# 1、同时发送两个queue

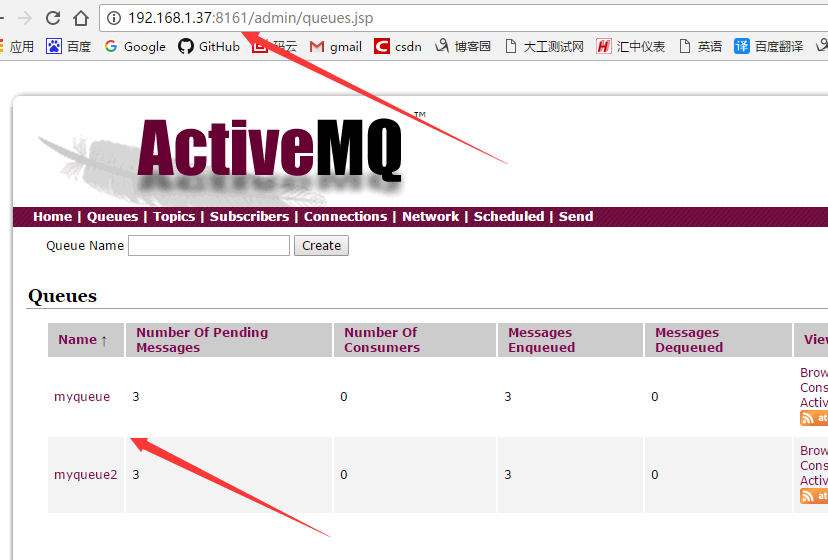
## 1、代码方式

### 1、同时发送两个queue，用逗号隔开

<http://www.cnblogs.com/xinhuaxuan/p/6143232.html>

|  |
| --- |
| Destination destination = session.createQueue("myqueue,myqueue2"); |

### 2、运行这个生产者，会发现产生了两个queue



### 3、消费者也可以利用上面的接收两个（当然也可分别接收啦）

|  |
| --- |
| Destination destination = session.createQueue("myqueue,myqueue2"); |

## 2、xml方式

### 1、activemq.xml中配置拦截器 destinationInterceptors

|  |
| --- |
| <destinationInterceptors>  <virtualDestinationInterceptor>  <virtualDestinations>  <compositeQueue name="MY.QUEUE">  <forwardTo>  <queue physicalName="my-queue" />  <queue physicalName="my-queue2" />  </forwardTo>  </compositeQueue>  </virtualDestinations>  </virtualDestinationInterceptor>  </destinationInterceptors> |

### 2、生产者代码，使用上面的compositeQueue

|  |
| --- |
| Destination destination = session.createQueue("MY.QUEUE"); |

### 3、消费者能够成功接收上面的两个queue

# 2、删除不使用的队列，

## 解释：在ActiveMQ的queue在不使用之后，可以通过web控制台或者JMX方式来删除掉，当然，也可以通过配置，使得broker可以自动探测到无用的队列并删除掉，回收响应资源。

## 1、参数解释：

### schedulePeriodForDestinationPurge: 设置多长时间检查一次，这里是10秒。默认是0秒

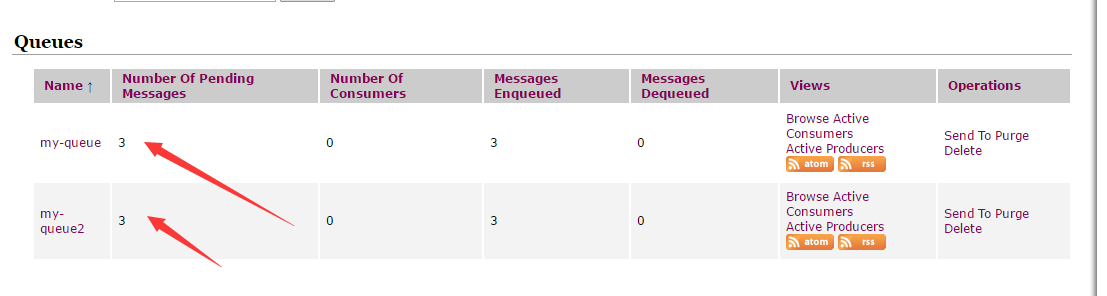
### inactiveTimoutBeforeGC: 设置当Destination为空后，多长时间被删除，这里是30秒。默认是60秒

### gcInactiveDestinations:设置删除掉不活动的队列，默认为false

## 2、配置如下 2秒检查一次

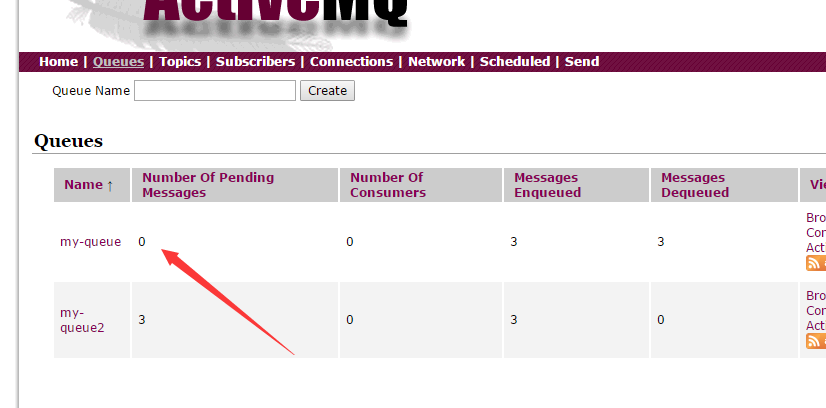
|  |
| --- |
| <broker xmlns="http://activemq.apache.org/schema/core"  brokerName="localhost"  dataDirectory="${activemq.data}"  schedulePeriodForDestinationPurge="1000"  >  <destinationPolicy>  <policyMap>  <policyEntries>  <policyEntry topic=">" >      <pendingMessageLimitStrategy>  <constantPendingMessageLimitStrategy limit="1000"/>  </pendingMessageLimitStrategy>  </policyEntry>  <policyEntry queue=">" gcInactiveDestinations="true" inactiveTimeoutBeforeGC="5000" />  </policyEntries>  </policyMap>  </destinationPolicy> |

## 3、测试，运行生产者，会产生下面的2条，这两条有数据是不会消失的。

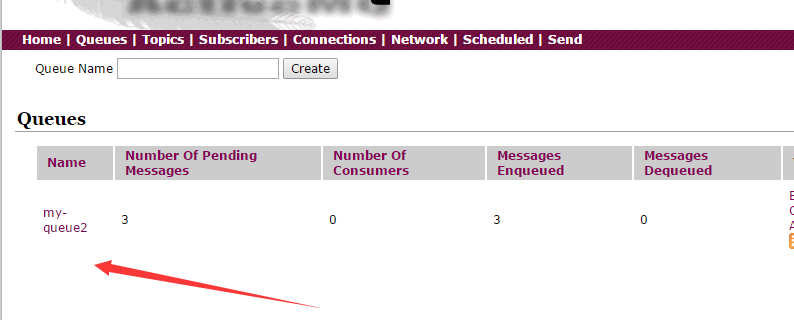


## 4、运行消费者，先消化一个my-queue.3秒内观察浏览器，会发现有0个消息的my-queue，但是3秒之后会被删除

### 1、3秒内，存在这个my-queue



### 2、3秒后，只剩下一个了哦(当然要记得关闭消费者客户端，防止一直去读消息，因为我这里用了线程，创建了多个消费者)



# 3、如何使用虚拟topic（把1.2的配置注释了）消费者为队列queue

### 第一：对于消息发布者来说，就是一个正常的topic,名称以VirtualTopic.开始，比如VirtualTopic.Orders,代码示例如下：

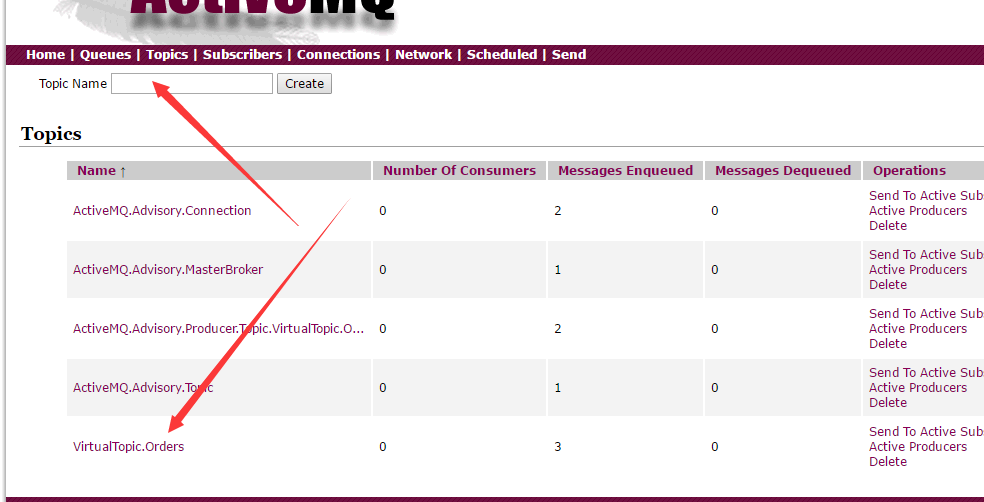
#### destination = session.createTopic("VirtualTopic.Orders");

### 第二：对于消息接收端来说，是个队列，不同应用里使用不同的前缀作为队列名称，即可表明自己的身份即可实现消费端应用分组，

#### 例如Consumer.A.VirtualTopic.Orders说明它是名称为A的消费端，同理Consumer.B VirtualTopic.Orders说明是一名称为B的消费端。可以在同一个应用中使用多个消费者消费这个队列

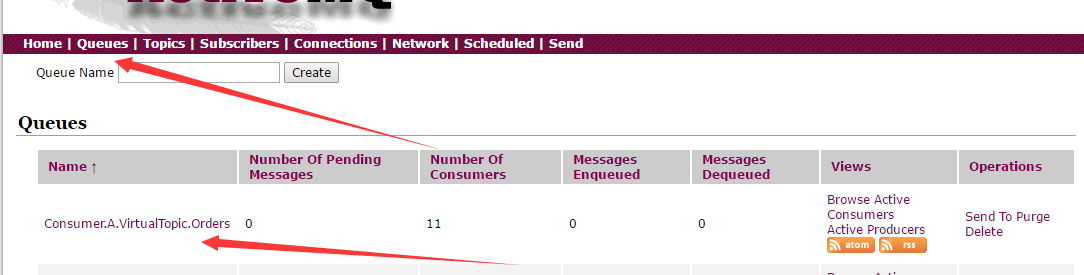
## 1、创建一个生产者

|  |
| --- |
| Destination destination = session.createTopic("VirtualTopic.Orders"); |

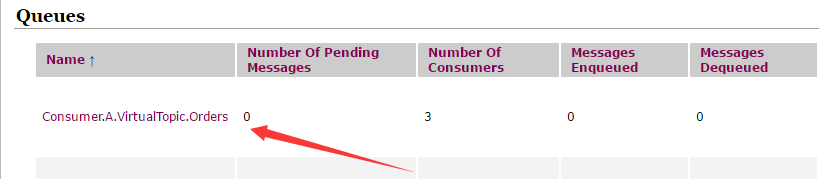


## 2、消费者队列创建，然后运行，但是会发现没有收到消息，所以先关闭，这样就是订阅了topic

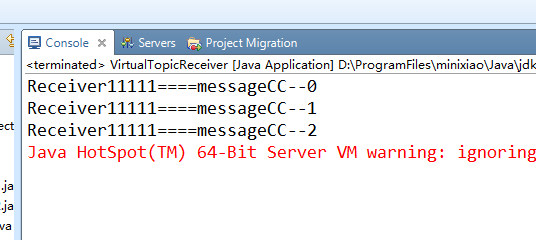
|  |
| --- |
| Destination destination = session.createQueue("Consumer.A.VirtualTopic.Orders"); |

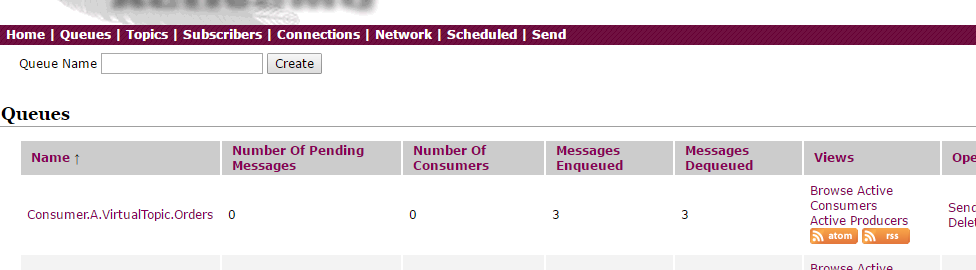


## 3、再次运行消费者，发现还是没有数据，



## 4、这个时候不关闭消费者，运行生产者，就会发现我们将消费拿到了（这个其实就是非持久化的topic一样，必须消费者在线）





# 4、修改消费者虚拟主机前缀，拦截器配置

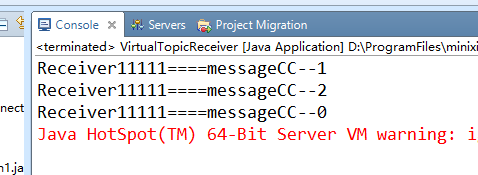
## 1、activemq.xml中配置

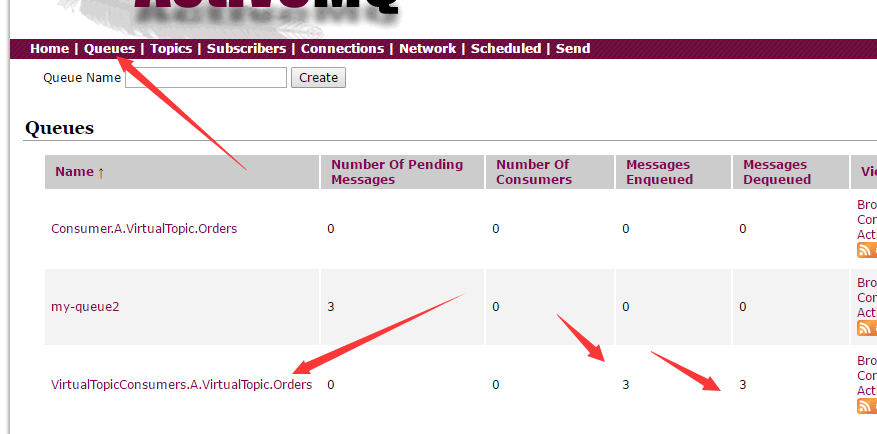
|  |
| --- |
| <destinationInterceptors>  <virtualDestinationInterceptor>  <virtualDestinations>  <compositeQueue name="MY.QUEUE">  <forwardTo>  <queue physicalName="my-queue" />  <queue physicalName="my-queue2" />  </forwardTo>  </compositeQueue>  <virtualTopic name=">" prefix="VirtualTopicConsumers.\*." selectorAware="false" />  </virtualDestinations>  </virtualDestinationInterceptor>  </destinationInterceptors> |

## 2、生产者VirtualTopic.Orders不变，消费者如下

|  |
| --- |
| /\*\*  \* 自定义的前缀 拦截器中配置  \*/  Destination destination = session.createQueue("VirtualTopicConsumers.A.VirtualTopic.Orders"); |

## 3、运行消费者，再运行生产者，可以收到数据





# 5、queue发送给topic

## 1、activemq中配置，Broker会把发送到某一个队列上的所有消息转发到一个名称类似的topic,因此监控程序只需要订阅这个mirroredqueue topic.为启用MirroredQueue，

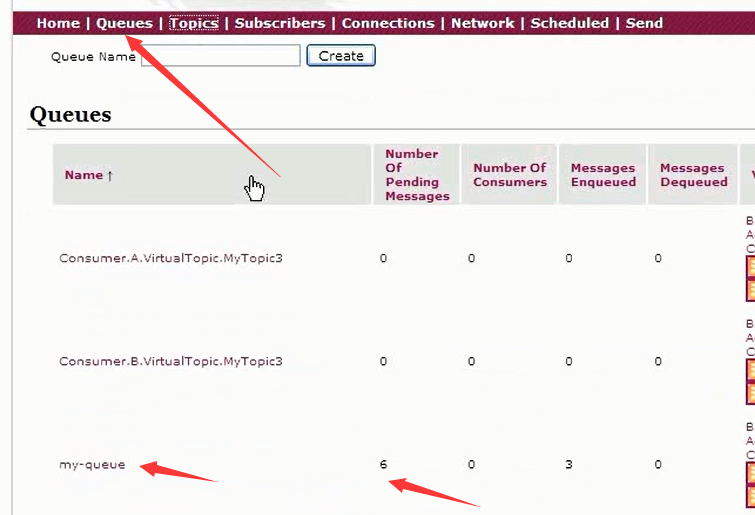
|  |
| --- |
| <broker xmlns="http://activemq.apache.org/schema/core"  brokerName="localhost"  dataDirectory="${activemq.data}"  schedulePeriodForDestinationPurge="2000"  useMirroredQueue="true"  > |

## 2、通过destinationInterceptors设置其属性，如mirrortopic的前缀，缺省是VritualTopic.Mirror.

|  |
| --- |
| <destinationInterceptors>  <mirroredQueue copyMessage="true" postfix=".qmirror" prefix="" /> |

## 3、创建生产者 my-queue

### 1、queue 浏览器中查看



### 2、topic 浏览器中查看



## 4.消费者中获取topic

|  |
| --- |
| destination = session.createTopic("my-queue.qmirror"); |



