消费端的确认数据方式 ack

1.2 Consumer Ack

ack指Acknowledge，确认。 表示消费端收到消息后的确认方式。

有三种确认方式：

- 自动确认：acknowledge="none"
- 手动确认：acknowledge="manual"
- 根据异常情况确认：acknowledge="auto"，（这种方式使用麻烦，不作讲解）

其中自动确认是指，当消息一旦被Consumer接收到，则自动确认收到，并将相应 message 从 RabbitMQ 的消息缓存中移除。但是在实际业务处理中，很可能消息接收到，业务处理出现异常，那么该消息就会丢失。如果设置了手动确认方式，则需要在业务处理成功后，调用channel.basicAck()，手动签收，如果出现异常，则调用channel.basicNack()方法，让其自动重新发送消息。

1.2 Consumer Ack 小结

- 在rabbit:listener-container标签中设置acknowledge属性，设置ack方式 none：自动确认，manual：手动确认
- 如果在消费端没有出现异常，则调用channel.basicAck(deliveryTag,false);方法确认签收消息
- 如果出现异常，则在catch中调用 basicNack或 basicReject，拒绝消息，让MQ重新发送消息。

代码

rabbitmq.properties

```
rabbitmq.host=172.16.98.133
rabbitmq.port=5672
rabbitmq.username=guest
rabbitmq.password=guest
rabbitmq.virtual-host=/
```

spring-rabbitmq-consumer.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:rabbit="http://www.springframework.org/schema/rabbit"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/context
       https://www.springframework.org/schema/context/spring-context.xsd
       http://www.springframework.org/schema/rabbit
       http://www.springframework.org/schema/rabbit/spring-rabbit.xsd">
    <!--加载配置文件-->
    <context:property-placeholder location="classpath:rabbitmq.properties"/>

    <!-- 定义rabbitmq connectionFactory -->
    <rabbit:connection-factory id="connectionFactory" host="${rabbitmq.host}"
                               port="${rabbitmq.port}"
                               username="${rabbitmq.username}"
                               password="${rabbitmq.password}"
                               virtual-host="${rabbitmq.virtual-host}"/>


    <context:component-scan base-package="com.itheima.listener" />

    <!--定义监听器容器-->
<!--    <rabbit:listener-container connection-factory="connectionFactory" acknowledge="manual"
prefetch="1" >-->
    <rabbit:listener-container connection-factory="connectionFactory" acknowledge="manual" >
        <rabbit:listener ref="ackListener" queue-names="test_queue_confirm"></rabbit:listener>
       <!-- <rabbit:listener ref="qosListener" queue-names="test_queue_confirm">
</rabbit:listener>-->
        <!--定义监听器，监听正常队列-->
        <!--<rabbit:listener ref="dlxListener" queue-names="test_queue_dlx"></rabbit:listener>--
>

        <!--延迟队列效果实现：  一定要监听的是 死信队列！！！-->
<!--        <rabbit:listener ref="orderListener" queue-names="order_queue_dlx">
</rabbit:listener>-->
    </rabbit:listener-container>

</beans>
```

AckListener

```
package com.itheima.listener;

import com.rabbitmq.client.Channel;
import org.springframework.amqp.core.Message;
import org.springframework.amqp.core.MessageListener;
import org.springframework.amqp.rabbit.listener.api.ChannelAwareMessageListener;
import org.springframework.stereotype.Component;
```

```java
import java.io.IOException;

/**
 * Consumer ACK机制：
 *  1. 设置手动签收。acknowledge="manual"
 *  2. 让监听器类实现ChannelAwareMessageListener接口
 *  3. 如果消息成功处理，则调用channel的 basicAck()签收
 *  4. 如果消息处理失败，则调用channel的basicNack()拒绝签收，broker重新发送给consumer
 *
 *
 */
```