84_代码实战: 再次基于 MQ 的延时消息实现订单自动完成功能

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: <u>www.ruyuan2020.com</u> (建议 PC 端访问)

1、开篇

上节课首先讲述了支付订单到完成订单的业务流程,为了完成这一过程需要引入 Job 不断检查订单状态和支付的相关时间参数,这样会带来系统性能的问题。又通过取消订单也遇到类似的场景,联想到使用 RocketMQ 进行异步操作,从而提高性能也解耦了支付订单和完成订单的功能,于是引出了队列处理支付订单的方案。该方案中有两条执行流程,分别使用了延迟消息和及时消息来完成。

本节课会针对本节课将方案落地,带大家进行编码实现。今天课程的内容包括以下 几个部分:

- OrderServiceImpl 修改
- OrderDelayMessageListener 修改
- OrderMessageListener 修改

2、OrderServiceImpl 修改

订单功能入口都没有发生改变,主要的修改还是集中于 OrderServiceImpl,只需要调整其中支付订单和完成订单的业务逻辑即可。

如图 1 所示,在 OrderServiceImpl 类中,找到 payOrder 方法,在更新订单状态为已支付以后,会调用 orderEventManager 中的 payOrderEvent 方法将订单作为消息发送到 RocketMQ 队列。后续会有消息的消费者通过监听器去监听该消息。

```
▶ interceptor
                                                                               return updateRow:
   ims jms
                                                  120
121
                                                                       }
       © • MgConsumerConfiguration
       © 16 MqDelayConsumerConfigurati
© 16 MqProducerConfiguration
                                                                       public int payOrder(Integer orderId) {
    Order order = orderMapper.findById(orderId);
    if (!OrderStatusEnum.WAITING_FOR_PAY.getStatus().equals(order.getStatus())){
        throw new BusinessException(EducationBusinessErrorCodeEnum.ORDER_CANT_PAID.getMsg());
}
                                                  123 o 124
       © 'a OrderDelayMessageListener
       © • OrderMessageListener
  mapper mapper
  service
    ▼ 🛅 impl
                                                  128
                                                                                // 更新订单支付时间和订单状态
           © 'a ConsumerServiceImpl
                                                                              int updateRow = orderMapper.payOrder(orderId, OrderStatusEnum.PAID.getStatus());
if (updateRow > 0) {
    // 影响行数大于0, 发送支付订单的消息
          © 1 CouponServiceInpl
© 2 OrderEvenM agerImpl
                                                  130
          C G OrderServiceImpl
                                                                                      orderEventManager.payOrderEvent(order);
      © 1 TeacherServiceImpl
1 ConsumerService
                                                  134
135
136
137
138 ®Î
                                                                                return updateRow:
       CouponService

    OrderEventManager

    OrderService
    TeacherService

                                                                       @Override
public int publishComment(Integer orderId, Integer score, String commentContent) {
    // 根据id查询订单信息
    Order order = orderMapper.findById(orderId);
    if (order.getCommentId() != null) {
▶ □ utils
validator
```

图 1 支付订单之后发送支付订单消息

支付订单的消息根据上节课描述的业务流程来看,它是一个延迟消息,因此该消息会被延迟消费。与之不同的是完成订单的业务,如图 2 所示,在 OrderServiceImpl 类中找到 finishOrder 方法,在将订单状态更新为等待评论之后,也会发送消息。这个消息应该是一个即时消息,也会有消费者通过监听器进行监听。

```
OrderDelayMessageLis
                                            174
175
176 1
  © 🕆 OrderMessageListener
                                                               mappe
© CouponServiceImpl
© OrderEvenManagerImpl
© OrderServiceImpl
                                            180
181
182
183
184
185
                                                                     }
order.setStatus(OrderStatusEnum.WAITING_FOR_COMMENT.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 {
    // 影响行数大于0 发送完成订单的消息
    orderEventManager.orderFinishEvent(order);
      TeacherServiceImple
  1 GonsumerService
  1 Consumerative
Couponervice
CorderEventManager
CorderService
TeacherService
                                           186
187
188
189
190
191
                                                                        return undateRow:
utils
                                                                3
■ validato
```

图 2 完成订单以后发送即时消息

3、OrderEventManagerImpl 修改

由于 OrderServiceImp 中是调用的 OrderEventManagerImpl 进行消息的发送,接下来看看 OrderEventManagerImpl 需要做哪些调整。如图 3 所示,在 OrderEventManagerImpl 有一个 payOrderEvent 方法,显然是通过 sendOrderDelayMessage 进行延迟消息的发送,其中消息类型为 "FINISH_DELAY",也就是完成延迟。另外一个方法 orderFinishEvent,调用的 sendOrderMessage 方法用来发送即时消息,消息的类型是 "WX_FINISHED_ORDER"是完成订单消息。

```
© % OrderController
© % TeacherController
                                                    54

55 e1 ±

58

59

60

61 e1 ±

64

65

66 e1 =

67

68

69
                                                                         @Override
                                                                         public void createOrderEvent(Order order) { this.sendOrderDelayMessage(DelayMessageTypeEnum.PAY_DELAY,order); }
► 🛅 dao
⊳ 🛅 dto
event
exception
handler
                                                                         public void cancelOrderEvent(Order order) { this.sendOrderMessage(MessageTypeEnum.WX_CANCEL_ORDER, order); }
                                                                        @Override
public void payOrderEvent(Order order) {
    this.sendOrderDelayMessage(DelayMessageTypeEnum.FINISH_DELAY, order);
}
   interceptor
⊭ 🛅 jms
        MgConsumerConfiguration

    MqDelayConsumerConfiguration
    MqDelayConsumerConfiguration
    MqProducerConfiguration
    OrderDelayMessageListener
    OrderMessageListener
                                                    70
71
91
74
75
76
77
78
79
80
81
82
83
                                                                        public void orderFinishEvent(Order order) { this.sendOrderMessage(MessageTypeEnum.WX_FINISHED_ORDER, order); }
► 🛅 mapper
© 16 ConsumerServiceImpl
© 16 CouponServiceImpl
© 16 OrderEvenManagerImpl
                                                                          * <u>@param</u> messageTypeEnum 订单消息类型
* <u>@param</u> order 订单信息
                                                                        private void sendOrderMessage(MessageTypeEnum messageTypeEnum, Order order) {
    OrderMessageDTO orderMessageDTO = new OrderMessageDTO();
    orderMessageDTO.setMessageType(messageTypeEnum);
    orderMessageDTO.setContent(JSON.toJSONString(order));
           C G OrderServiceImpl
           © 1 TeacherServiceImpl

    ConsumerService
    CouponService
    CouponService
    OrderEventManager

                                                    84
85
                                                                                Message message = new Message();
```

图 3 发送两种类型的消息

4、OrderDelayMessageListener 修改

我们在业务中调用发送消息的代码,也介绍了根据不同的业务场景使用了不同的消息类型:延迟消息和及时消息。接下来介绍一下监听器部分的修改,由于消息的消费者都会定义一个消息监听器,当消息进入 RocketMQ 队列之后,监听器就会工作将监听到的消息进行处理。

如图 4 所示,OrderDelayMessageListener 的监听器在取消订单的章节就给大家介绍过。这里我们需要修改 consumeMessage 方法,在原有的方法体中加入一个"else if"分支。该分支判断消息类型是"FINISH_DELAY"的情况,并且针对完成延迟的订单进行后续操作。

```
@Override
public ConsumeOrderlyStatus consumeMessage(List<MessageExt> list, ConsumeOrderlyContext consumeOrderlyContext) {
    for (MessageExt ext : List) {
        LOGGER.info("延时消息队列消息总数{}",list.size());
        String content = new String(ext.getBody(), StandardCharsets.UTF_8);
        LOGGER.info("received order delay message:{}", content);
        // 汀单消息
         ▼ 🛅 controller
                         © % ConsumerController
                         © 12 HealthController
Controller
Controller
Controller
Controller
Controller
Controller
                                                                                                                                                                                                                    パーリー・ファイス
OrderDelayMessageDTO orderMessage = JSON.parseObject(content, OrderDelayMessageDTO.class);
// 订单内容
                                                                                                                                                                                                                   // 订单内容
Order order = JSON.parseObject(orderMessage.getContent(), Order.class);
DelayMessageTypeEnum delayMessageType = orderMessage.getDelayMessageType();
                                                                                                                                                                                                                                   【
if (DelayMessageTypeEnum.PAY_DELAY.equals(delayMessageType))
LOGGER.info("订单支付超时,现在开始取消订单,订单信息{}", order);
                                                                                                                                                                                                                                                     LOGGER LING
() 订单数词
order = orderMapper.findById(order.getId());
if (!OrderStatusEnum.WAITING_FOR_PAY.getStatus().equals(order.getStatus())){
    throw new BusinessException(EducationBusinessErrorCodeEnum.ORDER_CANT_CANCEL.getMsg());
                                                                                                                                                                                                                                                         order.setStatus(OrderStatusEnum.CANCELED.getStatus());
© In MqDelayConsumerConfiguration

© In MqDelayConsumerConfiguration

© In MqProducerConfiguration

© In OrderDelayMessageListener

© In OrderMessageListener
                                                                                                                                                                                                                                                    order.setStatus(OrderStatusEnum.CANCELED.getStatus());
// 更新订单规制的和订单状态
int updateRow = orderMapper.cancelOrder(order.getId(), OrderStatusEnum.CANCELED.getStatus());
if (updateRow > 0) {
    // 订单取消 积分还原
    if (order.getDeductCredits() > 0) {
        LOGGER.info("订单取消,积分还原");
        consumerService.orderCancelInformUnDeductCredits(order);
    }
     S Corden/MessageListerer

Imapper

Ima
                                                                                                                                                                                                                                                                       }
// 订单取消 优惠券还原
if (!StringUtils.isEmpty(order.getConsumerCouponId())) {
            LOGGER.info("订单取消, 优惠券还原");
            couponService.orderCancelUpdateCouponStatus(order);
                                                                                                                                                                                                                                  } else if (DelayMessageTypeEnum.FINISH_DELAY.equals(delayMessageType)) {
    // 订单完成
    order = orderMapper.findById(order.getId());
    if (107derStatusEnum.PAID.getStatus().equals(order.getStatus())) {
        throw new BusinessException(EducationBusinessErrorCodeEnum.ORDER_CANT_FINISHED.getMsg());
    }
       ▶ □ utils▶ □ validator
                                                                                                                                                                                                                                                         order.setStatus(OrderStatusEnum.WAITING FOR COMMENT.getStatus()):
                                                                                                                                                                                                                                                     // 知本以半 史朝以半心心和語果以早的別問
int updateRow = orderMapper.finishOrder(order.getId(), OrderStatusEnum.WAITING_FOR_COMMENT.getStatus());
if (updateRow > 0) {
// 订单完成 积分増加
rprise Spring % 6: TODO
```

图 4 修改 OrderDelayMessageListener 类

如图 5 所示,在后续操作中会针对订单,进行积分增加和优惠券发送的操作。

```
} else if (DelayMessageTypeEnum.FINISH_DELAY.equals(delayMessageType)) {
   // 订单完成
   order = orderMapper.findById(order.getId());
   if (!OrderStatusEnum.PAID.getStatus().equals(order.getStatus())){
       throw new BusinessException(EducationBusinessErrorCodeEnum.ORDER_CANT_FINISHED.getMsg());
   order.setStatus(OrderStatusEnum.WAITING_FOR_COMMENT.getStatus());
    // 结束订单 更新订单状态和结束订单的时间
   int updateRow = orderMapper.finishOrder(order.getId(), OrderStatusEnum.WAITING_FOR_COMMENT.getStatus());
   if (updateRow > 0) { // 订单完成 积分增加
       LOGGER. info("订单完成,积分增加");
       if (order.getReceiveCredits() > 0) {
           consumerService.orderFinishInformReceiveCredits(order);
       // 订单完成 优惠券发放
       LOGGER.info("订单完成, 优惠券发放");
       if (order.getReceiveCouponId() != null) {
           couponService.orderFinishReceiverCoupon(order);
```

图 5 积分增加和优惠券发放

5、OrderMessageListener 修改

上面讲的是延迟消息监听器的修改,这里继续及时消息监听器的修改。如图 6 所

示,修改 OrderMessageListener 类的 consumeMessage 方法。依旧在原来的方

法体中加入一个"else if"分支,处理的内容和延迟消息相似,进行积分增加和优惠券发送。

```
@Override
public ConsumeOrderlyStatus consumeMessage(List<MessageExt> list, ConsumeOrderlyContext consumeOrderlyContext) {
    for (MessageExt ext : list) {
        String content = new String(ext.getBody(), StandardCharsets.UTF_8);
        LOGGER.info("received order message:{}", content);
        // 订单词是
        OrderMessageDTO orderMessage = JSON.parseObject(content, OrderMessageDTO.class);
        // 订单内容
        Order order = JSON.parseObject(orderMessage.getContent(), Order.class);
        MessageTypeEnum messageType = orderMessage.getMessageType();
▶ □ aspect▶ □ cache

▼ □ controller

         c & ConsumerController
         © % HealthController
© % JdbcDemoController
         © % OrderController
© % TeacherController
► 🛅 dao
► 🛅 dto
▶ □ enums
▶ □ event
                                                                                                  if (messageType.equals(MessageTypeEnum.WX_CANCEL_ORDER)) {
    // 订单取消 积分还原
    if (order.getDeductCredits() > 0) {
        LOGGER.info("订单取消, 积分还原");
        consumerService.orderCancelInformUnDeductCredits(order);
    }
   exception
   nandler handler
   interceptor
                                                                                                            © % MgDelavConsumerConfigurat
        © 16 MqProducerConfiguration
© 16 OrderDelayMessageLia
        © □ OrderMessageListener
► Imapper

■ service
■ impl
                                                                                                    © 'a ConsumerServiceImpl
             © 'a CouponServiceImpl
             © © OrderEvenManagerImpl
             © 'a OrderServiceImpl
© 'a TeacherServiceImpl
                                                                                                            }
// 汀单完成 优惠券发放
// 汀单完成 优惠券发放");
if (order.getReceiveeCouponId() != null) {
    couponService.orderrinishReceiverCoupon(order);
}

■ GonsumerService

        1 CouponService

    GrderEventManager
    GrderService

         TeacherService
                                                                                                    「
LOGGER.info("订单消息处理完成");
▶ □ utils▶ □ validator
                                                                                            return ConsumeOrderlyStatus.SUCCESS;
```

图 6 OrderMessageListener 修改

6、总结

本节课使用 RocketMQ 实现了异步的订单完成功能,设计到延迟消息和及时消息的部分。修改代码包括: OrderServiceImpl、OrderEventManagerImpl、

OrderDelayMessageListener、OrderMessageListener等类文件。下节课带大家 讲上述修改的代码进行测试。这里将代码给大家,下期见,拜拜。

友情提示:本章讲述代码只是部分核心代码,完整代码请查阅文末链接中代码,谢谢。