**[RabbitMQ防止消息丢失](http://www.cnblogs.com/Leo_wl/p/6581989.html)**

# **转载请注明出处**

# **0.目录**

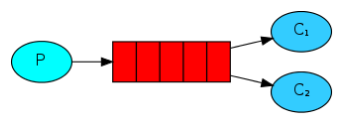
[RabbitMQ-从基础到实战（1）— Hello RabbitMQ](http://www.cnblogs.com/4----/p/6518801.html)

[RabbitMQ-从基础到实战（3）— 消息的交换](http://www.cnblogs.com/4----/p/6549865.html)

# **1.简介**

RabbitMQ中，消息丢失可以简单的分为两种：客户端丢失和服务端丢失。针对这两种消息丢失，RabbitMQ都给出了相应的解决方案。

# **2.防止客户端丢失消息**



如图，生产者P向队列中生产消息，C1和C2消费队列中的消息，默认情况下，RabbitMQ会平均的分发消费给C1C2（Round-robin dispatching），假设一个任务的执行时间非常长，在执行过程中，客户端挂了（连接断开），那么，该客户端正在处理且未完成的消息，以及分配给它还没来得及执行的消息，都将丢失。因为默认情况下，RabbitMQ分发完消息后，就会从内存中把消息删除掉。

# **3.消息确认（Message acknowledgment）**

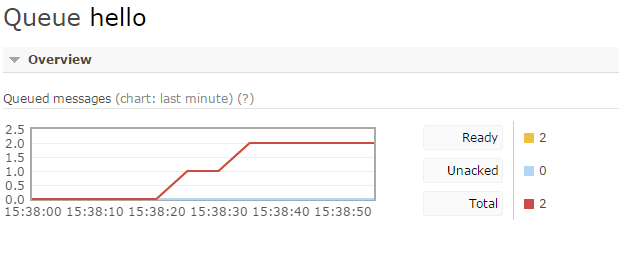
为了解决上述问题，RabbitMQ引入了消息确认机制，当消息处理完成后，给Server端发送一个确认消息，来告诉服务端可以删除该消息了，如果连接断开的时候，Server端没有收到消费者发出的确认信息，则会把消息转发给其他保持在线的消费者。

## **验证上述问题**

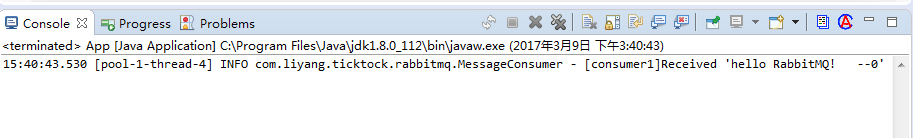
首先，我们验证上述问题（客户端丢失消息）是否真的存在，对Consumer进行如下改造。



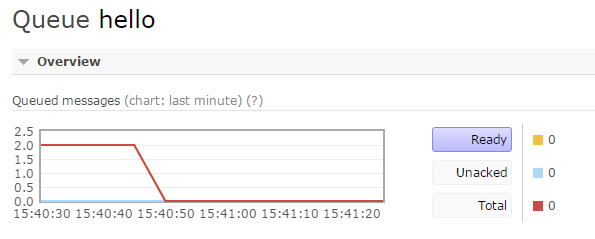
先生产两条消息



启动消费者，在消费者接收到消息，还没处理完成的时候，强制关掉



这时，观察控制台，发现两条消息都没有了，1条是在执行中丢失的，还有1条，已经分配给这个Consumer，还没来得及处理，也丢失了



这证明了上述问题是真的存在的，如果发生在生产环境，将产生难以预料的后果

## **引入消息确认机制**

为了方便观察，我们用CMD来运行Consumer，要通过maven打成可执行的JAR包，需要在pom.xml中增加如下配置

**[IMG_261](http://www.cnblogs.com/Leo_wl/p/javascript:void(0);)**

**IMG_262**

**<build>**

**<finalName>Consumer</finalName>**

**<plugins>**

**<plugin>**

**<artifactId>maven-assembly-plugin</artifactId>**

**<configuration>**

**<appendAssemblyId>false</appendAssemblyId>**

**<descriptorRefs>**

**<descriptorRef>jar-with-dependencies</descriptorRef>**

**</descriptorRefs>**

**<archive>**

**<manifest>**

**<mainClass>com.liyang.ticktock.rabbitmq.App</mainClass>**

**</manifest>**

**</archive>**

**</configuration>**

**<executions>**

**<execution>**

**<id>make-assembly</id>**

**<phase>package</phase>**

**<goals>**

**<goal>assembly</goal>**

**</goals>**

**</execution>**

**</executions>**

**</plugin>**

**<plugin>**

**<groupId>org.apache.maven.plugins</groupId>**

**<artifactId>maven-compiler-plugin</artifactId>**

**<configuration>**

**<source>1.8</source>**

**<target>1.8</target>**

**</configuration>**

**</plugin>**

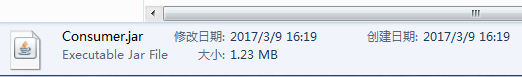
**</plugins>**

**</build>**

**IMG_263**

**[IMG_264](http://www.cnblogs.com/Leo_wl/p/javascript:void(0);)**

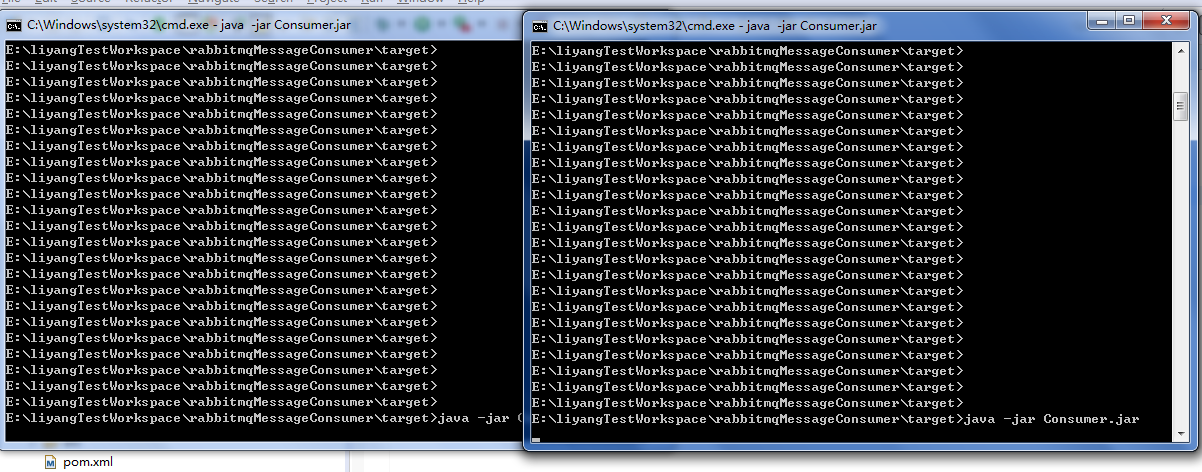
上述配置描述了最终打包名字、入口类路径、带上依赖包、使用1.8版本的JDK进行打包，配置完后，就可以通过maven的install方法，在target目录生成可执行的jar包，如果包大小很小，应检查配置，是不是没有带上依赖包



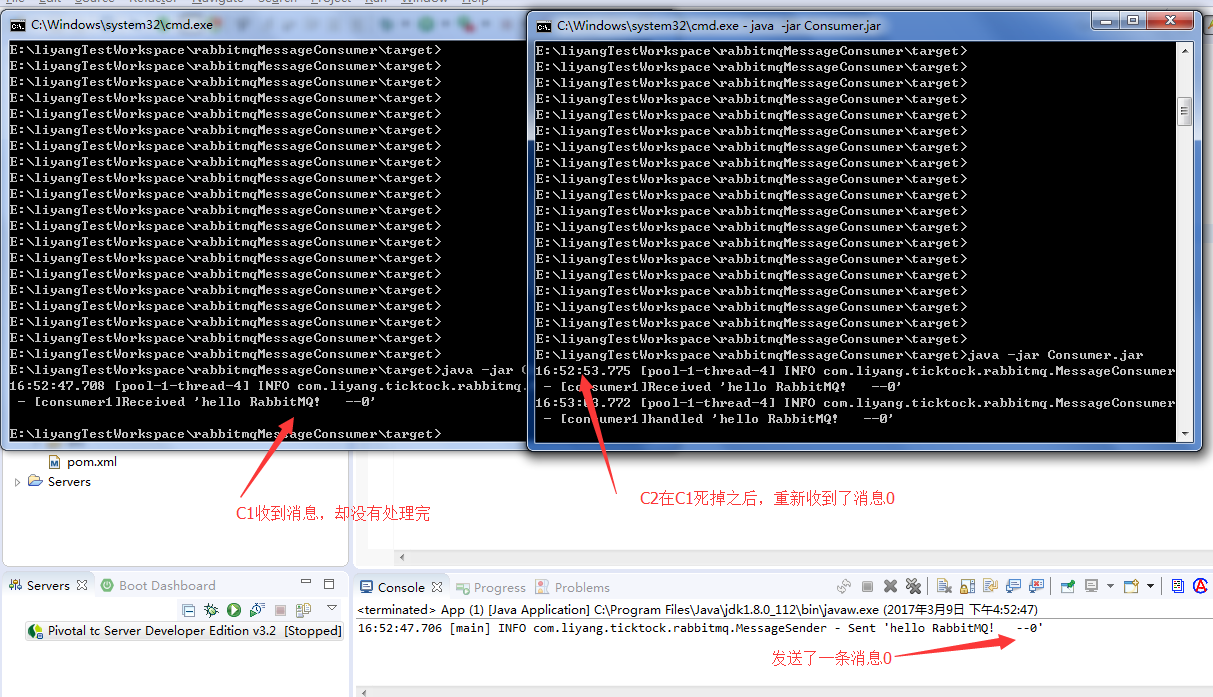
再次改造Consummer类



install成可执行jar包，通过cmd开启两个consumer



通过Sender发送一条消息，然后用Ctrl+C结束先收到消息的Consumer，发现另外一个Consumer接收到了未处理完的消息



问题得到了解决，现在消费者在执行过程中死掉也不会丢失消息了

看一下发送确认的方法

**[IMG_269](http://www.cnblogs.com/Leo_wl/p/javascript:void(0);)**

**IMG_270**

**1 /\*\***

**2 \* Acknowledge one or several received**

**3 \* messages. Supply the deliveryTag from the {@link com.rabbitmq.client.AMQP.Basic.GetOk}**

**4 \* or {@link com.rabbitmq.client.AMQP.Basic.Deliver} method**

**5 \* containing the received message being acknowledged.**

**6 \* @see com.rabbitmq.client.AMQP.Basic.Ack**

**7 \* @param deliveryTag the tag from the received 这个是RabbitMQ用来区分消息的，[文档在这](https://www.rabbitmq.com/amqp-0-9-1-reference.html" \l "basic.deliver.delivery-tag" \t "http://www.cnblogs.com/Leo_wl/p/_blank) 8 \* @param multiple true to acknowledge all messages up to and 为true的话，确认所有消息，为false只确认当前消息**

**9 \* including the supplied delivery tag; false to acknowledge just**

**10 \* the supplied delivery tag.**

**11 \* @throws java.io.IOException if an error is encountered**

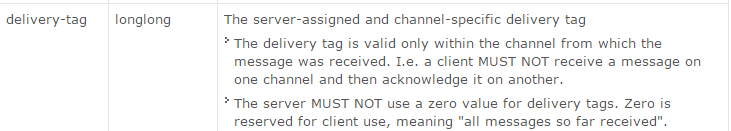
**12 \*/**

**13 void basicAck(long deliveryTag, boolean multiple) throws IOException;**

**IMG_271**

**[IMG_272](http://www.cnblogs.com/Leo_wl/p/javascript:void(0);)**

在官方文档中，这样描述deliveryTag



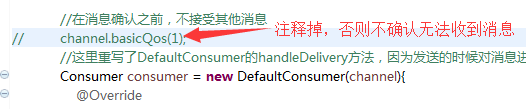
简单来说，就是RabbitMQ内部用来区分消息的一个标签，从envelope中获取就行了

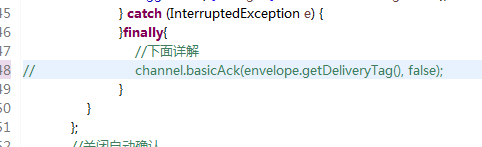
## **忘记确认将引起内存泄漏**

RabbitMQ只有在收到消费者确认后，才会从内存中删除消息，如果消费者忘了确认（更多情况是因为代码问题没有执行到确认的代码），将会导致内存泄漏

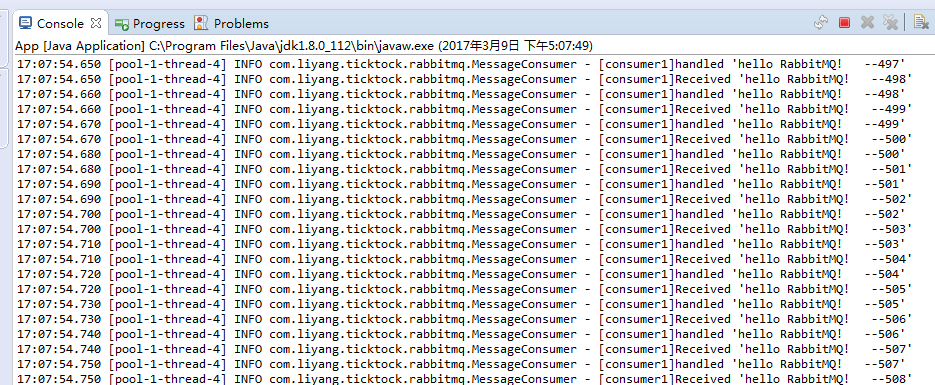
验证一下

注释掉Consumer中的确认代码

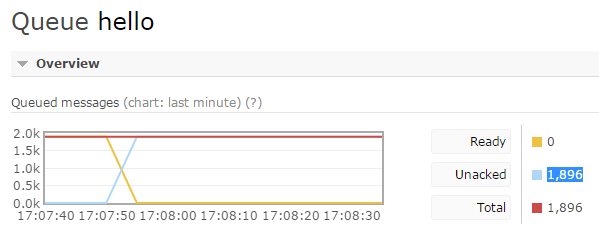


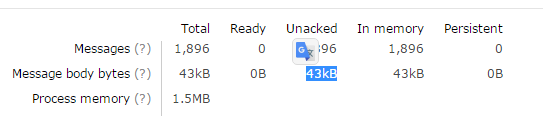


运行Sender和Consumer，不停的生产消费消息，发现消费者在正常的消费消息



查看控制台，发现已经被吃掉了43KB的内存，所以，在试用过程中，一定要保证消息确认在任何情况下都可以发出，否则即使消费者处理完成，RabbitMQ也不会把消息在内存中清除，在该消费者断开连接之后，还会把消息转发给其他消费者重新处理，将引发难以预计的问题





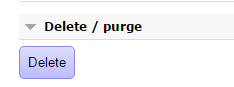
# **4.消息的持久化**

现在，消费者宕机已经无法影响到我们的消息了，但如果RabbitMQ重启了，消息依然会丢失。所幸的是，RabbitMQ提供了持久化的机制，将内存中的消息持久化到硬盘上，即使重启RabbitMQ，消息也不会丢失。但是，仍然有一个非常短暂的时间窗口（RabbitMQ收到消息还没来得及存到硬盘上）会导致消息丢失，如果需要严格的控制，可以参考[官方文档](https://www.rabbitmq.com/confirms.html)

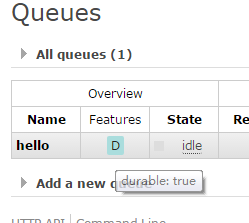
要使用RabbitMQ的消息持久化，在声明队列时设置一个参数即可

image

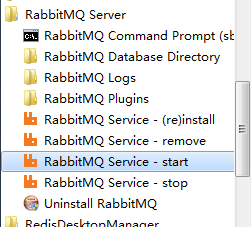
注意，RabbitMQ不允许对一个已经存在的队列用不同的参数重新声明，对于试图这么做的程序，会报错，所以，改动之前代码之前，要在控制台中把原来的队列删除



重新声明队列后，发现Durable为true



重启RabbitMQ



队列的消息没有丢失

