ActiveMQ教程

1. **概念**

ActiveMQ是Apache推出的一款开源的，完全支持JMS1.1和J2EE1.4规范的JMS provider实现的消息中间件（Message Oriented Middleware，MOM）

**功能**：

最主要的功能就是：实现JMS Provider，用来帮助实现高可用、高性能、可伸缩、易用和安全的企业级面向消息服务系统。

比如系统A和系统B，之前的系统流程比如A系统调用B系统的服务，A系统必须等待B系统将该请求处理完成后，才能继续进行下一步的操作。如果使用消息中间件，系统A可以将操作组装成一个消息，放入消息中间件中，然后系统A就继续干其他事，系统B去消息中间件拿消息，并且进行处理，两个系统之间就是完全解耦的状态，并且是异步的。

**特点**：

完全支持JMS1.1和JSEE1.4规范（持久化、XA消息、事务）

支持多种传送协议：in-VM、TCP、SSL、NIO、UDP、JGroups、JXTA

可插拔的体系结构，可以灵活定制，如：消息存储方式、安全管理等

很容易和Application Server集成使用

多种语言和协议编写客户端。语言：Java、C、C++、C#、Ruby、Perl、Python、PHP

从设计上保证了高性能的集群，客户端-服务器，点对点

可以很容易的和spring结合使用

支持通过JDBC和journal提供高速的消息持久化

支持与Axis的整合

1. **下载安装**

服务器端下载地址：

<http://activemq.apache.org/download.html>

下载并解压到需要安装的位置。

此时也就算默认安装好了

启动：

普通启动：终端命令进入到ActiveMQ/bin下面，然后输入./activemq start命令进入启动



启动并指定日志文件：./activemq start > /tmp/activemqlog

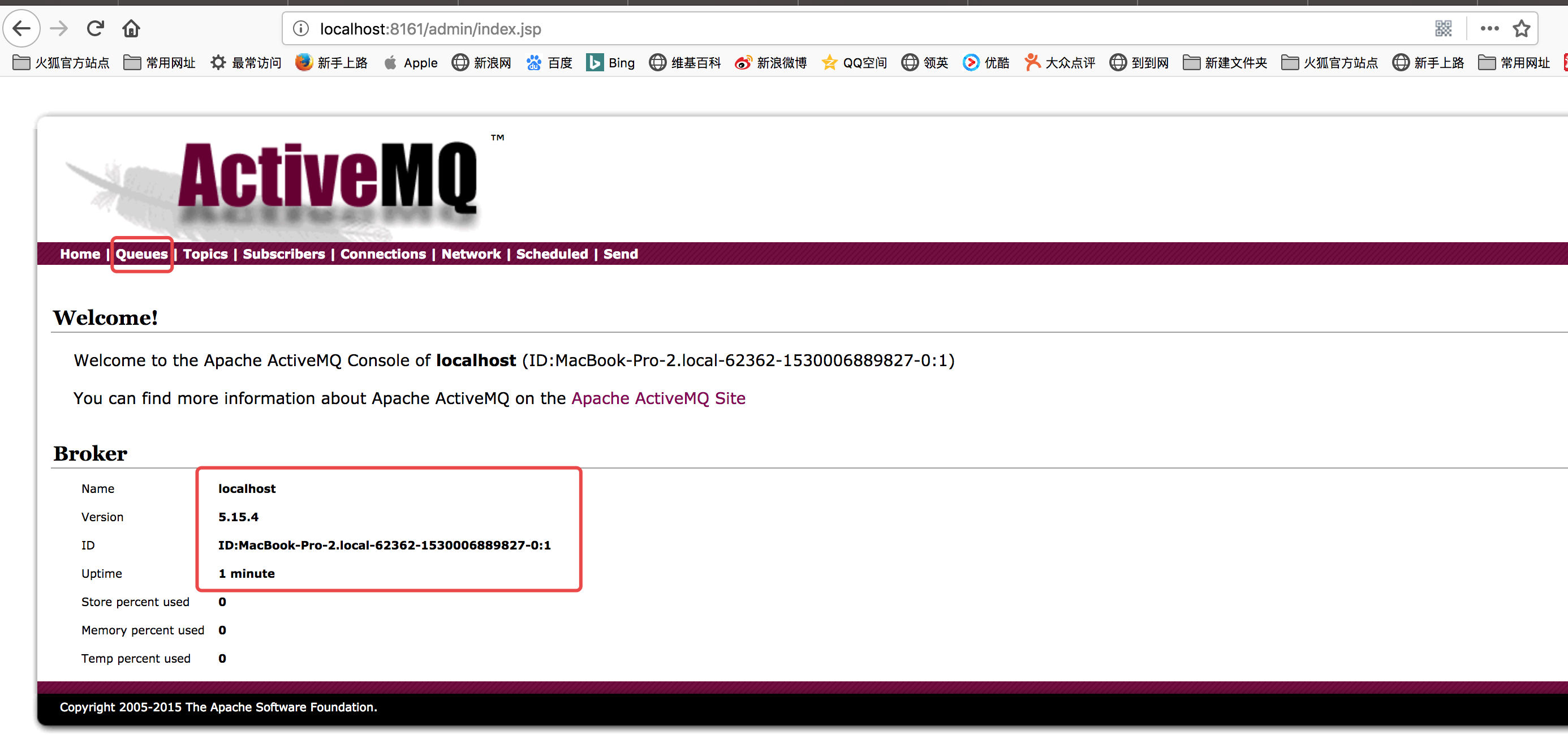
检查是否已经启动：

ActiveMQ默认采用61616端口提供JMS服务，使用8161端口提供管理控制台服务，执行以下命令以便检验是否已经成功启动ActiveMQ服务：

（1）查看61616端口是否打开：netstat –an | grep 61616

（2）也可以查看控制台输出或者日志文件

（3）直接访问ActiveMQ的管理页面：<http://loaclhost:8161/admin>,默认的账户和密码是admin/admin

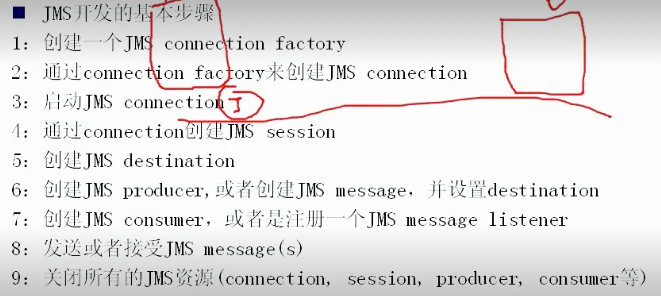


关闭ActiveMQ：./activemq stop

或者直接根据进程号去kill掉。

1. **实例demo**

开发流程：



消息生产者：

import javax.jms.Connection;

import javax.jms.ConnectionFactory;

import javax.jms.Destination;

import javax.jms.JMSException;

import javax.jms.MessageProducer;

import javax.jms.Session;

import javax.jms.TextMessage;

import org.apache.activemq.ActiveMQConnectionFactory;

public class QueueSend {

public static void main(String[] args) throws JMSException, InterruptedException {

//连接mq服务器

ConnectionFactory connFactory = new ActiveMQConnectionFactory("tcp://127.0.0.1:61616");

//建立连接

Connection connection = connFactory.createConnection();

//启动连接

connection.start();

//创建session，AUTO\_ACKNOWLEDGE是应答模式

Session session = connection.createSession(Boolean.TRUE, Session.AUTO\_ACKNOWLEDGE);

//创建队列，my-queue是队列名称

Destination destination = session.createQueue("my-queue");

//创建消息生产者

MessageProducer producer = session.createProducer(destination);

for (int i = 0; i < 3; i++) {

//创建消息

TextMessage message = session.createTextMessage("message--"+i);

Thread.sleep(1000);

//发送消息，此时消息不会出现在队列中，在控制台管理页面是看不见的

producer.send(message);

}

//提交消息，提交后，消息就入了队列，此时在管理页面可以见到消息

session.commit();

session.close();

connection.close();

}

}

消息消费者：

import javax.jms.Connection;

import javax.jms.ConnectionFactory;

import javax.jms.Destination;

import javax.jms.JMSException;

import javax.jms.MessageConsumer;

import javax.jms.Session;

import javax.jms.TextMessage;

import org.apache.activemq.ActiveMQConnectionFactory;

public class QueueReceiver {

public static void main(String[] args) throws JMSException, InterruptedException {

//连接mq服务器

ConnectionFactory connFactory = new ActiveMQConnectionFactory("tcp://127.0.0.1:61616");

//建立连接

Connection connection = connFactory.createConnection();

//启动连接

connection.start();

//创建session，AUTO\_ACKNOWLEDGE是应答模式

Session session = connection.createSession(Boolean.TRUE, Session.AUTO\_ACKNOWLEDGE);

//创建队列，my-queue是队列名称

Destination destination = session.createQueue("my-queue");

//创建消息消费者

MessageConsumer consumer = session.createConsumer(destination);

for (int i = 0; i < 3; i++) {

//创建消息

TextMessage message = (TextMessage) consumer.receive();

System.out.println(message.getText());

//如果不提交，则消息可以无限消费，也就是可以receive无数次，提交了，则消息消费后就会从队列消失

session.commit();

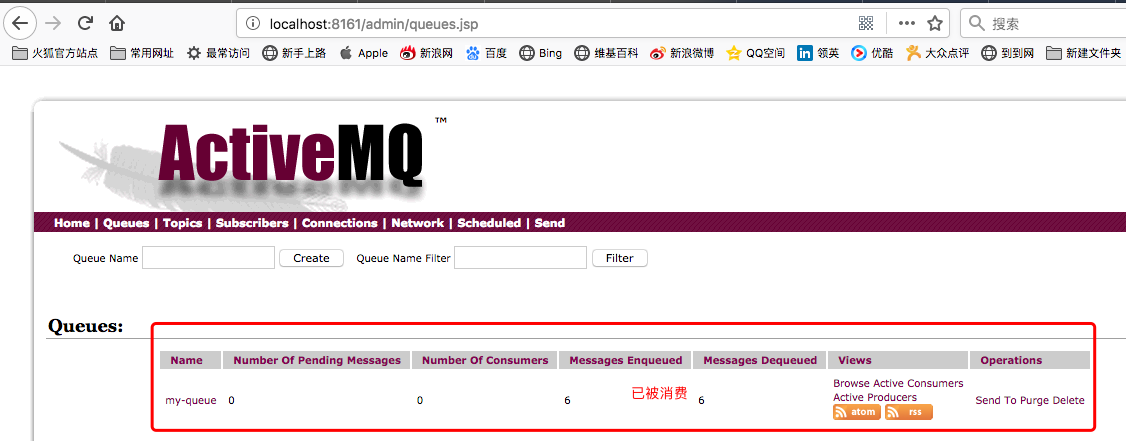
}

session.close();

connection.close();

}

}



1. **JMS**

概念：

JMS：Java message service，Java消息服务，是Java EE的一个技术。

JMS规范：JMS定义了Java中访问消息中间件的接口，并没有给予实现，实现JMS接口的消息中间件称为JMS Provider，例如ActiveMQ

JMS Provider：实现JMS接口和规范的消息中间件

JMS message：JMS的消息，JMS消息由以下三部分组成

（1）消息头：每个消息头字段都有相应的getter和setter方法

（2）消息属性：如果需要除消息头字段以外的值，那么可以使用消息属性

（3）消息体：封装具体的消息数据

JMS producer:消息生产者，创建和发送JMS消息的客户端应用

JMS consumer：消息消费者，接收和处理JMS消息的客户端应用

消息的消费可以采用以下两种方法之一：

（1）同步消费：通过调用消费者的receive方法从目的地中显式的提取消息，receive方法可以一直阻塞到消息到达。也就是消息消费者在没有消息的时候就一直阻塞，一直等，直到消息到达，则马上进行处理。

（2）异步消费：客户可以为消费者注册一个消息监听器，以定义在消息到达时所采取的动作。也就是说消息消费者在MQ上注册了一个消息监听器，当MQ收到消息时，通过消息监听器，回调触发消费者的具体方法。

JMS domains:消息传递域，JMS规范中定义了两种消息传递域：点对点简写为PTP，消息传递域和发布/订阅消息传递域（publish/subscribe，简写成pub/sub）。

**可靠性机制：**

消息接收确认

JMS消息只有在被确认之后，才认为已经被成功地消费了，消息的成功消费通常包含三个阶段：客户接收消息、客户处理消息、消息被确认（被确认的消息说明已经消费过了，会被移除消息队列）。

在事务会话中，当一个事务被提交的时候，确认自动发生。在非事务性会话中，消息何时被确认取决于创建会话时的应答模式（acknowledge mode）。该参数有以下三个可选值。

应答模式也就是如下红圈的地方：



Session.AUTO\_ACKNOWLEDGE：当客户成功的从receive方法返回的时候，或者从MessageListener.onMessage方法成功返回的时候，会话自动确认客户收到的消息。也就是说，消息确认后，会从队列中移除，不能再次消费。

Session.AUTO\_ACKNOWLEDGE：客户需要通过调用消息的acknowledge方法确认消息（如果不调用，消息可以无限制的消费），需要注意的是，在这种模式中，确认是在会话层上进行，确认一个被消费的消息将自动确认所有已被会话消费的消息。例如：如果一个消息消费者消费了10个消息，然后确认第5个消息，那么所有10个消息都被确认。

Session.DUPS\_OK\_ACKNOWLEDGE:该选择只是会话迟钝的确认消息的提交，如果JMS provider失败，那么可能会导致一些重复的消息（注：因为在这里是延迟提交确认，比如一条消息到队列，消息消费者已经消费了该条消息，但是还没提交确认，此时消息消费者挂了，下次消息消费者再启动的时候，就认为刚才那条消息没有被消费，因此还会消费一次，导致该消息重复处理。）。如果是重复的消息，那么JMS provider必须把消息头的JMSRedelivered字段设置为true。

**消息持久性：**

JMS支持以下两种消息提交模式

（1）PERSISTENT：指示JMS provider持久保存消息，以保证消息不会因为JMS provider的失败而丢失。

（2）NON\_PERSISTENT：不要求JMS provider持久保存消息

**消息优先级：**

可以使用消息优先级来指示JMS provider首先提交紧急的消息，优先级分10个级别，从0（最低）到9（最高）。如果不指定优先级，默认级别是1，需要注意的是，JMS provider并不一定保证按照优先级的顺序提交消息。

**消息过期：**

可以设置消息在一定时间后过期，默认是永不过期。

**消息的临时目的地：**

可以通过会话的createTemporaryQueue方法和createTemporaryTopic方法来创建临时目的地，他们的存在时间只限于创建他们的连接所保持的时间。只有创建该临时目的地的连接上的消息消费者才能够从临时目的地中提取消息。

**JMS PTP模型：**

JMS PTP（point-to-point）模型定义了客户端如何向队列发送消息，从队列接收消息，以及浏览队列中的消息。

PTP模型是基于队列的，生产者发消息到队列，消费者从队列接收消息，队列的存在使得消息的异步传输成为可能。和邮件系统中的邮箱一样，队列可以包含各种消息，JMS provider提供工具管理队列的创建、删除。

PTP的特点：

（1）如果在session关闭时，有一些消息已经被收到，但还没有被签收acknowledge，那么当消费者下次连接到相同的队列时，这些消息还会被再次接收。

（2）如果用户在receive方法中设置了消息选择条件，那么不符合条件的消息会留在队列中，不会被接收到。

（3）队列可以长久地保存消息直到消费者收到消息。消费者不需要因为担心消息会丢失而时刻和队列保持激活的连接状态，充分体现了异步传输模式的优势。

**JMS的pub/sub模型：**

JMS Pub/Sub模型定义了如何向一个内容节点发布和订阅消息，这些节点被称作topic。主题可以被认为是消息的传输中介，发布者（publisher）发布消息到主题，订阅者（subscribe）从主题订阅消息。主题使得消息订阅者和消息发布者保持相互独立，不需要接触即可保证消息的传送。

Pub/Sub的特点：

（1）消息订阅分为非持久订阅和持久订阅

非持久订阅只有当客户端处于激活状态，也就是和JMS provider保持连接状态才能收到发生到某个主题的消息，而当客户端处于离线状态，这个时间段发到主题的消息将会丢失，永远不会收到。

持久订阅时，客户端向jms注册一个识别自己身份的ID，当这个客户端处于离线时，JMS provider会为这个ID保存所有发送到主题的消息，当客户再次连接到JMS provider时，会根据自己的ID得到所有当自己处于离线时发送到主题的消息。

（2）如果用户在receive方法中设置了消息选择条件，那么不符合条件的消息不会被接收

（3）非持久订阅状态下，不能恢复或重新派送一个未签收的消息，只有持久订阅才能恢复或重新派送一个未签收的消息。

（4）当所有的消息必须被接收，则用持久订阅，当丢失消息能够被容忍，则用非持久订阅。

1. **Topic消息示例**

非持久的Topic消息的发送：基本上跟前面发送队列信息是一样的，只是把创建Destination的地方，由创建队列替换成创建Topic，例如：

Destination topic = session.createTopic("my-topic");

对于非持久的Topic消息的接收：

（1）必须要接收方在线，然后客户端再发送消息，接收方才能接收到消息

（2）同样把创建Destination的地方，由创建队列替换成创建Topic，例如：Destination topic = session.createTopic("my-topic");

（3）由于不知道客户端发送多少消息，因此改成while循环的方式接收，例如：

//接收消息

Message message = consumer.receive();

while (message!=null) {

TextMessage textMessage = (TextMessage)message;

System.out.println("接收到的Topic消息："+textMessage.getText());

message = consumer.receive(1000L);

}

持久的Topic消息：

发送端代码变化如下：

//创建消息生产者

MessageProducer producer = session.createProducer(destination);

producer.setDeliveryMode(DeliveryMode.PERSISTENT);

//启动连接

connection.start();

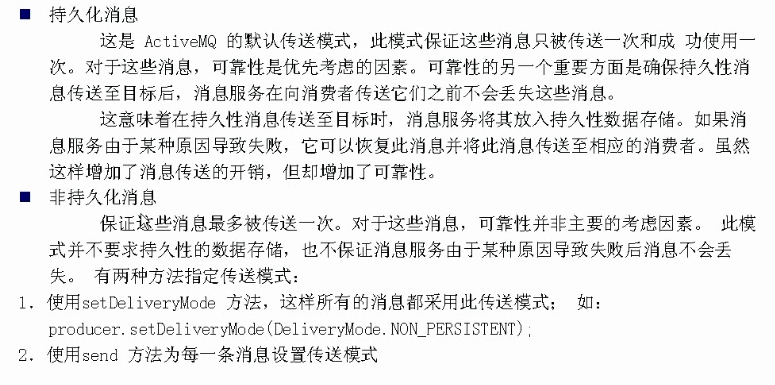
默认的Topic模式是非持久化的。启动连接记得一定要在设置持久化模式之后。

接收端代码变化如下：



这个消息可以给多个消费者使用，也就是一个消费者消费了消息，另外一个消费者还可以消费该消息。但是每个消费者对于同一条消息只能消费一次。

总结：



1. **用ActiveMQ构建应用**

Broker:相当于一个ActiveMQ服务器实例，如下启动activemq的命令也就是启动Broker

命令行启动参数示例如下：

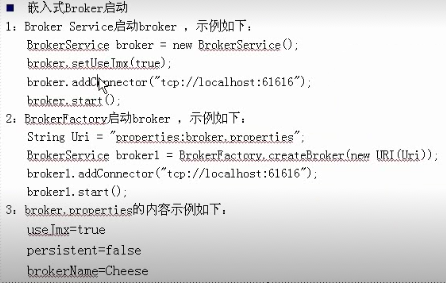
activemq start:使用默认的activemq.xml来启动，此配置文件在conf目录下。

Activemq start xbean:file:../conf/activemq.xml：使用指定的配置文件来启动activemq。

如果我们不指定file，也就是xbean:activemq.xml，那么xml必须在classpath下面。

**用activemq来构建Java应用：**

这里主要将用activemq broker作为独立的消息服务器来构建Java应用，activemq也支持在vm中通信基于嵌入式的broker，能够无缝的集成其他Java应用。此处也就是将broker的启动嵌入到Java程序中去，完全由Java程序去控制，因此在需要的时候我们可以启动broker，不需要的时候就可以把broker进行关闭。



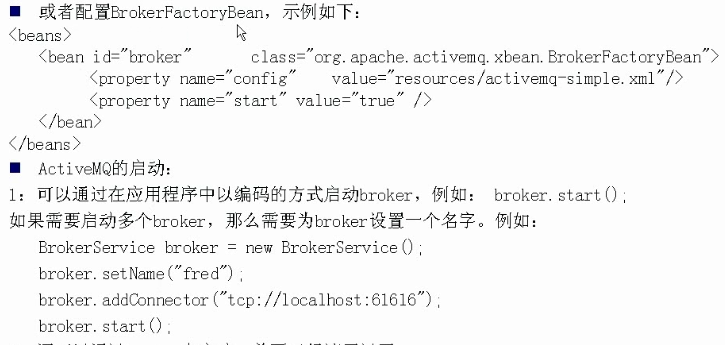
如上启动broker后，我们就可以进行收发消息了，注意收发消息的服务器地址应该改成broker的地址。注：broker就是一个activemq实例，所以可以用它收发消息。

**Spring集成broker：**

在spring的配置文件中添加如下配置：







**Spring结合ActiveMQ开发：**





1. **aa**