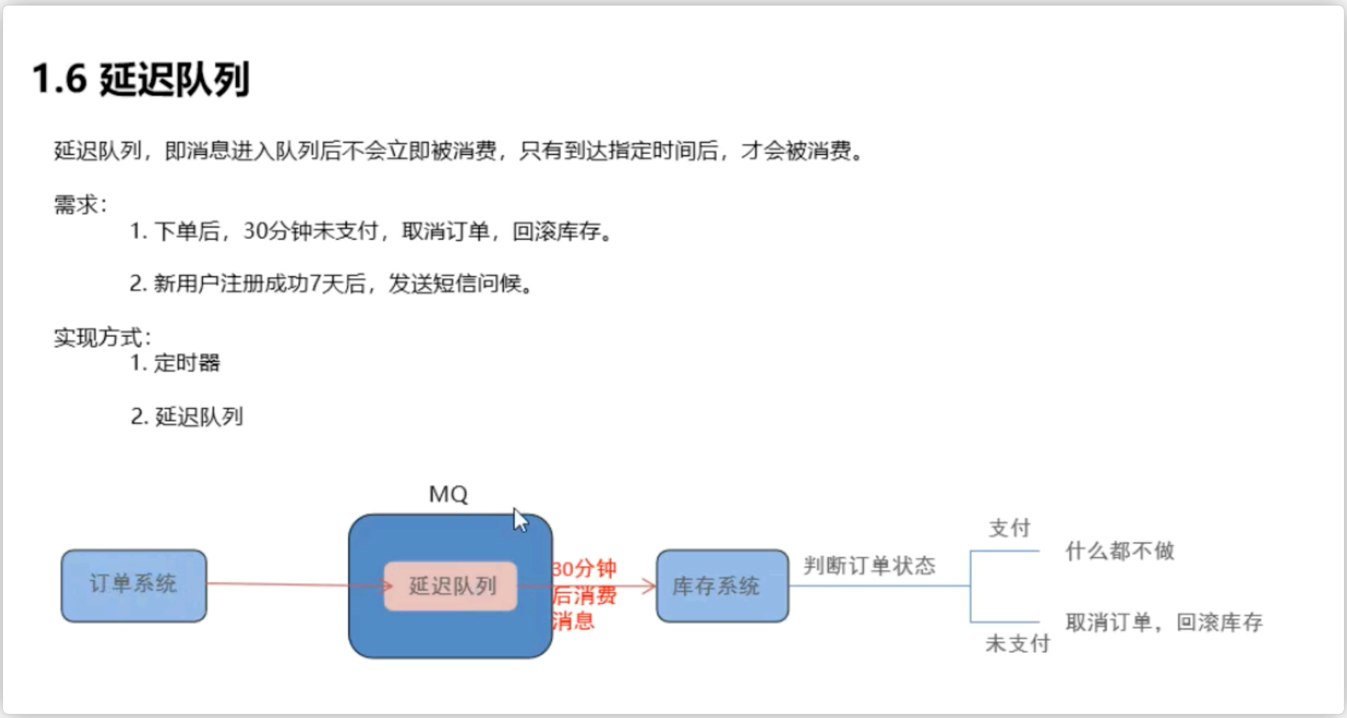
延迟队列



1.6 延迟队列

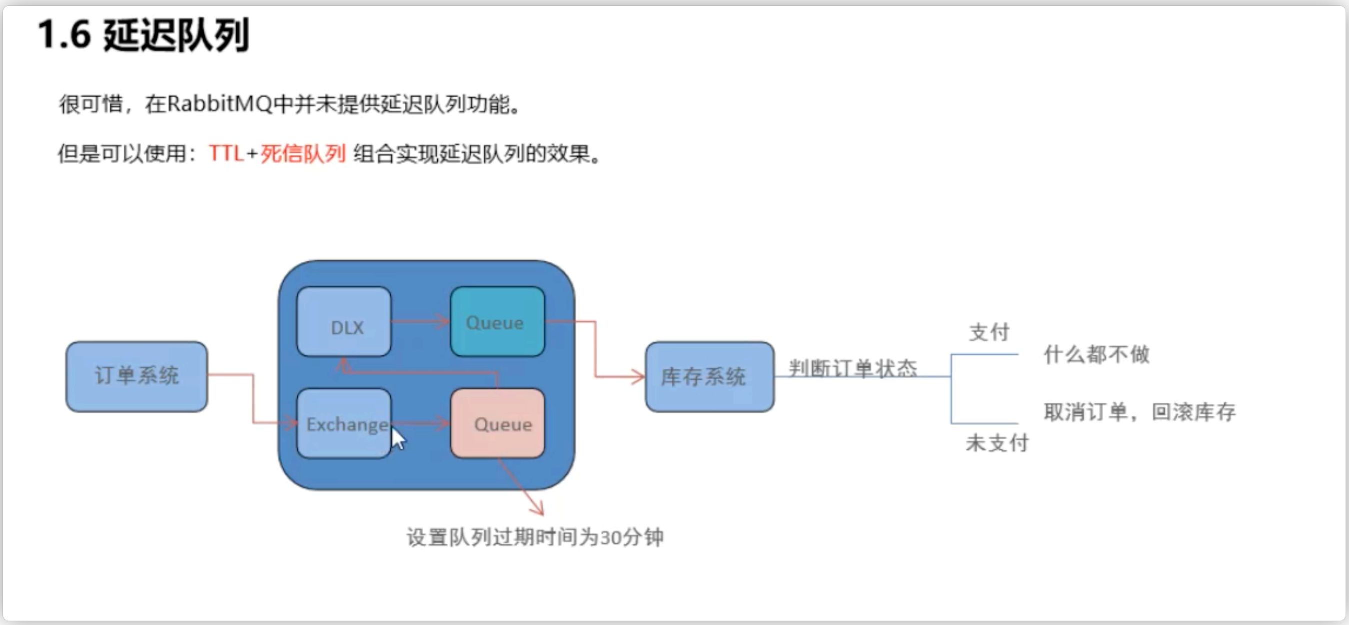
延迟队列，即消息进入队列后不会立即被消费，只有到达指定时间后，才会被消费。

需求：

1. 下单后，30分钟未支付，取消订单，回滚库存。
2. 新用户注册成功7天后，发送短信问候。

实现方式：

1. 定时器
2. 延迟队列



1.6 延迟队列

很可惜，在RabbitMQ中并未提供延迟队列功能。

但是可以使用：TTL+死信队列 组合实现延迟队列的效果。

springboot版

参考： https://www.jianshu.com/p/451958b1adca
https://www.jianshu.com/p/f6313814bf77

spring版

product

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"/>



    <!--
```

```
        延迟队列:
            1. 定义正常交换机 (order_exchange) 和队列(order_queue)
            2. 定义死信交换机 (order_exchange_dlx) 和队列(order_queue_dlx)
            3. 绑定,设置正常队列过期时间为30分钟
    -->
    <!-- 1. 定义正常交换机 (order_exchange) 和队列(order_queue)-->
    <rabbit:queue id="order_queue" name="order_queue">
        <!-- 3. 绑定,设置正常队列过期时间为30分钟-->
        <rabbit:queue-arguments>
            <entry key="x-dead-letter-exchange" value="order_exchange_dlx" />
            <entry key="x-dead-letter-routing-key" value="dlx.order.cancel" />
            <entry key="x-message-ttl" value="10000" value-type="java.lang.Integer" />

        </rabbit:queue-arguments>

    </rabbit:queue>
    <rabbit:topic-exchange name="order_exchange">
        <rabbit:bindings>
            <rabbit:binding pattern="order.#" queue="order_queue"></rabbit:binding>
        </rabbit:bindings>
    </rabbit:topic-exchange>

    <!--  2. 定义死信交换机 (order_exchange_dlx) 和队列(order_queue_dlx)-->
    <rabbit:queue id="order_queue_dlx" name="order_queue_dlx"></rabbit:queue>
    <rabbit:topic-exchange name="order_exchange_dlx">
        <rabbit:bindings>
            <rabbit:binding pattern="dlx.order.#" queue="order_queue_dlx"></rabbit:binding>
        </rabbit:bindings>
    </rabbit:topic-exchange>

</beans>
```

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"
```

```xml
         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>
```

OrderListener

```java
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 OrderListener 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...");
            System.out.println("根据订单id查询其状态...");
            System.out.println("判断状态是否为支付成功");
            System.out.println("取消订单，回滚库存....");
            //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);
        }
    }
}
```