RabbitMq特性

如何确保消息一定发送到Rabbitmq了？

默认情况下，我们不知道我们的消息到底有没有发送到rabbitmq当中，这肯定是不可取的， 假设我们是一个电商项目的话，用户下了订单，订单发送消息给库存，结果这个消息没发送到rabbitmq当中，但是订单还是下了，这时候因为没有消息，库存不会去减少库存， 这种问题是非常严重的， 所以接下来就讲一种解决方案： 失败回调。

失败回调， 顾名思义就是消息发送失败的时候会调用我们事先准备好的回调函数，并且把失败的消息和失败原因等返回过来。

具体操作：

**注意：使用失败回调也需要开启发送方确认模式**

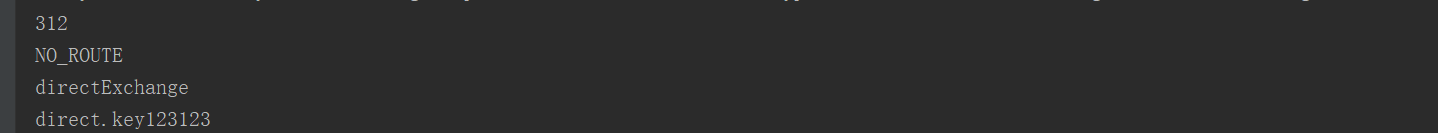
@Bean  
public RabbitTemplate rabbitTemplate(ConnectionFactory connectionFactory) {  
 RabbitTemplate template = new RabbitTemplate(connectionFactory);  
 //开启mandatory模式（开启失败回调）  
 template.setMandatory(true);  
 //指定失败回调接口的实现类  
 template.setReturnCallback(new MyReturnCallback());  
 return template;  
}

回调接口的实现类：

实现RabbitTemplate.ReturnCallback里面的returnedMessage方法即可

public class MyReturnCallback implements RabbitTemplate.ReturnCallback {  
  
 @Override  
 public void returnedMessage(Message message, int replyCode, String replyText, String exchange, String routingKey) {  
 System.out.println(message);  
 System.out.println(replyCode);  
 System.out.println(replyText);  
 System.out.println(exchange);  
 System.out.println(routingKey);  
 }  
}

模拟一个失败的发送： 当指定的交换机不能把消息路由到队列时（没有指定路由键或者指定的路由键没有绑定对应的队列或者压根就没有绑定队列都会失败）消息就会发送失败效果:



分别打印的是错误状态码，错误原因（这里的原因是不能路由）交换机名字 和路由键（还有个参数是你发送出去的消息因为太长了就没截图了.）

可能会想到了一个答案-----事务

没错事务的确能解决这个问题， 而且恰巧rabbitmq刚好也支持事物， 但是！ 事务非常影响rabbitmq的性能，有多严重？ 据了解，开启rabbitmq事物的话 对性能的影响超过100倍之多，也就是说开启事物后处理一条消息的时间是不开事物能处理100条， 那么这样是非常不合理的， 因为消息中间件的性能其实非常关键的（参考双11） 如果这样子做的话，虽然能确保消息100%投递成功，但是代价太大了！

那么除了事务还有什么解决方案吗？

rabbitmq其实还提供了一种解决方案，叫**发送方确认模式**，这种方式对性能的影响非常小，而且也能确定消息是否发送成功。

而且发送方确认模式一般也会和失败回调一起使用，这样就能确保消息100%投递了

发送方确认开启:

connectionFactory.setPublisherConfirms(true);

如果是yml配置的话：

spring:  
 rabbitmq:  
 publisher-confirms: true

和失败回调一样需要实现一个接口:

public class MyConfirmCallback implements RabbitTemplate.ConfirmCallback{  
  
 @Override  
 public void confirm(CorrelationData correlationData, boolean ack, String cause) {  
 System.out.println(correlationData);  
 System.out.println(ack);  
 System.out.println(cause);  
 }  
}

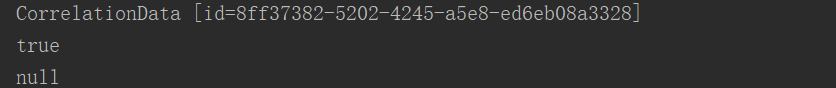
在RabbitmqTemplate设置一下

template.setConfirmCallback(new MyConfirmCallback());

而且我们可以在发送消息的时候附带一个CorrelationData参数，这个对象可以设置一个id，可以是你的业务id 方便进行对应的操作

CorrelationData correlationData = new CorrelationData(UUID.randomUUID().toString());  
rabbitTemplate.convertAndSend("directExchange", "direct.key123123", "hello",correlationData);

效果：



这里会把我们传入的那个业务id 以及ack（是否发送成功） 以及原因返回回来

但是要注意的是confirm模式的发送成功的意思是发送到RabbitMq（Broker）成功而不是发送到队列成功

所以才有了上面所说的那句要和失败回调结合使用，这样才能确认消息投递成功了

简单的总结一下就是confirm机制是确认我们的消息是否投递到了RabbitMq（Broker）上面，而mandatory是在我们的消息进入队列失败时候不会被遗弃（让我们自己进行处理）

那么上面就是rabbitmq在发送消息时我们可以做的一些处理， 接下来我们会讲到rabbitmq在接收（消费）消息时的一些特性