rabbitMq高级（ttl过期时间,死信队列,延时队列）一文透彻springboot篇

一、过期时间TTL

1、设置队列TTL

过期时间TTL表示可以对消息设置预期的时间，在这个时间内都可以被消费者接收获取；过了之后消息将自动被删除。RabbitMQ可以对**消息和队列**设置TTL。目前有两种方法可以设置。

- 第一种方法是通过队列属性设置，队列中所有消息都有相同的过期时间。
- 第二种方法是对消息进行单独设置，每条消息TTL可以不同。

如果上述两种方法同时使用，则消息的过期时间以两者之间TTL较小的那个数值为准。消息在队列的生存时间一旦超过设置的TTL值，就称为dead message被投递到死信队列， 消费者将无法再收到该消息。

复制代码

```
@Component
public class MyRabbitmqConfig {
//创建队列（my_ttl_queue），投递到该队列的消息如果没有消费都将在6秒之后被删除
    @Bean
    public Queue my_ttl_queue(){
        return QueueBuilder.durable("my_ttl_queue")
                .ttl(5000)
                .build();
    }

}
```

复制代码

```
@RunWith(SpringRunner.class)
@SpringBootTest
public class RabbitmqStudy2ApplicationTests {

    @Autowired
    RabbitTemplate rabbitTemplate;
    /**
     * 过期队列消息
     * 投递到该队列的消息如果没有消费都将在6秒之后被删除
     */
    @Test
```

```
    public void ttlQueueTest(){
        //路由键与队列同名
        rabbitTemplate.convertAndSend("my_ttl_queue", "发送到过期队列my_ttl_queue，5秒内不消费则不能再被消费
    }

}
```

效果如下（红线）：

Overview					Messages			Message rates		
Virtual host	**Name**	**Type**	Features	**State**	**Ready**	**Unacked**	**Total**	**incoming**	**deliver / get**	**ack**
/courses_selecting	**CourseUserConsumer**	classic	D AD	idle	0	0	0			
/courses_selecting	**delay.queue**	classic	D DLX DLK	idle	0	0	0	0.00/s	0.00/s	0.00/s
/courses_selecting	**my_ttl_queue**	classic	D TTL	idle	1	0	1	0.00/s	0.00/s	0.00/s
/courses_selecting	**process.queue**	classic	D	idle	0	0	0	0.00/s	0.00/s	0.00/s

Add a new queue

等待5秒过后，消息过期：

Overview					Messages			Message rates		
Virtual host	**Name**	**Type**	Features	**State**	**Ready**	**Unacked**	**Total**	**incoming**	**deliver / get**	**ack**
/courses_selecting	**CourseUserConsumer**	classic	D AD	idle	0	0	0			
/courses_selecting	**delay.queue**	classic	D DLX DLK	idle	0	0	0	0.00/s	0.00/s	0.00/s
/courses_selecting	**my_ttl_queue**	classic	D TTL	idle	0	0	0	0.00/s	0.00/s	0.00/s
/courses_selecting	**process.queue**	classic	D	idle	0	0	0	0.00/s	0.00/s	0.00/s

2、设置消息TTL

为了让效果更加直观，并证明之前说的消息的过期时间以队列ttl和消息ttl两者之间TTL较小的那个数值为准，我们登录客户端删除之前的队列，然后重新配置：

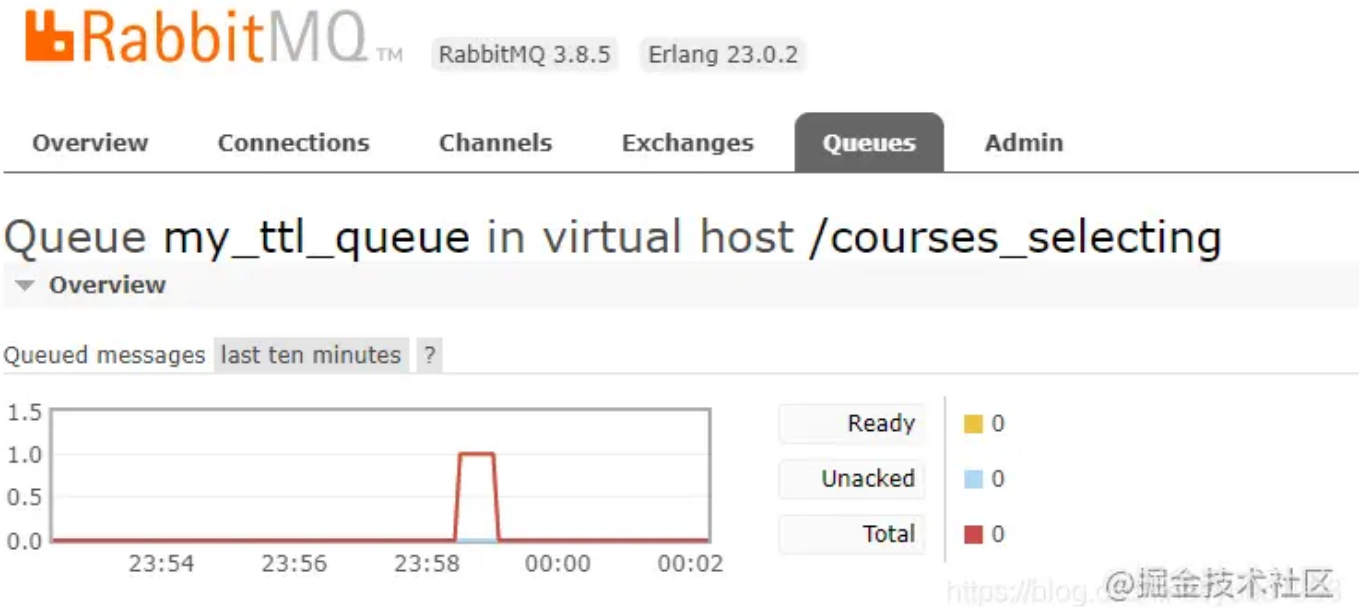
复制代码

```
@Component
public class MyRabbitmqConfig {

    @Bean
    public Queue my_ttl_queue(){
        return QueueBuilder.durable("my_ttl_queue")
                .ttl(500000)
                .build();
    }

}
```

复制代码

```
/**
 * 过期消息
 * 该消息投递任何交换机或队列中的时候；如果到了过期时间则将从该队列中删除
 */
@Test
public void ttlMessageTest(){
    MessageProperties messageProperties = new MessageProperties();
    //设置消息的过期时间，30秒
    messageProperties.setExpiration("30000");//字符串非负数字
    Message message = new Message("测试过期消息，30秒钟过期".getBytes(), messageProperties);
    //路由键与队列同名
    rabbitTemplate.convertAndSend("my_ttl_queue", message);
}
```

可以清楚看到，我们预设的在队列ttl为500秒，消息为30秒，由客户端可以清楚看到，消息在队列中躺了30秒，故可以证明前面的两者之间取TTL较小。



> 这里还有个思考：
>
> 我们也可以在配置中不设置队列的ttl，只设置消息的ttl，这样就能达到跟定时器一样的效果，确实可以这样，因为队列不设置ttl的话，表示在队列中永远存在，通过设置不同消息ttl达到定时消费，但是也可能存在一些情况导致消息不断堆积在队列中消费不了，所以我们是不是也可以设置一个保证业务不受影响的尽可能大的队列ttl?

二、死信队列

DLX，全称为Dead-Letter-Exchange，可以称之为死信交换机。当消息在一个队列中变成死信(dead message)之后，它能被重新发送到另一个交换机中，这个交换机就是DLX，绑定DLX的队列就称之为死信

队列。

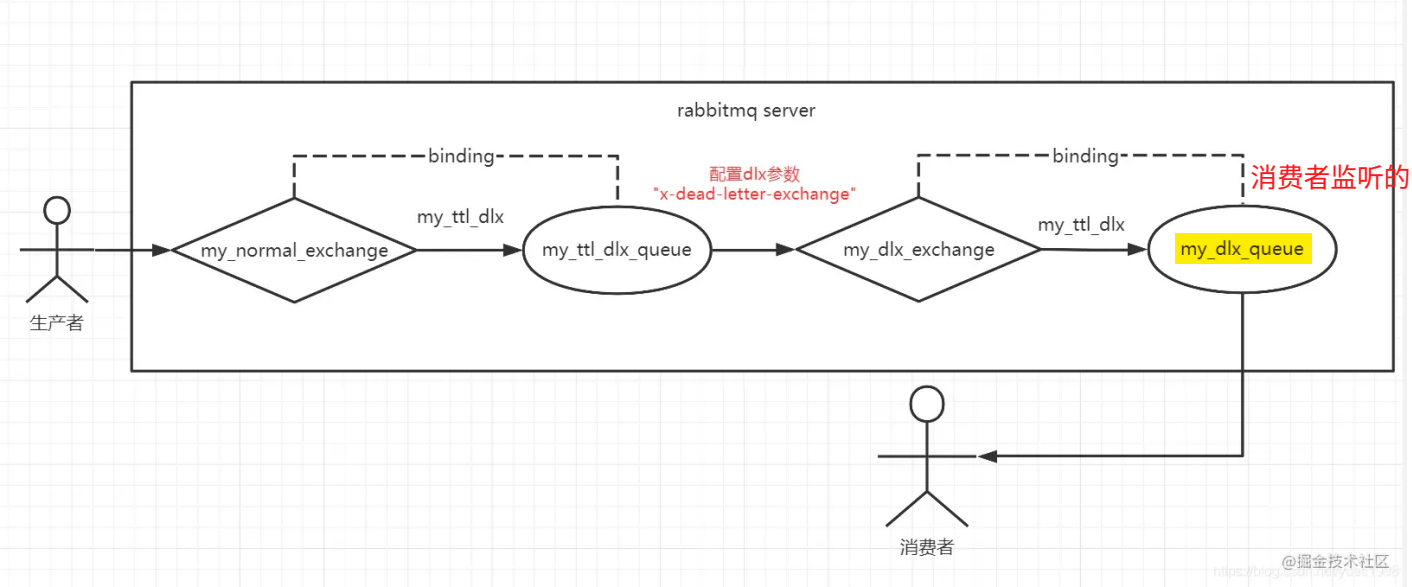
消息变成死信，可能是由于以下的原因：

- 消息被拒绝
- 消息过期
- 队列达到最大长度

DLX也是一个正常的交换机，和一般的交换机没有区别，它能在任何的队列上被指定，实际上就是设置某一个队列的属性。当这个队列中存在死信时，Rabbitmq就会自动地将这个消息重新发布到设置的DLX上去，进而被路由到另一个队列，即死信队列。

要想使用死信队列，只需要在定义队列的时候设置队列参数 `x-dead-letter-exchange` 指定交换机即可。

配置如下（专门画了张图，方便理解）



复制代码

```
//定义定向交换机中的持久化死信队列
@Bean
public Queue my_dlx_queue(){
    return QueueBuilder.durable("my_dlx_queue")
            .build();
}

//定义广播类型交换机
@Bean
public DirectExchange my_dlx_exchange(){
    return ExchangeBuilder.directExchange("my_dlx_exchange").build();
}

@Bean
public Binding dlx_exchange_to_queue(@Qualifier("my_dlx_queue")Queue my_dlx_queue,
                                     @Qualifier("my_dlx_exchange")DirectExchange my_dlx_exchange){
```

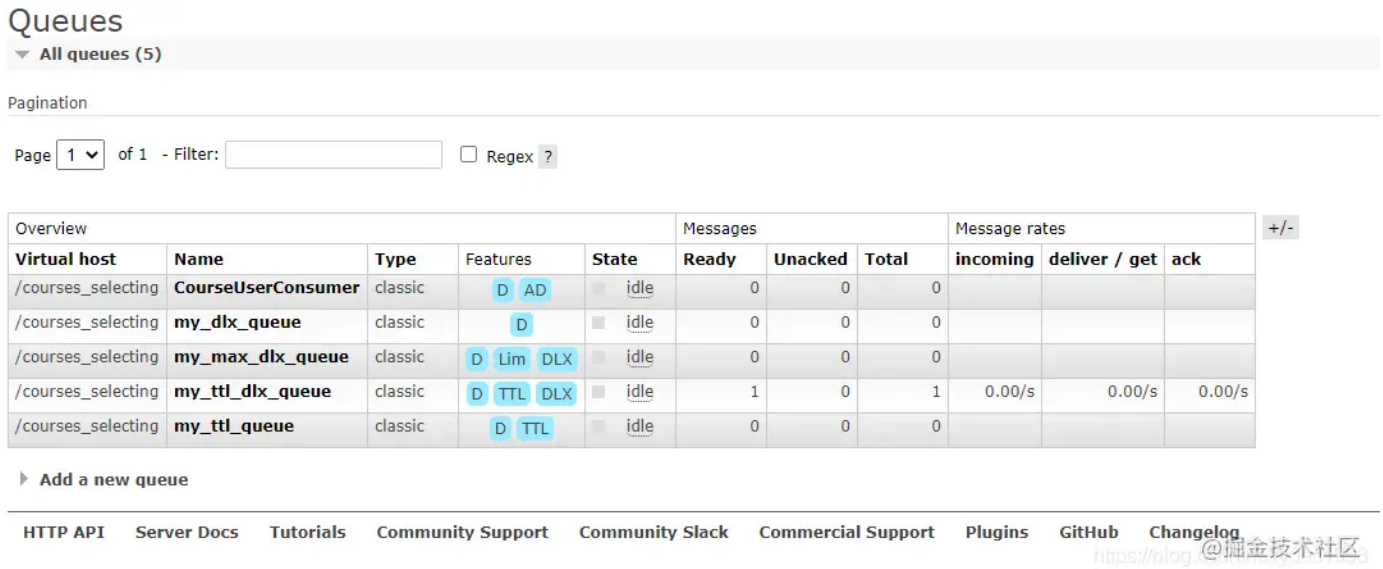
```
    //绑定路由键my_ttl_dlx，可以将过期的消息转移到my_dlx_queue队列
    return BindingBuilder.bind(my_dlx_queue).to(my_dlx_exchange).with("my_ttl_dlx");
}

//定义过期队列及其属性
@Bean
public Queue my_ttl_dlx_queue(){
    return QueueBuilder.durable("my_ttl_dlx_queue")
            .ttl(6000)//投递到该队列的消息如果没有消费都将在6秒之后被投递到死信交换机
            .deadLetterExchange("my_dlx_exchange")//设置当消息过期后投递到对应的死信交换机
            .build();
}

//定义定向交换机 根据不同的路由key投递消息
@Bean
public DirectExchange my_normal_exchange(){
    return ExchangeBuilder.directExchange("my_normal_exchange").build();
}

@Bean
public Binding normal_exchange_to_queue(@Qualifier("my_ttl_dlx_queue")Queue my_ttl_dlx_queue,
                                        @Qualifier("my_normal_exchange")DirectExchange my_normal_exchang
    //绑定路由键my_ttl_dlx，可以将过期的消息转移到my_dlx_queue队列
    return BindingBuilder.bind(my_ttl_dlx_queue).to(my_normal_exchange).with("my_ttl_dlx");
}
```

复制代码

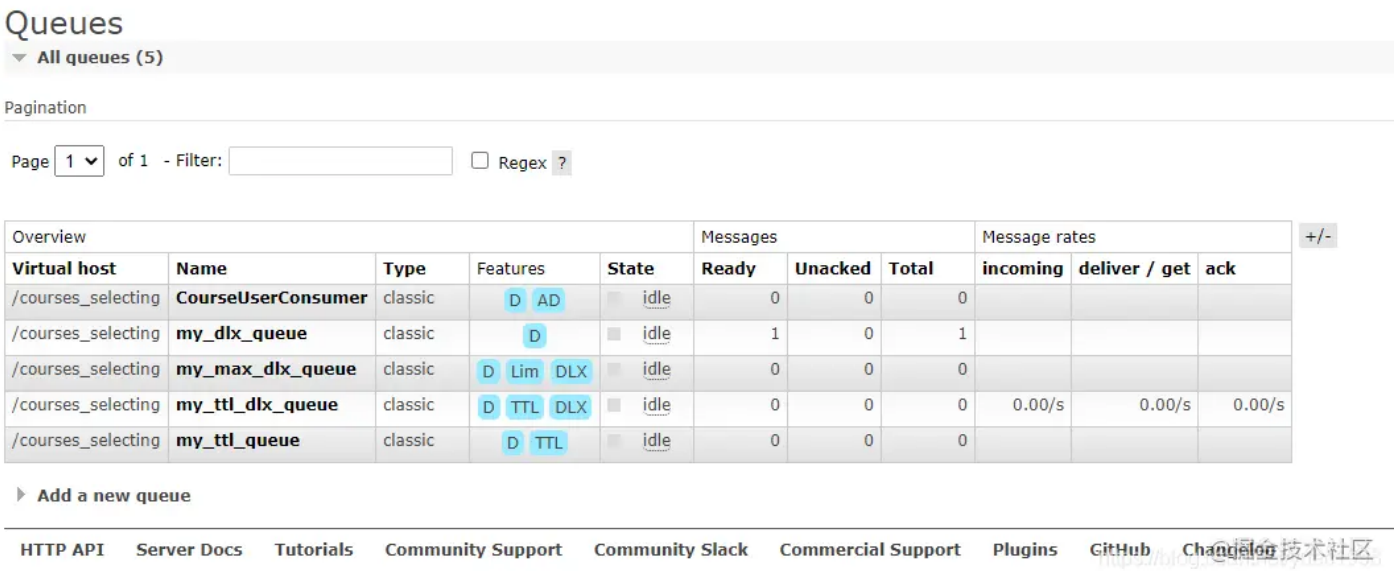
```
/**
 * 过期消息投递到死信队列
 * 投递到一个正常的队列，但是该队列有设置过期时间，到过期时间之后消息会被投递到死信交换机（队列）
 */
@Test
public void dlxTTLMessageTest(){
    rabbitTemplate.convertAndSend("my_normal_exchange", "my_ttl_dlx", "测试过期消息：6秒过期后会被投递到
}
```

实例结果也很明显，

1、现在my_ttl_dlx_queue停留了6秒



2、成为死信，路由到my_dlx_queue



下一篇，将介绍一下自动确认机制，以及mq在生产应用中会遇到的问题及解决方案