点对点消息模型

当需要将消息发布送给唯一的一个消息消费者时，就应该使用点对点模型。虽然可能或有多个消费者在队列中侦听统一消息，但是，只有一个且仅有一个消费者线程会接受到该消息。在p2p模型中，生产者称为发送者，而消费者则称为接受者。点对点模型最重要的特性如下：

* 消息通过称为队列的一个虚拟通道来进行交换。队列是生产者发送消息的目的地和接受者消费消息的消息源。
* 每条消息通仅会传送给一个接受者。可能会有多个接受者在一个队列中侦听，但是每个队列中的消息只能被队列中的一个接受者消费。
* 消息存在先后顺序。一个队列会按照消息服务器将消息放入队列中的顺序，把它们传送给消费者当消息已被消费时，就会从队列头部将它们删除。
* 生产者和消费者之间没有耦合。接受者和发送者可以在运行时动态添加，这使得系统的复杂性可以随着时间而增长或降低（这是消息传送系统的普遍特性）。

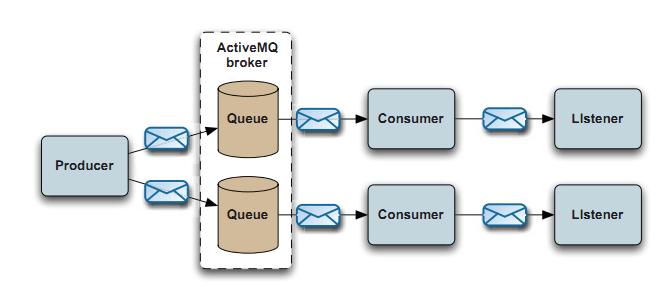
点对点消息传送模型有两种类型：异步即发即弃（fire-and-forget）处理和异步请求/应答处理。使用即发即弃处理时，消息生产者向某个队列发送一条消息，而且它并不会期望接受到一个响应（至少不是立刻接收到响应）。这类处理可用于触发一个事件，或者用于向接受者发出请求来执行一个并不需要响应的特定活动。

JMS(Java Message Service)，Java消息服务，是Java平台中面向消息中间件的API，用于在两个应用程序之间或者分布式系统中发布消息，进行一步通信。ActiveMQ是一个完全支持JMS规范的JMS Provider实现，其中涉及到概念：

* JMS producer，创建和发送JMS消息的客户端
* JMS consumer，接收和处理JMS消息的客户端
* JMS provider，JMS接口实现
* Connection Factory，客户端用Connection Factory创建Connection来连接JMS Provider
* Destination，用于记录消息的目的地址和接收地址

# ActiveMQ Broker安装和启动

ActiveMQ点对点模式模型图：



消息是点对点发送的，一个Queue只会发送到客户端。

1. 安装

下载安装包

*wget https://archive.apache.org/dist/activemq/5.15.2/apache-activemq-5.15.2-bin.tar.gz*

解压：

*tar –zxvf apache-activemq-5.15.2.tar.gz*

其目录结构如下所示：

├── activemq-all-5.15.2.jar

├── bin

│   ├── activemq

│   ├── activemq-diag

│   ├── activemq.jar

......

├── conf

│   ├── activemq.xml

│   ├── broker.ks

│   ├── broker-localhost.cert

│   ├── broker.ts

.....

├── lib

│   ├── activemq-broker-5.15.2.jar

│   ├── activemq-client-5.15.2.jar

│   ├── activemq-console-5.15.2.jar

│   ├── activemq-jaas-5.15.2.jar

│   ├── activemq-kahadb-store-5.15.2.jar

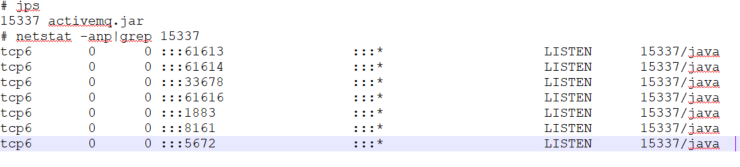
│   ├── activemq-openwire-legacy-5.15.2.jar

.....

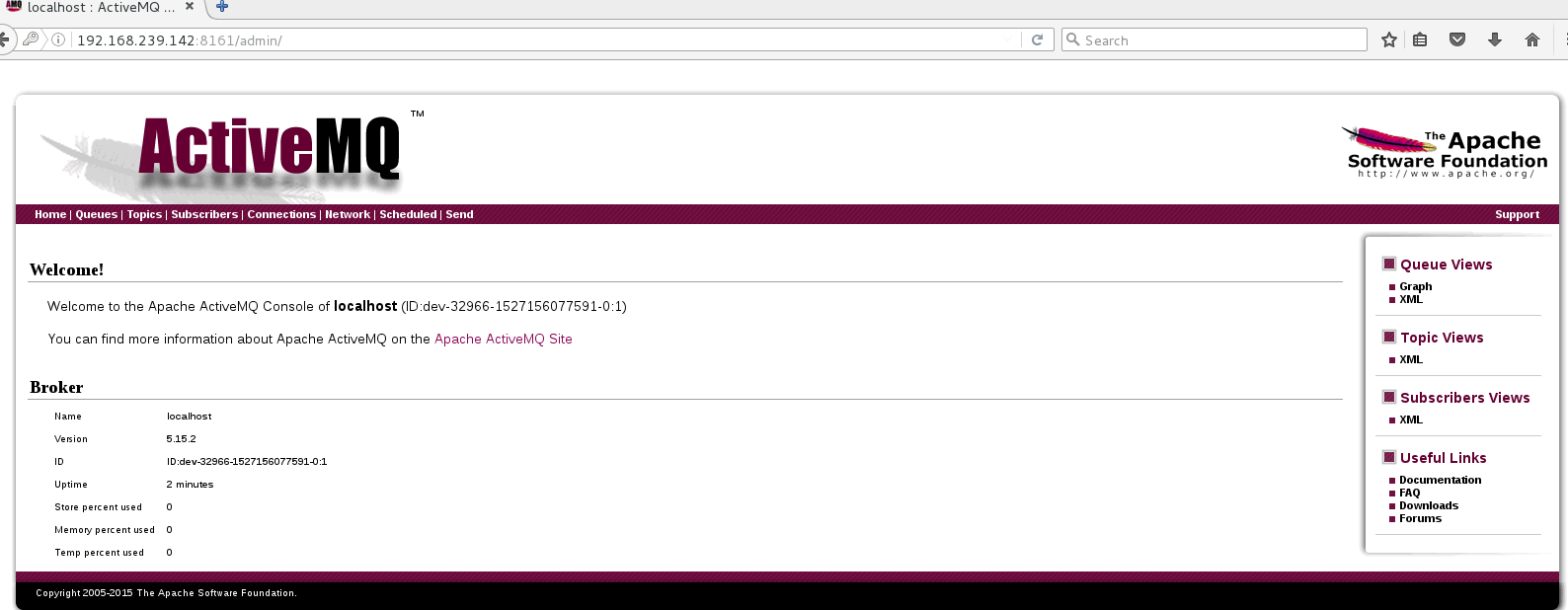
2）启动，解压后，即可进行启动，命令如下

*bin/activemq start*

启动如下：



启动完成后，打开管理页面localhost:8161，如下所示：



第一次打开是需要输入用户名和密码：admin/admin

# ActiveMQ Example

1. 消息生产者：JMSProducer

*public class JMSProducer {*

*private static String USERNAME = ActiveMQConnection.DEFAULT\_USER ;*

*private static String PASSWORD = ActiveMQConnection.DEFAULT\_PASSWORD;*

*private static String URL = ActiveMQConnection.DEFAULT\_BROKER\_URL ;*

*private static final int SENDNUM = 10;*

*public static void main(String[] args) {*

*System.out.println("url:" + URL);*

*ConnectionFactory factory;*

*Connection connection = null;*

*Session session;*

*Destination destination;*

*MessageProducer messageProduer;*

*factory = new ActiveMQConnectionFactory(JMSProducer.USERNAME,*

*JMSProducer.PASSWORD,*

*JMSProducer.URL);*

*try {*

*connection = factory.createConnection();*

*connection.start();*

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

*destination = session.createQueue("FirstQueue");*

*messageProduer = session.createProducer(destination);*

*sendMessage(session, messageProduer);*

*session.commit();*

*} catch (JMSException e) {*

*e.printStackTrace();*

*}*

*}*

*public static void sendMessage(Session session, MessageProducer mp) throws JMSException {*

*for (int i = 0 ; i < JMSProducer.SENDNUM ; i++) {*

*TextMessage message = session.createTextMessage("ActiveMQ Send:" + i);*

*System.out.println("Send Message:" + "ActiveMQ Message:" + i);*

*mp.send(message);*

*}*

*}*

*}*

1. 消息消费者: JMSConsumer

*public class JMSConsumer {*

*private static String USERNAME = ActiveMQConnection.DEFAULT\_USER ;*

*private static String PASSWORD = ActiveMQConnection.DEFAULT\_PASSWORD;*

*private static String URL = ActiveMQConnection.DEFAULT\_BROKER\_URL ;*

*public static void main(String[] args) {*

*ConnectionFactory factory;*

*Connection connection = null;*

*Session session;*

*Destination destination;*

*MessageConsumer messageConsumer;*

*factory = new ActiveMQConnectionFactory(JMSConsumer.USERNAME,*

*JMSConsumer.PASSWORD,*

*JMSConsumer.URL);*

*try {*

*connection = factory.createConnection();*

*connection.start();*

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

*destination = session.createQueue("FirstQueue");*

*messageConsumer = session.createConsumer(destination);*

*messageConsumer.setMessageListener(new Listener());*

*} catch (JMSException e) {*

*e.printStackTrace();*

*}*

*}*

*}*

*public class Listener implements MessageListener {*

*@Override*

*public void onMessage(Message message) {*

*try {*

*System.out.println("Receiver message:" + ((TextMessage)message).getText());*

*} catch ( JMSException e) {*

*e.printStackTrace();*

*}*

*}*

*}*

执行如下：

JobProducer:

*url:failover://tcp://localhost:61616*

*INFO | Successfully connected to tcp://localhost:61616*

*Send Message:ActiveMQ Message:0*

*Send Message:ActiveMQ Message:1*

*Send Message:ActiveMQ Message:2*

*Send Message:ActiveMQ Message:3*

*Send Message:ActiveMQ Message:4*

*Send Message:ActiveMQ Message:5*

*Send Message:ActiveMQ Message:6*

*Send Message:ActiveMQ Message:7*

*Send Message:ActiveMQ Message:8*

*Send Message:ActiveMQ Message:9*

JobConsumer

*INFO | Successfully connected to tcp://localhost:61616*

*Receiver message:ActiveMQ Send:0*

*Receiver message:ActiveMQ Send:1*

*Receiver message:ActiveMQ Send:2*

*Receiver message:ActiveMQ Send:3*

*Receiver message:ActiveMQ Send:4*

*Receiver message:ActiveMQ Send:5*

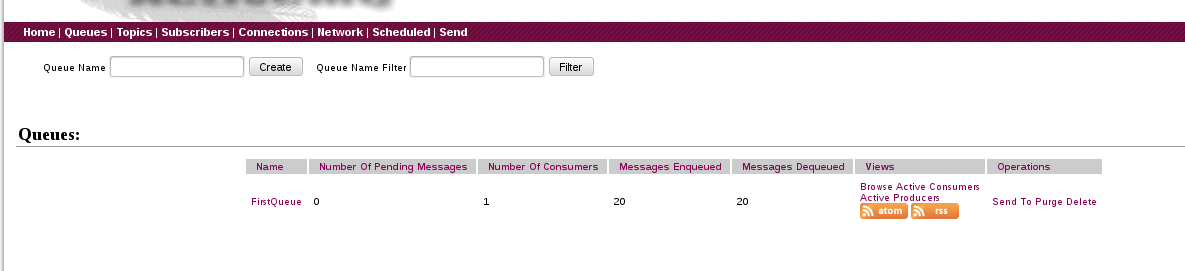
*Receiver message:ActiveMQ Send:6*

*Receiver message:ActiveMQ Send:7*

*Receiver message:ActiveMQ Send:8*

*Receiver message:ActiveMQ Send:9*

在Web上信息展示如下：



http://www.reader8.cn/jiaocheng/20120701/2216889.html

http://www.cnblogs.com/jahentao/p/activemq\_study\_1.html