boolean checked = Optional.ofNullable(param.getBoolean("checked")).orElse(false);

TbOrderInstance record = new TbOrderInstance();

record.setUpdateTime(new Date());

record.setStatus(checked ? 2 : 1);

TbOrderInstanceExample example = new TbOrderInstanceExample();

TbOrderInstanceExample.Criteria criteria = example.createCriteria();

criteria.andUserIdEqualTo(uid);

if(instanceid!=0){

criteria.andIdEqualTo(instanceid);

}

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

status.add(1);

status.add(2);

criteria.andStatusIn(status);

tbOrderInstanceMapper.updateByExampleSelective(record,example);

/\*\*

JsonArray instances = new JsonArray();

TbOrderInstanceExample query = new TbOrderInstanceExample();

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

ss.add(1);ss.add(2);

query.createCriteria().andUserIdEqualTo(uid).andStatusIn(ss);

List<TbOrderInstance> result = tbOrderInstanceMapper.selectByExample(query);

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

result.forEach(item -> {

JsonObject i = new JsonObject();

if (item.getId() != null) {

i.put("instanceid", item.getId());

}

if (item.getProductCode() != null) {

i.put("prodcode", item.getProductCode());

}

if (item.getProductName() != null) {

i.put("prodname", item.getProductName());

}

if (item.getProductPrice() != null) {

i.put("prodprice", item.getProductPrice().toString());

}

if (item.getProductTotal() != null) {

i.put("prodtotal", item.getProductTotal());

}

if (item.getPayFee() != null) {

i.put("payfee", item.getPayFee().toString());

}

if (item.getStatus() != null) {

i.put("checked", item.getStatus() == 1 ? "false" : "true");

}

instances.add(i);

});

}

\*/

message = respWriter.toSuccess(sn);

} catch (Exception e) {

throw e;

}

return message;

}

@Override

public String delCart(String sn, JsonObject param) {

// TODO delCart

String message = null;

try {

Integer instanceid = param.getInteger("instanceid");// 购物车中产品的ID

TbOrderInstance record = new TbOrderInstance();

record.setId(instanceid);

record.setUpdateTime(new Date());

record.setStatus(4);

tbOrderInstanceMapper.updateByPrimaryKeySelective(record);

message = respWriter.toSuccess(sn);

} catch (Exception e) {

throw e;

}

return message;

}

@Override

public String listCart(String sn, JsonObject param) {

// TODO listCart

String message = null;

try {

JsonArray instances = new JsonArray();

Integer uid = param.getInteger("uid");// 用户ID

Optional<Integer> status = Optional.ofNullable(param.getInteger("status"));

Integer pageNum = Optional.ofNullable(param.getInteger("pageNum")).orElse(1);// 页码

Integer pageSize = Optional.ofNullable(param.getInteger("pageSize")).orElse(Integer.MAX\_VALUE);// 每页显示条数

TbOrderInstanceExample example = new TbOrderInstanceExample();

TbOrderInstanceExample.Criteria criteria = example.createCriteria();

criteria.andUserIdEqualTo(uid);

if(status.isPresent()){

criteria.andStatusEqualTo(status.get());

} else {

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

s.add(1);

s.add(2);

criteria.andStatusIn(s);

}

Page<Object> page = PageHelper.startPage(pageNum, pageSize);

List<TbOrderInstance> result = tbOrderInstanceMapper.selectByExample(example);

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

result.forEach(item -> {

JsonObject i = new JsonObject();

if (item.getId() != null) {

i.put("instanceid", item.getId());

}

if (item.getProductCode() != null) {

i.put("prodcode", item.getProductCode());

}

if (item.getProductName() != null) {

i.put("prodname", item.getProductName());

}

if (item.getProductPrice() != null) {

i.put("prodprice", item.getProductPrice().intValue());

}

if (item.getProductTotal() != null) {

i.put("prodtotal", item.getProductTotal());

}

if (item.getPayFee() != null) {

i.put("payfee", item.getPayFee().toString());

}

if (item.getStatus() != null) {

i.put("checked", item.getStatus() == 1 ? Boolean.FALSE : Boolean.TRUE);

}

instances.add(i);

});

}

message = respWriter.toSuccess(new PageModel(page, instances), sn);

} catch (Exception e) {

throw e;

}

return message;

}

}

package com.chinamcom.framework.aftersale.service.impl;

import java.util.Date;

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.chinamcom.framework.aftersale.mapper.TbAftersaleImageboxMapper;

import com.chinamcom.framework.aftersale.mapper.TbAftersaleLogisticsMapper;

import com.chinamcom.framework.aftersale.mapper.TbAftersaleMapper;

import com.chinamcom.framework.aftersale.mapper.TbAftersaleProductMapper;

import com.chinamcom.framework.aftersale.model.TbAftersale;

import com.chinamcom.framework.aftersale.model.TbAftersaleExample;

import com.chinamcom.framework.aftersale.model.TbAftersaleImagebox;

import com.chinamcom.framework.aftersale.model.TbAftersaleImageboxExample;

import com.chinamcom.framework.aftersale.model.TbAftersaleLogistics;

import com.chinamcom.framework.aftersale.model.TbAftersaleLogisticsExample;

import com.chinamcom.framework.aftersale.model.TbAftersaleProduct;

import com.chinamcom.framework.aftersale.model.TbAftersaleProductExample;

import com.chinamcom.framework.aftersale.service.IAfterSaleService;

import com.chinamcom.framework.common.model.PageModel;

import com.chinamcom.framework.common.response.RespCode;

import com.chinamcom.framework.common.service.BaseService;

import com.chinamcom.framework.common.util.StringUtil;

import com.github.pagehelper.Page;

import com.github.pagehelper.PageHelper;

import io.vertx.core.json.JsonArray;

import io.vertx.core.json.JsonObject;

@Service("afterSaleService")

public class AfterSaleServiceImpl extends BaseService implements IAfterSaleService {

@Autowired

private TbAftersaleMapper tbAfterSaleMapper;

@Autowired

private TbAftersaleProductMapper tbAfterSaleProductMapper;

@Autowired

private TbAftersaleLogisticsMapper tbAfterSaleLogisticsMapper;

@Autowired

private TbAftersaleImageboxMapper tbAfterSaleImageboxMapper;

@Override

public String selectTbAfterSale(String sn, JsonObject param) {

String message = respWriter.toError(sn,RespCode.CODE\_504);

try {

JsonArray result = new JsonArray();

Integer uid = param.getInteger("uid");

Integer pageNum = Optional.ofNullable(param.getInteger("pageNum")).orElse(1);// 页码

Integer pageSize = Optional.ofNullable(param.getInteger("pageSize")).orElse(Integer.MAX\_VALUE);// 每页显示条数

TbAftersaleExample example = new TbAftersaleExample();

TbAftersaleExample.Criteria criteria = example.createCriteria();

if (uid != 0) {

criteria.andUidEqualTo(uid);

}

example.setOrderByClause("create\_time desc");

Page<Object> page = PageHelper.startPage(pageNum, pageSize);

List<TbAftersale> listTbAftersale = tbAfterSaleMapper.selectByExample(example);

if(!listTbAftersale.isEmpty()){

listTbAftersale.forEach(item->{

JsonObject sale = new JsonObject();

sale.put("aftercode", item.getAftercode());

sale.put("applytime", item.getCreateTime().getTime());

sale.put("aftertype", item.getAftertype() == 1 ? "退货" : item.getAftertype() == 2 ? "换货" : "维修");

sale.put("status", item.getStatus() == 0 ? "待受理" : item.getStatus() == 1 ? "已受理" : item.getStatus() == 2 ? "已完成" : "取消");

TbAftersaleProductExample pexample = new TbAftersaleProductExample();

pexample.createCriteria().andAfterIdEqualTo(item.getId());

List<TbAftersaleProduct> listTbAftersaleProduct = tbAfterSaleProductMapper.selectByExample(pexample);

if(!listTbAftersaleProduct.isEmpty()){

TbAftersaleProduct product = listTbAftersaleProduct.get(0);

sale.put("prodname", product.getProductName());

sale.put("prodcode", product.getProductCode());

sale.put("total", product.getQuantity().intValue());

}

TbAftersaleLogisticsExample lexample = new TbAftersaleLogisticsExample();

lexample.createCriteria().andAfterCodeEqualTo(item.getAftercode());

List<TbAftersaleLogistics> listTbAftersaleLogistics= tbAfterSaleLogisticsMapper.selectByExampleWithExpress(lexample);

if(!listTbAftersaleLogistics.isEmpty()){

TbAftersaleLogistics l = listTbAftersaleLogistics.get(0);

sale.put("customer", new JsonObject().put("name", l.getName()).put("phone", l.getPhone()).put("address", l.getProvince()+l.getCity()+l.getArea()+l.getAddress()));

}

result.add(sale);

});

message = respWriter.toSuccess(new PageModel(page, result), sn);

}

} catch (Exception e) {

throw e;

}

return message;

}

@Override

public String selectTbAfterSaleInfo(String sn, JsonObject param) {

String message = respWriter.toError(sn,RespCode.CODE\_504);

try {

JsonObject sale = new JsonObject();

Optional<Integer> afterid = Optional.ofNullable(param.getInteger("afterid"));

Optional<String> aftercode = Optional.ofNullable(param.getString("aftercode"));

TbAftersaleExample example = new TbAftersaleExample();

TbAftersaleExample.Criteria criteria = example.createCriteria();

if(afterid.isPresent()){

criteria.andIdEqualTo(afterid.get());

}

if(aftercode.isPresent()){

criteria.andAftercodeEqualTo(aftercode.get());

}

List<TbAftersale> listTbAftersale = tbAfterSaleMapper.selectByExample(example);

if (!listTbAftersale.isEmpty()) {

TbAftersale item = listTbAftersale.get(0);

sale.put("aftercode", item.getAftercode());

sale.put("applytime", item.getCreateTime().getTime());

sale.put("handletime", item.getHandleTime() != null ? item.getHandleTime().getTime() : null);

sale.put("finishtime", item.getFinishTime() != null ? item.getFinishTime().getTime() : null);

sale.put("updatetime", item.getUpdateTime() != null ? item.getUpdateTime().getTime() : null);

sale.put("aftertype", item.getAftertype() == 1 ? "退货" : item.getAftertype() == 2 ? "换货" : "维修");

sale.put("status", item.getStatus() == 0 ? "待受理" : item.getStatus() == 1 ? "已受理" : item.getStatus() == 2 ? "已完成" : "取消");

sale.put("reply", item.getReply());

TbAftersaleProductExample pexample = new TbAftersaleProductExample();

pexample.createCriteria().andAfterIdEqualTo(item.getId());

List<TbAftersaleProduct> listTbAftersaleProduct = tbAfterSaleProductMapper.selectByExample(pexample);

if (!listTbAftersaleProduct.isEmpty()) {

TbAftersaleProduct product = listTbAftersaleProduct.get(0);

sale.put("prodname", product.getProductName());

sale.put("prodcode", product.getProductCode());

sale.put("total", product.getQuantity().intValue());

}

TbAftersaleImageboxExample iexample = new TbAftersaleImageboxExample();

iexample.createCriteria().andAfterIdEqualTo(item.getId());

List<TbAftersaleImagebox> listTbAftersaleImagebox = tbAfterSaleImageboxMapper.selectByExample(iexample);

if (!listTbAftersaleImagebox.isEmpty()) {

JsonArray images = new JsonArray();

listTbAftersaleImagebox.forEach(image->{

images.add(image.getImageUri());

});

sale.put("images", images);

}

message = respWriter.toSuccess(sale, sn);

}

} catch (Exception e) {

throw e;

}

return message;

}

@Override

public String submitTbAfterSale(String sn, JsonObject param) {

String message = respWriter.toError(sn);

try {

Integer orderId = param.getInteger("orderid");// 订单ID

Integer uid = param.getInteger("uid");

Integer type = param.getInteger("type");// 1退货2换货3维修

String ordercode = param.getString("ordercode");// 订单编号

Optional<String> devicecode = Optional.ofNullable(param.getString("devicecode"));// 设备编号

Optional<Long> signtime = Optional.ofNullable(param.getLong("signtime"));// 签收日期

Optional<Integer> expresskey = Optional.ofNullable(param.getInteger("express"));// 收货ID

String reason = param.getString("reason");

String channel = Optional.ofNullable(param.getString("channel")).orElse("pc");

Optional<String> productType = Optional.ofNullable(param.getString("producttype"));

JsonArray products = param.getJsonArray("prods");// 商品信息

Optional<JsonArray> imageboxs = Optional.ofNullable(param.getJsonArray("imageboxs"));// 图片信息

TbAftersale sale = new TbAftersale();

sale.setAftercode(StringUtil.getAfterSaleCode());

sale.setAftertype(type);

sale.setUid(uid);

sale.setOrderId(orderId);

sale.setOrderCode(ordercode);

if(devicecode.isPresent()){

sale.setProductCode(devicecode.get());

}

if (productType.isPresent()) {

sale.setProductType(productType.get());

}

sale.setReason(reason);

sale.setChannel(channel);

tbAfterSaleMapper.insertSelective(sale);

LOG.info("build aftersale succeed.");

LOG.info("afterid:"+sale.getId());

products.forEach(item -> {

JsonObject prod = (JsonObject) item;

TbAftersaleProduct product = new TbAftersaleProduct();

product.setAfterId(sale.getId());

product.setAfterCode(sale.getAftercode());

if (prod.containsKey("code")) {

product.setProductCode(prod.getString("code"));

}

if (prod.containsKey("name")) {

product.setProductName(prod.getString("name"));

}

if(devicecode.isPresent()){

product.setDeviceCode(devicecode.get());

}

if(signtime.isPresent()){

product.setSignTime(new Date(signtime.get()));

}

product.setCreateTime(new Date());

product.setQuantity(prod.containsKey("total") ? prod.getInteger("total") : 1);

tbAfterSaleProductMapper.insertSelective(product);

LOG.info("build aftersale\_product succeed.");

});

if (imageboxs.isPresent()) {

imageboxs.get().forEach(image->{

TbAftersaleImagebox imagebox = new TbAftersaleImagebox();

imagebox.setImageUri(image.toString());

imagebox.setAfterId(sale.getId());

imagebox.setAfterCode(sale.getAftercode());

tbAfterSaleImageboxMapper.insertSelective(imagebox);

LOG.info("build aftersale\_imagebox succeed.");

});

}

//处理物流信息

TbAftersaleLogistics logistics = new TbAftersaleLogistics();

logistics.setAfterId(sale.getId());

logistics.setAfterCode(sale.getAftercode());

if(expresskey.isPresent()){

logistics.setExpressId(expresskey.get());

}

tbAfterSaleLogisticsMapper.insertSelective(logistics);

LOG.info("build aftersale\_logistics succeed.");

message = respWriter.toSuccess(new JsonObject().put("code", sale.getAftercode()),sn);

} catch (Exception e) {

throw e;

}

return message;

}

@Override

public String submitTbAfterSaleByDevice(String sn, JsonObject param) {

String message = respWriter.toError(sn);

try {

} catch (Exception e) {

throw e;

}

return message;

}

@Override

public String handleTbAfterSale(String sn, JsonObject param) {

String message = respWriter.toError(sn);

try {

Optional<Integer> afterid = Optional.ofNullable(param.getInteger("afterid"));

Optional<String> reply = Optional.ofNullable(param.getString("reply"));// 售后完成时的反馈信息

Optional<String> aftercode = Optional.ofNullable(param.getString("aftercode"));

Integer type = param.getInteger("type");

TbAftersale updatesale = new TbAftersale();

updatesale.setStatus(type);

if (type == 1) {

updatesale.setHandleTime(new Date());

url.append("info=" + info).append("&");

url.append("requestKey=" + requestKey).append("&");

url.append("currencyType=" + currencyType).append("&");

url.append("pageUrl=" + pageUrl).append("&");

url.append("bgUrl=" + bgUrl).append("&");

reqData.put("PAY\_REQ", url.toString());

threeCPayService.insertTbOrderPayment(sn,reqData);

url.append("sign=" + sign);

reply = respWriter.toSuccess(new JsonObject().put("payurl", new String(url.toString().getBytes(),"UTF-8")), sn);

} else {

reply = respWriter.toError(sn, "请求参数不符合规范，请重新提交.");

}

} catch (Exception e) {

LOG.error("跳转支付统一平台出现异常:" + e);

reply = respWriter.toError(sn, "支付失败,请重新提交.");

}

LOG.info("jumppay\_reply:" + reply);

message.reply(reply);

});

vertx.eventBus().consumer(ConsumerConstant.PAYMENT\_ACCEPTPAY, message -> {

LOG.info("acceptpay\_received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

String sn = reqData.getString("sn");

String reply = respWriter.toError(sn);

try {

threeCPayService.updateTbOrderPayment(sn, reqData);

reply = respWriter.toSuccess(sn);

} catch (Exception e) {

LOG.error("支付出现异常:" + e);

reply = respWriter.toError(sn, "支付失败,请重新提交.");

}

LOG.info("acceptpay\_reply:" + reply);

message.reply(reply);

});

}

}

package com.chinamcom.framework.user.verticle;

import java.io.ByteArrayOutputStream;

import java.util.ArrayList;

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Component;

import com.bjcathay.pay.service.util.ClientSignHelper;

import com.chinamcom.framework.common.constant.ConsumerConstant;

import com.chinamcom.framework.common.util.Base64Utils;

import com.chinamcom.framework.common.util.RedisClientUtils;

import com.chinamcom.framework.common.util.StringUtil;

import com.chinamcom.framework.common.util.VerifyCodeUtils;

import com.chinamcom.framework.common.verticle.BaseVerticle;

import com.chinamcom.framework.user.services.IUserService;

import io.vertx.core.http.HttpClientRequest;

import io.vertx.core.json.JsonObject;

import io.vertx.ext.auth.jwt.JWTAuth;

import io.vertx.redis.RedisClient;

/\*\*

\* 用户服务

\*

\* @author xuxg

\* @since 20160831

\*

\*/

@Component

public class UserVerticle extends BaseVerticle {

@Autowired

private IUserService userServiceImpl;

@Override

public void start() throws Exception {

JWTAuth jwt = JWTAuth.create(vertx, new JsonObject().put("keyStore",

new JsonObject().put("path", "keystore.jceks").put("password", "secret")));

vertx.eventBus().consumer(ConsumerConstant.ZM3C\_USER\_REGISTER, message -> {

LOG.info("received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

String sn = reqData.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<String> phone = Optional.ofNullable(reqData.getString("phone"));

Optional<String> password = Optional.ofNullable(reqData.getString("password"));

if (phone.isPresent() && password.isPresent()) {

reply = userServiceImpl.register(sn, reqData);

}

} catch (Exception e) {

LOG.error(e);

reply = respWriter.toError(sn, "register faild.");

}

LOG.info("reply:" + reply);

message.reply(reply);

});

vertx.eventBus().consumer(ConsumerConstant.ZM3C\_USER\_LOGIN\_CN, message -> {

LOG.info("received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

String sn = reqData.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<String> phone = Optional.ofNullable(reqData.getString("phone"));

Optional<String> password = Optional.ofNullable(reqData.getString("password"));

if (phone.isPresent() && password.isPresent()) {

reply = userServiceImpl.loginCn(jwt, sn, phone.get(), password.get());

}

} catch (Exception e) {

LOG.error(e);

reply = respWriter.toError(sn, "login faild.");

}

LOG.info("reply:" + reply);

message.reply(reply);

});

vertx.eventBus().consumer(ConsumerConstant.PASSWORD\_MODIFY, message -> {

LOG.info("modify\_password\_received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

String sn = reqData.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<String> phone = Optional.ofNullable(reqData.getString("phone"));

Optional<String> password = Optional.ofNullable(reqData.getString("password"));

if (phone.isPresent() && password.isPresent()) {

reply = userServiceImpl.modifyPass(sn, reqData);

}

} catch (Exception e) {

LOG.error("修改密码异常:" + e);

reply = respWriter.toError(sn, "修改密码失败，请重新操作.");

}

LOG.info("modify\_password\_reply:" + reply);

message.reply(reply);

});

vertx.eventBus().consumer(ConsumerConstant.SEND\_SMS, message -> {

LOG.info("sendsms received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

String sn = reqData.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<String> type = Optional.ofNullable(reqData.getString("type"));

Optional<String> phone = Optional.ofNullable(reqData.getString("phone"));

if (type.isPresent() && phone.isPresent()) {

String authcode = StringUtil.getSmsAuthCode();// 4位随机码

String content = StringUtil.getSmsContent(authcode);// 短信内容

String smssn = StringUtil.getSmsSn(phone.get());

reqData.put("authcode", authcode);

reqData.put("content", content);

reqData.put("channel", reqData.containsKey("channel") ? reqData.getString("channel") : "PC");

reqData.put("smssn", smssn);

reply = userServiceImpl.sendSMS(sn, reqData, 3);// 3分钟后失效

// 调用发送短信接口

String service = config.getProperty("send\_sms", "bop.orders.sendInfoToUser");

Integer port = Integer.valueOf(config.getProperty("send\_port", "8050"));

String host = config.getProperty("send\_host", "localhost");

String requestURI = config.getProperty("send\_uri", "/esb/json");

JsonObject sms = new JsonObject().put("delay\_minute", 0).put("send\_channel", "sms")

.put("phoneno", phone.get()).put("content", content);

JsonObject o = new JsonObject();

o.put("channel", "web");

o.put("platform\_type", "CUNT");

o.put("request\_data", sms);

o.put("ret\_type", "json");

o.put("serial\_number", sn);

o.put("service\_name", service);

HttpClientRequest request = vertx.createHttpClient().post(port, host, requestURI, ct -> {

int code = ct.statusCode();

LOG.info("sendsms\_reply\_code=" + code);

if (code == 200) {

ct.bodyHandler(body -> {

JsonObject result = body.toJsonObject();

LOG.info("sendsms\_reply\_result=" + result);

if (result.getString("ret\_code").equals("0")) {

userServiceImpl.updateSMS(sn, new JsonObject().put("smssn", smssn));

LOG.info("发送短信验证码成功.");

} else {

LOG.info("发送短信验证码失败.");

}

});

} else {

LOG.info("调用发送短信服务接口失败.");

}

});

request.end(o.toString());

} else {

reply = respWriter.toError(sn, "请求参数不符合规范，请重新提交.");

}

} catch (Exception e) {

LOG.error("发送短信异常：" + e);

reply = respWriter.toError(sn, "发送短信失败，请重新发送.");

}

LOG.info("sendsms reply:" + reply);

message.reply(reply);

});

vertx.eventBus().consumer(ConsumerConstant.GENERATE\_CODE, message -> {

LOG.info("generate\_code\_received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

String sn = reqData.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<Integer> uid = Optional.ofNullable(reqData.getInteger("uid"));

if (uid.isPresent()) {

int w = 200, h = 80;

String verifyCode = VerifyCodeUtils.generateVerifyCode(4);

ByteArrayOutputStream output = new ByteArrayOutputStream();

VerifyCodeUtils.outputImage(w, h, output, verifyCode);

String smssn = StringUtil.getSmsSn("10020");

// 验证码是否要保存到Redis

Optional<RedisClient> client = Optional.ofNullable(RedisClientUtils.getRedisClient(vertx));

if (client.isPresent()) {

client.get().setex(uid.get().toString() + "\_verifycode", 3600L, verifyCode, handler -> {

});

} else {

reqData.put("authcode", verifyCode);

reqData.put("content", "验证码为" + verifyCode);

reqData.put("channel", reqData.containsKey("channel") ? reqData.getString("channel") : "PC");

reqData.put("smssn", smssn);

reqData.put("phone", "10020");

reqData.put("type", "verifycode");

userServiceImpl.sendSMS(sn, reqData, 60);// 60分钟失效

}

reply = respWriter.toSuccess(new JsonObject().put("uid", uid.get()).put("code", verifyCode)

.put("smssn", smssn).put("url", Base64Utils.encode(output.toByteArray())), sn);

}

} catch (Exception e) {

LOG.error("生成验证码异常:" + e);

reply = respWriter.toError(sn, "生成验证码失败，请重新操作.");

}

LOG.info("generate\_code\_reply:" + reply);

message.reply(reply);

});

vertx.eventBus().consumer(ConsumerConstant.VERIFY\_CODE, message -> {

LOG.info("verify\_code\_received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

JsonObject reply = new JsonObject();

try {

String smssn = Optional.ofNullable(reqData.getString("smssn")).orElse("10020");

String code = Optional.ofNullable(reqData.getString("code")).orElse("10020");

Optional<Integer> uid = Optional.ofNullable(reqData.getInteger("uid"));

Optional<RedisClient> client = Optional.ofNullable(RedisClientUtils.getRedisClient(vertx));

if (client.isPresent()) {

client.get().get(uid.get().toString() + "\_verifycode", handler -> {

if (handler.succeeded()) {

if (code.equals(handler.result())) {

reply.put("checked", Boolean.TRUE);

} else {

reply.put("checked", Boolean.FALSE);

}

} else {

LOG.error("从Redis取验证码校异常,直接返回验证失败:" + handler.cause());

reply.put("checked", Boolean.FALSE);

}

LOG.info("verify\_code\_reply:" + reply);

message.reply(reply);

});

} else {

reply.put("checked", userServiceImpl.verifyCode(smssn, code));

LOG.info("verify\_code\_reply:" + reply);

message.reply(reply);

}

} catch (Exception e) {

LOG.error("验证码校验异常,直接返回验证失败:" + e);

reply.put("checked", Boolean.FALSE);

message.reply(reply);

}

});

vertx.eventBus().consumer(ConsumerConstant.BUILD\_SIGN\_CODE, message -> {

LOG.info("build\_sign\_received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

JsonObject reply = new JsonObject();

String result = respWriter.toError(reqData.getString("sn"));

try {

// 取签名

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

reqData.forEach(param -> {

String value = (String) param.getValue();

buildParam.add(value);

});

String sign = ClientSignHelper.getSign(config.getProperty("login.domainSecret", "test"), buildParam);

reply.put("sign", sign);

result = respWriter.toSuccess(reply, reqData.getString("sn"));

} catch (Exception e) {

LOG.error("获取签名异常:" + e);

}

LOG.info("build\_sign\_reply:" + result);

message.reply(result);

});

}

public void setUserServiceImpl(IUserService userServiceImpl) {

this.userServiceImpl = userServiceImpl;

}

}

package com.chinamcom.framework.user.verticle;

import java.util.Optional;

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

import org.springframework.stereotype.Component;

import com.chinamcom.framework.common.constant.ConsumerConstant;

import com.chinamcom.framework.common.verticle.BaseVerticle;

import com.chinamcom.framework.user.services.IExpressService;

import io.vertx.core.json.JsonObject;

/\*\*

\* 用户服务

\*

\* @author xuxg

\* @since 20160831

\*

\*/

@Component

public class ExpressVerticle extends BaseVerticle {

@Autowired

private IExpressService expressServiceImpl;

@Override

public void start() throws Exception {

vertx.eventBus().consumer(ConsumerConstant.EXPRESS\_ADD, message -> {

// TODO add

LOG.info("add received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

String sn = reqData.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<Integer> uid = Optional.ofNullable(reqData.getInteger("uid"));// 用户ID

Optional<String> name = Optional.ofNullable(reqData.getString("name"));// 收货人

Optional<String> phone = Optional.ofNullable(reqData.getString("phone"));// 手机号

Optional<String> zipcode = Optional.ofNullable(reqData.getString("zipcode"));// 邮编

Optional<String> address = Optional.ofNullable(reqData.getString("address"));// 街道地址

if (!uid.isPresent() || !name.isPresent() || !phone.isPresent() || !zipcode.isPresent()

|| !address.isPresent()) {

reply = respWriter.toError(sn, "请求参数不符合规范，请重新提交.");

} else {

reply = expressServiceImpl.addExpress(sn, reqData);

}

} catch (Exception e) {

LOG.error("添加收货地址异常：" + e);

reply = respWriter.toError(sn, "操作失败，请重新提交.");

}

LOG.info("add reply:" + reply);

message.reply(reply);

});

vertx.eventBus().consumer(ConsumerConstant.EXPRESS\_MOD, message -> {

// TODO mod

LOG.info("mod received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

String sn = reqData.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<Integer> eid = Optional.ofNullable(reqData.getInteger("eid"));// ID

Optional<Integer> uid = Optional.ofNullable(reqData.getInteger("uid"));// 用户ID

Optional<String> name = Optional.ofNullable(reqData.getString("name"));// 收货人

Optional<String> phone = Optional.ofNullable(reqData.getString("phone"));// 手机号

Optional<String> zipcode = Optional.ofNullable(reqData.getString("zipcode"));// 邮编

Optional<String> address = Optional.ofNullable(reqData.getString("address"));// 街道地址

if (!uid.isPresent() || !eid.isPresent() || !name.isPresent() || !phone.isPresent()

|| !zipcode.isPresent() || !address.isPresent()) {

reply = respWriter.toError(sn, "请求参数不符合规范，请重新提交.");

} else {

reply = expressServiceImpl.modExpress(sn, reqData);

}

} catch (Exception e) {

LOG.error("修改收货地址异常：" + e);

reply = respWriter.toError(sn, "操作失败，请重新提交.");

}

LOG.info("mod reply:" + reply);

message.reply(reply);

});

vertx.eventBus().consumer(ConsumerConstant.EXPRESS\_DEL, message -> {

// TODO del

LOG.info("del received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

String sn = reqData.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<Integer> eid = Optional.ofNullable(reqData.getInteger("eid"));// 地址ID

Optional<Integer> uid = Optional.ofNullable(reqData.getInteger("uid"));// 用户ID

if (!eid.isPresent() || !uid.isPresent()) {

reply = respWriter.toError(sn, "请求参数不符合规范，请重新提交.");

} else {

reply = expressServiceImpl.delExpress(sn, reqData);

}

} catch (Exception e) {

LOG.error("删除收货地址异常：" + e);

reply = respWriter.toError(sn, "操作失败，请重新提交.");

}

LOG.info("del reply:" + reply);

message.reply(reply);

});

vertx.eventBus().consumer(ConsumerConstant.EXPRESS\_DEF, message -> {

// TODO def

LOG.info("def received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

String sn = reqData.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<Integer> eid = Optional.ofNullable(reqData.getInteger("eid"));// 地址ID

Optional<Integer> uid = Optional.ofNullable(reqData.getInteger("uid"));// 用户ID

if (!eid.isPresent() || !uid.isPresent()) {

reply = respWriter.toError(sn, "请求参数不符合规范，请重新提交.");

} else {

reply = expressServiceImpl.defExpress(sn, reqData);

}

} catch (Exception e) {

LOG.error("收货地址默认设置异常：" + e);

reply = respWriter.toError(sn, "操作失败，请重新提交.");

}

LOG.info("def reply:" + reply);

message.reply(reply);

});

vertx.eventBus().consumer(ConsumerConstant.EXPRESS\_LIST, message -> {

// TODO list

LOG.info("list received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

String sn = reqData.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<Integer> uid = Optional.ofNullable(reqData.getInteger("uid"));// 用户ID

if (!uid.isPresent()) {

reply = respWriter.toError(sn, "请求参数不符合规范，请重新提交.");

} else {

reply = expressServiceImpl.listExpress(sn, reqData);

}

} catch (Exception e) {

LOG.error("检索收货地址异常：" + e);

reply = respWriter.toError(sn, "操作失败，请重新提交.");

}

LOG.info("list reply:" + reply);

message.reply(reply);

});

}

public void setExpressServiceImpl(IExpressService expressServiceImpl) {

this.expressServiceImpl = expressServiceImpl;

}

}

package com.chinamcom.framework.monitor.verticle;

import java.util.ArrayList;

import java.util.List;

import java.util.Optional;

import org.springframework.stereotype.Component;

import com.chinamcom.framework.common.constant.ConsumerConstant;

import com.chinamcom.framework.common.response.RespCode;

import com.chinamcom.framework.common.verticle.BaseVerticle;

import io.vertx.core.json.Json;

import io.vertx.core.json.JsonObject;

import io.vertx.ext.mongo.FindOptions;

import io.vertx.ext.mongo.MongoClient;

/\*\*

\* 日志监测

\*

\* @author xuxg

\* @since 20161123

\*

\*/

@Component

public class MonitorVerticle extends BaseVerticle {

private MongoClient client;

public MonitorVerticle(MongoClient client) {

this.client = client;

}

@Override

public void start() throws Exception {

/\*\*

\* 监测业务日志信息

\*/

vertx.eventBus().consumer(ConsumerConstant.MONITOR\_LOG, message -> {

JsonObject param = (JsonObject) message.body();

String sn = param.getString("sn");

LOG.info(">>>>>" + param);

JsonObject query = new JsonObject();

String path = Optional.ofNullable(param.getString("path")).orElse("api");

String[] paths = config.getProperty("log.path").split("@");

path = paths[0] + path.toUpperCase() + paths[1] + path.toLowerCase() + paths[2];

LOG.info(">>>>>path=" + path);

query.put("path", path);

if (param.containsKey("key")) {

String key = param.getString("key");

LOG.info("key=" + key);

query.put("\_id", key);

}

if (param.containsKey("host")) {

String host = param.getString("host");

LOG.info(">>>>>host=" + host);

query.put("host", host);

}

String findBeginDate = null;

if (param.containsKey("bDate")) {

String bdate = param.getString("bdate");

findBeginDate = bdate;

LOG.info(">>>>>bDate=" + findBeginDate);

}

String findEndDate = null;

if (param.containsKey("eDate")) {

String edate = param.getString("edate");

findBeginDate = edate;

LOG.info(">>>>>eDate=" + findEndDate);

}

// 分页

int skip = Optional.ofNullable(param.getInteger("skip")).orElse(0);

int limit = Optional.ofNullable(param.getInteger("limit")).orElse(10);

FindOptions options = new FindOptions();

options.setSkip(skip);

options.setLimit(limit);

options.setSort(new JsonObject().put("\_id", -1));

client.findWithOptions(config.getProperty("mongo.document", "log"), query, options, resultHandler -> {

if (resultHandler.succeeded()) {

List<JsonObject> data = resultHandler.result();

List<JsonObject> rData = new ArrayList<JsonObject>();

data.forEach(item -> {

JsonObject log = new JsonObject();

log.put("key", item.getJsonObject("\_id").getString("$oid"));

log.put("log", item.getString("message"));

log.put("version", item.getString("@version"));

log.put("date", item.getString("@timestamp").replaceAll("\\\"", ""));

log.put("path", item.getString("path"));

log.put("host", item.getString("host"));

rData.add(log);

});

LOG.info(">>>>>log=" + Json.encode(rData));

message.reply(respWriter.toSuccess(rData, sn));

}

message.reply(respWriter.toError(sn, RespCode.CODE\_504));

});

});

}

}

package com.chinamcom.framework.aftersale.model;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

public class TbAftersaleImageboxExample {

/\*\*

\* Table: tb\_aftersale\_imagebox

@mbggenerated 2016-11-09 13:35:22

\*/

protected String orderByClause;

/\*\*

\* Table: tb\_aftersale\_imagebox

@mbggenerated 2016-11-09 13:35:22

\*/

protected boolean distinct;

/\*\*

\* Table: tb\_aftersale\_imagebox

@mbggenerated 2016-11-09 13:35:22

\*/

protected List<Criteria> oredCriteria;

public TbAftersaleImageboxExample() {

oredCriteria = new ArrayList<Criteria>();

}

public void setOrderByClause(String orderByClause) {

this.orderByClause = orderByClause;

}

public String getOrderByClause() {

return orderByClause;

}

public void setDistinct(boolean distinct) {

this.distinct = distinct;

}

public boolean isDistinct() {

return distinct;

}

public List<Criteria> getOredCriteria() {

return oredCriteria;

}

public void or(Criteria criteria) {

oredCriteria.add(criteria);

}

public Criteria or() {

Criteria criteria = createCriteriaInternal();

oredCriteria.add(criteria);

return criteria;

}

public Criteria createCriteria() {

Criteria criteria = createCriteriaInternal();

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

oredCriteria.add(criteria);

}

return criteria;

}

protected Criteria createCriteriaInternal() {

Criteria criteria = new Criteria();

return criteria;

}

public void clear() {

oredCriteria.clear();

orderByClause = null;

distinct = false;

}

/\*\*

\* [STRATO MyBatis Generator]

\* Table: tb\_aftersale\_imagebox

@mbggenerated 2016-11-09 13:35:22

\*/

protected abstract static class GeneratedCriteria {

protected List<Criterion> criteria;

protected GeneratedCriteria() {

super();

criteria = new ArrayList<Criterion>();

}

public boolean isValid() {

return criteria.size() > 0;

}

public List<Criterion> getAllCriteria() {

return criteria;

}

public List<Criterion> getCriteria() {

return criteria;

}

protected void addCriterion(String condition) {

if (condition == null) {

throw new RuntimeException("Value for condition cannot be null");

}

criteria.add(new Criterion(condition));

}

protected void addCriterion(String condition, Object value, String property) {

if (value == null) {

throw new RuntimeException("Value for " + property + " cannot be null");

}

criteria.add(new Criterion(condition, value));

}

protected void addCriterion(String condition, Object value1, Object value2, String property) {

if (value1 == null || value2 == null) {

throw new RuntimeException("Between values for " + property + " cannot be null");

}

criteria.add(new Criterion(condition, value1, value2));

}

public Criteria andIdIsNull() {

addCriterion("id is null");

return (Criteria) this;

}

public Criteria andIdIsNotNull() {

addCriterion("id is not null");

return (Criteria) this;

}

public Criteria andIdEqualTo(Integer value) {

addCriterion("id =", value, "id");

return (Criteria) this;

}

public Criteria andIdNotEqualTo(Integer value) {

addCriterion("id <>", value, "id");

return (Criteria) this;

}

public Criteria andIdGreaterThan(Integer value) {

addCriterion("id >", value, "id");

return (Criteria) this;

}

public Criteria andIdGreaterThanOrEqualTo(Integer value) {

addCriterion("id >=", value, "id");

return (Criteria) this;

}

public Criteria andIdLessThan(Integer value) {

addCriterion("id <", value, "id");

return (Criteria) this;

}

public Criteria andIdLessThanOrEqualTo(Integer value) {

addCriterion("id <=", value, "id");

return (Criteria) this;

}

public Criteria andIdIn(List<Integer> values) {

addCriterion("id in", values, "id");

return (Criteria) this;

}

public Criteria andIdNotIn(List<Integer> values) {

addCriterion("id not in", values, "id");

return (Criteria) this;

}

public Criteria andIdBetween(Integer value1, Integer value2) {

addCriterion("id between", value1, value2, "id");

return (Criteria) this;

}

public Criteria andIdNotBetween(Integer value1, Integer value2) {

addCriterion("id not between", value1, value2, "id");

return (Criteria) this;

}

public Criteria andAfterIdIsNull() {

addCriterion("after\_id is null");

return (Criteria) this;

}

public Criteria andAfterIdIsNotNull() {

addCriterion("after\_id is not null");

return (Criteria) this;

}

public Criteria andAfterIdEqualTo(Integer value) {

addCriterion("after\_id =", value, "afterId");

return (Criteria) this;

}

public Criteria andAfterIdNotEqualTo(Integer value) {

addCriterion("after\_id <>", value, "afterId");

return (Criteria) this;

}

public Criteria andAfterIdGreaterThan(Integer value) {

addCriterion("after\_id >", value, "afterId");

return (Criteria) this;

}

public Criteria andAfterIdGreaterThanOrEqualTo(Integer value) {

addCriterion("after\_id >=", value, "afterId");

return (Criteria) this;

}

public Criteria andAfterIdLessThan(Integer value) {

addCriterion("after\_id <", value, "afterId");

return (Criteria) this;

}

public Criteria andAfterIdLessThanOrEqualTo(Integer value) {

addCriterion("after\_id <=", value, "afterId");

return (Criteria) this;

}

public Criteria andAfterIdIn(List<Integer> values) {

addCriterion("after\_id in", values, "afterId");

return (Criteria) this;

}

public Criteria andAfterIdNotIn(List<Integer> values) {

addCriterion("after\_id not in", values, "afterId");

return (Criteria) this;

}

public Criteria andAfterIdBetween(Integer value1, Integer value2) {

addCriterion("after\_id between", value1, value2, "afterId");

return (Criteria) this;

}

public Criteria andAfterIdNotBetween(Integer value1, Integer value2) {

addCriterion("after\_id not between", value1, value2, "afterId");

return (Criteria) this;

}

public Criteria andAfterCodeIsNull() {

addCriterion("after\_code is null");

return (Criteria) this;

}

public Criteria andAfterCodeIsNotNull() {

addCriterion("after\_code is not null");

return (Criteria) this;

}

public Criteria andAfterCodeEqualTo(String value) {

addCriterion("after\_code =", value, "afterCode");

return (Criteria) this;

}

public Criteria andAfterCodeNotEqualTo(String value) {

addCriterion("after\_code <>", value, "afterCode");

return (Criteria) this;

}

public Criteria andAfterCodeGreaterThan(String value) {

addCriterion("after\_code >", value, "afterCode");

return (Criteria) this;

}

public Criteria andAfterCodeGreaterThanOrEqualTo(String value) {

addCriterion("after\_code >=", value, "afterCode");

return (Criteria) this;

}

public Criteria andAfterCodeLessThan(String value) {

addCriterion("after\_code <", value, "afterCode");

return (Criteria) this;

}

public Criteria andAfterCodeLessThanOrEqualTo(String value) {

addCriterion("after\_code <=", value, "afterCode");

return (Criteria) this;

}

public Criteria andAfterCodeLike(String value) {

addCriterion("after\_code like", value, "afterCode");

return (Criteria) this;

}

public Criteria andAfterCodeNotLike(String value) {

addCriterion("after\_code not like", value, "afterCode");

return (Criteria) this;

}

public Criteria andAfterCodeIn(List<String> values) {

addCriterion("after\_code in", values, "afterCode");

return (Criteria) this;

}

public Criteria andAfterCodeNotIn(List<String> values) {

addCriterion("after\_code not in", values, "afterCode");

return (Criteria) this;

}

public Criteria andAfterCodeBetween(String value1, String value2) {

addCriterion("after\_code between", value1, value2, "afterCode");

return (Criteria) this;

}

public Criteria andAfterCodeNotBetween(String value1, String value2) {

addCriterion("after\_code not between", value1, value2, "afterCode");

return (Criteria) this;

}

public Criteria andImageUriIsNull() {

addCriterion("image\_uri is null");

return (Criteria) this;

}

public Criteria andImageUriIsNotNull() {

addCriterion("image\_uri is not null");

return (Criteria) this;

}

public Criteria andImageUriEqualTo(String value) {

addCriterion("image\_uri =", value, "imageUri");

return (Criteria) this;

}

public Criteria andImageUriNotEqualTo(String value) {

addCriterion("image\_uri <>", value, "imageUri");

return (Criteria) this;

}

public Criteria andImageUriGreaterThan(String value) {

addCriterion("image\_uri >", value, "imageUri");

return (Criteria) this;

}

public Criteria andImageUriGreaterThanOrEqualTo(String value) {

addCriterion("image\_uri >=", value, "imageUri");

return (Criteria) this;

}

public Criteria andImageUriLessThan(String value) {

addCriterion("image\_uri <", value, "imageUri");

return (Criteria) this;

}

public Criteria andImageUriLessThanOrEqualTo(String value) {

addCriterion("image\_uri <=", value, "imageUri");

return (Criteria) this;

}

public Criteria andImageUriLike(String value) {

addCriterion("image\_uri like", value, "imageUri");

return (Criteria) this;

}

public Criteria andImageUriNotLike(String value) {

addCriterion("image\_uri not like", value, "imageUri");

return (Criteria) this;

}

public Criteria andImageUriIn(List<String> values) {

addCriterion("image\_uri in", values, "imageUri");

return (Criteria) this;

}

public Criteria andImageUriNotIn(List<String> values) {

addCriterion("image\_uri not in", values, "imageUri");

return (Criteria) this;

}

public Criteria andImageUriBetween(String value1, String value2) {

addCriterion("image\_uri between", value1, value2, "imageUri");

return (Criteria) this;

}

public Criteria andImageUriNotBetween(String value1, String value2) {

addCriterion("image\_uri not between", value1, value2, "imageUri");

return (Criteria) this;

}

public Criteria andCreateTimeIsNull() {

addCriterion("create\_time is null");

return (Criteria) this;

}

public Criteria andCreateTimeIsNotNull() {

addCriterion("create\_time is not null");

return (Criteria) this;

}

public Criteria andCreateTimeEqualTo(Date value) {

addCriterion("create\_time =", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeNotEqualTo(Date value) {

addCriterion("create\_time <>", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeGreaterThan(Date value) {

addCriterion("create\_time >", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeGreaterThanOrEqualTo(Date value) {

addCriterion("create\_time >=", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeLessThan(Date value) {

addCriterion("create\_time <", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeLessThanOrEqualTo(Date value) {

addCriterion("create\_time <=", value, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeIn(List<Date> values) {

addCriterion("create\_time in", values, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeNotIn(List<Date> values) {

addCriterion("create\_time not in", values, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeBetween(Date value1, Date value2) {

addCriterion("create\_time between", value1, value2, "createTime");

return (Criteria) this;

}

public Criteria andCreateTimeNotBetween(Date value1, Date value2) {

addCriterion("create\_time not between", value1, value2, "createTime");

return (Criteria) this;

}

}

/\*\*

\* This class corresponds to the database table tb\_aftersale\_imagebox

@mbggenerated do\_not\_delete\_during\_merge 2016-11-09 13:35:22

\*/

public static class Criteria extends GeneratedCriteria {

protected Criteria() {

super();

}

}

/\*\*

\* [STRATO MyBatis Generator]

\* Table: tb\_aftersale\_imagebox

@mbggenerated 2016-11-09 13:35:22

\*/

public static class Criterion {

private String condition;

private Object value;

private Object secondValue;

private boolean noValue;

private boolean singleValue;

private boolean betweenValue;

private boolean listValue;

private String typeHandler;

public String getCondition() {

return condition;

}

public Object getValue() {

return value;

}

public Object getSecondValue() {

return secondValue;

}

public boolean isNoValue() {

return noValue;

}

public boolean isSingleValue() {

return singleValue;

}

public boolean isBetweenValue() {

return betweenValue;

}

public boolean isListValue() {

return listValue;

}

public String getTypeHandler() {

return typeHandler;

}

protected Criterion(String condition) {

super();

this.condition = condition;

this.typeHandler = null;

this.noValue = true;

}

protected Criterion(String condition, Object value, String typeHandler) {

super();

this.condition = condition;

this.value = value;

this.typeHandler = typeHandler;

if (value instanceof List<?>) {

this.listValue = true;

} else {

this.singleValue = true;

}

}

protected Criterion(String condition, Object value) {

this(condition, value, null);

}

protected Criterion(String condition, Object value, Object secondValue, String typeHandler) {

super();

this.condition = condition;

this.value = value;

this.secondValue = secondValue;

this.typeHandler = typeHandler;

this.betweenValue = true;

}

protected Criterion(String condition, Object value, Object secondValue) {

this(condition, value, secondValue, null);

}

}

}

package com.chinamcom.framework.monitor.verticle;

import java.util.ArrayList;

import java.util.List;

import org.apache.curator.framework.CuratorFramework;

import org.springframework.stereotype.Component;

import com.chinamcom.framework.common.constant.ConsumerConstant;

import com.chinamcom.framework.common.verticle.BaseVerticle;

import io.vertx.core.json.JsonArray;

import io.vertx.core.json.JsonObject;

/\*\*

\* 服务监控

\*

\* @author xuxg

\* @since 20161123

\*

\*/

@Component

public class ServiceVerticle extends BaseVerticle {

private String zkPath = config.getProperty("zk.path", "/");

private CuratorFramework curator;

public ServiceVerticle(CuratorFramework curator) {

this.curator = curator;

}

@Override

public void start() throws Exception {

vertx.eventBus().consumer(ConsumerConstant.MONITOR\_SERVICE, message -> {

JsonObject param = (JsonObject) message.body();

String sn = param.getString("sn");

String reply = null;

List<JsonObject> services = new ArrayList<JsonObject>();

try {

List<String> nodeList = curator.getChildren().forPath(zkPath + "/asyncMultiMap/\_\_vertx.subs");

if (!nodeList.isEmpty()) {

nodeList.forEach(item -> {

JsonObject node = new JsonObject();

node.put("service", item);

try {

List<String> hostList = curator.getChildren()

.forPath(zkPath + "/asyncMultiMap/\_\_vertx.subs/" + item);

if (!hostList.isEmpty()) {

JsonArray hosts = new JsonArray();

node.put("stat", 1);

hostList.forEach(host -> {

hosts.add(new String(host.getBytes()));

});

node.put("host", hosts);

} else {

node.put("stat", 0);

}

} catch (Exception e) {

}

services.add(node);

});

}

reply = respWriter.toSuccess(services, sn);

} catch (Exception e) {

LOG.error(e);

reply = respWriter.toError(sn);

}

message.reply(reply);

});

}

}

package com.chinamcom.framework.order.verticle;

import java.util.Optional;

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

import org.springframework.stereotype.Component;

import com.chinamcom.framework.common.constant.ConsumerConstant;

import com.chinamcom.framework.common.response.RespCode;

import com.chinamcom.framework.common.verticle.BaseVerticle;

import com.chinamcom.framework.order.services.ITbOrderService;

import io.vertx.core.json.JsonArray;

import io.vertx.core.json.JsonObject;

/\*\*

\* 订单服务

\*

\* @author xuxg

\* @since 20160901

\*

\*/

@Component

public class OrderVerticle extends BaseVerticle {

@Autowired

private ITbOrderService tbOrderService;

@Override

public void start() throws Exception {

vertx.eventBus().consumer(ConsumerConstant.ZM3C\_ORDER\_ADD, message -> {

LOG.info("received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

String sn = reqData.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<Integer> uid = Optional.ofNullable(reqData.getInteger("uid"));// 用户ID

if (uid.isPresent()) {

if (reqData.containsKey("product")) {// 直接下订单

// 调用产品服务，检索产品相关信息

JsonObject product = reqData.getJsonObject("product");

product.put("sn", sn);

product.put("prodcode", product.getString("code"));

product.put("prod", 1);

vertx.eventBus().send(ConsumerConstant.ZM3C\_PRODUCT\_QUERY, product, replyHandler -> {

if (replyHandler.succeeded()) {

JsonObject data = new JsonObject(replyHandler.result().body().toString());

LOG.info(data);

Integer code = data.getInteger("code");

if (code == 200) {

JsonArray prod = data.getJsonArray("respData");

reqData.getJsonObject("product").put("price",

prod.getJsonObject(0).getFloat("prodprice"));

reqData.getJsonObject("product").put("name",

prod.getJsonObject(0).getString("prodname"));

reqData.getJsonObject("product").put("id", prod.getJsonObject(0).getInteger("id"));

message.reply(tbOrderService.submitTbOrderInfo(sn, reqData));

} else {

LOG.info("未检索到产品信息，不能正常提交订单.");

message.reply(respWriter.toError(sn));

}

} else {

LOG.info("产品服务异常，未检索到产品信息，不能正常提交订单.");

message.reply(respWriter.toError(sn));

}

});

} else {// 从购物车下订单

reply = tbOrderService.submitTbOrderInfoByCart(sn, reqData);

LOG.info("reply:" + reply);

message.reply(reply);

}

} else {

reply = respWriter.toError(sn, "请核查请求参数是否符合规范.");

LOG.info("reply:" + reply);

message.reply(reply);

}

} catch (Exception e) {

LOG.error("提交订单异常：" + e);

reply = respWriter.toError(sn, "提交订单失败，请重新提交.");

message.reply(reply);

}

});

vertx.eventBus().consumer(ConsumerConstant.ZM3C\_ORDER\_LIST, message -> {

LOG.info("received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

String sn = reqData.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<Integer> uid = Optional.ofNullable(reqData.getInteger("uid"));// 用户ID

if (uid.isPresent()) {

reply = tbOrderService.selectTbOrderList(sn, reqData);

} else {

reply = respWriter.toError(sn, "请核查请求参数是否符合规范.");

}

} catch (Exception e) {

LOG.error("检索订单异常：" + e);

reply = respWriter.toError(sn, "检索订单失败，请重新提交.");

}

LOG.info("reply:" + reply);

message.reply(reply);

});

vertx.eventBus().consumer(ConsumerConstant.ZM3C\_ORDER\_INFO, message -> {

LOG.info("received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

String sn = reqData.getString("sn");

String reply = respWriter.toError(sn);

try {

JsonObject result = new JsonObject();

Optional<Integer> uid = Optional.ofNullable(reqData.getInteger("uid"));// 用户ID

Optional<Integer> orderid = Optional.ofNullable(reqData.getInteger("orderid"));// 订单ID

Optional<String> ordercode = Optional.ofNullable(reqData.getString("ordercode"));// 订单编号

if (uid.isPresent() && (orderid.isPresent() || ordercode.isPresent())) {

tbOrderService.selectTbOrderInfo(reqData, result);

if (result.containsKey("express")) {

vertx.eventBus().send(ConsumerConstant.EXPRESS\_LIST, new JsonObject()

.put("express", result.getInteger("express")).put("sn", sn).put("uid", uid.get()),

replyHandler -> {

if (replyHandler.succeeded()) {

JsonObject data = new JsonObject(replyHandler.result().body().toString());

LOG.info("express repley :" + data);

if (data.getInteger("code") == 200) {

JsonArray respData = data.getJsonArray("respData");

JsonObject ex = respData.getJsonObject(0);

result.remove("express");

ex.remove("userid");

ex.remove("provid");

ex.remove("cityid");

ex.remove("areaid");

ex.remove("createtime");

ex.remove("updatetime");

ex.remove("status");

result.put("express", ex);

message.reply(respWriter.toSuccess(result, sn));

} else {

LOG.info("没有取得地址信息，直接返回订单信息.");

message.reply(respWriter.toSuccess(result, sn));

}

} else {

LOG.info("调用地址服务异常，直接返回订单信息.");

message.reply(respWriter.toSuccess(result, sn));

}

});

} else {

reply = respWriter.toSuccess(result, sn);

LOG.info("reply:" + reply);

message.reply(reply);

}

} else {

reply = respWriter.toError(sn, RespCode.CODE\_505);

message.reply(reply);

}

} catch (Exception e) {

LOG.error("检索订单明细异常:" + e);

reply = respWriter.toError(sn, "检索订单明细失败.");

message.reply(reply);

}

});

vertx.eventBus().consumer(ConsumerConstant.ZM3C\_ORDER\_UPDATE, message -> {

LOG.info("update\_received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

String sn = reqData.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<String> ordercode = Optional.ofNullable(reqData.getString("ordercode"));// 订单编号

if (ordercode.isPresent()) {

reply = tbOrderService.updateTbOrderInfo(sn, reqData);

} else {

reply = respWriter.toError(sn, "请核查请求参数是否符合规范.");

}

} catch (Exception e) {

LOG.error("更新订单异常：" + e);

reply = respWriter.toError(sn, "更新订单失败，请重新提交.");

}

LOG.info("update\_reply:" + reply);

message.reply(reply);

});

}

}

package com.chinamcom.framework.order.verticle;

import java.util.Optional;

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

import org.springframework.stereotype.Component;

import com.chinamcom.framework.common.constant.ConsumerConstant;

import com.chinamcom.framework.common.response.RespCode;

import com.chinamcom.framework.common.verticle.BaseVerticle;

import com.chinamcom.framework.order.services.ICartService;

import io.vertx.core.json.JsonArray;

import io.vertx.core.json.JsonObject;

/\*\*

\* 购物车服务

\*

\* @author xuxg

\* @since 20160901

\*

\*/

@Component

public class CartVerticle extends BaseVerticle {

@Autowired

private ICartService cartServiceImpl;

@Override

public void start() throws Exception {

vertx.eventBus().consumer(ConsumerConstant.CART\_INCR, message -> {

LOG.info("cart\_incr\_received:" + message.body());

JsonObject param = (JsonObject) message.body();

String sn = param.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<Integer> instanceid = Optional.ofNullable(param.getInteger("instanceid"));// 购物车中产品的ID

Optional<Integer> uid = Optional.ofNullable(param.getInteger("uid"));// 用户ID

if (instanceid.isPresent() && uid.isPresent()) {

reply = cartServiceImpl.incrCart(sn, param);

} else {

reply = respWriter.toError(sn, RespCode.CODE\_505);

}

} catch (Exception e) {

LOG.error("购物车数量增加异常：" + e);

reply = respWriter.toError(sn, "操作失败，请重新提交.");

}

LOG.info("cart\_incr\_reply:" + reply);

message.reply(reply);

});

vertx.eventBus().consumer(ConsumerConstant.CART\_DECR, message -> {

LOG.info("cart\_decr\_received:" + message.body());

JsonObject param = (JsonObject) message.body();

String sn = param.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<Integer> instanceid = Optional.ofNullable(param.getInteger("instanceid"));// 购物车中产品的ID

Optional<Integer> uid = Optional.ofNullable(param.getInteger("uid"));// 用户ID

if (instanceid.isPresent() && uid.isPresent()) {

reply = cartServiceImpl.decrCart(sn, param);

} else {

reply = respWriter.toError(sn, RespCode.CODE\_505);

}

} catch (Exception e) {

LOG.error("购物车数量减少异常：" + e);

reply = respWriter.toError(sn, "操作失败，请重新提交.");

}

LOG.info("cart\_decr\_reply:" + reply);

message.reply(reply);

});

vertx.eventBus().consumer(ConsumerConstant.CART\_DEL, message -> {

LOG.info("cart\_del\_received:" + message.body());

JsonObject param = (JsonObject) message.body();

String sn = param.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<Integer> instanceid = Optional.ofNullable(param.getInteger("instanceid"));// 购物车中产品的ID

Optional<Integer> uid = Optional.ofNullable(param.getInteger("uid"));// 用户ID

if (instanceid.isPresent() && uid.isPresent()) {

reply = cartServiceImpl.delCart(sn, param);

} else {

reply = respWriter.toError(sn, RespCode.CODE\_505);

}

} catch (Exception e) {

LOG.error("删除购物车异常：" + e);

reply = respWriter.toError(sn, "操作失败，请重新提交.");

}

LOG.info("cart\_del\_reply:" + reply);

message.reply(reply);

});

vertx.eventBus().consumer(ConsumerConstant.CART\_ACCOUNT, message -> {

LOG.info("cart\_account\_received:" + message.body());

JsonObject param = (JsonObject) message.body();

String sn = param.getString("sn");

String reply = respWriter.toError(sn);

try {

JsonObject buy = new JsonObject();

Optional<Integer> uid = Optional.ofNullable(param.getInteger("uid"));// 用户ID

if (uid.isPresent()) {

cartServiceImpl.account(buy, param);

if (buy.containsKey("product")) {

vertx.eventBus().send(ConsumerConstant.EXPRESS\_LIST, param, ctx -> {

if (ctx.succeeded()) {

JsonObject o = new JsonObject(ctx.result().body().toString());

buy.put("address",

o.containsKey("respData") ? o.getJsonArray("respData") : new JsonArray());

} else {

LOG.info("调用地址服务出现异常:" + ctx.cause());

}

String r = respWriter.toSuccess(buy, sn);

LOG.info("cart\_account\_reply:" + r);

message.reply(r);

});

} else {

reply = respWriter.toError(sn, "请选择购物车中的商品.");

LOG.info("cart\_account\_reply:" + reply);

message.reply(reply);

}

} else {

reply = respWriter.toError(sn, "请求参数不符合规范，请重新提交.");

LOG.info("cart\_account\_reply:" + reply);

message.reply(reply);

}

} catch (Exception e) {

LOG.error("结算异常：" + e);

reply = respWriter.toError(sn, "结算失败，请重新提交.");

message.reply(reply);

}

});

vertx.eventBus().consumer(ConsumerConstant.CART\_CHECK, message -> {

LOG.info("cart\_account\_received:" + message.body());

JsonObject param = (JsonObject) message.body();

String sn = param.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<Integer> instanceid = Optional.ofNullable(param.getInteger("instanceid"));// 购物车中产品的ID

Optional<Integer> uid = Optional.ofNullable(param.getInteger("uid"));// 用户ID

if (instanceid.isPresent() && uid.isPresent()) {

reply = cartServiceImpl.checked(sn, param);

} else {

reply = respWriter.toError(sn, RespCode.CODE\_505);

}

} catch (Exception e) {

LOG.error("选中购物车产品异常：" + e);

reply = respWriter.toError(sn, "操作失败，请重新提交.");

}

LOG.info("cart\_account\_reply:" + reply);

message.reply(reply);

});

vertx.eventBus().consumer(ConsumerConstant.CART\_LIST, message -> {

LOG.info("cart\_list\_received:" + message.body());

JsonObject param = (JsonObject) message.body();

String sn = param.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<Integer> uid = Optional.ofNullable(param.getInteger("uid"));// 用户ID

if (uid.isPresent()) {

reply = cartServiceImpl.listCart(sn, param);

} else {

reply = respWriter.toError(sn, RespCode.CODE\_505);

}

} catch (Exception e) {

LOG.error("购物车列表异常：" + e);

reply = respWriter.toError(sn, "操作失败，请重新提交.");

}

LOG.info("cart\_list\_reply:" + reply);

message.reply(reply);

});

vertx.eventBus().consumer(ConsumerConstant.CART\_ADD, message -> {

LOG.info("cart\_add\_received:" + message.body());

JsonObject param = (JsonObject) message.body();

String sn = param.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<Integer> uid = Optional.ofNullable(param.getInteger("uid"));// 用户ID

Optional<String> code = Optional.ofNullable(param.getString("code"));// 产品编号

if (uid.isPresent() && code.isPresent()) {

// 调用产品服务，检索产品相关信息

JsonObject product = new JsonObject();

product.put("sn", sn);

product.put("prodcode", code.get());

product.put("prod", 1);

vertx.eventBus().send(ConsumerConstant.ZM3C\_PRODUCT\_QUERY, product, replyHandler -> {

if (replyHandler.succeeded()) {

JsonObject data = new JsonObject(replyHandler.result().body().toString());

LOG.info(data);

if (data.getInteger("code") == 200) {

JsonArray prod = data.getJsonArray("respData");

param.put("price", prod.getJsonObject(0).getFloat("prodprice"));

param.put("name", prod.getJsonObject(0).getString("prodname"));

param.put("id", prod.getJsonObject(0).getInteger("id"));

message.reply(cartServiceImpl.addCart(sn, param));

} else {

LOG.info("未检索到产品信息，加入购物车失败.");

message.reply(respWriter.toError(sn));

}

} else {

LOG.info("产品服务异常，加入购物车失败.");

message.reply(respWriter.toError(sn));

}

});

} else {

reply = respWriter.toError(sn, RespCode.CODE\_505);

message.reply(reply);

}

} catch (Exception e) {

LOG.error("加入购物车异常：" + e);

reply = respWriter.toError(sn, "操作失败，请重新提交.");

message.reply(reply);

}

});

}

}

package com.chinamcom.framework.aftersale.verticle;

import java.util.Optional;

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

import org.springframework.stereotype.Component;

import com.chinamcom.framework.aftersale.service.IAfterSaleService;

import com.chinamcom.framework.common.constant.ConsumerConstant;

import com.chinamcom.framework.common.response.RespCode;

import com.chinamcom.framework.common.verticle.BaseVerticle;

import io.vertx.core.eventbus.Message;

import io.vertx.core.json.JsonObject;

import io.vertx.rx.java.ObservableFuture;

import io.vertx.rx.java.RxHelper;

import rx.Observable;

/\*\*

\* 售后服务

\*

\* @author xuxg

\* @since 20160926

\*

\*/

@Component

public class AfterSaleVerticle extends BaseVerticle {

@Autowired

private IAfterSaleService afterSaleService;

@Override

public void start() throws Exception {

vertx.eventBus().consumer(ConsumerConstant.AFTERSALE\_LIST, message -> {

LOG.info("received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

String sn = reqData.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<Integer> uid = Optional.ofNullable(reqData.getInteger("uid"));

if (uid.isPresent()) {

reply = afterSaleService.selectTbAfterSale(sn, reqData);

} else {

reply = respWriter.toError(sn, RespCode.CODE\_505);

}

} catch (Exception e) {

LOG.error(e);

reply = respWriter.toError(sn, "检索售后服务信息异常,请重新提交.");

}

LOG.info("reply:" + reply);

message.reply(reply);

});

vertx.eventBus().consumer(ConsumerConstant.AFTERSALE\_INFO, message -> {

LOG.info("received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

String sn = reqData.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<Integer> uid = Optional.ofNullable(reqData.getInteger("uid"));

Optional<Integer> afterid = Optional.ofNullable(reqData.getInteger("afterid"));

Optional<String> aftercode = Optional.ofNullable(reqData.getString("aftercode"));

if (uid.isPresent() && (afterid.isPresent() || aftercode.isPresent())) {

reply = afterSaleService.selectTbAfterSaleInfo(sn, reqData);

} else {

reply = respWriter.toError(sn, RespCode.CODE\_505);

}

} catch (Exception e) {

LOG.error("检索售后服务详情出现异常：" + e);

reply = respWriter.toError(sn, "检索售后服务详情.");

}

LOG.info("reply:" + reply);

message.reply(reply);

});

vertx.eventBus().consumer(ConsumerConstant.AFTERSALE\_DEVICECHECK, message -> {

LOG.info(ConsumerConstant.AFTERSALE\_DEVICECHECK + ".received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

String sn = reqData.getString("sn");

try {

Optional<Integer> uid = Optional.ofNullable(reqData.getInteger("uid"));

Optional<String> devicescode = Optional.ofNullable(reqData.getString("devicescode"));// 设备编号

Optional<Long> date = Optional.ofNullable(reqData.getLong("date"));// 购买时期

Optional<String> code = Optional.ofNullable(reqData.getString("code"));// 验证码

Optional<String> smssn = Optional.ofNullable(reqData.getString("smssn"));

if (!uid.isPresent() || !devicescode.isPresent() || !date.isPresent() || !code.isPresent()

|| !smssn.isPresent()) {

message.reply(respWriter.toError(sn, RespCode.CODE\_505));

} else {

ObservableFuture<Message<Object>> authobservable = RxHelper.observableFuture();

ObservableFuture<Message<Object>> prodobservable = RxHelper.observableFuture();

vertx.eventBus().send(ConsumerConstant.ZM3C\_DEVICES\_QUERY, reqData, prodobservable.toHandler());// 通设备编号查询产品信息

vertx.eventBus().send(ConsumerConstant.VERIFY\_CODE, reqData, authobservable.toHandler());// 校验验证码是否正确

Observable.zip(authobservable, prodobservable, (b1, b2) -> new Object[] { b1.body(), b2.body() })

.subscribe(o -> {

JsonObject checked = new JsonObject(o[0].toString());

JsonObject prods = new JsonObject(o[1].toString());

checked.put("checked", true);// 测试

LOG.info(checked);

if (!checked.getBoolean("checked")) {

message.reply(respWriter.toError(sn, RespCode.CODE\_507));

}

if (prods.getInteger("code") != 200) {

message.reply(respWriter.toError(sn, RespCode.CODE\_504));

} else {

message.reply(respWriter.toSuccess(sn));

}

}, err -> {

message.reply(respWriter.toError(sn, "通过设备编号申请售后失败,请重新提交."));

});

}

} catch (Exception e) {

LOG.error("通过设备编号申请售后异常：" + e);

message.reply(respWriter.toError(sn, "通过设备编号申请售后失败,请重新提交."));

}

});

vertx.eventBus().consumer(ConsumerConstant.AFTERSALE\_DEVICE, message -> {

LOG.info(ConsumerConstant.AFTERSALE\_DEVICE + ".received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

String sn = reqData.getString("sn");

try {

Optional<Integer> uid = Optional.ofNullable(reqData.getInteger("uid"));

Optional<String> devicescode = Optional.ofNullable(reqData.getString("devicescode"));// 设备编号

Optional<Long> date = Optional.ofNullable(reqData.getLong("date"));// 购买时期

if (!uid.isPresent() || !devicescode.isPresent() || !date.isPresent()) {

message.reply(respWriter.toError(sn, RespCode.CODE\_505));

} else {

ObservableFuture<Message<Object>> prodobservable = RxHelper.observableFuture();

ObservableFuture<Message<Object>> expressobservable = RxHelper.observableFuture();

vertx.eventBus().send(ConsumerConstant.ZM3C\_DEVICES\_QUERY, reqData, prodobservable.toHandler());// 通设备编号查询产品信息

vertx.eventBus().send(ConsumerConstant.EXPRESS\_LIST, reqData, expressobservable.toHandler());// 通过用户ID取地址信息

Observable.zip(prodobservable, expressobservable, (b1, b2) -> new Object[] { b1.body(), b2.body() })

.subscribe(o -> {

JsonObject prods = new JsonObject(o[0].toString());

JsonObject exprs = new JsonObject(o[1].toString());

if (prods.getInteger("code") == 200) {

prods.getJsonObject("respData").put("express", exprs.getJsonArray("respData"));

message.reply(respWriter.toSuccess(prods, sn));

} else {

message.reply(respWriter.toError(sn, RespCode.CODE\_504));

}

}, err -> {

message.reply(respWriter.toError(sn, "通过设备编号申请售后失败,请重新提交."));

});

}

} catch (Exception e) {

LOG.error("通过设备编号申请售后异常：" + e);

message.reply(respWriter.toError(sn, "通过设备编号申请售后失败,请重新提交."));

}

});

vertx.eventBus().consumer(ConsumerConstant.AFTERSALE\_APPLEY, message -> {

LOG.info("received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

String sn = reqData.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<Integer> uid = Optional.ofNullable(reqData.getInteger("uid"));

Optional<Integer> type = Optional.ofNullable(reqData.getInteger("type"));// 1退货2换货3维修

Optional<String> ordercode = Optional.ofNullable(reqData.getString("ordercode"));// 订单编号

Optional<String> prodcode = Optional.ofNullable(reqData.getString("prodcode"));// 产品编号

Optional<Integer> express = Optional.ofNullable(reqData.getInteger("express"));// 收货ID

Optional<String> reason = Optional.ofNullable(reqData.getString("reason"));// 产品编号

if (!uid.isPresent() || !type.isPresent() || !ordercode.isPresent() || !prodcode.isPresent()

|| !reason.isPresent() || (type.get() != 1 && !express.isPresent())) {

reply = respWriter.toError(sn, RespCode.CODE\_505);

message.reply(reply);

} else {

// 检索订单信息

vertx.eventBus().send(ConsumerConstant.ZM3C\_ORDER\_LIST, reqData, replyHandler -> {

if (replyHandler.succeeded()) {

JsonObject data = new JsonObject(replyHandler.result().body().toString());

LOG.info("order repley :" + data);

if (data.getInteger("code") == 200) {

JsonObject o = data.getJsonObject("respData").getJsonArray("data").getJsonObject(0);

message.reply(afterSaleService.submitTbAfterSale(sn, reqData

.put("prods", o.getJsonArray("prods")).put("orderid", o.getInteger("oid"))));

} else {

LOG.info("取得订单信息失败，直接返回申请售后失败.");

message.reply(respWriter.toError(sn));

}

} else {

LOG.info("调用订单服务异常，直接返回申请售后失败.");

message.reply(respWriter.toError(sn));

}

});

}

} catch (Exception e) {

LOG.error(e);

reply = respWriter.toError(sn, "申请售后失败,请重新提交.");

message.reply(reply);

}

});

vertx.eventBus().consumer(ConsumerConstant.AFTERSALE\_ACCEPT, message -> {

LOG.info("received:" + message.body());

JsonObject reqData = (JsonObject) message.body();

String sn = reqData.getString("sn");

String reply = respWriter.toError(sn);

try {

Optional<Integer> afterid = Optional.ofNullable(reqData.getInteger("afterid"));// 售后服务ID

Optional<String> aftercode = Optional.ofNullable(reqData.getString("aftercode"));

Optional<Integer> type = Optional.ofNullable(reqData.getInteger("type"));// 值为1时客服受理售后服务；值为2时售后完成；值为3客服取消售后服务

if ((!afterid.isPresent() && !aftercode.isPresent()) || !type.isPresent()) {

message.reply(respWriter.toError(sn, RespCode.CODE\_505));

} else {

switch (type.get()) {

case 1:

reply = afterSaleService.handleTbAfterSale(sn, reqData);

break;

case 2:

reply = afterSaleService.handleTbAfterSale(sn, reqData);

break;

case 3:

reply = afterSaleService.handleTbAfterSale(sn, reqData);

break;

default:

reply = respWriter.toError(sn, RespCode.CODE\_505);

break;

}

}

} catch (Exception e) {

LOG.error(e);

reply = respWriter.toError(sn, "申请售后失败,请重新提交.");

}

LOG.info("reply:" + reply);

message.reply(reply);

});

}

}