# 什么是消息中间件：

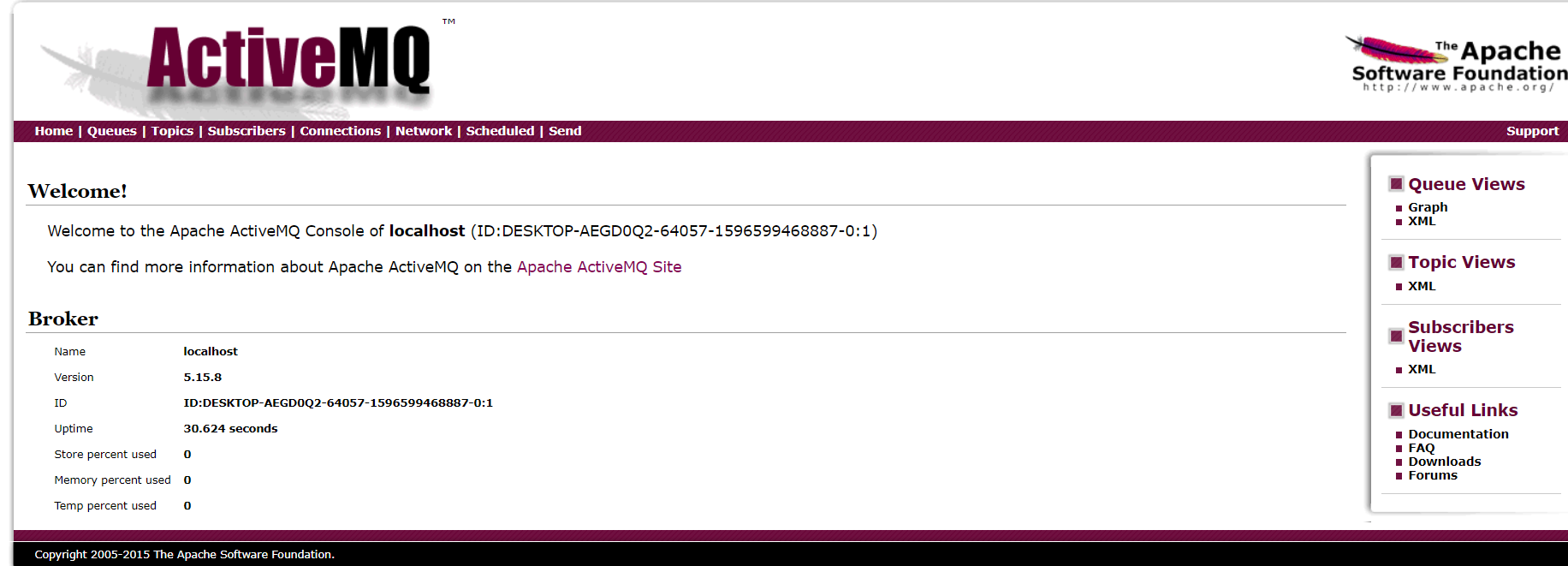
是一种订阅发布模式工具组件，用来派发消息：

原文：

什么是消息中间件呢？  
以站长的公众号为例，如果某学员订阅（关注）了这个公众号，每当站长发布新教程的时候，都可以在这个公众号得到通知，这就是一种广播订阅模式。  
而公众号如何实现这一点呢？ 就可以通过 消息中间件 来轻松实现。  
站长把最新的教程信息 发给 消息中间件服务器， 学员手机上的微信里的消息中间件客户端，就会自动去把消息获取出来显示，这样站长就达到了教程广播的效果了。

# ActiveMQ：

安装好之后是这样的：



## 队列模式（随机分配）：

ActiveMQ有两种模式，一种是队列模式。“随机匹配”，一种是主题模式。“人人都有”

队列模式的意思是消息全部在同一个队列当中，“随机分发”到消费端。意思是不同消费端收到的东西是不一样的，队列模式也叫分食模式。“随机匹配”

而主题模式是传统的信息广播订阅发布模式，一个消息从生产端出来会传播到所有的消费端。“人人都有”

步骤就是：

1.生产端打开端口对话，定义destination类型（队列还是主题），定义消息名称

2.消费端打开相同的端口，定义相同的destination类型和相同的消息名，并监听来自生产端的信息。

3.生产端发消息，消费端按队列类型还是主题类型接受消息。

原文：

activeMQ 有两种模式，分别是队列模式和主题模式。

队列模式，其实就是分食模式。 比如生产方发了 10条消息到 activeMQ 服务器， 而此时有多个 消费方，那么这些消费方就会瓜分这些10条消息，一条消息只会被一个消费方得到。

主题模式，就是订阅模式。 比如生产方发了10条消息，而此时有多个消费方，那么多个消费方都能得到这 10条消息，就如同订阅公众号那样。  
  
而本知识点会演示如何开发队列模式。

和Kafka一样，需要生产者和消费者。

原来的生产者就是Flume，在这里生产者是Producer:



还要有消费者，在原来的Kafka中，消费者是Spark，这次是Consumer：

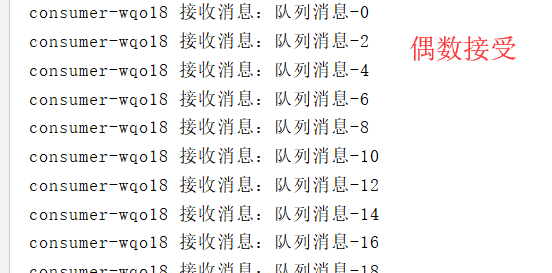


监听器：

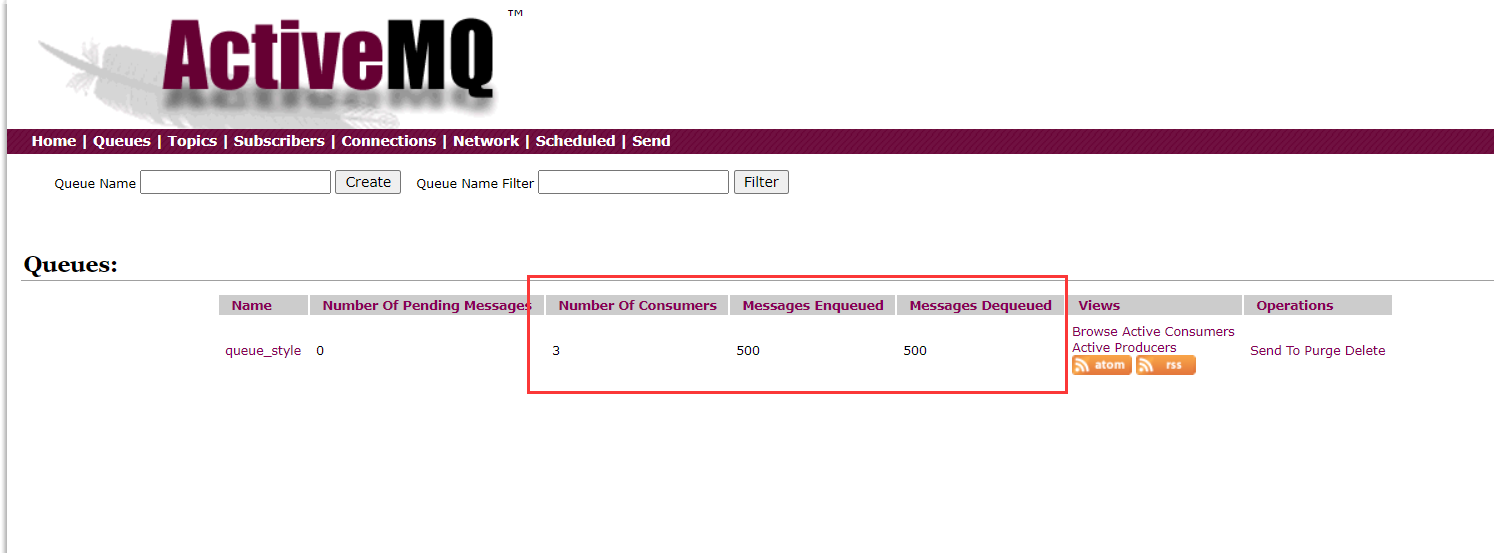


结果:打开两个Consumer监听，再开启一个Producer，会发现发送的100个数字中，奇数的部分由一个consumer接收到了，偶数的部分由另一个consumer接收到了，这就体现了分食模式。

当然，如果开很多consumer的话，就不是这种奇数偶数分配了，但也是分配。



ActiveMQ控制台的信息:



显示了consumer的个数和消息的收发总数。

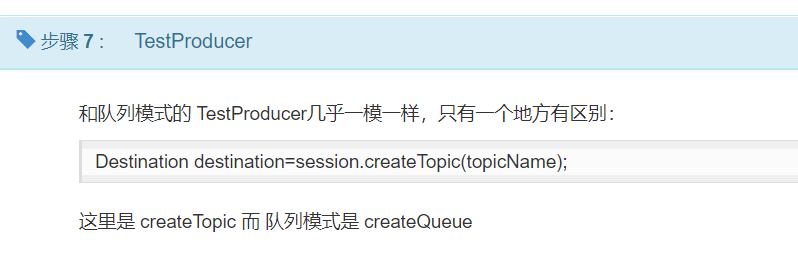
## 主题模式（人人都有）：

一个消息发出去，所有的消费端都能收到同样的信息，不用争抢的方式，订阅发布。

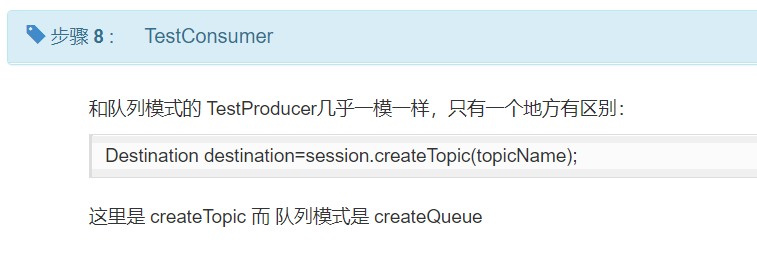
原文：

前面学习了队列模式，现在来学习主题模式。  
主题模式就是每个订阅了的消费者，都可以获取所有的消息，而不像队列模式那样要争抢~

主题模式的producer编写和队列模式几乎一摸一样，都是打开端口的对话，然后发消息，除了模式不一样外：



Consumer也一样，就是对Destination的创建变为Topic模式：



结果就是生产者往端口里发东西，两个consumer都接收到了所有内容。

## 3.Spring的模式：

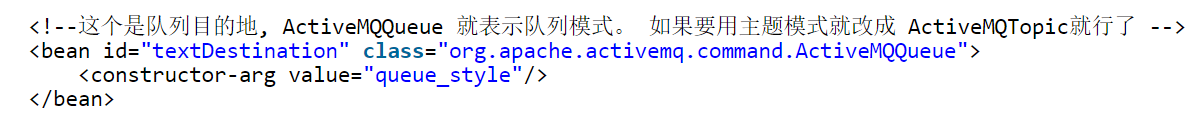
[https://how2j.cn/k/message/message-activemq-spring/2028.html#nowhere](https://how2j.cn/k/message/message-activemq-spring/2028.html" \l "nowhere)

要配置去这看，pom文件啥的还比较多。

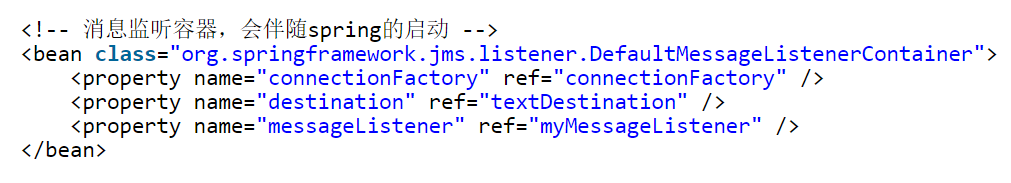
用这样的方式，利用Spring的自动注入，可以不用自己写很多链接端口的代码了



修改模式的地方：



Spring在这里默认定义了一个consumer监听器，意思是一开始consumer就固定有一个了（producer没有），在自定义consumer的话就在一个的基础上往上加。



生产者：





消费者：

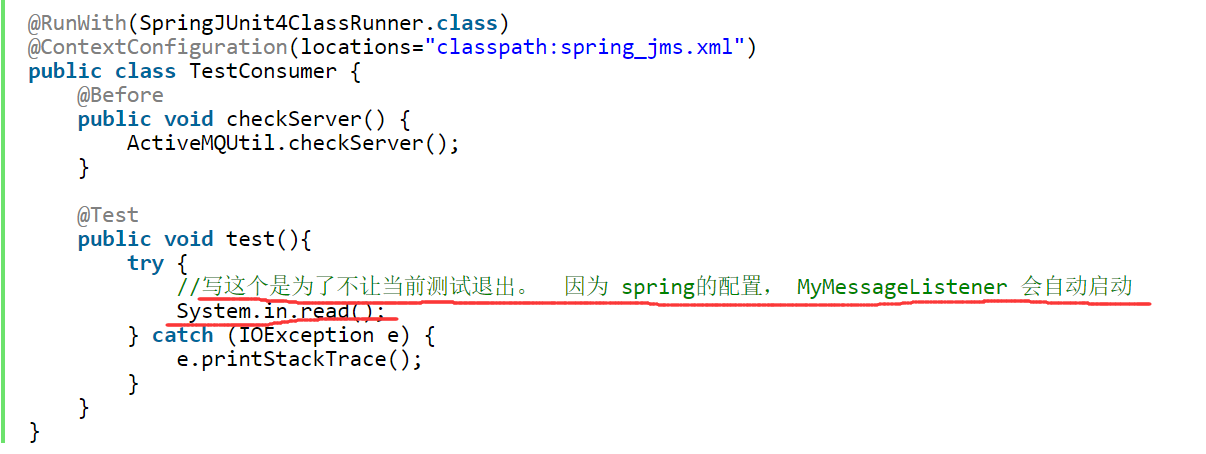


在Spring\_jms配置文件中,定义了消费者容器，专门用来管理一个自带的默认监听器和注册的自定义consumers（下面就一共两个监听器），只要Spring运行，监控器容器中的consumer就会保持监听。

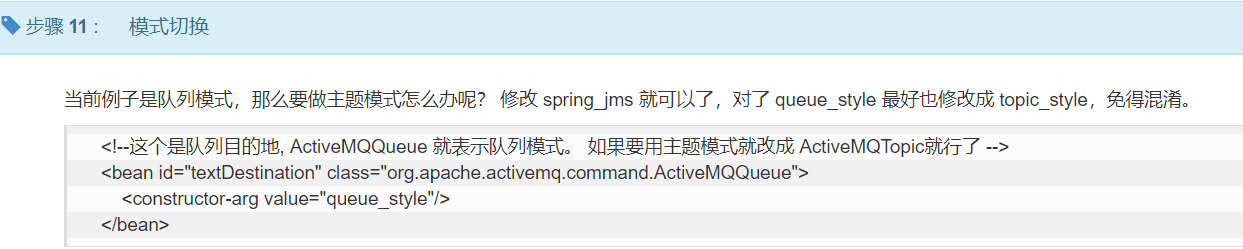


测试消费者:

消费者测试类实际上只是保持Spring框架的运行，没干啥实际业务操作，因为是在spring\_jms配置文件中定义的容器，只要框架运行，consumer监听者们就会不断保持监听。



Queue和Topic的模式切换



# RabbitMQ：

与activemq不一样， rabbitmq 使用的是一种叫做 AMQP 的协议来通信。 AMQP 是 dvanced Message Queuing Protocol 的缩写。协议内容我们就没必要深入研究了，简单地说，通过这种协议，可以处理更为复杂的业务需求~  
以下就是基于 AMQP 这种协议，可以实现的各种模式

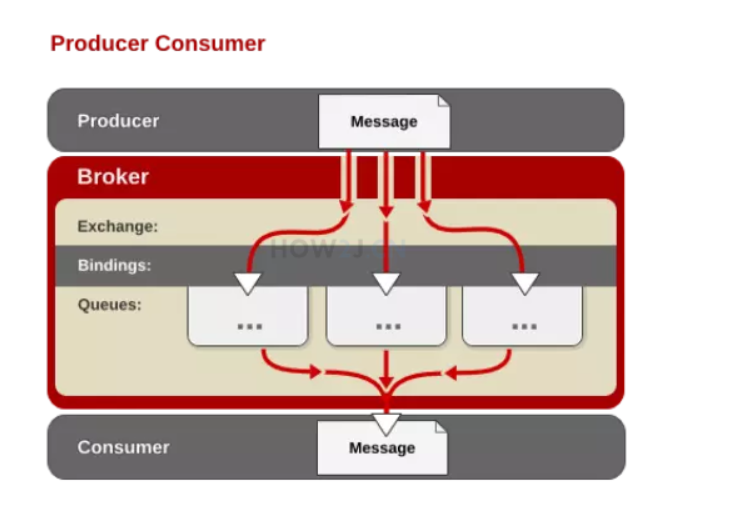
## 多了一层交换机Exchange：

与 ActiveMQ 拿到消息就直接放在队列等待消费者拿走不同， Rabbit 拿到消息之后，会先交给 交换机 （Exchange）, 然后交换机再根据预先设定的不同绑定( Bindings )策略，来确定要发给哪个队列。

原来ActiveMQ是：生产者—>消息队列—>消费者

RabbitMQ：生产者 🡪 交换机🡪 消息队列 🡪 消费者

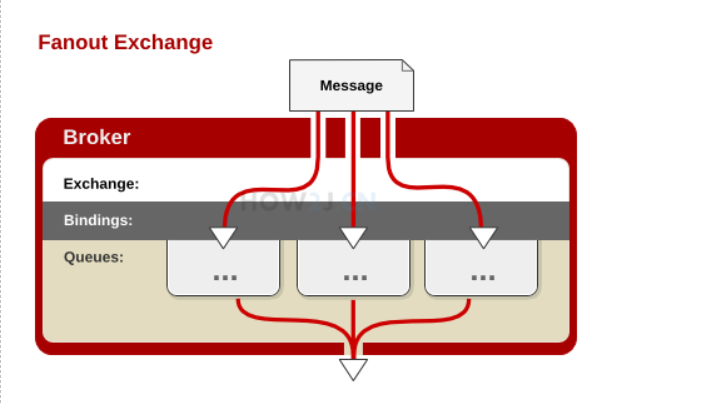
如图所示，比起 ActiveMQ 多了 Exchange 和 Bindings。  
正式由于有了 Exchange 和 Bindings， RabbitMQ 就可以灵活地支撑各种模式。



RabbitMQ提供了四种Exchange模式：fanout,direct,topic,header 。 header模式在实际使用中较少，这里只讨论前三种模式.

### Fanout 广播模式:

就是广播模式，消息得到后散发到所有队列。



生产端：元素有两个，channel，交换机。通过channel去发消息。



消费端： 元素有三个：channel，交换机，和临时队列。Channel和交换机绑定，channel又和临时队列绑定，通过channel来收消息。

应该是，channel通过临时队列，向交换机中取数据。

识别名称是通过命名交换机完成的。而没有交换机就命名channel名做识别。



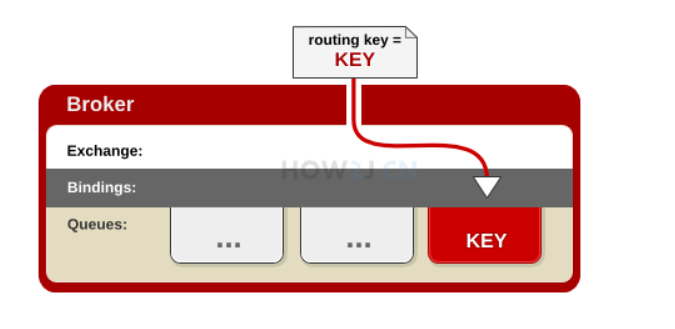
监听consumer：只要频道有消息过来，就触发回调函数。



结果就是正常的广播收发。

### Dirct 指定模式

交换机将消息指定交给某个队列。其他队列收不到。



生产端：



消费端:



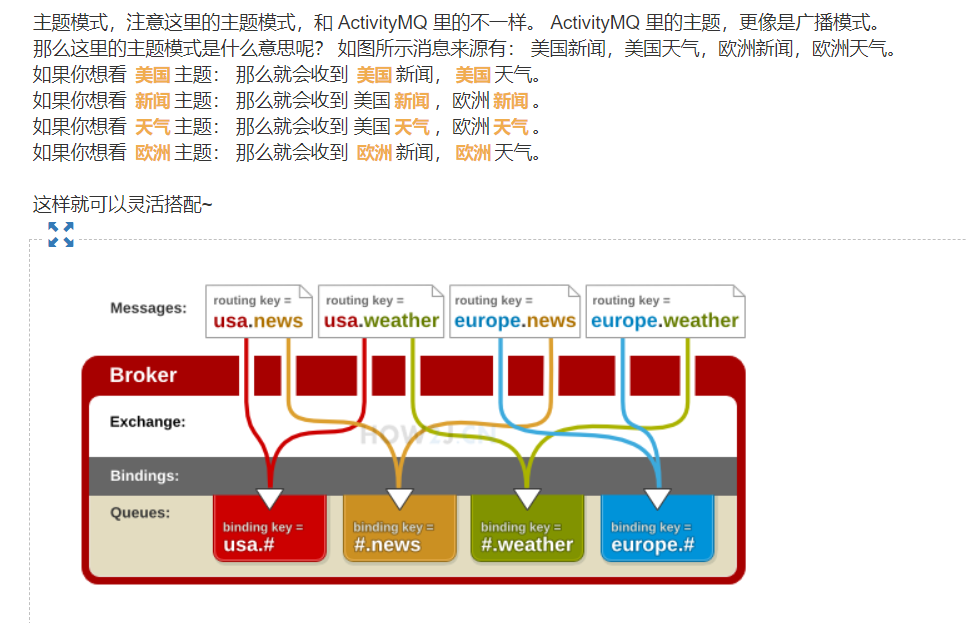
打开两台consumer，一台producer，效果就是两台consumer分食了producer的内容。

和ActiveMQ中的队列模式一样（分食模式）

### Topic 关键词模式

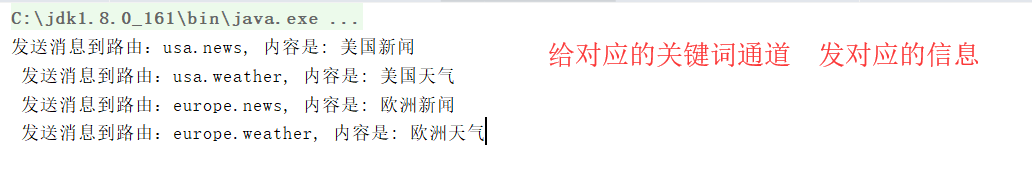
与ActiveMQ不一样，AMQ里面是广播模式叫做Topic。

Rabbit的topic模式很灵活，可以通过关键词来获取相对的信息进入消息队列：



Producer:





Consumer:

给交换机设置关键词

