

[1、概述](#)

[1-1、设置队列TTL](#) ▼

[代码设置](#)

[测试类](#)

[1-2、设置消息TTL](#)

RabbitMQ高级-过期时间TTL

 飞哥 VIP

分类: [学习笔记](#) 创建时间: 2021/03/05 23:59 ☒ 字体 ☐ 皮肤

1、概述

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。TTL可以针对消息设置，也可以针对队列设置。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TTL，该消息将被标记为dead message被投递到死信队列，消费者将无法再收到该消息。

1-1、 设置队列TTL

代码设置





```

1. package com.xuexiangban.rabbitmq.ttl;
2.
3. import com.rabbitmq.client.AMQP;
4. import com.rabbitmq.client.Channel;
5. import com.rabbitmq.client.Connection;
6. import com.rabbitmq.client.ConnectionFactory;
7. import org.springframework.amqp.core.Message;
8. import org.springframework.amqp.core.MessageProperties;
9.
10. import java.util.HashMap;
11. import java.util.Map;
12.
13. /**
14.  * @author: 学相伴-飞哥
15.  * @description: Producer 简单队列生产者
16.  * @Date : 2021/3/2
17.  */
18. public class Producer {
19.
20.     public static void main(String[] args) {
21.
22.         // 1: 创建连接工厂
23.         ConnectionFactory connectionFactory = new ConnectionFactory();
24.         // 2: 设置连接属性
25.         connectionFactory.setHost("47.104.141.27");
26.         connectionFactory.setPort(5672);
27.         connectionFactory.setVirtualHost("/");
28.         connectionFactory.setUsername("admin");
29.         connectionFactory.setPassword("admin");
30.
31.         Connection connection = ,
32.         Channel channel = null;
33.         try {
34.             // 3: 从连接工厂中获取连接
35.             connection = connectionFactory.newConnection("生产者");
36.             // 4: 从连接中获取通道channel
37.             channel = connection.createChannel();
38.             // 5: 申明队列queue存储消息
39.             /*
40.              * 如果队列不存在，则会创建
41.              * Rabbitmq不允许创建两个相同的队列名称，否则会报错。
42.              *
43.              * @params1: queue 队列的名称
44.              * @params2: durable 队列是否持久化
45.              * @params3: exclusive 是否排他，即是否私有的，如果为true,会对当前队列加
46.              * 锁，并且连接自动关闭
47.              * @params4: autoDelete 是否自动删除，当最后一个消费者断开连接之后是否自动删除
48.              * @params5: arguments 可以设置队列附加参数，设置队列的有效期，消息的最大有效期等等。
49.              */
50.             Map<String, Object> args2 = new HashMap<>();
51.             args2.put("x-message-ttl", 5000);
52.             channel.queueDeclare("ttl.queue", true, false, false, args2);
53.
54.             // 6: 准备发送消息的内容
55.             String message = "你好，学相伴！！！";
56.
57.             Map<String, Object> headers = new HashMap<String, Object>();
58.             headers.put("x", "1");
59.             headers.put("y", "1");
60.             AMQP.BasicProperties basicProperties = new AMQP.BasicProperties().builder()
61.                 .deliveryMode(2) // 传送方式
62.                 .priority(1)
63.                 .contentEncoding("UTF-8") // 编码方式
64.                 .expiration("3000") // 过期时间
65.                 .headers(headers).build(); //自定义属性
66.
67.             // 7: 发送消息给中间件rabbitmq-server

```

[1、概述](#)[1-1、设置队列TTL](#) [代码设置](#)[测试类](#)[1-2、设置消息TTL](#)

```
67.      // @params1: 交换机exchange
68.      // @params2: 队列名称/routing
69.      // @params3: 属性配置
70.      // @params4: 发送消息的内容
71.      for (int i = 0; i <100 ; i++) {
72.          channel.basicPublish("", "ttl.queue", basicProperties, message.g
73.              System.out.println("消息发送成功!");
74.              Thread.sleep(1000);
75.          }
76.      } catch (Exception ex) {
77.          ex.printStackTrace();
78.          System.out.println("发送消息出现异常...");
79.      } finally {
80.          // 7: 释放连接关闭通道
81.          if (channel != null && channel.isOpen()) {
82.              try {
83.                  channel.close();
84.              } catch (Exception ex) {
85.                  ex.printStackTrace();
86.              }
87.          }
88.
89.          if (connection != null && connection.isOpen()) {
90.              try {
91.                  connection.close();
92.              } catch (Exception ex) {
93.                  ex.printStackTrace();
94.              }
95.          }
96.      }
97.  }
98. }
```

1、概述

1-1、设置队列TTL

代码设置

测试类

1-2、设置消息TTL

测试类



```
1. package com.xuexiangban.rabbitmq.ttl;
2.
3. import com.rabbitmq.client.*;
4.
5. import java.io.IOException;
6.
7. /**
8.  * @author: 学相伴-飞哥
9.  * @description: Consumer
10.  * @Date : 2021/3/2
11.  */
12. public class Consumer {
13.
14.     public static void main(String[] args) {
15.         // 1: 创建连接工厂
16.         ConnectionFactory connectionFactory = new ConnectionFactory();
17.         // 2: 设置连接属性
18.         connectionFactory.setHost("47.104.141.27");
19.         connectionFactory.setPort(5672);
20.         connectionFactory.setVirtualHost("/");
21.         connectionFactory.setUsername("admin");
22.         connectionFactory.setPassword("admin");
23.
24.         Connection connection = null;
25.         Channel channel = null;
26.         try {
27.             // 3: 从连接工厂中获取连接
28.             connection = connectionFactory.newConnection("消费者");
29.             // 4: 从连接中获取通道
30.             channel = connection.createChannel();
31.             // 5: 申明队列queue
32.             /*
33.              * 如果队列不存在，则会创建
34.              * Rabbitmq不允许创建两个相同的队列名称，否则会报错。
35.              *
36.              * @params1: queue 队列的名称
37.              * @params2: durable 队列是否持久化
38.              * @params3: exclusive 是否排他，即是否私有的，如果为true,会对当前队列加
39.              * 锁，并且连接自动关闭
40.              * @params4: autoDelete 是否自动删除，当最后一个消费者断开连接之后是否自动删除
41.              * @params5: arguments 可以设置队列附加参数，设置队列的有效期，消息的最大有效期等等。
42.              */
43.             // 这里如果queue已经被创建过一次了，可以不需要定义
44.             //channel.queueDeclare("queue1", false, false, false, null);
45.
46.             // 6: 定义接受消息的回调
47.             Channel finalChannel = channel;
48.             finalChannel.basicConsume("ttl.queue", true, new DefaultConsumer(channel) {
49.                 @Override
50.                 public void handleDelivery(String consumerTag, Envelope envelope, AMQP.BasicProperties properties, byte[] body) throws IOException {
51.                     System.out.println(properties);
52.                     System.out.println("获取的消息是：" + new String(body, "UTF-8"));
53.                 }
54.             });
55.
56.             System.out.println("开始接受消息");
57.             System.in.read();
58.         } catch (Exception ex) {
59.             ex.printStackTrace();
60.             System.out.println("发送消息出现异常...");
61.         } finally {
62.             // 7: 释放连接关闭通道
63.             if (channel != null && channel.isOpen()) {
64.                 try {
65.                     channel.close();
66.                 } catch (Exception ex) {
```

[1、概述](#)

[1-1、设置队列TTL](#)

[代码设置](#)

[测试类](#)

[1-2、设置消息TTL](#)



```
66.         ex.printStackTrace();
67.     }
68. }
69. if (connection != null && connection.isOpen()) {
70.     try {
71.         connection.close();
72.     } catch (Exception ex) {
73.         ex.printStackTrace();
74.     }
75. }
76. }
77. }
78. }
```

1、概述

1-1、设置队列TTL ▼

代码设置

测试类

1-2、设置消息TTL

参数 x-message-ttl 的值 必须是非负 32 位整数 (0 <= n <= 2^32-1)，以毫秒为单位表示 TTL 的值。这样，中的当前 消息 将最多只存活 6 秒钟。

OverviewConnectionsChannelsExchangesQueuesAdmin

Queues

▶ All queues (11)

Overview				Messages			Message rate
Virtual host	Name	Features	State	Ready	Unacked	Total	incoming d
/itcast	my_ttl_queue	D TTL	idle	0	0	0	0.00/s

如果队列设置了过期时间，那么会有标识

<>

1-2、设置消息TTL

消息的过期时间；只需要在发送消息（可以发送到任何队列，不管该队列是否属于某个交换机）的时候设置过期时间
如下方法发送消息并设置过期时间到队列：



```
1. package com.xuexiangban.rabbitmq.ttl;
2.
3. import com.rabbitmq.client.AMQP;
4. import com.rabbitmq.client.Channel;
5. import com.rabbitmq.client.Connection;
6. import com.rabbitmq.client.ConnectionFactory;
7.
8. import java.util.HashMap;
9. import java.util.Map;
10.
11. /**
12.  * @author: 学相伴-飞哥
13.  * @description: Producer 简单队列生产者
14.  * @Date : 2021/3/2
15.  */
16. public class MessageTTLProducer {
17.
18.     public static void main(String[] args) {
19.
20.         // 1: 创建连接工厂
21.         ConnectionFactory connectionFactory = new ConnectionFactory();
22.         // 2: 设置连接属性
23.         connectionFactory.setHost("47.104.141.27");
24.         connectionFactory.setPort(5672);
25.         connectionFactory.setVirtualHost("/");
26.         connectionFactory.setUsername("admin");
27.         connectionFactory.setPassword("admin");
28.
29.         Connection connection = null;
30.         Channel channel = null;
31.         try {
32.             // 3: 从连接工厂中获取连接
33.             connection = connectionFactory.newConnection("生产者");
34.             // 4: 从连接中获取通道channel
35.             channel = connection.createChannel();
36.             // 5: 申明队列queue存储消息
37.             /*
38.              * 如果队列不存在，则会创建
39.              * Rabbitmq不允许创建两个相同的队列名称，否则会报错。
40.              *
41.              * @params1: queue 队列的名称
42.              * @params2: durable 队列是否持久化
43.              * @params3: exclusive 是否排他，即是否私有的，如果为true,会对当前队列加
44.              锁，并且连接自动关闭
45.              * @params4: autoDelete 是否自动删除，当最后一个消费者断开连接之后是否自动删除
46.              * @params5: arguments 可以设置队列附加参数，设置队列的有效期，消息的最大
47.              过期等等。
48.              */
49.             channel.queueDeclare("ttl.queue2", true, false, false, null);
50.             // 6: 准备发送消息的内容
51.             String message = "你好，学相伴!!!";
52.
53.             Map<String, Object> headers = new HashMap<String, Object>();
54.             headers.put("x", "1");
55.             headers.put("y", "1");
56.             AMQP.BasicProperties basicProperties = new AMQP.BasicProperties().builder()
57.                 .deliveryMode(2) // 传送方式
58.                 .priority(1)
59.                 .contentEncoding("UTF-8") // 编码方式
60.                 .expiration("5000") // 过期时间
61.                 .headers(headers).build(); //自定义属性
62.
63.             // 7: 发送消息给中间件rabbitmq-server
64.             // @params1: 交换机exchange
65.             // @params2: 队列名称/routing
66.             // @params3: 属性配置
67.             // @params4: 发送消息的内容
68.             for (int i = 0; i < 10 ; i++) {
```

[1、概述](#)

[1-1、设置队列TTL](#)

[代码设置](#)

[测试类](#)

[1-2、设置消息TTL](#)



```
67.         channel.basicPublish("", "ttl.queue2", basicProperties, message.getBytes());
68.         System.out.println("消息发送成功!");
69.     }
70. } catch (Exception ex) {
71.     ex.printStackTrace();
72.     System.out.println("发送消息出现异常...");
73. } finally {
74.     // 7: 释放连接关闭通道
75.     if (channel != null && channel.isOpen()) {
76.         try {
77.             channel.close();
78.         } catch (Exception ex) {
79.             ex.printStackTrace();
80.         }
81.     }
82.
83.     if (connection != null && connection.isOpen()) {
84.         try {
85.             connection.close();
86.         } catch (Exception ex) {
87.             ex.printStackTrace();
88.         }
89.     }
90. }
91. }
92. }
```

1、概述

1-1、设置队列TTL

代码设置

测试类

1-2、设置消息TTL

/	sms.fanout.queue	classic	D	idle	10	0	10	
/	sms.topic.queueu	classic	D	idle	0	0	0	
/	ttl.queue	classic	D	TTL	0	0	0	
/	ttl.queue2	classic	D					00/s
/	weixin.direct.queue	classic	D	idle	0	0	0	
/	weixin.fanout.queue	classic	D	idle	10	0	10	
/	weixin.topic.queue	classic	D	idle	0	0	0	
/	weixin.topic.queueu	classic	D	idle	40	0	40	

expiration 字段以微秒为单位表示 TTL 值。且与 x-message-ttl 具有相同的约束条件。因为 expiration 字段: broker 将只会接受以字符串形式表达的数字。

当同时指定了 queue 和 message 的 TTL 值，则两者中较小的那个才会起作用。