# ActiveMQ从入门到精通（一）

原文地址: <http://www.jianshu.com/p/ecdc6eab554c>

这是关于消息中间件ActiveMQ的一个系列专题文章，将涵盖JMS、ActiveMQ的初步入门及API详细使用、两种经典的消息模式（PTP and Pub/Sub）、与Spring整合、ActiveMQ集群、监控与配置优化等。话不多说，我们来一起瞧一瞧！

# JMS

首先来说较早以前，也就是没有JMS的那个时候，很多应用系统存在一些缺陷：

1.通信的同步性

client端发起调用后，必须等待server处理完成并返回结果后才能继续执行

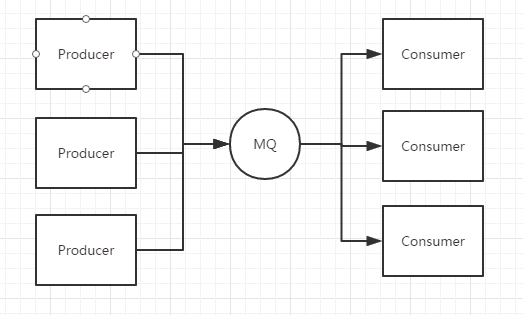
2.client 和 server 的生命周期耦合太高

client进程和server服务进程都必须可用，如果server出现问题或者网络故障，那么client端会收到异常

3.点对点通信

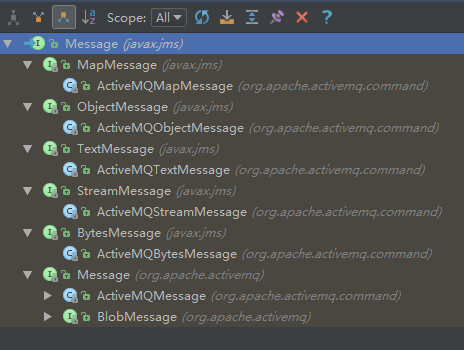
client端的一次调用只能发送给某一个单独的服务对象，无法一对多

JMS，即Java Message Service，通过面向消息中间件（MOM：Message Oriented Middleware）的方式很好的解决了上面的问题。大致的过程是这样的：发送者把消息发送给消息服务器，消息服务器将消息存放在若干队列/主题中，在合适的时候，消息服务器会将消息转发给接受者。在这个过程中，发送和接受是异步的，也就是发送无需等待，而且发送者和接受者的生命周期也没有必然关系；在pub/sub模式下，也可以完成一对多的通信，即让一个消息有多个接受者。



JMS

需要注意的是，JMS只是定义了Java访问消息中间件的接口，其实就是在包javax.jms中，你会发现这个包下除了异常定义，其他都是interface。我们可以扫一眼，比如Message：



Message接口

我想你应该发现了，JMS只给出接口，然后由具体的中间件去实现，比如ActiveMQ就是实现了JMS的一种Provider，还有阿里巴巴的RocketMQ（后续专题中在为大家介绍）。这些消息中间件都符合JMS规范。说起规范，自然要定义一些术语：

Provider/MessageProvider：生产者

Consumer/MessageConsumer：消费者

PTP：Point To Point，点对点通信消息模型

Pub/Sub：Publish/Subscribe，发布订阅消息模型

Queue：队列，目标类型之一，和PTP结合

Topic：主题，目标类型之一，和Pub/Sub结合

ConnectionFactory：连接工厂，JMS用它创建连接

Connnection：JMS Client到JMS Provider的连接

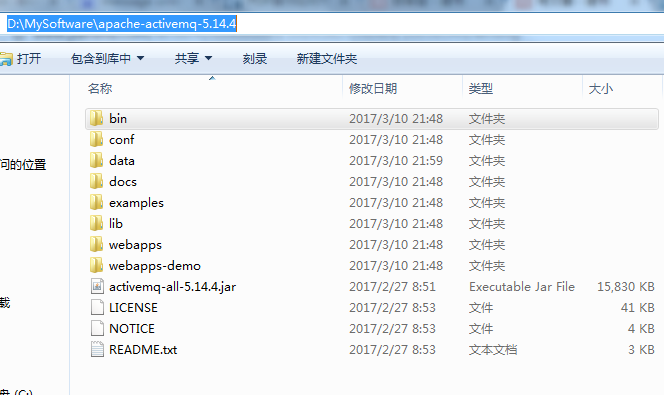
Destination：消息目的地，由Session创建

Session：会话，由Connection创建，实质上就是发送、接受消息的一个线程，因此生产者、消费者都是Session创建的

初步来看，Session非常核心，因为很多东西都是它创建的，在后文中可以通过代码来进一步认识这些术语。

# ActiveMQ QuickStart

ActiveMQ是Apache出品的，非常流行的消息中间件，可以说要掌握消息中间件，需要从ActiveMQ开始，要掌握更加强大的RocketMQ，也需要ActiveMQ的基础，因此我们来搞定它吧。官网地址：http://activemq.apache.org/，目前最新的版本是5.14.4，我这边将以最新版来讲解。这篇文章主要是ActiveMQ的初步，因此我这边暂时用windows版本，后期采用Linux。



ActiveMQ目录结构

bin下面存放的是ActiveMQ的启动脚本activemq.bat，注意分32、64位

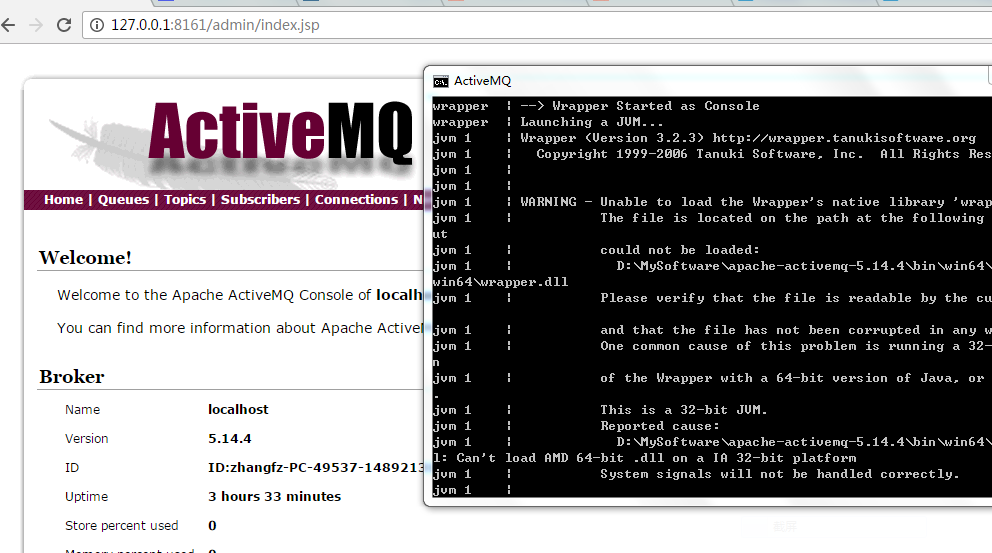
conf里面是配置文件，重点关注的是activemq.xml、jetty.xml、jetty-realm.properties。在登录ActiveMQ Web控制台需要用户名、密码信息；在JMS CLIENT和ActiveMQ进行何种协议的连接、端口是什么等这些信息都在上面的配置文件中可以体现。

data目录下是ActiveMQ进行消息持久化存放的地方，默认采用的是kahadb，当然我们可以采用leveldb，或者采用JDBC存储到MySQL，或者干脆不使用持久化机制。

webapps，注意ActiveMQ自带Jetty提供Web管控台

lib中ActiveMQ为我们提供了分功能的JAR包，当然也提供了activemq-all-5.14.4.jar

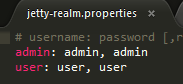
在JDK安装没有问题的情况下，直接activemq.bat启动它，并访问Web控制台！



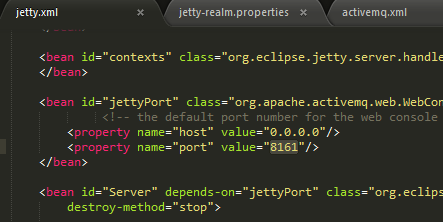
ActiveMQ Start

到这里，ActiveMQ就已经启动了，So easy~

访问ActiveMQ web控制台的用户名、密码在哪里配置的？URL当中的端口是在哪里配置的？



username/pwd 4 access web

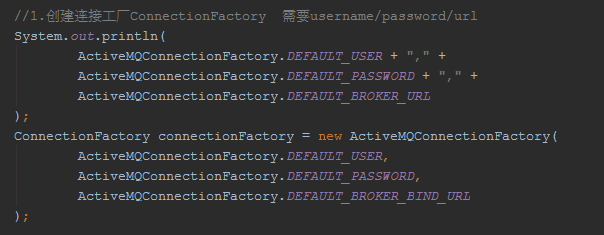


port 4 web console

# Write Code 4 ActiveMQ

来一个HelloWorld级别的例子，来感受下ActiveMQ。具体来说，我这边会写一个生产者用于发送消息，一个消费者用于接收消息。实际上，JMS是有“套路”的，下面我将以生产者为例详细说明。

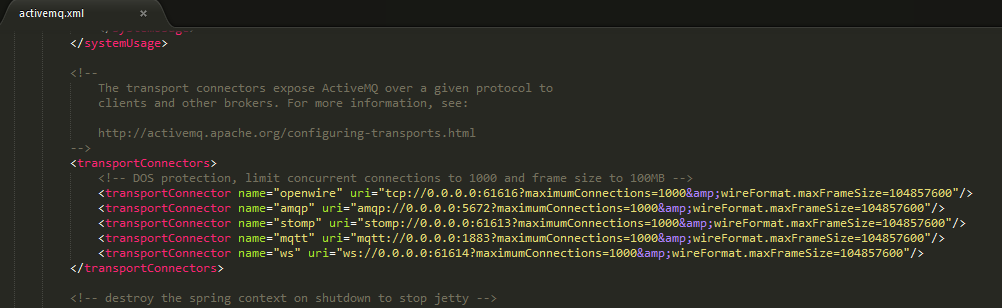
## 第一步：创建ConnectionFactory连接工厂



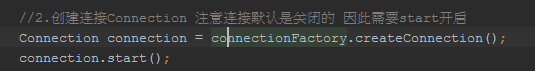
ConnectionFactory

实际上，这里是存在安全隐患的，也就是任何人一旦知道MQ的地址，就可以连接访问了，我们可以在activemq.xml中配置指定的用户、密码才能访问ActiveMQ。

关于broker\_bind\_url，默认就是tcp://localhost:61616，说明是采用TCP协议，61616端口。其实对于ActiveMQ不仅仅支持TCP协议，还有其他协议，开启了多个端口。



## 第二步：创建Connection



Connection

Connection就代表了应用程序和消息服务器之间的通信链路。获得了连接工厂后，就可以创建Connection。

事实上，ConnectionFactory存在重载方法：

Connection createConnection(String username,String password)

也就是说我们也可以在这里指定用户名、密码进行验证

## 第三步：创建Session



Session

Session，用于发送和接受消息，而且是单线程的，支持事务的。如果Session开启事务支持，那么Session将保存一组信息，要么commit到MQ，要么回滚这些消息。Session可以创建MessageProducer/MessageConsumer。

## 第四步：创建Destination

IMG_267

Destination

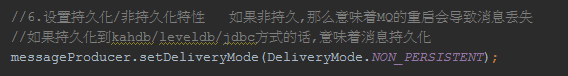
所谓消息目标，就是消息发送和接受的地点，要么queue，要么topic。

## 第五步：创建MessageProducer

IMG_268

MessageProducer

## 第六步：设置持久化方式



持久化方式设置

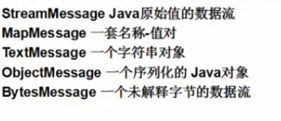
## 第七步：定义消息对象，并发送



Message

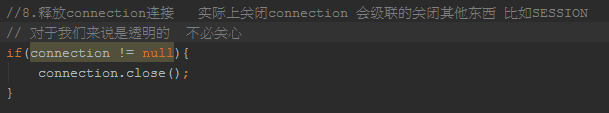
生产者和消费者之间传递的对象，由3个主要部分构成：

消息头（路由）+消息属性（消息选择器，以后介绍）+消息体（JMS规范的5种类型消息）



消息类型

## 第八步：释放连接

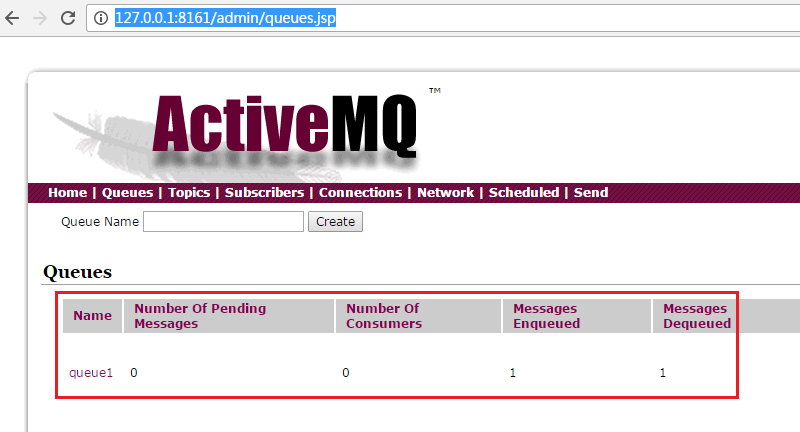


release resource

必须close connection，只有这样ActiveMQ才会释放资源！

消费者的代码和上面非常类似，只不过就是创建MessageConsumer进行receive而已，注意receive()/receive(long)/receiveNoWait()，这些说明消费者可以采用阻塞模式、非阻塞模式接受消息。

程序运行后，我们来看一下管控台：



ActiveMQ Web Info

[Messages Enqueued](http://127.0.0.1:8161/admin/queues.jsp" \t "http://www.jianshu.com/p/_blank)：表示生产了多少条消息，记做P

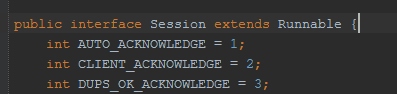
[Messages Dequeued](http://127.0.0.1:8161/admin/queues.jsp" \t "http://www.jianshu.com/p/_blank)：表示消费了多少条消息，记做C

[Number Of Consumers](http://127.0.0.1:8161/admin/queues.jsp" \t "http://www.jianshu.com/p/_blank)：表示在该队列上还有多少消费者在等待接受消息

[Number Of Pending Messages](http://127.0.0.1:8161/admin/queues.jsp" \t "http://www.jianshu.com/p/_blank)：表示还有多少条消息没有被消费，实际上是表示消息的积压程度，就是P-C

# 在说说Session

在通过Connection创建Session的时候，需要设置2个参数，一个是否支持事务，另一个是签收的模式。我们重点说一下签收模式：



签收模式

什么是签收？通俗点说，就是消费者接受到消息后，需要告诉消息服务器，我收到消息了。当消息服务器收到回执后，本条消息将失效。因此签收将对PTP模式产生很大影响。如果消费者收到消息后，并不签收，那么本条消息继续有效，很可能会被其他消费者消费掉！

AUTO\_ACKNOWLEDGE：表示在消费者receive消息的时候自动的签收

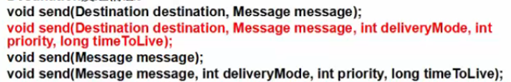
CLIENT\_ACKNOWLEDGE：表示消费者receive消息后必须手动的调用acknowledge()方法进行签收

DUPS\_OK\_ACKNOWLEDGE：签不签收无所谓了，只要消费者能够容忍重复的消息接受，当然这样会降低Session的开销

在实际中，我们应该采用哪种签收模式呢？CLIENT\_ACKNOWLEDGE，采用手动的方式较自动的方式可能更好些，因为接收到了消息，并不意味着成功的处理了消息，假设我们采用手动签收的方式，只有在消息成功处理的前提下才进行签收，那么只要消息处理失败，那么消息还有效，仍然会继续消费，直至成功处理！

# 关于消息的priority/ttl/deliveryMode

消息有优先级及存活时间，在MessageProducer进行send的时候，存在多个重载方法，我们来看一下：



send

在上面的code当中，我们创建生产者的时候，指定了Destination，设置了持久化方式，实际上这些都可以不必指定的，而是到send的时候指定。而且在实际业务开发中，往往根据各种判断，来决定将这条消息发往哪个Queue，因此往往不会在MessageProducer创建的时候指定Destination。

TTL，消息的存活时间，一句话：生产者生产了消息，如果消费者不来消费，那么这条消息保持多久的有效期

priority，消息优先级，0-9。0-4是普通消息，5-9是加急消息，消息默认级别是4。注意，消息优先级只是一个理论上的概念，并不能绝对保证优先级高的消息一定被消费者优先消费！也就是说ActiveMQ并不能保证消费的顺序性！

deliveryMode，如果不指定，默认是持久化的消息。如果可以容忍消息的丢失，那么采用非持久化的方式，将会改善性能、减少存储的开销。

# ActiveMQ从入门到精通（二）

[](http://www.jianshu.com/u/cb569cce501b)

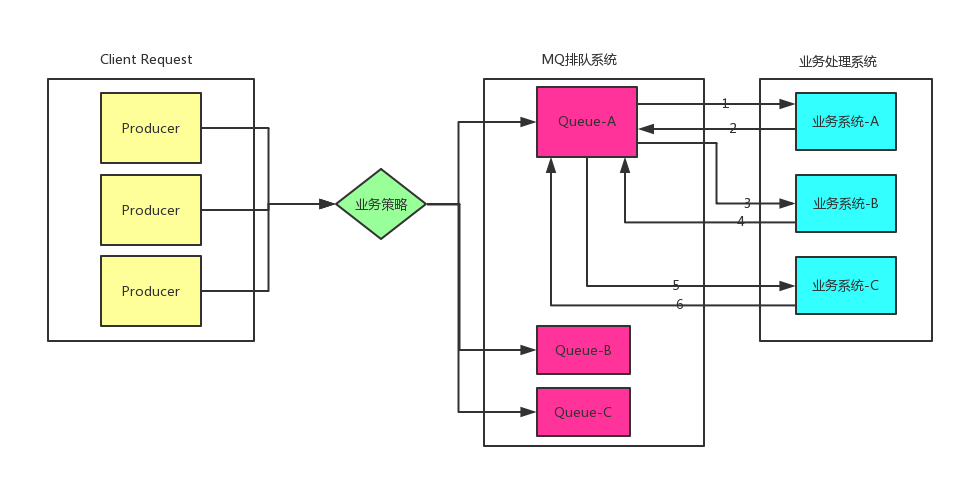
作者 [张丰哲](http://www.jianshu.com/u/cb569cce501b)

2017.03.19 14:19 字数 2713 阅读 364评论 2喜欢 17

接上一篇[《ActiveMQ从入门到精通（一）》](http://www.jianshu.com/p/ecdc6eab554c" \t "http://www.jianshu.com/p/_blank)，本篇主要讨论的话题是：消息的顺序消费、JMS Selectors、消息的同步/异步接受方式、Message、P2P/PubSub、持久化订阅、持久化消息到MySQL以及与Spring整合等知识。

# 消息的顺序消费

在上一篇文章中，我们已经明确知道了ActiveMQ并不能保证消费的顺序性，即便我们使用了消息优先级。而在实际开发中，有些场景又是需要对消息进行顺序消费的，比如：用户从下单、到支付、再到发货等。如果使用ActiveMQ该如何保证消费的顺序性呢？



消息消费顺序性

首先来说，在实际中，我们并不需要的是对全部消息的全局有序消费，我们仅仅需要的是局部业务有序性消费。比如说，我们仅仅需要的是一个用户的下订单、支付、发货这个过程的3条消息有序消费。

比如，我们可以根据用户ID简单做一个HASH，将消息定位到不同的队列上，也就意味着同一个用户的消息将发往同一个队列。这样做的好处在于，多个队列之间可以并行处理。

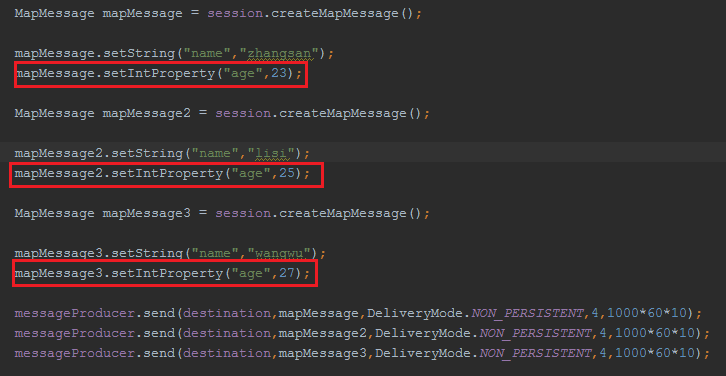
然后，在队列上可以对一段时间上的消息按照用户分组进行排序，这只是一个少量消息的局部排序而已，比如Queue-A上有一个用户的3条消息（订单消息msg1、支付消息msg2、发货消息msg3），那么，msg1将交给订单业务系统，处理完成后，msg2交给支付系统，处理完成后，msg3交给发货系统。**虽然这个处理过程是同步的（一条消息处理完，在接着处理），但是它的并发性，系统的处理能力并没有下降！为什么这么说呢？**

假设，msg1/msg2/msg3处理各需要0.1S，如果订单业务系统、支付系统、发货系统并没有分开，而是一个“大系统”，那么显然订单业务在0.1S完成后，需要等待后面的支付、发货逻辑处理完才能继续工作，意味着订单业务干了0.1S的活，等了0.2S，导致在0.3秒内订单业务只处理了1条消息。而现在这3个系统是分开的，那么在0.3S内，订单业务系统可以处理3条消息，而且没有业务系统闲着！

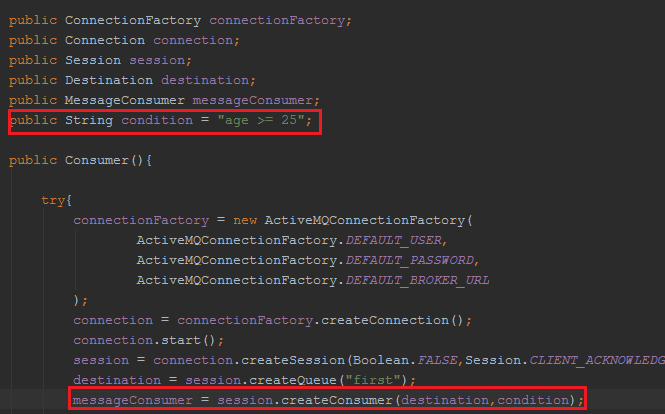
实际上，RocketMQ在消费顺序性这块要比ActiveMQ要强大些，后期在RocketMQ专题中再为大家介绍。

# JMS Selectors

JMS Selectors，即消息选择器。在[《ActiveMQ从入门到精通（一）》](http://www.jianshu.com/p/ecdc6eab554c" \t "http://www.jianshu.com/p/_blank)中，介绍过消息的组成部分，其中谈到消息对象有消息属性，用于消息选择器。我们来看一个代码片段，你就会明白：



生产者片段



消费者片段

需要注意一下几点：

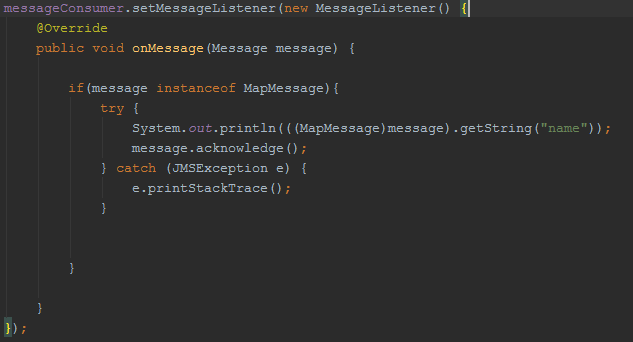
第一，生产者端需要设置消息属性，一定要注意的是setXxxProperty(filed,value)

第二，给出条件，其实本质上就是SQL92语法

第三，创建消费者的时候，指定条件即可

# 消息的同步 AND 异步 接受

消息的接受，我们已经知道，可以通过消费者的receive()/receive(long time)/receiveNoWait()，这种方式是client端主动接受消息，可以理解为消息的同步接受。要知道这种同步的消息接受方式，是让我们很难受的，我们不得不写一个死循环来不断接受消息。那么有没有一种比较优雅的方式，比如我们设置一个类似消息监听的机制，一旦队列上有消息了，那么回调我们的message handler进行处理呢？



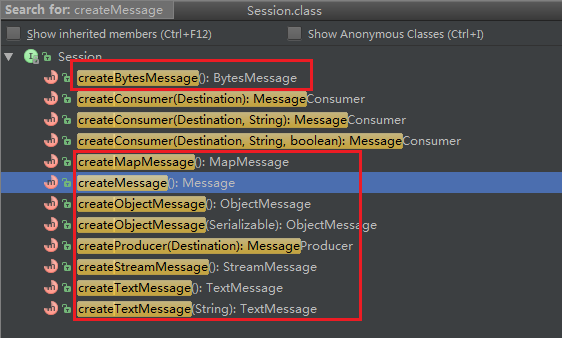
Message Listener

消息的异步接受是指当消息到达时，ActiveMQ主动通知客户端。可以通过注册一个实现了MessageListener接口的对象到MessageConsumer。MessageListener只有一个必须要实现的方法，即onMessage。在发往Destination的消息时，会调用该方法。

这种异步接受“貌似”是ActiveMQ主动的推送消息给消费者，其本质还是消费者轮询消息服务器导致的，只不过这个过程被封装了！

# Message

JMS程序的核心在于，生产和消费的消息能够被其他程序所使用到。JMS Message是一个既简单又不乏灵活的基本格式，由消息头、属性、消息体3部分组成。



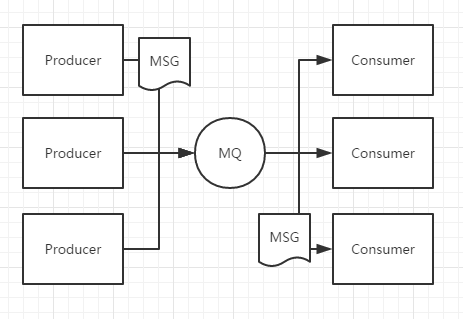
Message

注意，在消费者端，我们接受到消息后，一般需要通过instanceof来判断类型后在进行处理！

在ActiveMQ中，还存在一类临时消息，就是通过创建临时队列/临时主题，如果Connection一旦关闭，那么临时目标就关闭，消息内容也就消失。了解下即可，实际中并不适用。

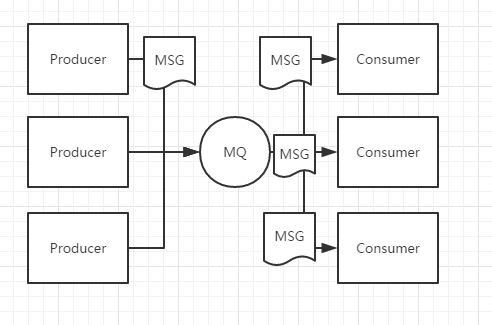
# P2P or Pub/Sub

上2张图，你就会明白这2种模式的区别了。



P2P

生产者端发送一条消息，消费者端只会有一个消费者消费这个消息。好像打电话，一对一通信！



Pub/Sub

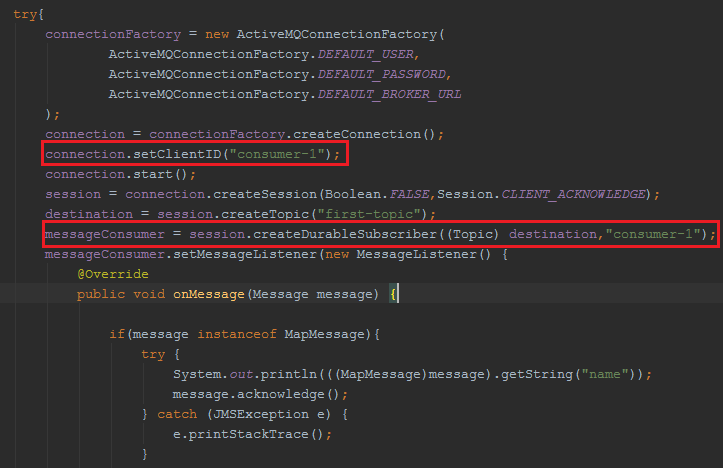
一对多通信，发送一条消息，所有订阅了该目标的消费者都会收到消息。

P2P、Pub/Sub在代码上的区别点仅仅在于，目标类型的创建是createQueue or createTopic，其他一切照旧！

对于订阅模式，对订阅者提出了特殊的要求，要想收到消息，必须先订阅，而且订阅进程必须一直处于运行状态！实际上，有时候消费者重启了下，那么这个消费者将丢失掉一些消息，那么能否避免这样的情况呢？ActiveMQ已经替我们想好了，就是持久化订阅！

# 持久化订阅

所谓持久化订阅，打个比方，就是说跟MQ打声招呼，即便我不在，那么给我发送的消息暂存在MQ，等我来了，再给我发过来。说白了，持久化订阅，需要给MQ备个案（你是谁，想在哪个Topic上搞特殊化）！看一个代码片段：

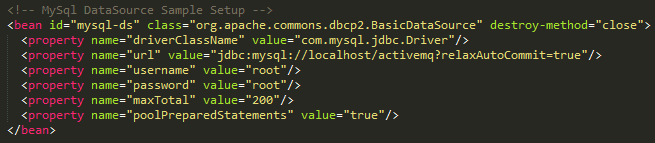


持久化订阅机制

每一个持久化订阅者都应该有一个唯一的ID作为标示以及要在哪个Topic上进行持久化订阅，一旦这些信息告知MQ之后，那么以后不论持久化订阅者在不在线，那么他的消息会暂存在MQ，以后都会发给他！

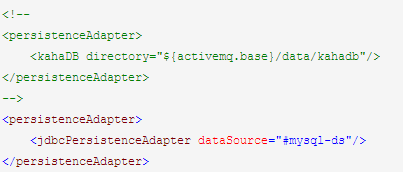
# 持久化消息到MySQL

在前文中已经提及默认情况下，ActiveMQ是开启持久化消息机制的，并且是持久化到kahadb的，但是"很可惜"kahadb对我们不是很友好的可视化，其实ActiveMQ提供了配置的方式让我们来选择持久化消息到哪里，这里我以到MySQL为例来说明。(实际上ActiveMQ已经在conf配置文件中提供了相应的例子，我这里就简单说明下)



在activemq.xml的<broker>节点中增加MySQL信息

注意到这个bean的id，这个是要被引用的。

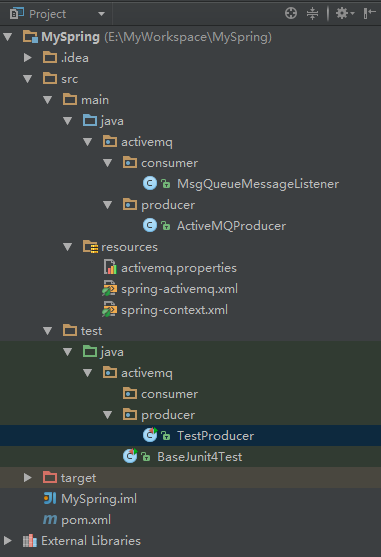


注释kahadb，启用持久化到MySQL配置

实际中，我们会持久化到哪里呢？一般情况下，比如到kahadb，比如到leveldb，因为这些数据库的性能要较MySQL更高些，我们并不关心消息的“可视化”，更加关心的是消息在持久化的同时更加高效！

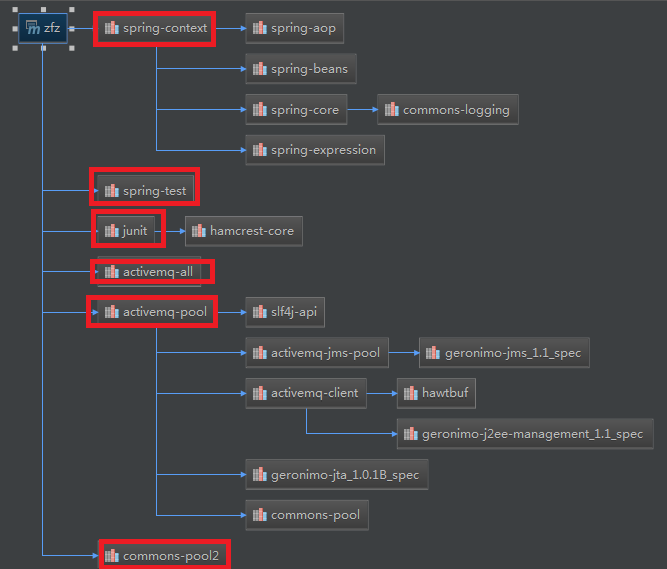
# 与Spring整合

这里我将为大家演示Spring和ActiveMQ整合的核心要素。采用Spring，不要Web容器，不涉及Spring-MVC，而且在这里我将采用JUnit + Spring-Test来进行测试！在文章末尾我将提供源码下载。OK，先来看一眼工程截图：



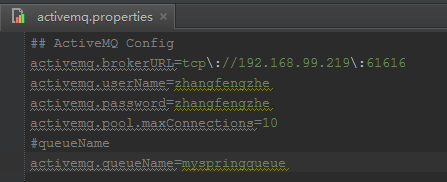
工程结构

**第一步：POM.XML配置**

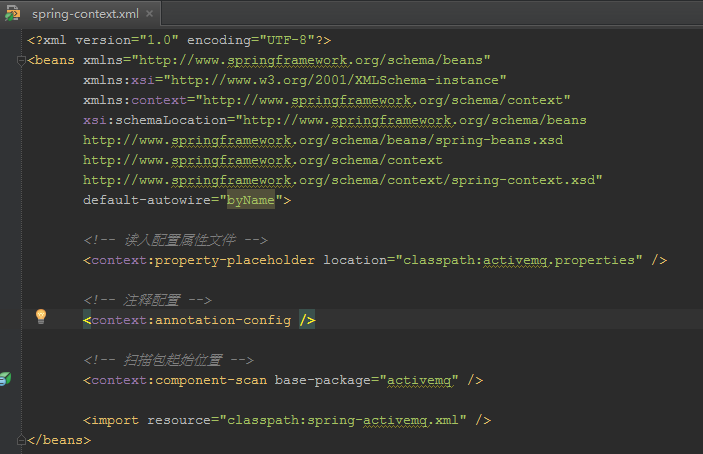


maven dependency tree

**第二步：MQ信息配置文件、Spring配置文件**

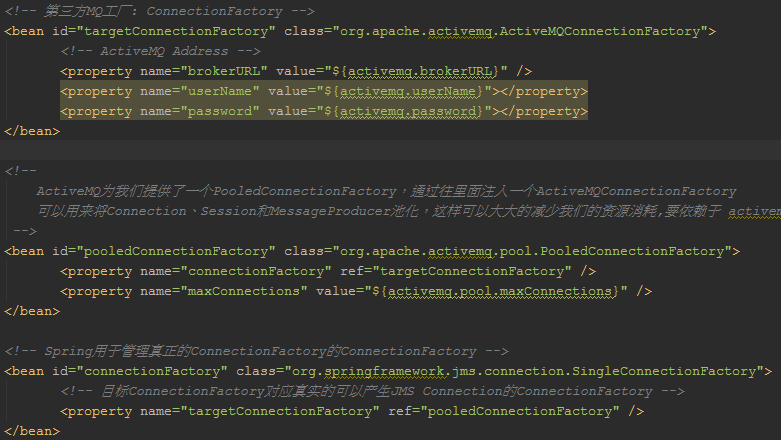


activemq.properties



spring-context.xml

下面我们重点关注spring-activemq.xml：



ConnectionFactory

注意从ActiveMQConectionFactory到PooledConnectionFactory，到Spring提供的SingleConnectionFactory，就是一个适配的过程。



生产者、消费者配置

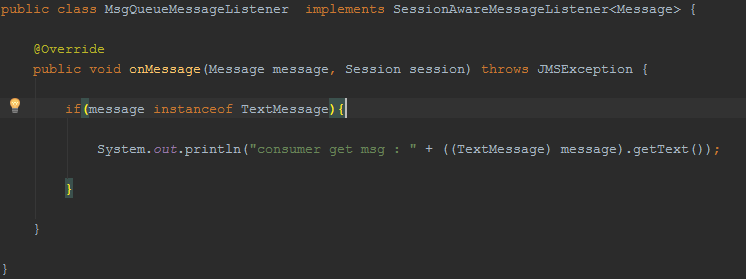
注意Spring的套路经常是这样的，提供XxxTemplate，比如HibernateTemplate，对于JMS，提供了JmsTemplate。

生产者应该持有JmsTemplate进行发送消息。

消费者，提供监听器、监听的目的地、连接工厂即可。

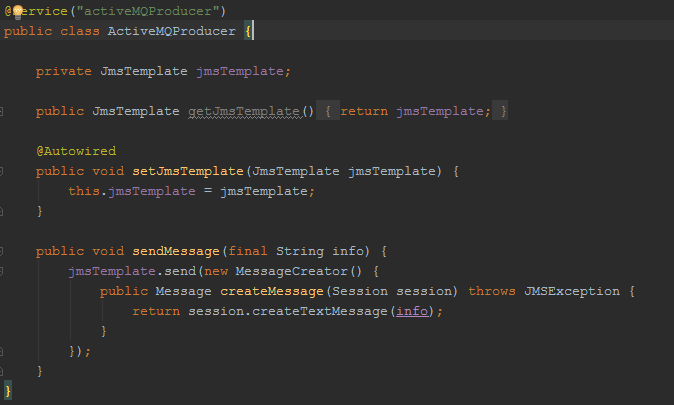
上面的配置，只是一个非常简单的示例，比如是发送到队列，还是发送到主题，事务的配置，签收机制的配置，ttl/priority等配置在后文通过看一下源码，你就会知道该如何配置了。

**第三步：消费者实现监听器**



spring提供的接口

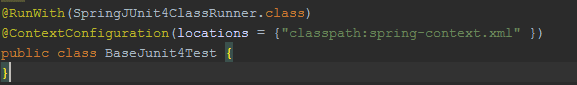
**第四步：生产者**



通过注入拿到JmsTemplate

**第五步：利用Junit4 + SpringTest方式进行测试**

我们以前在测试Spring这一块，大都是通过手动编码的方式（加载XML，setter/getter bean）进行，这里我将为大家介绍一种全新的方式测试Spring程序！

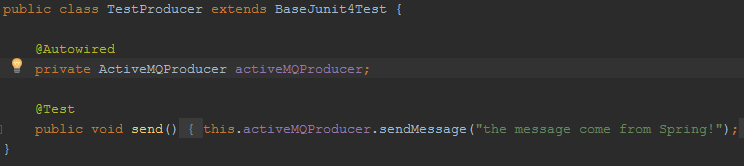


测试基类

为什么要提供一个测试基类呢？因为我们可能有很多个测试类，如果有了这个基类，其他测试类继承它，就可以自动获得测试基类的属性了。

@RunWith  指明采用SpringJunit4进行测试

@ContextConfiguration 告诉配置文件在哪里

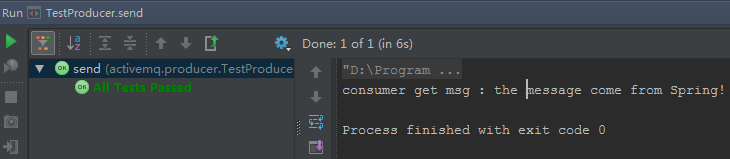


生产者测试类

发现没有，这样写Junit单元测试，和以前感觉不一样！

其实，SpringTest + Junit4还提供了很多功能强大的地方，比如可以设置数据库事务。如果我们在测试的过程结束后，希望回滚数据库的话，很简单，只需要在相应方法上打上注解即可。

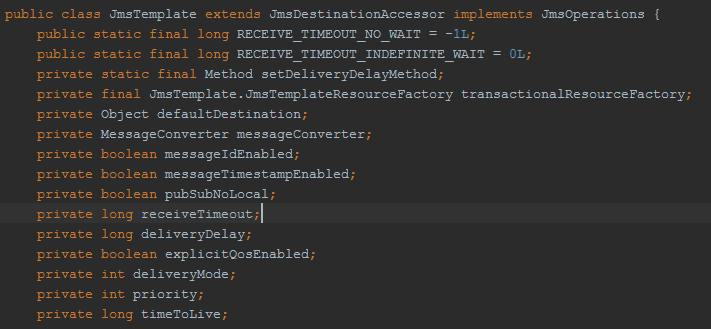
**运行结果**



Test Result

# JmsTemplate

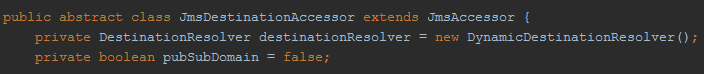
看一下属性：



JmsTemplate

很多属性，是不是很熟悉呢？

JmsTemplate的父类中有一个重要属性：



pubSubDomain

默认情况下，是P2P模式，如果将这个属性配置成true，那么将是主题模式。

**OK，到这里这篇博客的内容就介绍完毕了，下一篇是关于ActiveMQ集群方面的知识，See you again.**

本篇博客工程代码下载地址：

<http://pan.baidu.com/s/1i4HEZsx> 密码：h2hh

# ActiveMQ从入门到精通（三）

[](http://www.jianshu.com/u/cb569cce501b)

作者 [张丰哲](http://www.jianshu.com/u/cb569cce501b)

2017.03.25 20:25 字数 951 阅读 204评论 2喜欢 10

这是ActiveMQ系列的最后一篇文章，主要是关于ActiveMQ集群，这里采用的方式是：Zookeeper+LevelDB+ActiveMQ。前面2篇博客地址如下：[《ActiveMQ从入门到精通（一）》](http://www.jianshu.com/p/ecdc6eab554c" \t "http://www.jianshu.com/p/_blank)、[《ActiveMQ从入门到精通（二）》](http://www.jianshu.com/p/f7a7105b3c27" \t "http://www.jianshu.com/p/_blank)。

# 利用Zookeeper实现ActiveMQ的高可用

话不多说，先来看一张ActiveMQ官方提供的架构图：



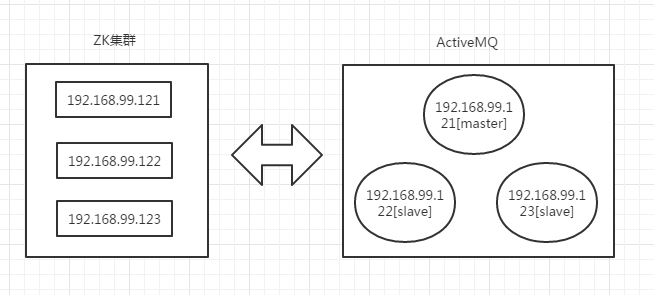
Zookeeper + ActiveMQ

Master/Slave broker的信息要注册到ZK

注意到只有Master对外提供了服务，Slave是待机状态。当Master出现故障，ZK内部的选举机制，会让一个Slave升级成Master对外提供服务。

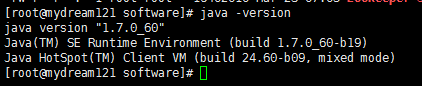
关于ZK，以后博主会有专题进行介绍，这里不做详细讲解，只需要了解通过ZK的协调、选举、容错机制等会实现ActiveMQ的高可用。

既然要做到高可用，那么ZK也得是高可用的，所以这里的搭建方案是这样的：



搭建方案

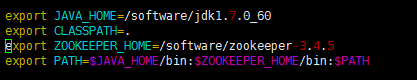
**第一步：JDK环境**



jdk环境

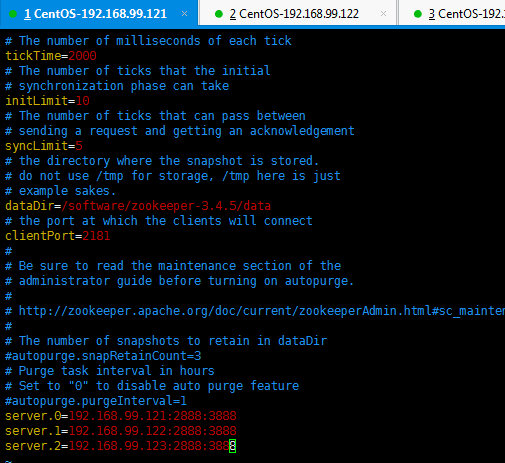
保证这3台机器都安装了JDK，并配置了JAVA环境变量。

**第二步：配置Zookeeper**



zk环境变量

为什么要配置ZK环境变量呢？很简单，我想在命令行下直接使用ZK相关的命令，而不是进入到安装ZK目录下的bin，更不想用绝对路径。



zoo.cfg

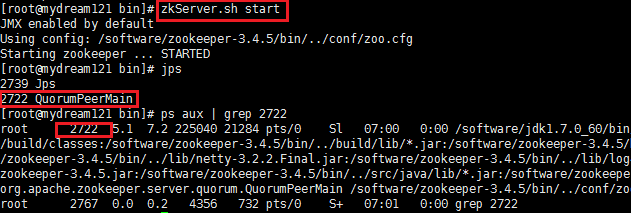
注意dataDir目录的指定；注意2181是外部访问ZK的端口；

2888:3888是ZK集群内部通信（比如ZK原子广播消息）的端口，注意server.X的定义，这是将ZK集群中的实例进行编号，实际上需要在dataDir目录中新建myid文件，并与之保持一致。

IMG_262

myid文件

**第三步：启动ZK**



zkServer start

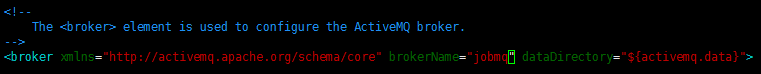
可以通过netstat命令查看2722进程，发现ZK的端口是2181，这和zoo.cfg的配置是一致的。让3台机器的ZK都启动起来。



zk status

**第四步：ActiveMQ主从配置**

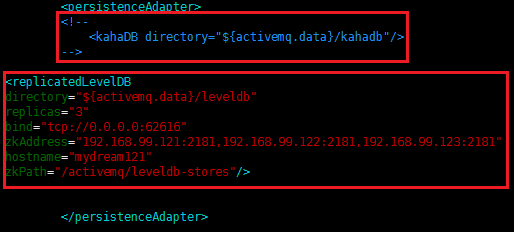
注意了，由于我将在3台物理机上搭建一台Master，2台Slave，因此我这边不需要对端口配置文件进行改动。比如WEB管控台的jetty.xml。



activemq.xml:brokerName

3台机器应该对外只有一个统一的名称，就是这个brokerName。3台机器都修改成一个名称即可。

这里持久化，我将采用LevelDB，因此需要修改持久化配置：



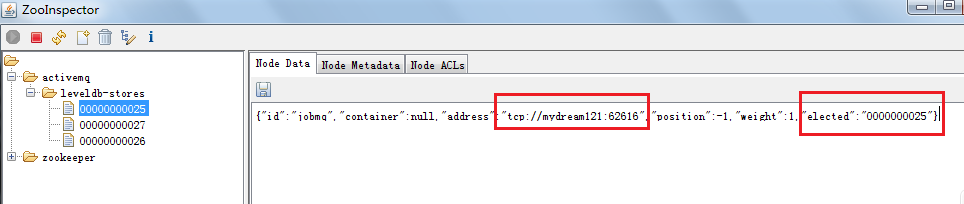
leveldb

注意bind地址，其实是ActiveMQ集群内部通信的TCP端口，和ActiveMQ对外提供的消息端口（默认61616）不要搞混了。

hostname即本机的主机名称。

给出ZK集群的列表以及zkPath。zkPath下面其实存放着ActiveMQ的节点，在后续你会看到。

启动3台机器上的ActiveMQ，然后利用ZooInspector你可以看到：



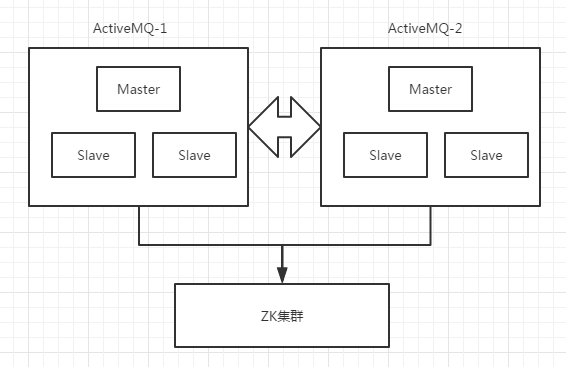
121节点是Master

此时此刻，基于ZK的ActiveMQ的高可用方案就做好了。那么JAVA端访问ActiveMQ有什么变化么？其实就是在创建ConnectionFactory的时候给定的URL有变化：

**failover:(tcp://192.168.99.121:61616,tcp://192.168.99.122:61616,tcp://192.168.99.123:61616)?Randomize=false**

就是一个失败转移协议！

上面只是做了一个ActiveMQ的高可用方案，那么ActiveMQ集群呢？其实所谓的ActiveMQ集群就是多个ActiveMQ高可用之间产生关联：



ActiveMQ集群

高可用的ActiveMQ-1,ActiveMQ-2,...,ActiveMQ-N就可以组建ActiveMQ集群

在配置上很简单，其实就是ActiveMQ-1要知道ActiveMQ-2的信息而已，这里大家可以自行百度，不在演示了。

整个ActiveMQ从入门到精通系列就结束了，希望对大家有帮助！