82 代码实战:编写代码实现支付订单和完成订单功能

儒猿架构官网上线,内有石杉老师架构课最新大纲,儒猿云平台详细介绍,敬请浏览

官网: www.ruyuan2020.com (建议 PC 端访问)

1、开篇

上周的课程介绍了互联网教育系统取消订单的功能,其中描述了主动取消订单和超时取消订单的场景,并且使用 RocketMQ 的延迟消息完成了超时取消订单的功能。本周的课程为延续后面的业务流程,订单创建之后会经历支付订单、完成订单、评论订单的环节,依旧会用到 RocketMQ 的相关技术。今天课程的内容包括以下几个部分:

- 修改 OrderController
- 修改 OrderService 和 OrderServiceImpl
- 修改 OrderMapper 接口与实现

2、修改 OrderController

由于订单的 Controller、Service 以及 Mapper 在前面几节课中都已经创建了,本节课的代码修改基本是在原有的基础上面进行调整。首先来看 OrderController 的部分,如图 1 所示,在 OrderController 类中增加了三个入口函数的调用,分别是 payOrder、finish 和 comment。其中 payOrder 方法是用来订单支付操作的,它会去调用 orderService 中的 payOrder 方法完成订单支付的功能。

finishOrder 方法会调用 orderService 中的 finishOrder 方法传入 orderId 用以完成订单操作。最后是 publishComment 方法,它是用来评论订单的,同样也会调用 orderService 中的 publishComment 方法。

```
spring
mapring
maprin
                                                                                                                          * @return 结果
                                                                                                                                                                                          public CommonResponse payOrder(Integer orderId) { return CommonResponse.success(orderService.payOrder(orderId)); }
                                                                                                                                                                                                  * 结束订单
                                                                                                                                                                                                        <u>@param</u> orderId 订单id
                                                                                                                                                                                               * @param orde
* @return 结果
   ▶ 🖿 dao
                                                                                                                                                                                         #C@RequestMapping("/finish") 完成
public CommonResponse finishOrder(Integer orderId) {
    return CommonResponse.success(orderService.finishOrder(orderId));
  b dao
b dto
b enums
b event
b exception
handler
           interceptor
                                                                                                                                                                                               * 订单发布评论
    imsimsimapper
                                                                                                                                                                                          * @param orderId 订单id

    CommentMapper
    ConsumerMapper

                                                                                                                                                                                               * @param score 评价
* @param commentContent 评论信息
* @return 结果

    Consumer Mapper
    CouponMapper
    OrderMapper
    TeacherMapper

                                                                                                                                                                                        ▼ laservice
▼ laservice
                               © LouponServiceImpl
                                 © • OrderEvenManagerImpl
                                  © № OrderServiceImpl
                                 © • TeacherServiceImpl
```

图 1 OrderController 修改

3、修改 OrderService 和 OrderServiceImpl

在 OrderController 中大量地用到了 OrderService 中的方法,由于 OrderService 是一个接口其具体的实现都放在 OrderServiceImpl 中,因此我们将目光聚焦到 OrderServiceImpl 上。如图 2 所示,在 OrderServiceImpl 中对应着 OrderController 中调用的 3 个方法。payOrder 方法会判断只有订单状态为"等待支付",就会调用 orderMapper 将订单的状态修改为"已支付"。

publishComment 方法会通过 buiderComment 方法将订单的评论信息进行封装,然后将该评论信息通过 commentMapper 保存到数据库中。

finishOrder 方法先会判断是否为已支付状态,在已支付状态的情况下才对订单的状态进行修改,将其修改为"等待评论"。随后通过 consumerService 中的 orderFinishInformReceiveCredits 方法更新积分信息,也就是增加消费者的积分。也会通过 couponService 中的 orderFinishReceiverCoupon 方法更新优惠券信息,也就是给消费者发放优惠券。

```
© 16 HealthController
© 16 JdbcDemoController
© 16 OrderController
© 16 TeacherController
                                                                    public int payOrder(Integer orderId) {
    Order order = orderMapper.findById(orderId);
    if (!OrderStatusEnum.MAITING.FOR.PAY.getStatus().equals(order.getStatus())){
        throw new BusinessException(EducationBusinessErrorCodeEnum.ORDER_CANT_PAID.getMsg());
    }
}
                                               dao dto enums
                                                                             ?
// 更新订单支付时间和订单状态
int <mark>updateRow</mark> = orderMapper.payOrder(orderId, OrderStatusEnum.PAID.getStatus());
return updateRow;
    nandler handler
                                                                     @Override
public int publishComment(Integer orderId, Integer score, String commentContent) { 评论
    interceptor
    ims jms
                                                                            // 根据id查询订单信息
Order order = orderMapper.findById(orderId);
if (order.getCommentId() != null) {
                                                                                  根据id查询订单信息
        mapper

1 % CommentMapper
1 % ConsumerMapper
1 % CouponMapper
1 % OrderMapper
1 % TeacherMapper
                                                                                   (order.getCommentId() != null) {
  throw new BusinessException(EducationBusinessErrorCodeEnum.CANT_COMMENT.getMsg());
                                                                              // 封装订单的评论信息
Comment comment = this.builderComment(orderId, score, commentContent, order);
// 添加评论
                                                                             commentMapper.add(comment);
// 更新订单评论信息
       © is ConsumerServicelmpl
© is CouponServicelmpl
© is OrderServicelmpl
© is TeacherServicelmpl
© is TeacherServicelmpl
© is ConsumerService
© is CouponService
© is CouponService
© is TeacherService
Use TeacherService
                                                                             return orderMapper.publishComment(comment.getId(), OrderStatusEnum.FINISHED.getStatus(), orderId);
                                                                     private Comment builderComment(Integer orderId, Integer score, String commentContent, Order order) {...}
                                                                     utilsvalidator
                                                                             order.setStatus(OrderStatusEnum.WAITING_FOR_COMMENT.getStatus());
                                                                            order.setstatus(rderstatuschum.wm.rlmo_rum_cumnenr.getstatus());
// 结束订单 更新订单状态和结束订单的时间
int updateRow = orderMapper.finishOrder(orderId, OrderStatusEnum.WAITING_FOR_COMMENT.getStatus());
if (updateRow > 0) {
    // 订单完成 积分增加
    if (order.getReceiveCredits() > 0) {
        consumerService.orderFinishInformReceiveCredits(order);
entMapper.xml
imerMapper.xml
inMapper.xml
dapper.xml
erMapper.xml
in.properties
perties
properties
iml
config.xml
                                                                                    // 订单完成 优惠券发放
if (order.getReceiveCouponId() != null) {
    couponService.orderFinishReceiverCoupon(order);
```

图 2 OrderServiceImpl 修改

4、修改 OrderMapper 接口与实现

说完了 OrderController 和 OrderService 就顺势来到了 OrderMapper,一般而言 OrderMapper 也是一个通过 Mybatis 访问数据库的接口,与之相对应地有一个 orderMapper.xml 文件是用来对接口进行实现的,说白了就是存放接口方法中访问数据库的 SQL 语句。

如图 3 所示,可以看到 payOrder 方法是通过 update t order 表对 status 进行更新的。

finishOrder 是通过 update t_order 表完成订单"完成"状态的更新。最后 publishComment 是通过 update t_order 表中的 comment_id 字段对评论进行更新的,在做这个操作之前会 先 insert t comment 记录到表中然后获取新增的 comment id 再到 t order 中更新。

图 3 OrderMapper.xml 修改

5、总结

本节课延续上节课创建订单的后续流程,完成了对应的代码实现。主要介绍了OrderController、OrderServiceImpl、OrderMapper 中对应代码的修改,其中包括支付订单、完成订单、评论订单的业务代码。下期会思考上述代码会有哪些问题,并提出基于MQ对完成订单模块异步化改造的方案,从而提升系统性能。这里将代码给大家,下期见,拜拜。