【第52话：京东二面死信队列】

Hello 小伙伴们，这节课给大家带来一个我们在实际授课过程中，一名同学在京东二面被问到的问题：“请说一下你的项目哪里使用死信队列”。

想要回答清楚这个问题，我们就必须先知道什么是死信队列。

想要明白死信队列，先给小伙伴们讲讲什么是死信消息（Dead-Letter Message）。以RabbitMQ进行说明：在RabbitMQ中，队列存储的消息如果在第一次没有被成功消费，会基于ACK进行消息重试，当重试次数达到最大后，表明消费者无法消费消息，会把消息放入到特殊的队列，这个消息就是死信消息。而专门存储死信消息的队列就是死信队列（Dead-Letter Queue）。

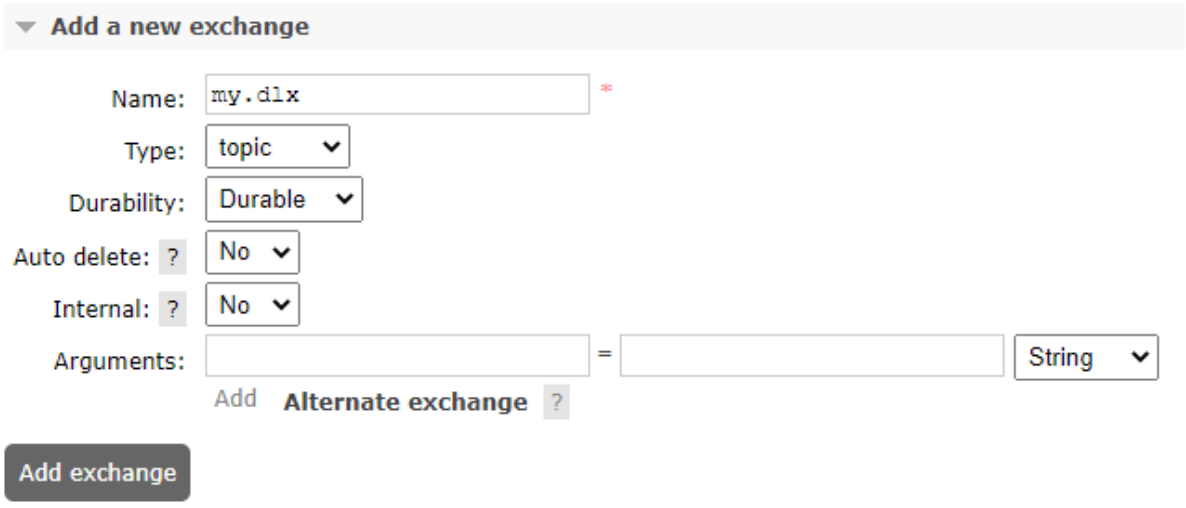
总结出来，下面几种情况的消息都会变成死信消息：

- 重试次数达到最大上限
- 消息被拒绝（basicNack 或 basicReject）或消息没有重新排队（requeue = false）
- 消息过期。这种智能出现在设置了过期时间（TTL）的消息中。
- 队列内消息达到最大长度。

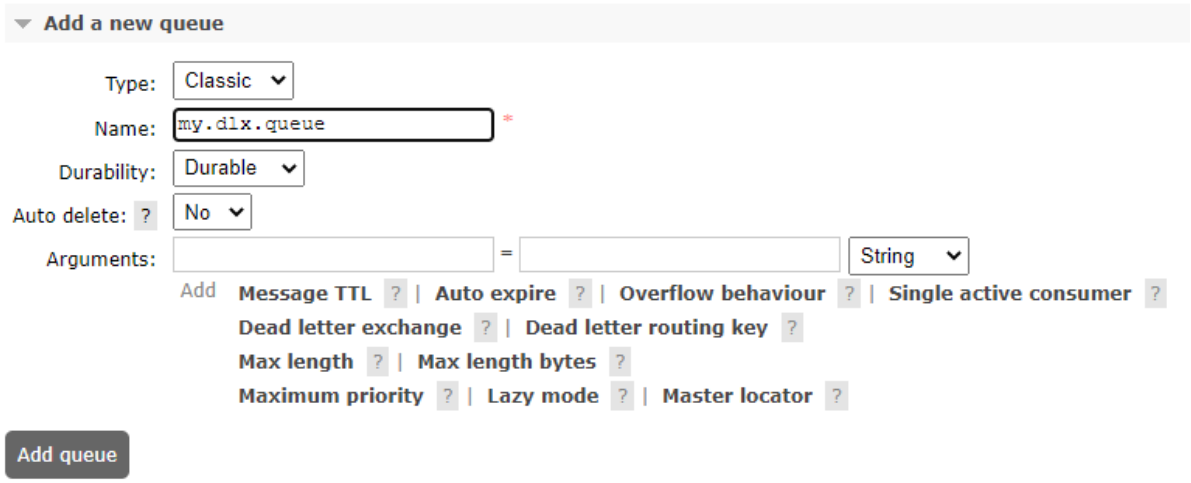
实践是检验真理的唯一标准，为了能够让小伙伴们更加清晰的理解清楚死信队列，我们上代码。

代码演示死信队列的使用

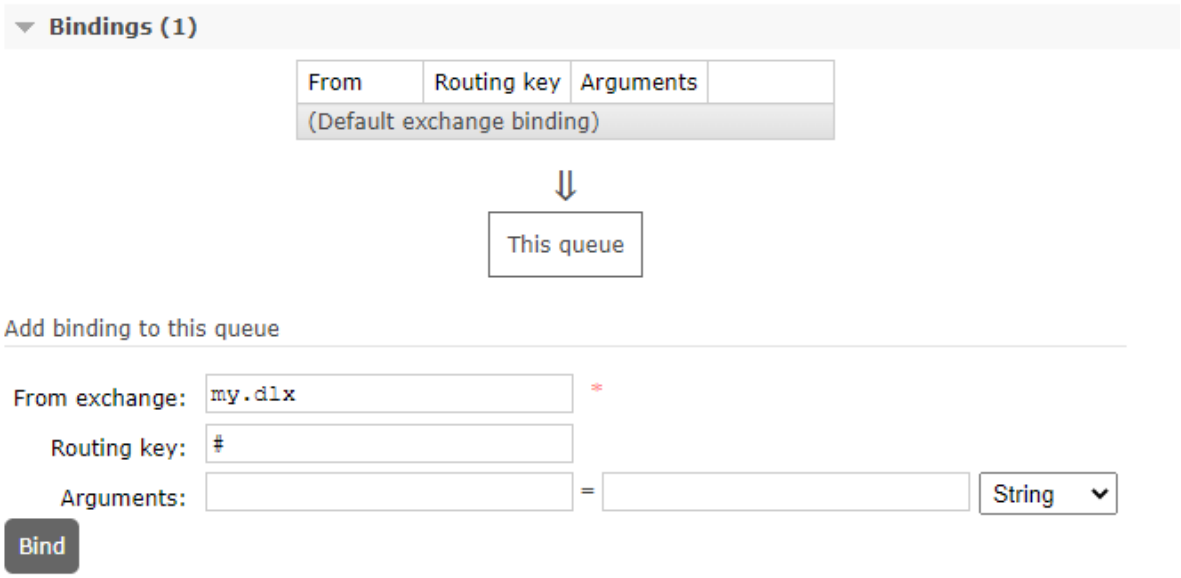
1. 先创建个DLX（死信交换器），虽然这里面叫做DLX，实际上就是一个普通的Topic交换器。交换器的名称任意，这里我命名为my.dlx。



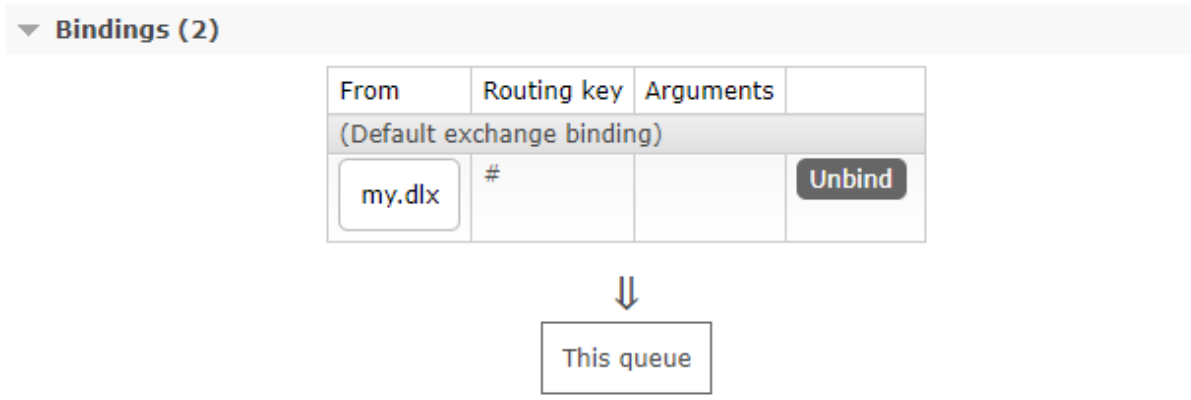
2. 创建DLQ（死信队列）并绑定到DLX（死信交换器）上。这里的DLQ就是一个普通的队列。因为它最后存储的是死信消息，所以称为死信队列



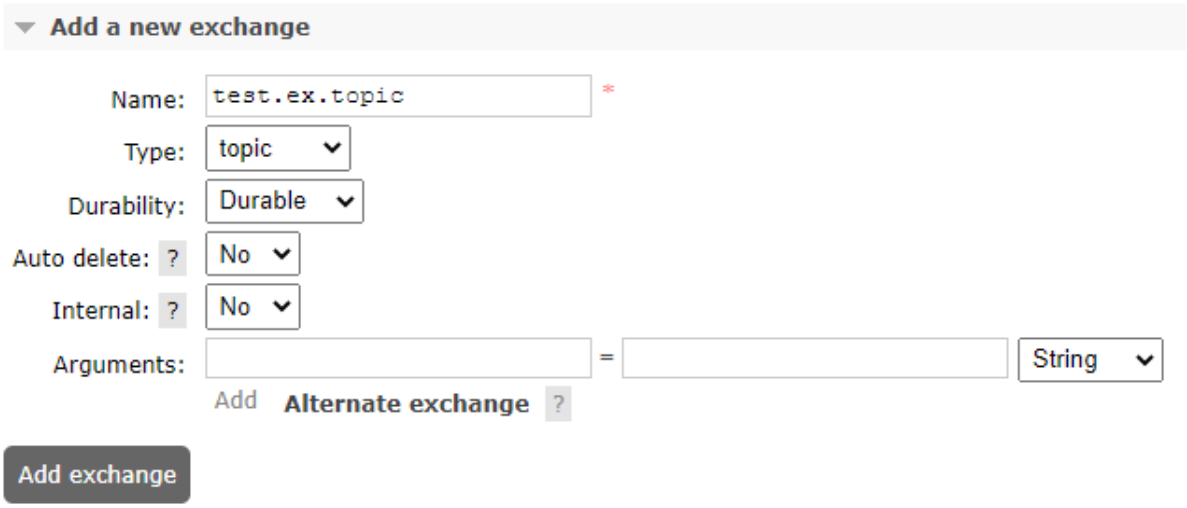
3. 把DLQ绑定到DLX上.



结果如下:

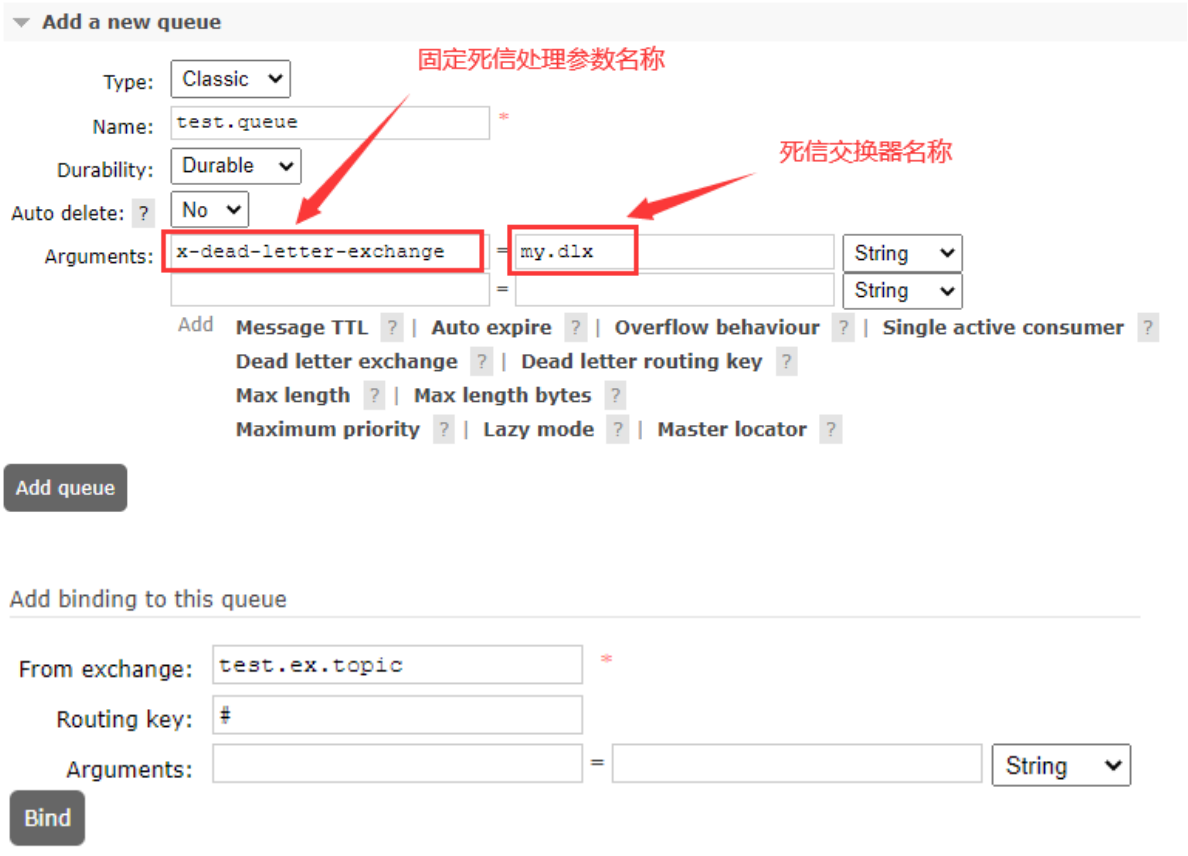


4. 创建任意交换器,这个交换器负责测试正常消息的。



5. 创建任意队列并绑定到交换器

创建任意队列，绑定到的交换器。注意：必须设置队列参数 x-dead-letter-exchange，此参数用于绑定死信处理逻辑，即信息成为死信后，投递到哪个死信交换器。



6. 创建死信队列消费者

创建消费者类：com.bjsxt.consumer.DLXMessageConsumer

```
@Component
public class DLXMessageConsumer {
    @RabbitListener(bindings = {
            @QueueBinding(
                    value = @Queue(name = "my.dlx.queue", autoDelete = "false"),
                    exchange = @Exchange(name = "my.dlx", type = "topic"),
                    key = {"#"}
```

```
            )
    })
    public void onMessage(String messageBody, Channel channel,
                          @Header(AmqpHeaders.DELIVERY_TAG) long deliveryTag){
        System.out.println("处理死信队列中的消息：" + messageBody);

        try {
            channel.basicAck(deliveryTag, true);
        } catch (IOException e){
            e.printStackTrace();
        }
    }
}
```

7. 编辑发送消息测试方法

在测试类中测试发送消息的方法

```
    @Test
    public void testDLX(){
        String messageBody = "投递到test.queue队列中的消息，等待超时后投递到DLX中再处
理";
        MessageProperties messageProperties =
                new MessageProperties();
        // 设置消息持久化
        messageProperties.setDeliveryMode(MessageDeliveryMode.PERSISTENT);
        // 设置消息体字符集
        messageProperties.setContentEncoding("UTF-8");
        // 设置消息超时时间，单位毫秒，参数类型是字符串
        messageProperties.setExpiration("30000");
        // 基于字符串消息体内容和消息参数，创建要传递的消息对象。
        Message message = new Message(
                        messageBody.getBytes(),
                        messageProperties
                );
        // 发送消息到队列，等待超时后消息成为死信，并转投到死信队列后，被相应消费者处理。
        rabbitOperations.send(
                "test.ex.topic",
                "routing.key",
                message
        );
    }
```

8. 观察结果

消息发送到普通队列 test.queue

test.queue	classic	D DLX Args	idle	1	0	1	0.20/s	0.00/s	0.00/s

消息具体内容如下：

The server reported 0 messages remaining.

Exchange	test.ex.topic
Routing Key	routing.key
Redelivered	○
Properties	expiration: 30000 priority: 0 delivery_mode: 2 headers: content_encoding: UTF-8 content_type: application/octet-stream
Payload 79 bytes Encoding: string	投递到test.queue队列中的消息，等待超时后投递到DLX中再处理

超时后消息投递到死信队列中：

my.dlx.queue	classic	D Args	idle	1	0	1		0.00/s	0.00/s

启动死信队列消费者消费死信：

```
2021-11-12 14:24:04.130  INFO 20348 --- [           main] o.s.a.r.c.CachingConnectionFactory
处理死信队列中的消息：投递到test.queue队列中的消息，等待超时后投递到DLX中再处理
2021-11-12 14:24:04.206  INFO 20348 --- [           main] com.bjsxt.AmqpConsumerApp
```

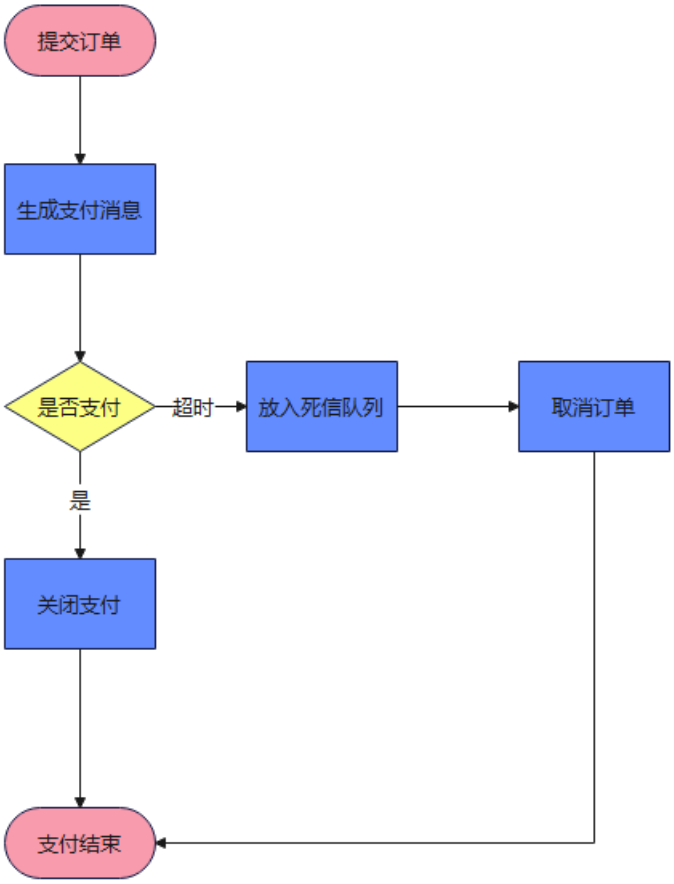
消费结束后，死信队列状态如下：

my.dlx.queue	classic	D Args	idle	0	0	0		0.00/s	0.00/s

通过上面效果，小伙伴们应该明白死信队列的工作流程了。

所以，对于一些需要延迟消费的消息，如：电商系统中用于关闭超时未支付的订单、铁路售票系统中恢复超时未支付的车票等。都可以使用死信队列。

在项目中使用死信队列完成订单超时支付流程图



在提交订单后就生成支付消息。如果用户支付了会 获取消息。消息正常被消费。如果超过30分钟或45分钟用户没有支付，消息进入到死信队列。监听死信队列的消费者完成未支付逻辑。