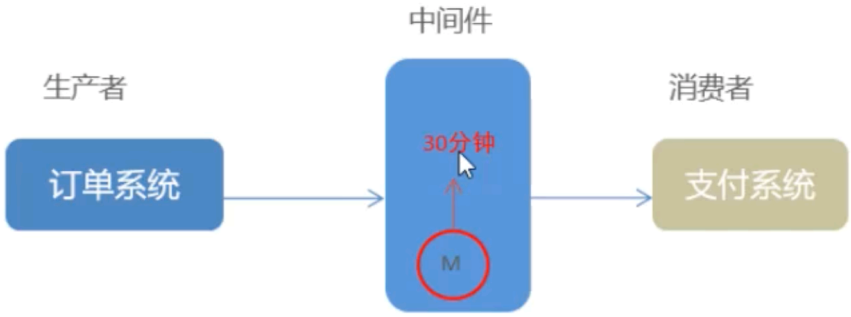
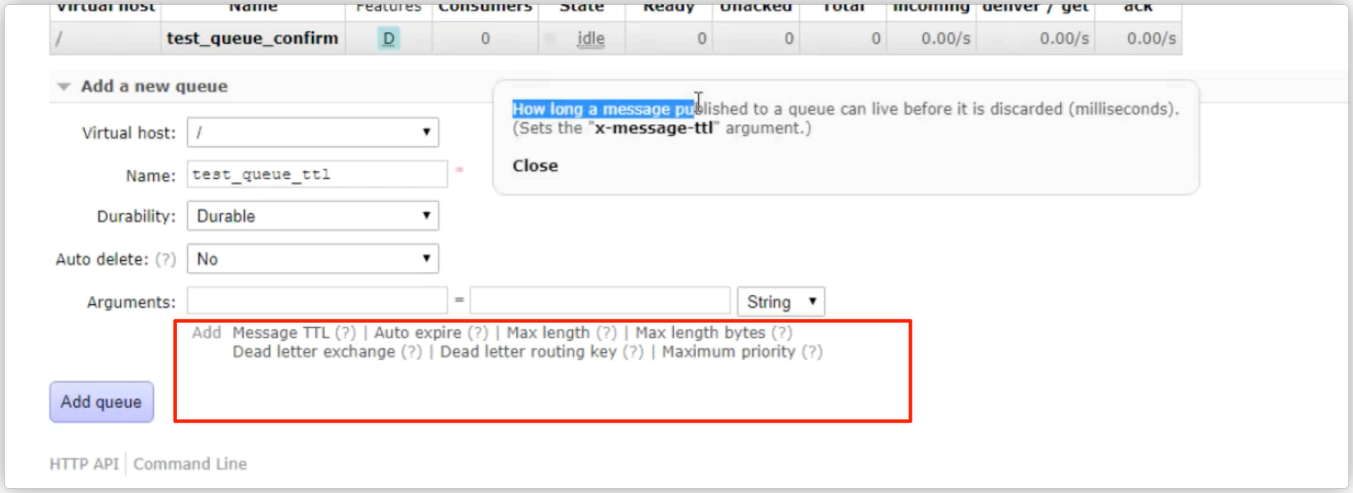
1.4 TTL

- TTL 全称 Time To Live（存活时间/过期时间）。
- 当消息到达存活时间后，还没有被消费，会被自动清除。
- RabbitMQ可以对消息设置过期时间，也可以对整个队列（Queue）设置过期时间。





- 设置队列过期时间使用参数：x-message-ttl，单位：ms(毫秒)，会对整个队列消息统一过期。
- 设置消息过期时间使用参数：expiration。单位：ms(毫秒)，当该消息在队列头部时（消费时），会单独判断这一消息是否过期。
- 如果两者都进行了设置，以时间短的为准。

```java
/**
 * TTL:过期时间
 *  1. 队列统一过期
 *
 *  2. 消息单独过期
 *
 *
 * 如果设置了消息的过期时间，也设置了队列的过期时间，它以时间短的为准。
 * 队列过期后，会将队列所有消息全部移除
 * 消息过期后，只有消息在队列顶端，才会判断其是否过期(移除掉)
 *
 */
@Test
public void testTtl() {
```

消息成为死信的三种情况：

1. 队列消息长度到达限制；
2. 消费者拒接消费消息，basicNack/basicReject,并且不把消息重新放入原目标队列,requeue=false；
3. 原队列存在消息过期设置，消息到达超时时间未被消费；

代码

spring版

rabbitmq.properties

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

spring-rabbitmq-producer.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}"
                           publisher-confirms="true"
                           publisher-returns="true"
/>
<!--定义管理交换机、队列-->
<rabbit:admin connection-factory="connectionFactory"/>

<!--定义rabbitTemplate对象操作可以在代码中方便发送消息-->
<rabbit:template id="rabbitTemplate" connection-factory="connectionFactory"/>



<!--ttl-->
<rabbit:queue name="test_queue_ttl" id="test_queue_ttl">
    <!--设置queue的参数-->
    <rabbit:queue-arguments>
        <!--x-message-ttl指队列的过期时间-->
        <entry key="x-message-ttl" value="100000" value-type="java.lang.Integer"></entry>
    </rabbit:queue-arguments>

</rabbit:queue>

<rabbit:topic-exchange name="test_exchange_ttl" >
    <rabbit:bindings>
        <rabbit:binding pattern="ttl.#" queue="test_queue_ttl"></rabbit:binding>
    </rabbit:bindings>
</rabbit:topic-exchange>

<!--
    死信队列:
        1. 声明正常的队列(test_queue_dlx)和交换机(test_exchange_dlx)
        2. 声明死信队列(queue_dlx)和死信交换机(exchange_dlx)
        3. 正常队列绑定死信交换机
            设置两个参数:
                * x-dead-letter-exchange: 死信交换机名称
```

```
                    * x-dead-letter-routing-key：发送给死信交换机的routingkey
    -->

    <!--
        1. 声明正常的队列(test_queue_dlx)和交换机(test_exchange_dlx)
    -->

    <rabbit:queue name="test_queue_dlx" id="test_queue_dlx">
        <!--3. 正常队列绑定死信交换机-->
        <rabbit:queue-arguments>
            <!--3.1 x-dead-letter-exchange：死信交换机名称-->
            <entry key="x-dead-letter-exchange" value="exchange_dlx" />

            <!--3.2 x-dead-letter-routing-key：发送给死信交换机的routingkey-->
            <entry key="x-dead-letter-routing-key" value="dlx.hehe" />

            <!--4.1 设置队列的过期时间 ttl-->
            <entry key="x-message-ttl" value="10000" value-type="java.lang.Integer" />
            <!--4.2 设置队列的长度限制 max-length -->
            <entry key="x-max-length" value="10" value-type="java.lang.Integer" />
        </rabbit:queue-arguments>
    </rabbit:queue>
    <rabbit:topic-exchange name="test_exchange_dlx">
        <rabbit:bindings>
            <rabbit:binding pattern="test.dlx.#" queue="test_queue_dlx"></rabbit:binding>
        </rabbit:bindings>
    </rabbit:topic-exchange>


    <!--
       2. 声明死信队列(queue_dlx)和死信交换机(exchange_dlx)
   -->

    <rabbit:queue name="queue_dlx" id="queue_dlx"></rabbit:queue>
    <rabbit:topic-exchange name="exchange_dlx">
        <rabbit:bindings>
            <rabbit:binding pattern="dlx.#" queue="queue_dlx"></rabbit:binding>
        </rabbit:bindings>
    </rabbit:topic-exchange>

</beans>
```

ProducerTest

```
/**
     * TTL:过期时间
     *  1. 队列统一过期
     *
     *  2. 消息单独过期
     *
     *
```

```
     * 如果设置了消息的过期时间，也设置了队列的过期时间，它以时间短的为准。
     * 队列过期后，会将队列所有消息全部移除。
     * 消息过期后，只有消息在队列顶端，才会判断其是否过期(移除掉)
     *
     */
    @Test
    public void testTtl() {


      /*  for (int i = 0; i < 10; i++) {
            // 发送消息
            rabbitTemplate.convertAndSend("test_exchange_ttl", "ttl.hehe", "message ttl....");
        }*/

      // 消息后处理对象，设置一些消息的参数信息
        MessagePostProcessor messagePostProcessor = new MessagePostProcessor() {

            @Override
            public Message postProcessMessage(Message message) throws AmqpException {
                //1.设置message的信息
                message.getMessageProperties().setExpiration("5000");//消息的过期时间
                //2.返回该消息
                return message;
            }
        };


        //消息单独过期
        //rabbitTemplate.convertAndSend("test_exchange_ttl", "ttl.hehe", "message
ttl....",messagePostProcessor);


        for (int i = 0; i < 10; i++) {
            if(i == 5){
                //消息单独过期
                rabbitTemplate.convertAndSend("test_exchange_ttl", "ttl.hehe", "message
ttl....",messagePostProcessor);
            }else{
                //不过期的消息
                rabbitTemplate.convertAndSend("test_exchange_ttl", "ttl.hehe", "message
ttl....");

            }

        }


    }


    /**
     * 发送测试死信消息:
```

```
     *  1. 过期时间
     *  2. 长度限制
     *  3. 消息拒收
     */
    @Test
    public void testDlx(){
        //1. 测试过期时间，死信消息
        //rabbitTemplate.convertAndSend("test_exchange_dlx","test.dlx.haha","我是一条消息，我会死
吗？");

        //2. 测试长度限制后，消息死信
       /* for (int i = 0; i < 20; i++) {
            rabbitTemplate.convertAndSend("test_exchange_dlx","test.dlx.haha","我是一条消息，我会死
吗？");
        }*/

        //3. 测试消息拒收
        rabbitTemplate.convertAndSend("test_exchange_dlx","test.dlx.haha","我是一条消息，我会死
吗？");

    }
```

customer

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>
```

DlxListener

```
package com.itheima.listener;

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



@Component
public class DlxListener implements ChannelAwareMessageListener {

    @Override
    public void onMessage(Message message, Channel channel) throws Exception {
        long deliveryTag = message.getMessageProperties().getDeliveryTag();

        try {
            //1.接收转换消息
            System.out.println(new String(message.getBody()));

            //2. 处理业务逻辑
```

```
            System.out.println("处理业务逻辑...");
            int i = 3/0;//出现错误
            //3. 手动签收
            channel.basicAck(deliveryTag,true);
        } catch (Exception e) {
            //e.printStackTrace();
            System.out.println("出现异常，拒绝接受");
            //4.拒绝签收，不重回队列 requeue=false
            channel.basicNack(deliveryTag,true,false);
        }
    }
}
```