尚品汇商城复习

版本：V 1.0

订单模块

一、业务介绍

订单业务在整个电商平台中处于核心位置，也是比较复杂的一块业务。是把“物”变为“钱”的一个中转站。

整个订单模块一共分四部分组成:

1. 结算:

   展示一个页面,根据用户 id 去查询出用户收货人和收货地址,让用户选择.获取到购物车中所有的选中的商品(购买清单).

   要解决 用户通过浏览器的回退页面无刷新的情况下,重复提交订单.

   使用流水号校验的方式解决

   要解决 用户通过浏览器的回退页面刷新的情况下,重复提交订单.

   当用户下单成功,删除购物车中所有选中的商品.

2. 下单:

   校验流水号---通过提交订单,不通过----给提示,不能重复提交

   下单之前 校验库存\校验价格

   保存订单信息,删除流水号

   发送一个 延迟消息,开始定时,时间到了 自动关单.

3. 对接支付服务

   跳转到支付页面.去支付

   支付成功,要拿到支付发送过来的消息,去修改订单状态----改为已支付

定时关单:

当延迟消息时间到了,订单模块消费这个消息,去校验这笔订单有没有支付,判断在支付宝有没有交易记录.都没有的情况下才能关单.还以关闭支付宝支付.

4. 对接库存管理系统

订单修改完状态后(已支付),会给库存发送消息,库存去判断这笔订单中的商品是否同一个库中,如果不在同一个库 需要调用订单服务进行拆单. 然后去锁定库存,扣减库存.

库存服务做完自己的事,给订单发送一个消息.订单拿到这个消息后去修改订单状态,改为待发货状态.

当库存那边 打包 叫快递,发货.发完货之后 是不是还会给订单发送消息,消息中装了订单 id+物流单号.订单拿到消息后可以更新状态---已发货,可以给用户展示物流信息.

二、结算页

入口：购物车点击计算按钮

分析

分析页面需要的数据：

1、得到用户信息

2、购物车中选择的商品列表

3、收货人信息

三、下单

1、数据结构

orderInfo ： 订单表

orderDetail ： 订单明细

id	主键。自动生成
consignee	收货人名称。页面获取
consignee_tel	收货人电话。页面获取
deliveryAddress	收货地址。页面获取
total_amount	总金额。计算
order_status	订单状态，用于显示给用户查看。设定初始值。
userId	用户 Id。从拦截器已放到请求属性中。
payment_way	支付方式（网上支付、货到付款）。页面获取
orderComment	订单状态。页面获取
out_trade_no	第三方支付编号。按规则生成
create_time	创建时间。设当前时间
expire_time	默认当前时间+1 天
process_status	订单进度状态，程序控制、 后台管理查看。设定初始值，
tracking_no	物流编号,初始为空，发货后补充
parent_order_id	拆单时产生，默认为空

id	主键，自动生成
order_id	订单编号，主表保存后给从表

sku_id	商品 id 页面传递
sku_name	商品名称，后台添加
img_url	图片路径，后台添加
order_price	商品单价，从页面中获取，并验价。
sku_num	商品个数，从页面中获取

2、分析下单：

1. 保存单据前要做校验：验库存、验证价格。
2. 保存单据: orderInfo orderDetail。
3. 保存以后把购物车中的商品删除。
4. 重定向到支付页面。

3、如何解决用户利用浏览器刷新和回退重复提交订单？

在进入结算页面是我们生产了一个流水号，然后保存到结算页面的隐藏元素中一份，Redis 中存一份，每次用户提交订单时都检查 reids 中流水号与页面提交的是否相符，如果相等可以提交，当订单保存以后把后台的流水号删除掉。那么第二次用户用同一个页面提交的话流水号就会匹配失败，无法重复保存订单。

3.1 修改结算页增加流水号的生成

实现类

```
@Override
public String getTradeNo(String userId) {
    // 定义key
    String tradeNoKey = "user:" + userId + ":tradeCode";
    // 定义一个流水号
  String tradeNo = UUID.randomUUID().toString().replace("-", "");
    redisTemplate.opsForValue().set(tradeNoKey, tradeNo);
    return tradeNo;
}
```

```
@Override
public boolean checkTradeCode(String userId, String tradeCodeNo) {
    // 定义key
    String tradeNoKey = "user:" + userId + ":tradeCode";
    String redisTradeNo = (String)
redisTemplate.opsForValue().get(tradeNoKey);
    return tradeCodeNo.equals(redisTradeNo);
}
//删除流水号
@Override
public void deleteTradeNo(String userId) {
    // 定义key
    String tradeNoKey = "user:" + userId + ":tradeCode";
    // 删除数据
  redisTemplate.delete(tradeNoKey);
}
```

3.2 OrderController 中的 submitOrder 方法中

```
/**
 * 提交订单
 * @param orderInfo
 * @param request
 * @return
 */
@PostMapping("auth/submitOrder")
public Result submitOrder(@RequestBody OrderInfo orderInfo,
HttpServletRequest request) {
    // 获取到用户Id
    String userId = AuthContextHolder.getUserId(request);
    orderInfo.setUserId(Long.parseLong(userId));
    // 获取前台页面的流水号
    String tradeNo = request.getParameter("tradeNo");
    // 调用服务层的比较方法
    boolean flag = orderService.checkTradeCode(userId, tradeNo);
    if (!flag) {
        // 比较失败!
        return Result.fail().message("不能重复提交订单！");
    }
    //  删除流水号
    orderService.deleteTradeNo(userId);
    // 验证库存:
    List<OrderDetail> orderDetailList =
orderInfo.getOrderDetailList();
```

```
    for (OrderDetail orderDetail : orderDetailList) {
        // 验证库存:
        boolean result =
orderService.checkStock(orderDetail.getSkuId(),
orderDetail.getSkuNum());
        if (!result) {
            return Result.fail().message(orderDetail.getSkuName() + "
库存不足!");
        }
        // 验证价格:
        BigDecimal skuPrice =
productFeignClient.getSkuPrice(orderDetail.getSkuId());
        if (orderDetail.getOrderPrice().compareTo(skuPrice) != 0) {
            // 重新查询价格!
            cartFeignClient.loadCartCache(userId);
            return Result.fail().message(orderDetail.getSkuName() + "
价格有变动!");
        }
    }
    // 验证通过, 保存订单!
    Long orderId = orderService.saveOrderInfo(orderInfo);
    return Result.ok(orderId);
}
```

4、验库存

通过 restful 接口查询商品是否有库存

一般电商系统的商品库存，都不由电商系统本身来管理，由另外一套仓库管理系统，或者进销存系统来管理，电商系统通过第三方接口调用该系统。

由于库管系统可能是异构的系统，所以不在微服务体系之内。只支持 restful 风格的 webservice 调用和消息队列的调用。

详见《库存管理系统手册》

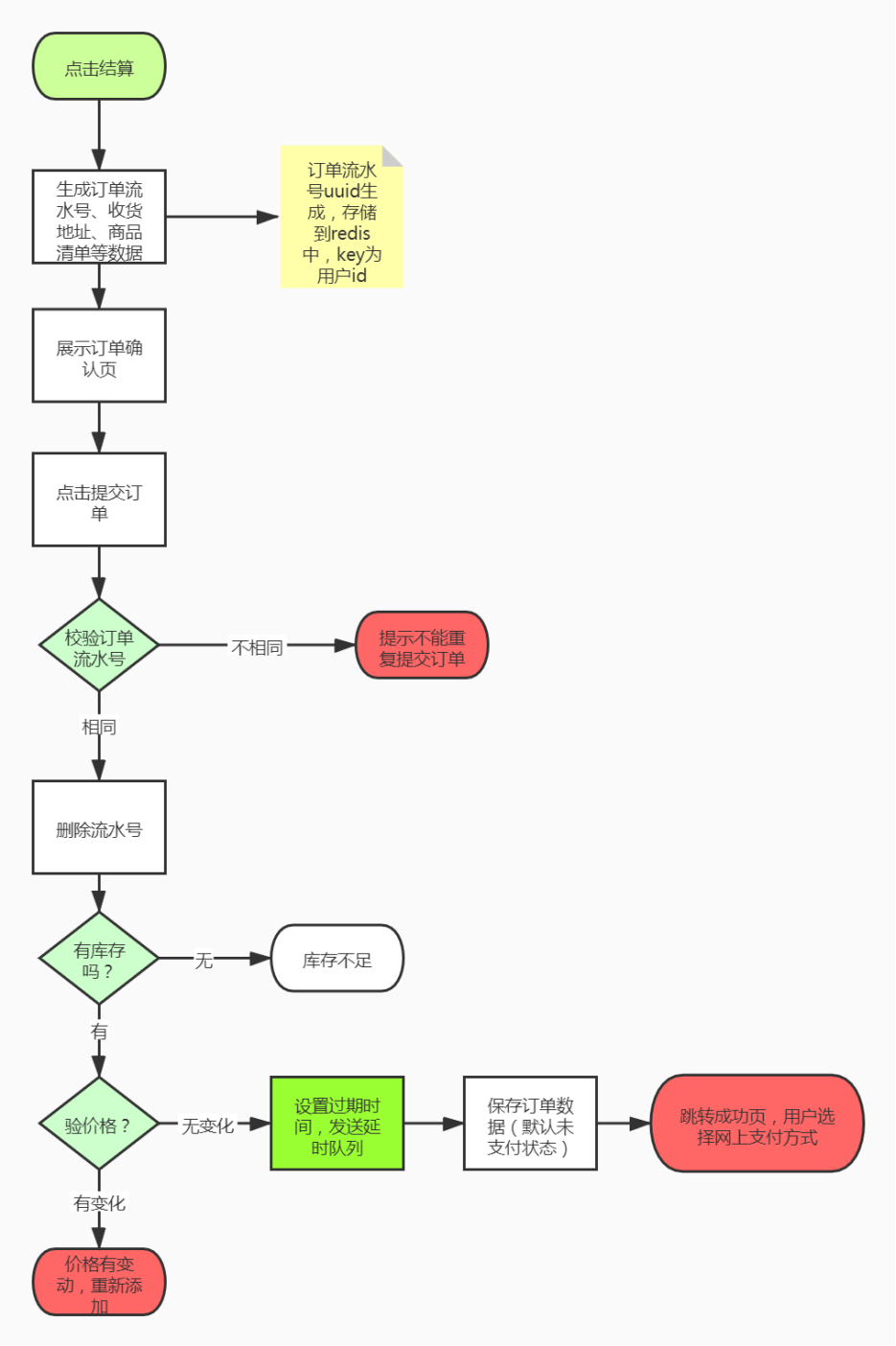
根据手册中的接口文档，编写调用代码。

查询库存接口

接口	/hasStock
请求参数	skuId： 商品 skuId num： 商品数量
请求方式	get
例	/hasStock?skuId=10221&num=2

四、业务流程和话术

1、订单业务流程图



2、业务话术（自己总结）

当用户点击结算的时候，这块我们用网关全局过滤器先判断用户是否登录，如果用户没有登录，则跳转到登陆页面，让用户去登录，登录成功之后，跳转到订单结算页面（这块在跳转到登录页面的时候，把之前的请求地址保存下来，作为参数进行跳转，在登录成功之后，查看是否有请求参数，如果有就跳转到对应的 url，如果值为 null，跳转到首页）；如果用户登录了，则跳转到订单结算页面；

当跳转到订单结算页面的时候，首先对收货人地址进行管理，其次选择支付方式，一期的时候只提供了支付宝支付（微信、支付宝），确认订单信息，然后提交数据到后台，生成对应的订单表、订单详情表和订单物流表（当订单生成的时候，我们要调用对应的库存系统针对订单的商品数量进行验库存，还要进行验价格）。当订单创建成功之后，自动跳转到成功页面（将订单数据和到期时间传递过去）。

这块我们设置的订单的有效时间为 24 小时（这个时间可以自己定，只要合理就行），因为我们利用延时队列实现定时消息发送，消费者到时间后监听到消息，进行订单校验，如果订单是未支付状态，把订单状态修改为关闭订单。

3、常见面试问题：

1、订单表结构（存了什么信息）？

主要的.

2、订单业务流程是什么样的？（订单你是怎么做的？）

3、订单有效期是多久，怎么取消订单？

合理就行 30 分钟 2 小时 24 小时 10 秒.

延迟消息.使用延迟插件.

4、怎么防止订单重复提交？

回退刷新----删除购物车选中商品

回退无刷新----流水号

5、订单超卖问题怎么解决的？

现在库存中这个商品就 1 件了,多人同时下单,你怎么解决?

咱们的:

没解决,多人都能下单成功. 超卖了.

这种 只要不是秒杀,不是绝版商品,不是限量的 可以.

好处: 库存不积压,损耗小. 不压 资金,提高现金流.

拼多多: 2 天内发货

小程序的商城: 以前微商。

淘宝

就要解决:

解决并发下单,针对订单购买的商品.

1、 分布式锁

2、 数据库锁

悲观锁:
select 库存量 from 库存表 where skuId=? for update;

事务提交或者回滚的时候。

乐观锁 :

在数据库加个字段 version

Select 库存量, version from 库存表 where skuId=?

Version =1

Update 库存表 set (库存量-购买量, version+1) where skuId=? and verison=查出来的 version 值