77_代码实战：实现通过延时消息取消订单、退回优惠券、退回积分

1.开篇

上节课为了解决大量未支付超时订单扫表性能问题，基于 RocketMQ 的延时消息进行优化。在创建订单的时候发送延迟消息（30 分钟），消费者在接受到延迟消息的时候再判断是否支付，未支付则取消订单。本节课会带大家通过代码的方式实现上述功能。今天课程的内容包括以下几个部分：

- 消息队列的配置
- 消息队列的服务
- 消息队列的应用

2.消息队列的配置

首先需要在 pom 文件中引入 RocketMQ 的依赖。如图 1 所示，在 pom.xml 文件中加入对应的依赖项。

图 1 pom 依赖项

再到 application.properties 文件中加入和 RocketMQ 相关配置项。如图 2 所示，添加 name server 的服务器地址；topic 名字、producer group 名字、consumer

group 名字；延迟的 topic 名字、延迟的 consumer group 名字；还有延迟等级信息。

图 2 application.properties 配置信息

这些信息会在下面的消息队列服务中被使用到。

3.消息队列的服务

有了 RocketMQ 的基本配置以后再来看看对应的生产者和消费者的服务是如何编写的。

首先需要对传递的消息进行定义，由于需要适应两种场景：主动取消订单和超时取消订单，所以这里我们分别建立两个消息的实体类。

如图 3 所示，在 spring/dto 下面建立 OrderMessageDTO 类，里面会定义消息类型和消息内容。

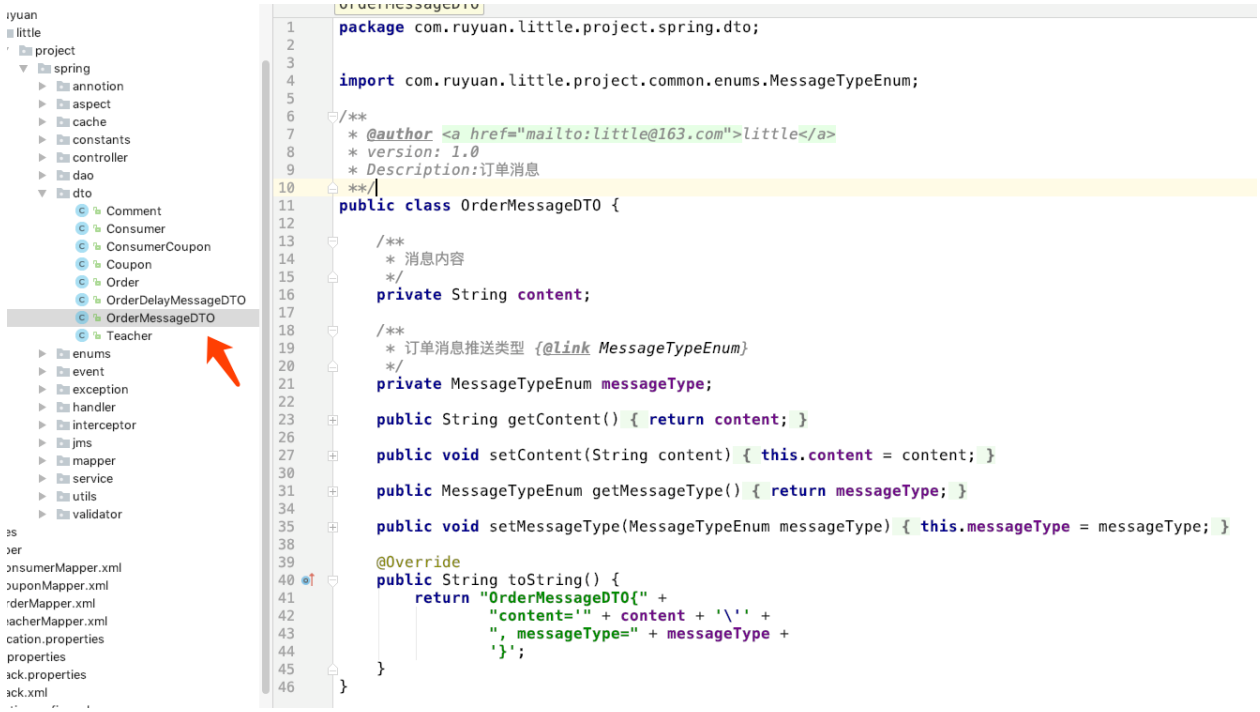


图 3 OrderMessageDTO

然后是 OrderDelayMessageDTO 的定义，内容和 OrderMessageDTO 保持一致。

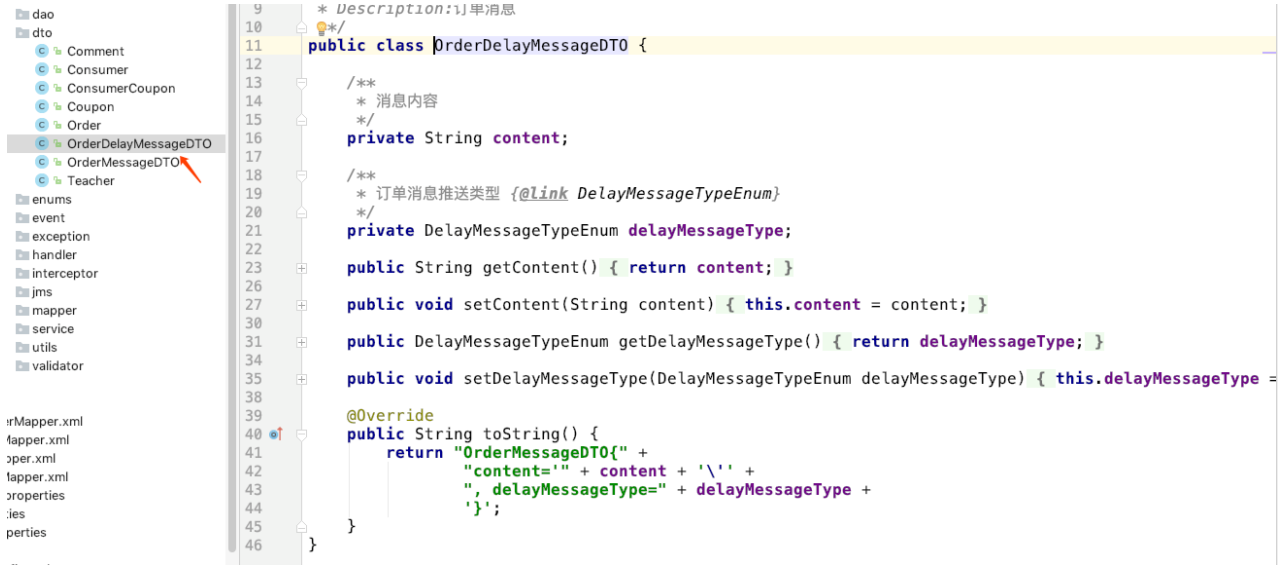


图 4 OrderDelayMessageDTO

如图 5 所示，在 spring/jms 目录下面创建 MqProducerConfiguration 类文件，它主要对消息生产者（Producer）的初始化和销毁做了定义。Init 方法主要为消息生产者创建生产者租，同时制定 name server 地址，并且启动这个生产者。而 destroy 方法则是用来关闭生产者服务。

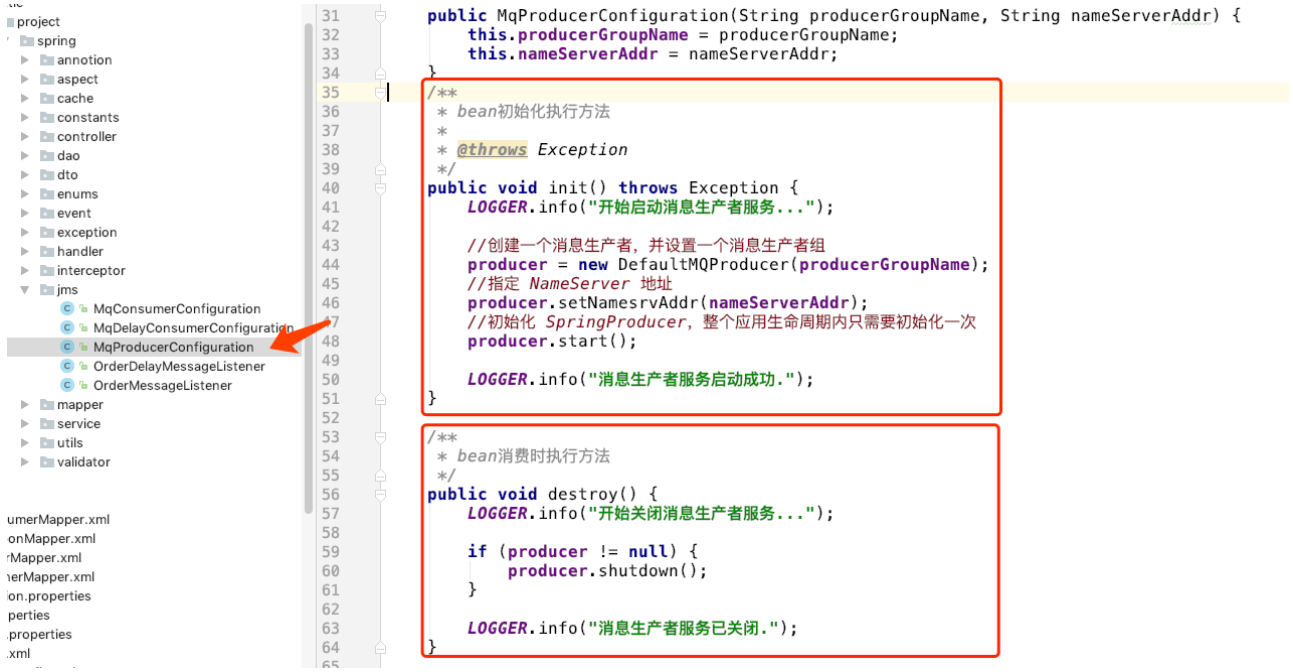


图 5 MqProducerConfiguration

定义完了生产者再来看看消费者的定义，如图 6 所示，MqConsumerConfiguration 类中对消息消费者做了定义，init 方法中定义了消费者组、name server 地址、并且设置消费消息的方式（从队列头部开始消费）、订阅的 topic 信息。最重要的是注册消息的监听器，这里将监听器指定为 orderMessageListener，后面会介绍这个监听器的职能，最后就是启动消费者服务。与 init 方法对应的是 destroy 方法用来停止消费者服务。

图 6 MqConsumerConfiguration

这里需要说明一下由于有两个取消订单场景：主动取消订单和超时取消订单。上面介绍的消费者 MqConsumerConfiguration 属于主动取消订单的功能。我们将为超时取消订单建立单独的消息消费者和监听器。

说完了主动取消订单的消费者，再来看看对应的监听器。如图 7 所示，OrderMessageListener 实现了 MessageListenerOrderly 并且 override 了 consumeMessage 方法。方法中对消息内容进行解析，在获取 orderMessage 之后判断消息类型是否是取消订单，如果是的话对积分也存在扣减的情况，那么就退还积分；如果同时也适用了优惠券，对优惠券也实现回退的操作。

图 7 OrderMessageListener

说完了主动取消订单的场景，再来看看超时订单的场景，它与前者的实现方式非常相似。

如图 8 所示，MqDelayConsumerConfiguration 中也定义了 init 和 destroy 方法。Init 中定义延迟消费者组、name server 地址、延迟消费的消费方式、订阅的 topic 信息。也注册了消息的监听器，这里将监听器指定为 orderDelayMessageListener。与 init 方法对应的是 destroy 方法用来停止消费者服务。

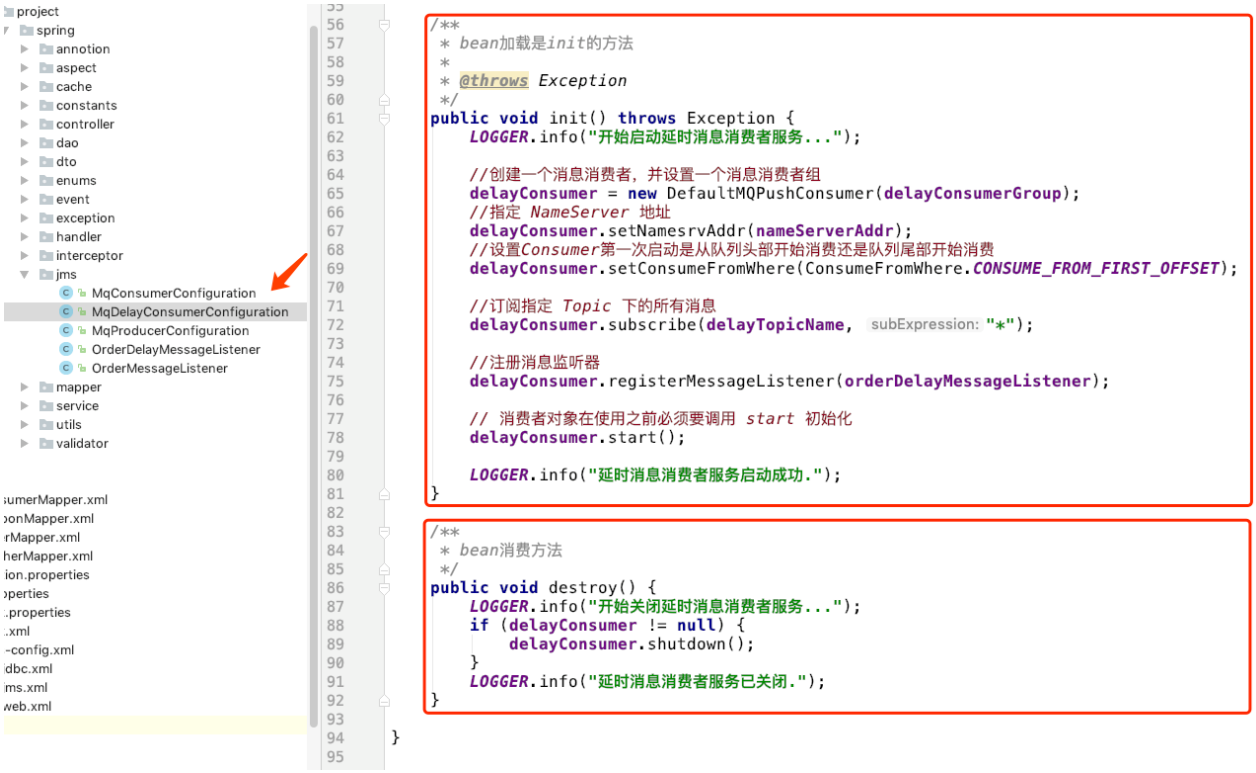


图 8 MqDelayConsumerConfiguration

最后来到 OrderDelayMessageListener 的部分，如图 9 所示，他是延迟消息消息费者的监听器，实现了 MessageListenerOrderly 并且 override 了 consumeMessage 方法。方法中对消息内容进行解析，在获取 orderMessage 之后判断消息类型是否是支付超时订单，如果是的话将订单取消并且更新订单状态。后面处理积分和优惠券的方式与主动取消订单如出一辙。

图 9 OrderDelayMessageListener

定义完消息体、消息生产者、消息消费者以及监听器以后，还需要对相关的 Bean 实例化。如图 10 所示，在 resources 目录下面建立 spring-jms.xml 文件对相关 Bean 进行实例化。我们将红框的部分从上往下介绍。

- 对订单消息 OrderMessageListener 实例化，它是主动取消订单的监听器。
- 对 MqConsumerConfiguration 订单消息消费者进行初始化配置，指明初始化方法“init”和销毁方法“destroy”。以及初始化所需要的参数 name server address、consumer group name 和 topic name。最重要的是指定监听器，也就是上面实例化的 OrderMessageListener。
- 对 OrderDelayMessageListener 进行实例化，这个是延迟消息的监听器。
- 对 MqDelayConsumerConfiguration 延迟订单消息消费者进行初始化配置，指明初始化方法 “init”和销毁方法“destroy”。 以及初始化所需要的参数 name server address、consumer group name 和 topic name。指定监听器是 OrderDelayMessageListener。
- 对 MqProducerConfiguration 订单消息生产者配置，也需要对“init”和“destroy”进行指定。同时制定 name server addres 和 producer group name 的信息。

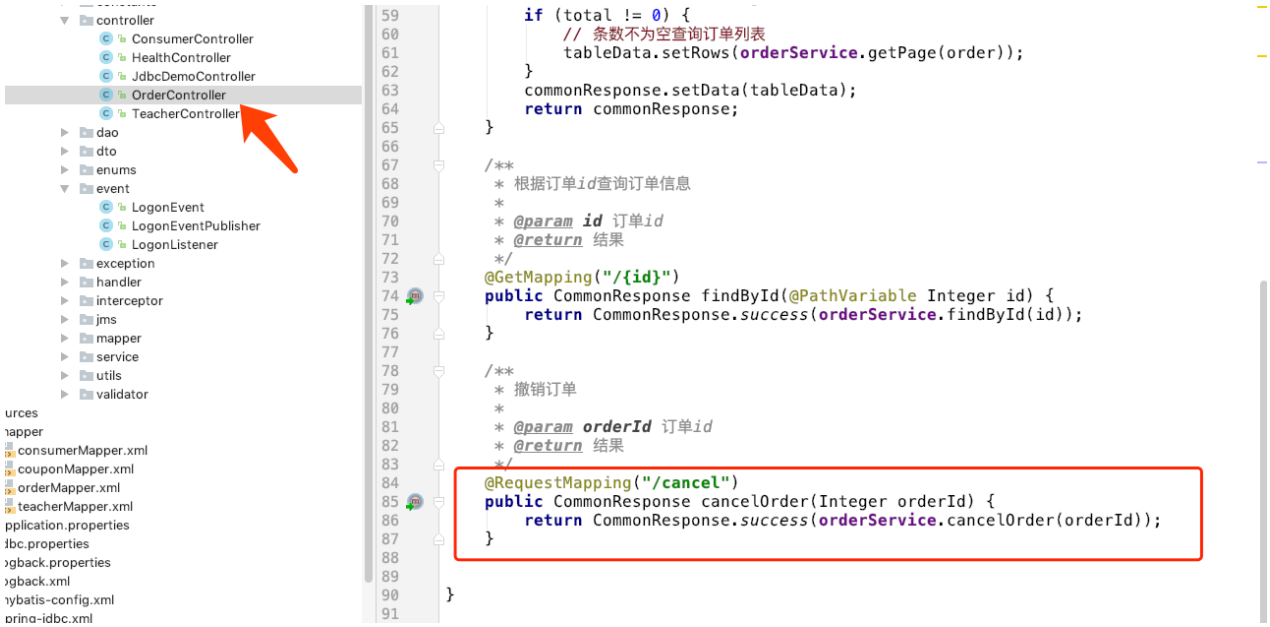
另外，本文件中读取的参数信息例如：“${rocketmq.namesrv.address}”这样的信息都是来自于 application.properties 文件中。

图 10 spring-jms.xml

4.消息队列的应用

消息队列应用的场景有主动取消和延迟取消，本节的重点在超时取消。下面先将主动取消的入口告诉大家，然后在转入超时取消的部分。

在 spring/controller 下面的 OrderController 中，加入 cancelOrder 方法，该方法调用 OrderService 中的 cancelOrder 对订单进行取消操作。



如图 11 OrderController 的 cancelOrder 方法

如图 12 所示，在 OrderServiceImpl 中的 cancelOrder 方法主要是更新订单状态，同时发送取消订单的消息。后面就是 OrderMessageListener 监听到该消息进行后续的取消订单处理。

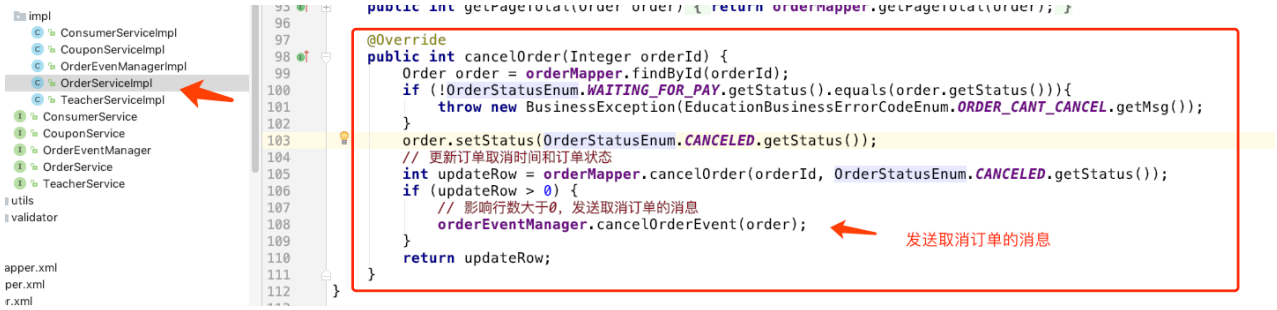


图 12 OrderServiceImpl 中的 cancelOrder 方法

说完了主动 取消订单再来看看超时取消订单的处理过程。如图 13 所示，在 OrderController 中有 add 方法用来创建订单。这里也是直接调用 OrderService 中的 createOrder 方法，传入 Order 对象作为输入参数。

图 13 OrderController 中的 add 创建订单

如图 14 所示，在 OrderServiceImpl 中的 createOrder 方法体中执行的内容和上周的基本一样，只是加入了一条发送消息的调用。这里调用了 OrderEventManager 中的 createOrderEvent 方法，传入的就是 Order 实体。

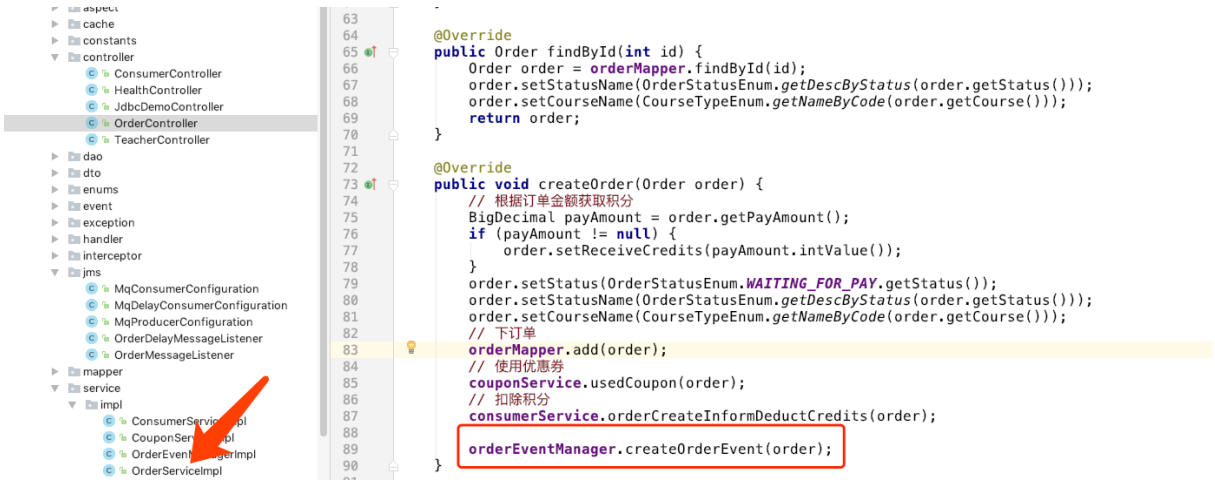


图 14 OrderServiceImpl 中的 createOrder

如图 15 所示，在 spring/service/impl 下面创建 OrderEventManagerImpl 类并且定义 createOrderEvent 方法，该方法会调用 sendOrderMessage 的私有方法。该方法定义了要发送 OrderMessageDTO，设置好 topic 和消息类型，通过 MQProducerConfiguration 生成的消息生产者发送这个延迟消息。发送延迟消息的内容和主动取消订单都是一样的，唯一不同就是会设置延迟消息的等级 OrderDelayLevel。

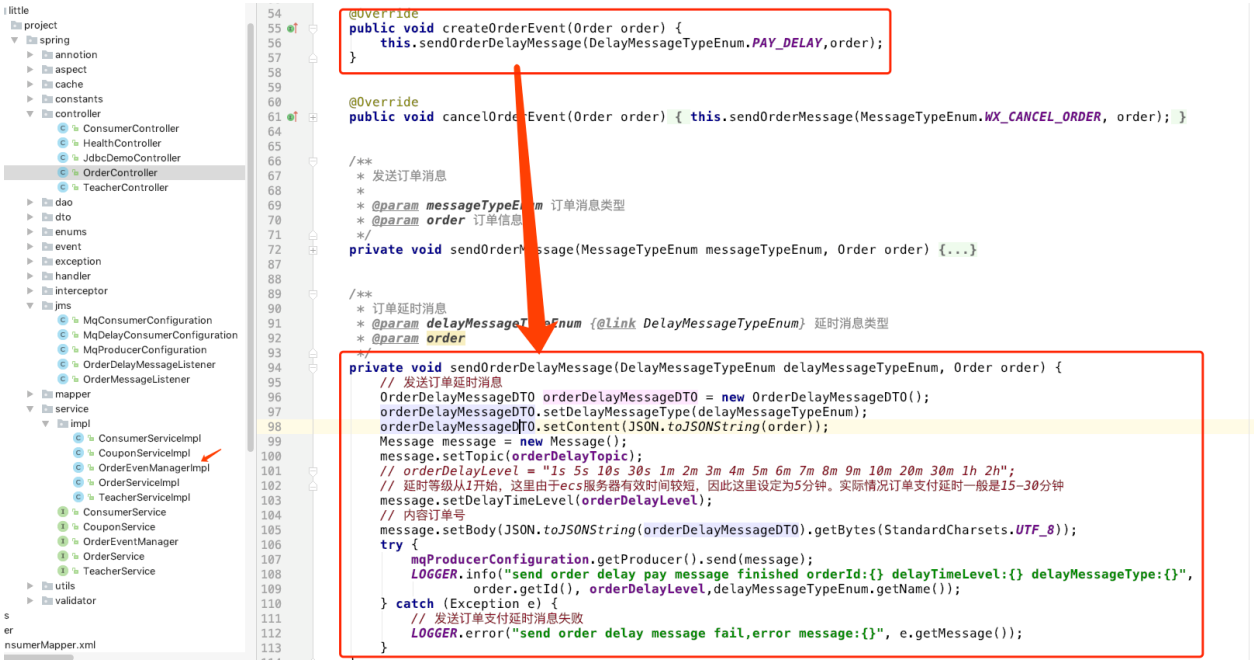


图 15 OrderEventManagerImpl 中的 createOrderEvent 方法

说明一下这里的 orderDelayLevel 是从 application.properties 中读取的，其值为 9. 这里的 9 是时间对应延迟时间的位移量，如果设置 8 就对应 4m，也就是 4 分钟。我们将每个延迟时间对应的 level 值的对应关系放到如下表格中，给大家参考，在测试的时候可以调整具体时间。

时间	1s	5s	10s	30s	1m	2m	3m	4m	5m	6m	7m	8m
Level	1	2	3	4	5	6	7	8	9	10	11	12
9m	10m	20m	30m	1h	2h							
13	14		16	17	18							

5.总结

本节课介绍了主动取消订单和延迟取消订单的代码实施，其中包括消息队列的配置：pom 文件和 appliaction.properties 文件；消息队列的服务：消息生产者、消息消费者、消息监听器；消息队列的应用：OrderServiceImpl 和 OrderEventManagerImpl。下节课会带大家进行取消订单业务代码的测试。这里将代码给大家，下期见，拜拜。