# 安装ActiveMQ

将ActiveMQ的压缩包上传到/usr/local/software下



解压

去解压后的bin目录下执行：

启动

./activemq start

关闭：

./activemq stop

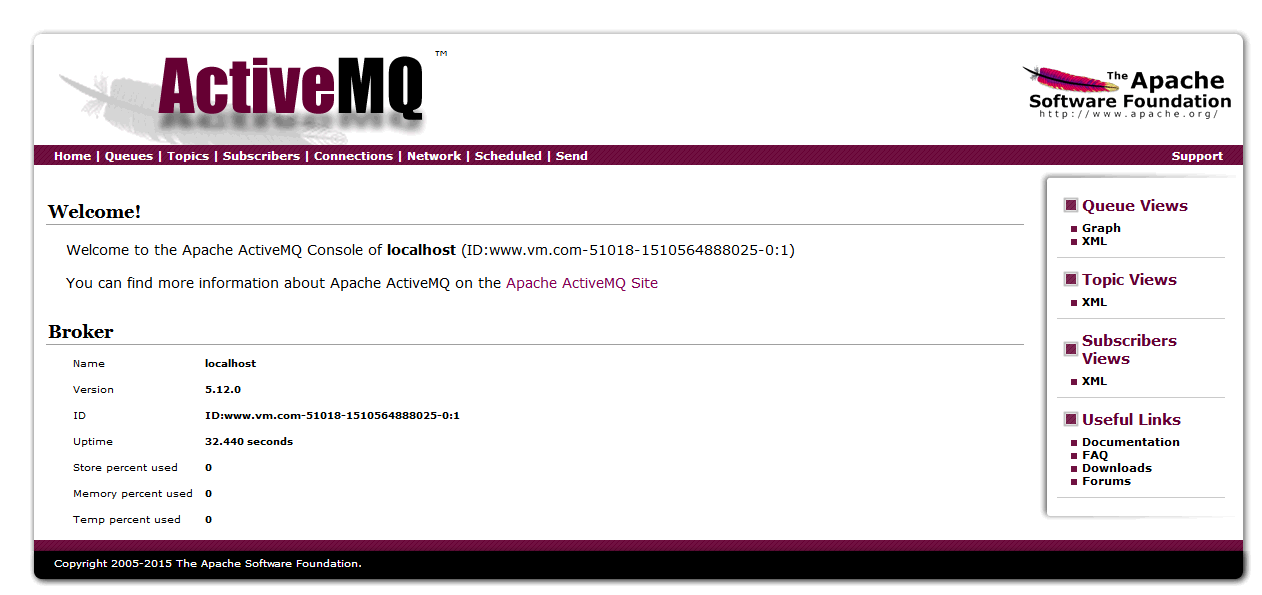
查看状态：

./activemq status

访问服务器的8161端口

http://www.vm.com:8161/admin/

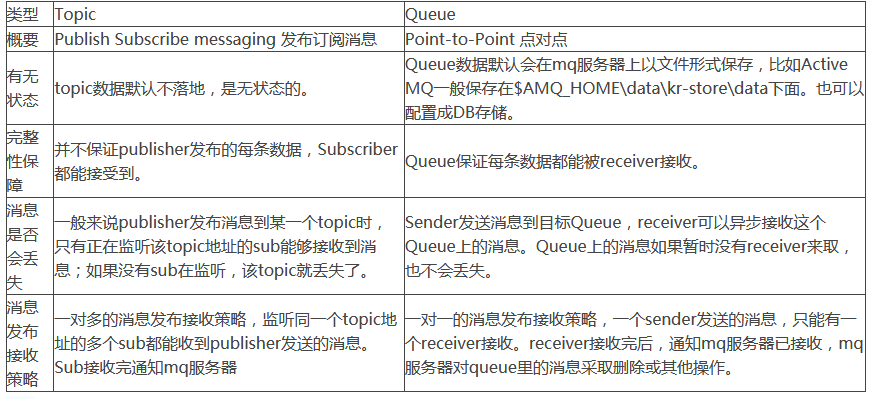
用户名密码都是admin



# ActiveMQ的使用

使用ActiveMQ需要引入jar包

<dependency>  
 <groupId>org.apache.activemq</groupId>  
 <artifactId>activemq-all</artifactId>  
 <version>5.11.2</version>  
</dependency>

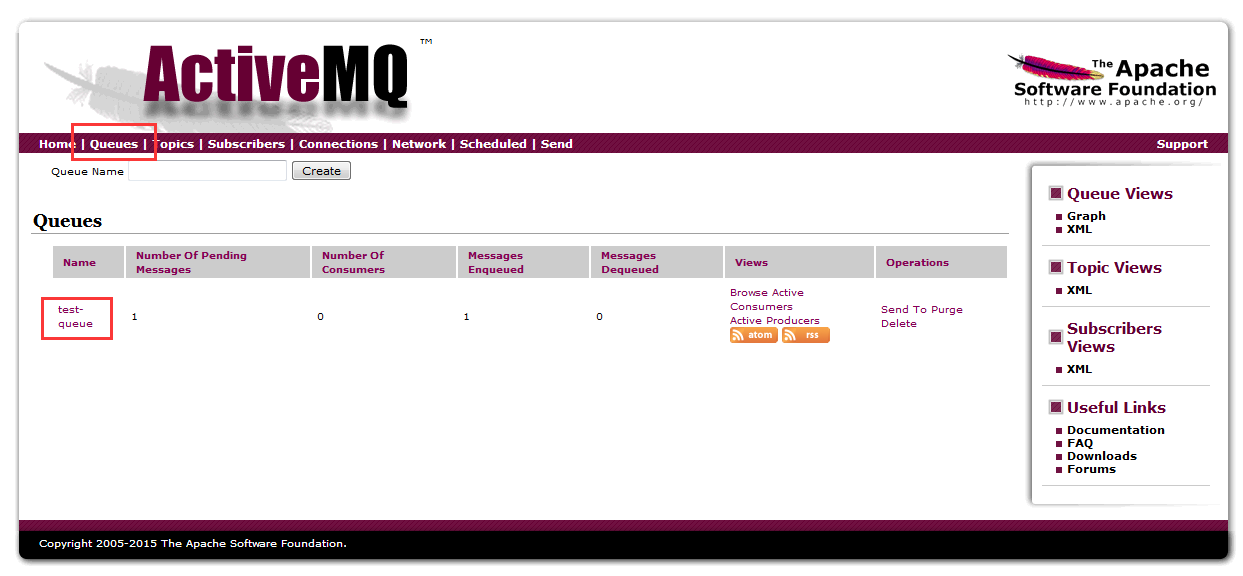


## Queue

### Producer

**@Test  
public void** testQueueProducer() **throws Exception {  
 // 第一步：创建ConnectionFactory对象，需要指定服务端ip及端口号。  
 //brokerURL服务器的ip及端口号** ConnectionFactory **connectionFactory = new** ActiveMQConnectionFactory(**"tcp://www.vm.com:61616"**);  
 **// 第二步：使用ConnectionFactory对象创建一个Connection对象。** Connection **connection = connectionFactory**.**createConnection**();  
 **// 第三步：开启连接，调用Connection对象的start方法。  
 connection**.**start**();  
 **// 第四步：使用Connection对象创建一个Session对象。  
 //第一个参数：是否开启事务。true：开启事务，第二个参数忽略。  
 //第二个参数：当第一个参数为false时，才有意义。消息的应答模式。1、自动应答2、手动应答。一般是自动应答。** Session **session = connection**.**createSession**(**false**, Session.***AUTO\_ACKNOWLEDGE***);  
 **// 第五步：使用Session对象创建一个Destination对象（topic、queue），此处创建一个Queue对象。  
 //参数：队列的名称。** Queue **queue = session**.**createQueue**(**"test-queue"**);  
 **// 第六步：使用Session对象创建一个Producer对象。** MessageProducer **producer = session**.**createProducer**(**queue**);  
 **// 第七步：创建一个Message对象，创建一个TextMessage对象。** TextMessage **textMessage = session**.**createTextMessage**(**"hello activeMq,this is my first test."**);  
 **// 第八步：使用Producer对象发送消息。  
 producer**.**send**(**textMessage**);  
 **// 第九步：关闭资源。  
 producer**.**close**();  
 **session**.**close**();  
 **connection**.**close**();  
**}**

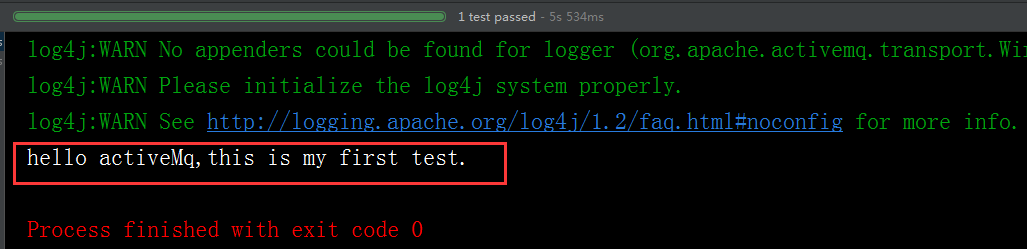
去管理界面查看：



### Consumer

**@Test  
public void** testQueueConsumer() **throws Exception {  
 // 第一步：创建一个ConnectionFactory对象。** ConnectionFactory **connectionFactory = new** ActiveMQConnectionFactory(**"tcp://www.vm.com:61616"**);  
 **// 第二步：从ConnectionFactory对象中获得一个Connection对象。** Connection **connection = connectionFactory**.**createConnection**();  
 **// 第三步：开启连接。调用Connection对象的start方法。  
 connection**.**start**();  
 **// 第四步：使用Connection对象创建一个Session对象。** Session **session = connection**.**createSession**(**false**, Session.***AUTO\_ACKNOWLEDGE***);  
 **// 第五步：使用Session对象创建一个Destination对象。和发送端保持一致queue，并且队列的名称一致。** Queue **queue = session**.**createQueue**(**"test-queue"**);  
 **// 第六步：使用Session对象创建一个Consumer对象。** MessageConsumer **consumer = session**.**createConsumer**(**queue**);  
 **// 第七步：接收消息。  
 consumer**.**setMessageListener**(**new** MessageListener() **{  
  
 @Override  
 public void** onMessage(Message **message**) **{  
 try {** TextMessage **textMessage =** (TextMessage) **message**;  
 **String text = null**;  
 **//取消息的内容  
 text = textMessage**.**getText**();  
 **// 第八步：打印消息。  
 System**.***out***.println(**text**);  
 **} catch** (**JMSException e**) **{  
 e**.printStackTrace();  
 **}  
 }  
 }**); **//线程要等待  
 System**.***in***.**read**();  
 **// 第九步：关闭资源  
 consumer**.**close**();  
 **session**.**close**();  
 **connection**.**close**();  
**}**

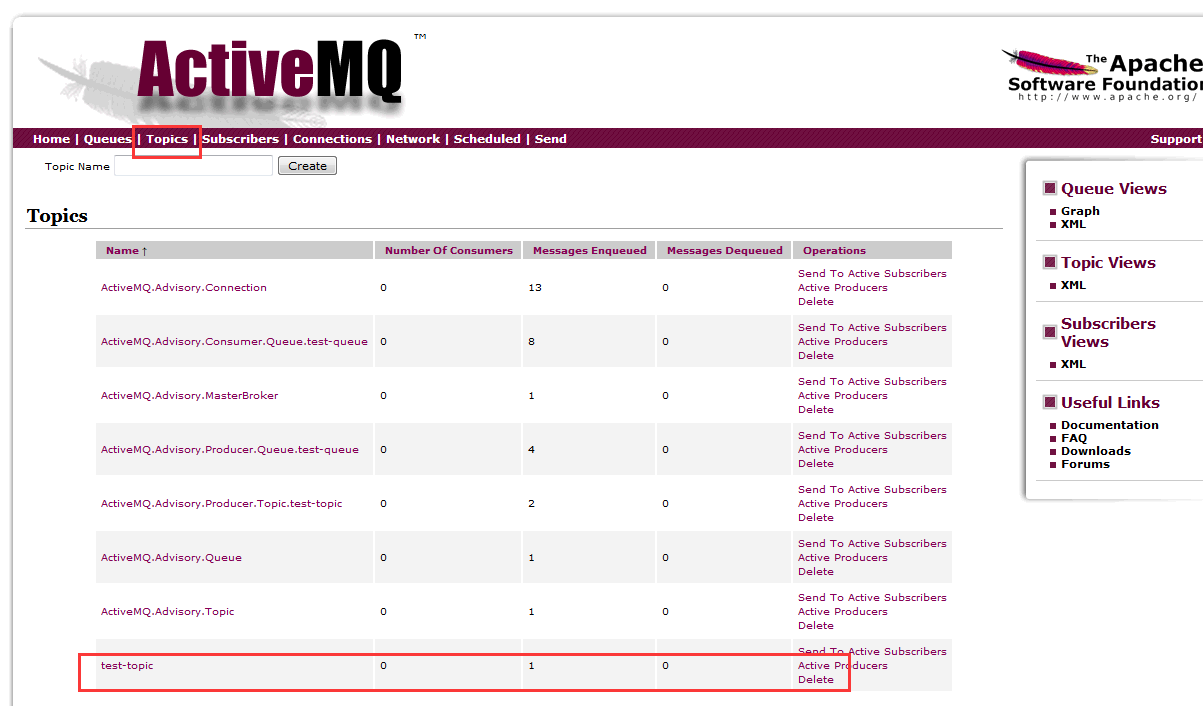
控制台：



## Topic

### Producer

**@Test  
 public void** testTopicProducer() **throws Exception {  
 // 第一步：创建ConnectionFactory对象，需要指定服务端ip及端口号。  
 // brokerURL服务器的ip及端口号** ConnectionFactory **connectionFactory = new** ActiveMQConnectionFactory(**"tcp://www.vm.com:61616"**);  
 **// 第二步：使用ConnectionFactory对象创建一个Connection对象。** Connection **connection = connectionFactory**.**createConnection**();  
 **// 第三步：开启连接，调用Connection对象的start方法。  
 connection**.**start**();  
 **// 第四步：使用Connection对象创建一个Session对象。  
 // 第一个参数：是否开启事务。true：开启事务，第二个参数忽略。  
 // 第二个参数：当第一个参数为false时，才有意义。消息的应答模式。1、自动应答2、手动应答。一般是自动应答。** Session **session = connection**.**createSession**(**false**, Session.***AUTO\_ACKNOWLEDGE***);  
 **// 第五步：使用Session对象创建一个Destination对象（topic、queue），此处创建一个topic对象。  
 // 参数：话题的名称。** Topic **topic = session**.**createTopic**(**"test-topic"**);  
 **// 第六步：使用Session对象创建一个Producer对象。** MessageProducer **producer = session**.**createProducer**(**topic**);  
 **// 第七步：创建一个Message对象，创建一个TextMessage对象。  
*/\*  
 \* TextMessage message = new ActiveMQTextMessage(); message.setText(  
 \* "hello activeMq,this is my first test.");  
 \*/*** TextMessage **textMessage = session**.**createTextMessage**(**"hello activeMq,this is my topic test"**);  
 **// 第八步：使用Producer对象发送消息。  
 producer**.**send**(**textMessage**);  
 **// 第九步：关闭资源。  
 producer**.**close**();  
 **session**.**close**();  
 **connection**.**close**();  
 **}**



### Consumer

**@Test  
public void** testTopicConsumer() **throws Exception {  
 // 第一步：创建一个ConnectionFactory对象。** ConnectionFactory **connectionFactory = new** ActiveMQConnectionFactory(**"tcp://www.vm.com:61616"**);  
 **// 第二步：从ConnectionFactory对象中获得一个Connection对象。** Connection **connection = connectionFactory**.**createConnection**();  
 **// 第三步：开启连接。调用Connection对象的start方法。  
 connection**.**start**();  
 **// 第四步：使用Connection对象创建一个Session对象。** Session **session = connection**.**createSession**(**false**, Session.***AUTO\_ACKNOWLEDGE***);  
 **// 第五步：使用Session对象创建一个Destination对象。和发送端保持一致topic，并且话题的名称一致。** Topic **topic = session**.**createTopic**(**"test-topic"**);  
 **// 第六步：使用Session对象创建一个Consumer对象。** MessageConsumer **consumer = session**.**createConsumer**(**topic**);  
 **// 第七步：接收消息。  
 consumer**.**setMessageListener**(**new** MessageListener() **{  
  
 @Override  
 public void** onMessage(Message **message**) **{  
 try {** TextMessage **textMessage =** (TextMessage) **message**;  
 **String text = null**;  
 **// 取消息的内容  
 text = textMessage**.**getText**();  
 **// 第八步：打印消息。  
 System**.***out***.println(**text**);  
 **} catch** (**JMSException e**) **{  
 e**.printStackTrace();  
 **}  
 }  
 }**);  
 **System**.***out***.println(**"topic的消费端03。。。。。"**);  
 **// 等待键盘输入,消费者再发消息，它可以收到  
 System**.***in***.**read**();  
 **// 第九步：关闭资源  
 consumer**.**close**();  
 **session**.**close**();  
 **connection**.**close**();  
**}**

# 整合Spring

activeMQ整合Spring需要引入jar包：

<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-jms</artifactId>  
 <version>4.3.11.RELEASE</version>  
</dependency>  
<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context-support</artifactId>  
 <version>4.3.11.RELEASE</version>  
</dependency>

## 生产者

spring-activemq-producer.xml

*<?*xml version="1.0" encoding="UTF-8"*?>*<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">  
 <!-- 产生Connection的ConnectionFactory，由对应的 JMS服务厂商提供 -->  
 <bean id="targetConnectionFactory" class="org.apache.activemq.ActiveMQConnectionFactory">  
 <property name="brokerURL" value="tcp://www.vm.com:61616" />  
 </bean>  
 <!-- Spring用于管理真正的ConnectionFactory的ConnectionFactory -->  
 <bean id="connectionFactory"  
 class="org.springframework.jms.connection.SingleConnectionFactory">  
 <!-- 目标ConnectionFactory对应真实的可以产生JMS Connection的ConnectionFactory -->  
 <property name="targetConnectionFactory" ref="targetConnectionFactory" />  
 </bean>  
 <!-- 配置生产者 -->  
 <!-- Spring提供的JMS工具类，它可以进行消息发送、接收等 -->  
 <bean id="jmsTemplate" class="org.springframework.jms.core.JmsTemplate">  
 <!-- 这个connectionFactory对应的是我们定义的Spring提供的那个ConnectionFactory对象 -->  
 <property name="connectionFactory" ref="connectionFactory" />  
 </bean>  
 <!--这个是队列目的地，点对点的 -->  
 <bean id="queueDestination" class="org.apache.activemq.command.ActiveMQQueue">  
 <constructor-arg>  
 <value>spring-queue</value>  
 </constructor-arg>  
 </bean>  
 <!--这个是主题目的地，一对多的 -->  
 <bean id="topicDestination" class="org.apache.activemq.command.ActiveMQTopic">  
 <constructor-arg value="topic" />  
 </bean>  
</beans>

测试方法

**@Test  
public void** testQueueProducer() **throws Exception {  
 // 第一步：初始化一个spring容器** ApplicationContext **applicationContext = new** ClassPathXmlApplicationContext(**"classpath:spring-activemq-producer.xml"**);  
 **// 第二步：从容器中获得JMSTemplate对象。  
 JmsTemplate jmsTemplate = applicationContext**.**getBean**(**JmsTemplate**.**class**);  
 **// 第三步：从容器中获得一个Destination对象** Queue **queue =** (Queue) **applicationContext**.**getBean**(**"queueDestination"**);  
 **// 第四步：使用JMSTemplate对象发送消息，需要知道Destination  
 jmsTemplate**.send(**queue**, **new** MessageCreator() **{  
 @Override  
 public** Message createMessage(Session **session**) **throws JMSException {** TextMessage **textMessage = session**.**createTextMessage**(**"spring activemq test"**);  
 **return textMessage**;  
 **}  
 }**);  
**}**

## 消费者

**import javax.jms.JMSException**;  
**import javax.jms.Message**;  
**import javax.jms.MessageListener**;  
**import javax.jms.TextMessage**;  
  
**public class MyMessageListener implements MessageListener {  
  
 @Override  
 public void** onMessage(**Message message**) **{  
  
 try {  
 TextMessage textMessage =** (**TextMessage**) **message**;  
 **//取消息内容  
 String text = textMessage**.**getText**();  
 **System**.***out***.println(**text**);  
 **} catch** (**JMSException e**) **{  
 e**.printStackTrace();  
 **}  
 }  
  
}**

spring-activemq-consumer.xml

*<?*xml version="1.0" encoding="UTF-8"*?>*<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">  
 <!-- 真正可以产生Connection的ConnectionFactory，由对应的 JMS服务厂商提供 -->  
 <bean id="targetConnectionFactory" class="org.apache.activemq.ActiveMQConnectionFactory">  
 <property name="brokerURL" value="tcp://www.vm.com:61616" />  
 </bean>  
 <!-- Spring用于管理真正的ConnectionFactory的ConnectionFactory -->  
 <bean id="connectionFactory"  
 class="org.springframework.jms.connection.SingleConnectionFactory">  
 <!-- 目标ConnectionFactory对应真实的可以产生JMS Connection的ConnectionFactory -->  
 <property name="targetConnectionFactory" ref="targetConnectionFactory" />  
 </bean>  
 <!--这个是队列目的地，点对点的 -->  
 <bean id="queueDestination" class="org.apache.activemq.command.ActiveMQQueue">  
 <constructor-arg>  
 <value>spring-queue</value>  
 </constructor-arg>  
 </bean>  
 <!--这个是主题目的地，一对多的 -->  
 <bean id="topicDestination" class="org.apache.activemq.command.ActiveMQTopic">  
 <constructor-arg value="topic" />  
 </bean>  
 <!-- 接收消息 -->  
 <!-- 配置监听器 -->  
 <bean id="myMessageListener" class="MyMessageListener" />  
 <!-- 消息监听容器 -->  
 <bean class="org.springframework.jms.listener.DefaultMessageListenerContainer">  
 <property name="connectionFactory" ref="connectionFactory" />  
 <property name="destination" ref="queueDestination" />  
 <property name="messageListener" ref="myMessageListener" />  
 </bean>  
</beans>

测试方法：

**@Test  
public void** testQueueConsumer() **throws Exception {  
 //初始化spring容器** ApplicationContext **applicationContext = new** ClassPathXmlApplicationContext(**"classpath:spring-activemq-consumer.xml"**);  
 **//等待  
 System**.***in***.**read**();  
**}**