# ActiveMQ介绍

ActiveMQ 是Apache出品，最流行的，能力强劲的开源消息总线。ActiveMQ 是一个完全支持JMS1.1和J2EE 1.4规范的 JMS Provider实现,尽管JMS规范出台已经是很久的事情了,但是JMS在当今的J2EE应用中间仍然扮演着特殊的地位。

主要特点：

1. 多种语言和协议编写客户端。语言: Java, C, C++, C#, Ruby, Perl, Python, PHP。应用协议: OpenWire,Stomp REST,WS Notification,XMPP,AMQP

2. 完全支持JMS1.1和J2EE 1.4规范 (持久化,XA消息,事务)

3. 对Spring的支持,ActiveMQ可以很容易内嵌到使用Spring的系统里面去,而且也支持Spring2.0的特性

4. 通过了常见J2EE服务器(如 Geronimo,JBoss 4, GlassFish,WebLogic)的测试,其中通过JCA 1.5 resource adaptors的配置,可以让ActiveMQ可以自动的部署到任何兼容J2EE 1.4 商业服务器上

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

6. 支持通过JDBC和journal提供高速的消息持久化

7. 从设计上保证了高性能的集群,客户端-服务器,点对点

8. 支持Ajax

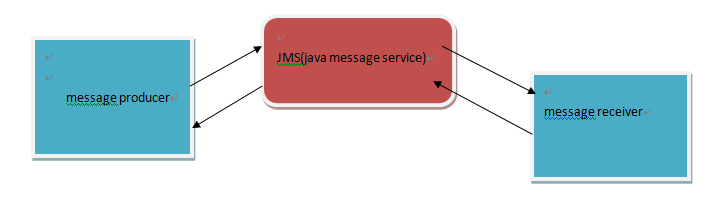
9. 支持与Axis的整合

10. 可以很容易得调用内嵌JMS provider,进行测试

## JMS介绍

jms即Java消息服务（Java Message Service）应用程序接口是一个Java平台中关于面向消息中间件（MOM）的API，用于在两个应用程序之间，或分布式系统中发送消息，进行异步通信。Java消息服务是一个与具体平台无关的API，绝大多数MOM提供商都对JMS提供支持。

JMS（Java Messaging Service）是Java平台上有关面向消息中间件的技术规范，它便于消息系统中的Java应用程序进行消息交换,并且通过提供标准的产生、发送、接收消息的接口简化企业应用的开发，翻译为Java消息服务。我们在应用jms的结构为：

****

**JMS有以下元素组成**

**JMS提供者**

连接面向消息中间件的，JMS接口的一个实现。提供者可以是Java平台的JMS实现，也可以是非Java平台的面向消息中间件的适配器。

**JMS客户**

　生产或消费消息的基于Java的应用程序或对象。

**JMS生产者**

　　创建并发送消息的JMS客户。

**JMS消费者**

**接收消息的JMS客户**。

**JMS消息**

　包括可以在JMS客户之间传递的数据的对象

**JMS队列**

　一个容纳那些被发送的等待阅读的消息的区域。队列暗示，这些消息将按照顺序发送。一旦一个消息被阅读，该消息将被从队列中移走。

**JMS主题**

　一种支持发送消息给多个订阅者的机制。

**JMS模型**

Java消息服务应用程序结构支持两种模型：即点对点或队列模型，发布者/订阅者模型。

**在点对点或队列模型**下，一个生产者向一个特定的队列发布消息，一个消费者从该队列中读取消息。这里，生产者知道消费者的队列，并直接将消息发送到消费者的队列。这种模式被概括为：只有一个消费者将获得消息。生产者不需要在接收者消费该消息期间处于运行状态，接收者也同样不需要在消息发送时处于运行状态。每一个成功处理的消息都由接收者签收。

**发布者／订阅者模型**支持向一个特定的消息主题发布消息。0或多个订阅者可能对接收来自特定消息主题的消息感兴趣。在这种模型下，发布者和订阅者彼此不知道对方。这种模式好比是匿名公告板。这种模式被概括为：多个消费者可以获得消息.在发布者和订阅者之间存在时间依赖性。发布者需要建立一个订阅（subscription），以便客户能够购订阅。订阅者必须保持持续的活动状态以接收消息，除非订阅者建立了持久的订阅。在那种情况下，在订阅者未连接时发布的消息将在订阅者重新连接时重新发布。

JMS现在有两种传递消息的方式。标记为NON\_PERSISTENT的消息最多投递一次，而标记为PERSISTENT的消息将使用暂存后再转送的机理投递。如果一个JMS服务离线，那么持久性消息不会丢失但是得等到这个服务恢复联机时才会被传递。所以默认的消息传递方式是非持久性的。即使使用非持久性消息可能降低内务和需要的存储器，并且这种传递方式只有当你不需要接收所有的消息时才使用。

虽然JMS规范并不需要JMS供应商实现消息的优先级路线，但是它需要递送加快的消息优先于普通级别的消息。JMS定义了从0到9的优先级路线级别，0是最低的优先级而9则是最高的。更特殊的是0到4是正常优先级的变化幅度，而5到9是加快的优先级的变化幅度。举例来说： topicPublisher.publish (message, DeliveryMode.PERSISTENT, 8, 10000); //Pub-Sub 或 queueSender.send(message,DeliveryMode.PERSISTENT, 8, 10000);//P2P 这个代码片断，有两种消息模型，映射递送方式是持久的，优先级为加快型，生存周期是10000 (以毫秒度量 )。如果生存周期设置为零，这则消息将永远不会过期。当消息需要时间限制否则将使其无效时，设置生存周期是有用的。

JMS定义了五种不同的消息正文格式，以及调用的消息类型，允许你发送并接收以一些不同形式的数据，提供现有消息格式的一些级别的兼容性。

　　· StreamMessage -- Java原始值的数据流

　　· MapMessage--一套名称-值对

　　· TextMessage--一个字符串对象

　　· ObjectMessage--一个序列化的 Java对象

　　· BytesMessage--一个未解释字节的数据流

**JMS应用程序接口**

**ConnectionFactory 接口**（连接工厂）

用户用来创建到JMS提供者的连接的被管对象。JMS客户通过可移植的接口访问连接，这样当下层的实现改变时，代码不需要进行修改。 管理员在JNDI名字空间中配置连接工厂，这样，JMS客户才能够查找到它们。根据消息类型的不同，用户将使用队列连接工厂，或者主题连接工厂。

**Connection 接口**（连接）

连接代表了应用程序和消息服务器之间的通信链路。在获得了连接工厂后，就可以创建一个与JMS提供者的连接。根据不同的连接类型，连接允许用户创建会话，以发送和接收队列和主题到目标。

**Destination 接口**（目标）

目标是一个包装了消息目标标识符的被管对象，消息目标是指消息发布和接收的地点，或者是队列，或者是主题。JMS管理员创建这些对象，然后用户通过JNDI发现它们。和连接工厂一样，管理员可以创建两种类型的目标，点对点模型的队列，以及发布者／订阅者模型的主题。

**MessageConsumer 接口**（消息消费者）

由会话创建的对象，用于接收发送到目标的消息。消费者可以同步地（阻塞模式），或异步（非阻塞）接收队列和主题类型的消息。

**MessageProducer 接口**（消息生产者）

由会话创建的对象，用于发送消息到目标。用户可以创建某个目标的发送者，也可以创建一个通用的发送者，在发送消息时指定目标。

**Message 接口**（消息）

是在消费者和生产者之间传送的对象，也就是说从一个应用程序创送到另一个应用程序。一个消息有三个主要部分：

消息头（必须）：包含用于识别和为消息寻找路由的操作设置。

一组消息属性（可选）：包含额外的属性，支持其他提供者和用户的兼容。可以创建定制的字段和过滤器（消息选择器）。

一个消息体（可选）：允许用户创建五种类型的消息（文本消息，映射消息，字节消息，流消息和对象消息）。

消息接口非常灵活，并提供了许多方式来定制消息的内容。

**Session 接口**（会话）

表示一个单线程的上下文，用于发送和接收消息。由于会话是单线程的，所以消息是连续的，就是说消息是按照发送的顺序一个一个接收的。会话的好处是它支持事务。如果用户选择了事务支持，会话上下文将保存一组消息，直到事务被提交才发送这些消息。在提交事务之前，用户可以使用回滚操作取消这些消息。一个会话允许用户创建消息生产者来发送消息，创建消息消费者来接收消息。

connection.createSession(paramA,paramB);

paramA 取值有 : true or false 表示是否支持事务

paramB 取值有:Session.AUTO\_ACKNOWLEDGE，Session.CLIENT\_ACKNOWLEDGE，DUPS\_OK\_ACKNOWLEDGE，SESSION\_TRANSACTED

createSession(paramA,paramB);

paramA是设置事务的，paramB设置acknowledgment mode

paramA设置为false时：paramB的值可为Session.AUTO\_ACKNOWLEDGE，Session.CLIENT\_ACKNOWLEDGE，DUPS\_OK\_ACKNOWLEDGE其中一个。

paramA设置为true时：paramB的值忽略， acknowledgment mode被jms服务器设置为SESSION\_TRANSACTED 。

Session.AUTO\_ACKNOWLEDGE为自动确认，客户端发送和接收消息不需要做额外的工作。

Session.CLIENT\_ACKNOWLEDGE为客户端确认。客户端接收到消息后，必须调用javax.jms.Message的acknowledge方法。jms服务器才会删除消息。

DUPS\_OK\_ACKNOWLEDGE允许副本的确认模式。一旦接收方应用程序的方法调用从处理消息处返回，会话对象就会确认消息的接收；而且允许重复确认。在需要考虑资源使用时，这种模式非常有效。

**JMS提供者实现**

要使用Java消息服务，你必须要有一个JMS提供者，管理会话和队列。现在既有开源的提供者也有专有的提供者。

　　开源的提供者包括：

　　Apache ActiveMQ

　　JBoss 社区所研发的 HornetQ

　　Joram

　　Coridan的MantaRay

　　The OpenJMS Group的OpenJMS

　　专有的提供者包括：

　　BEA的BEA WebLogic Server JMS

　　TIBCO Software的EMS

　　GigaSpaces Technologies的GigaSpaces

　　Softwired 2006的iBus

　　IONA Technologies的IONA JMS

　　SeeBeyond的IQManager（2005年8月被Sun Microsystems并购）

　　webMethods的JMS+ -

　　my-channels的Nirvana

　　Sonic Software的SonicMQ

　　SwiftMQ的SwiftMQ

　　IBM的WebSphere MQ

### ActiveMQ例子

在 Java 里有 JMS 的多个实现。其中 apache 下的 ActiveMQ 就是不错的选择。

用 ActiveMQ 最好还是了解下 JMS

|  |  |  |
| --- | --- | --- |
| **JMS 公共** | **点对点域** | **发布/订阅域** |
| ConnectionFactory | QueueConnectionFactory | TopicConnectionFactory |
| Connection | QueueConnection | TopicConnection |
| Destination | Queue | Topic |
| Session | QueueSession | TopicSession |
| MessageProducer | QueueSender | TopicPublisher |
| MessageConsumer | QueueReceiver | TopicSubscriber |

JMS 定义了两种方式：Quere（点对点）；Topic（发布/订阅）。

ConnectionFactory 是连接工厂，负责创建Connection。

Connection 负责创建 Session。

Session 创建 MessageProducer（用来发消息） 和 MessageConsumer（用来接收消息）。

Destination 是消息的目的地。

Jms 发送 代码

public static void main(String[] args) throws Exception {

ConnectionFactory connectionFactory = new ActiveMQConnectionFactory();

Connection connection = connectionFactory.createConnection();

connection.start();

//（parameter a:是否支持事务，parameter b:消息应答模式）

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

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

//创建什么什么模式的生产者，可选参数：

[Queue](http://www.ostools.net/uploads/apidocs/javaEE6/javax/jms/Queue.html), [TemporaryQueue](http://www.ostools.net/uploads/apidocs/javaEE6/javax/jms/TemporaryQueue.html" \o "interface in javax.jms), [TemporaryTopic](http://www.ostools.net/uploads/apidocs/javaEE6/javax/jms/TemporaryTopic.html" \o "interface in javax.jms), [Topic](http://www.ostools.net/uploads/apidocs/javaEE6/javax/jms/Topic.html)

MessageProducer producer = session.createProducer(destination);

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

MapMessage message = session.createMapMessage();

message.setLong("count", new Date().getTime());

Thread.sleep(1000);

//通过消息生产者发出消息

producer.send(message);

}

session.commit();

session.close();

connection.close();

}

Jms 接收代码：

public static void main(String[] args) throws Exception {

ConnectionFactory connectionFactory = new ActiveMQConnectionFactory();

Connection connection = connectionFactory.createConnection();

connection.start();

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

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

MessageConsumer consumer = session.createConsumer(destination);

/\*//listener 方式 ，消费者必须要重写MessageListener的onMessage（）来监听是否有消息可以接收

consumer.setMessageListener(new MessageListener() {

public void onMessage(Message msg) {

MapMessage message = (MapMessage) msg;

//TODO something....

System.out.println(" 收到消息：" + new Date(message.getLong("count")));

session.commit();

}

});

Thread.sleep(30000);

\*/

int i=0;

while(i<3) {

i++;

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

session.commit();

//TODO something....

System.out.println("收到消 息：" + new Date(message.getLong("count")));

}

session.close();

connection.close();

}

启动 JmsReceiver 和 JmsSender 可以在看输出三条时间信息。当然 Jms 还指定有其它格式的数据，如 TextMessage

结合 Spring 的 JmsTemplate 方便用：

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-2.5.xsd">

<!-- 在非 web / ejb 容器中使用 pool 时，要手动 stop，spring 不会为 你执行 destroy-method 的方法

<bean id="jmsFactory" class="org.apache.activemq.pool.PooledConnectionFactory" destroy-method="stop">

<property name="connectionFactory">

<bean class="org.apache.activemq.ActiveMQConnectionFactory">

<property name="brokerURL" value="tcp://localhost:61616" />

</bean>

</property>

</bean>

-->

<bean id="jmsFactory" class="org.apache.activemq.ActiveMQConnectionFactory">

<property name="brokerURL" value="tcp://localhost:61616" />

</bean>

<bean id="jmsTemplate" class="org.springframework.jms.core.JmsTemplate">

<property name="connectionFactory" ref="jmsFactory" />

<property name="defaultDestination" ref="destination" />

<property name="messageConverter">

<bean class="org.springframework.jms.support.converter.SimpleMessageConverter" />

</property>

</bean>

<bean id="destination" class="org.apache.activemq.command.ActiveMQQueue">

<constructor-arg index="0" value="my-queue" />

</bean>

</beans>

**sender**：

public static void main(String[] args) {

ApplicationContext ctx = new FileSystemXmlApplicationContext("classpath:app\*.xml");

JmsTemplate jmsTemplate = (JmsTemplate) ctx.getBean("jmsTemplate");

jmsTemplate.send(new MessageCreator() {

public Message createMessage(Session session) throws JMSException {

MapMessage mm = session.createMapMessage();

mm.setLong("count", new Date().getTime());

return mm;

}

});

}

**receiver：**

public static void main(String[] args) {

ApplicationContext ctx = new FileSystemXmlApplicationContext("classpath:app\*.xml");

JmsTemplate jmsTemplate = (JmsTemplate) ctx.getBean("jmsTemplate");

while(true) {

Map<String, Object> mm = (Map<String, Object>) jmsTemplate.receiveAndConvert();

System.out.println("收到消 息：" + new Date((Long)mm.get("count")));

}

}

注意：直接用 Jms 接口时接收了消息后要提交一下，否则下次启动接收者时还可以收到旧数据。有了 JmsTemplate 就不用自己提交 session.commit() 了。如果使用了 PooledConnectionFactory 要把 apache-activemq-5.3.0\lib\optional\activemq-pool-5.3.0.jar 加到 classpath

### 入门手册

#### 启动ActiveMQ

用activemq start命令启动ActiveMQ服务器。

在浏览器输入<http://localhost:8061/>并点击回车，可以看到如下界面：

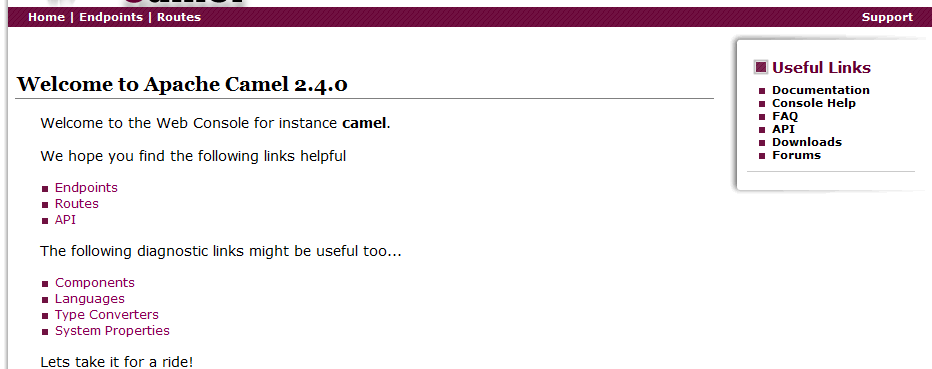


#### 总体页面浏览

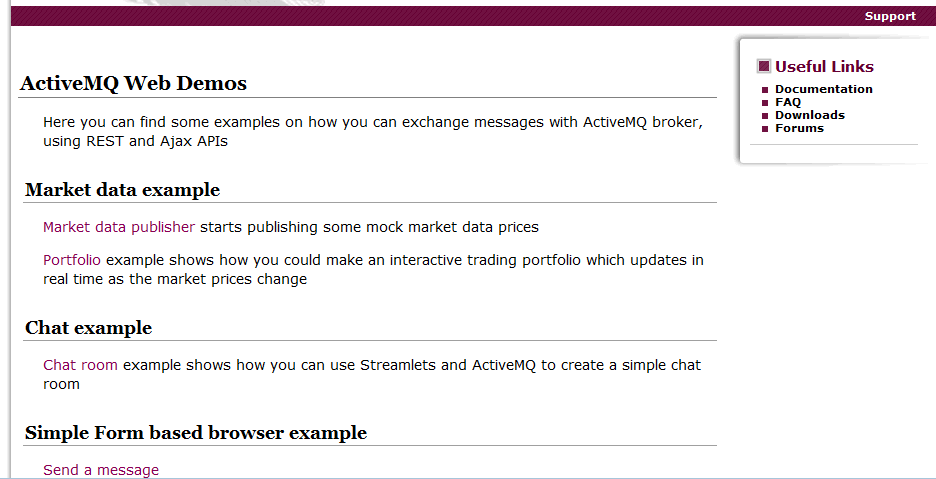
点击，会看到如下界面：



点击：

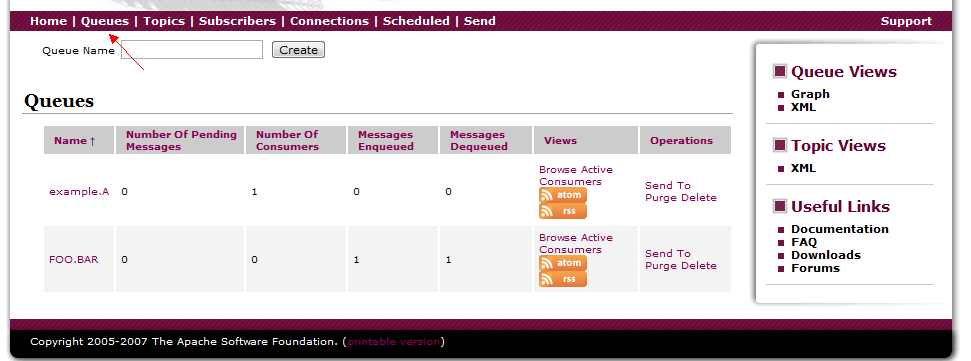


点击

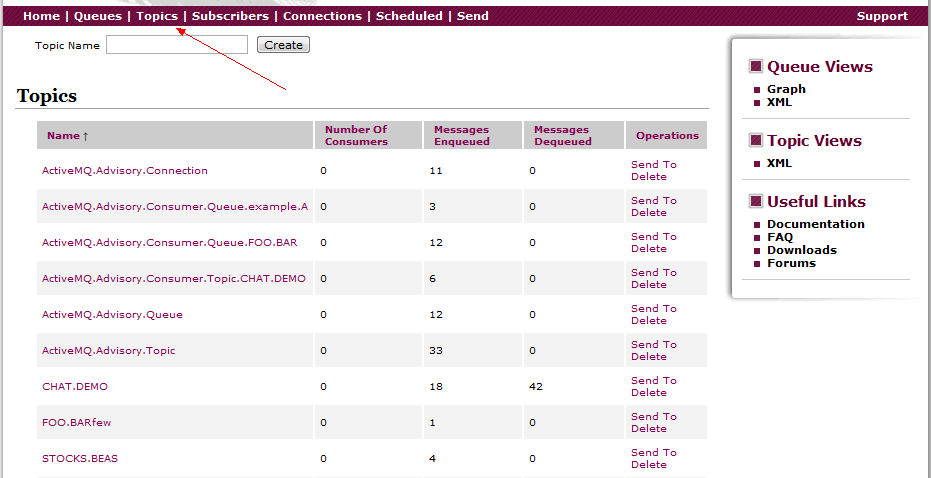


#### ActiveMQ管理界面

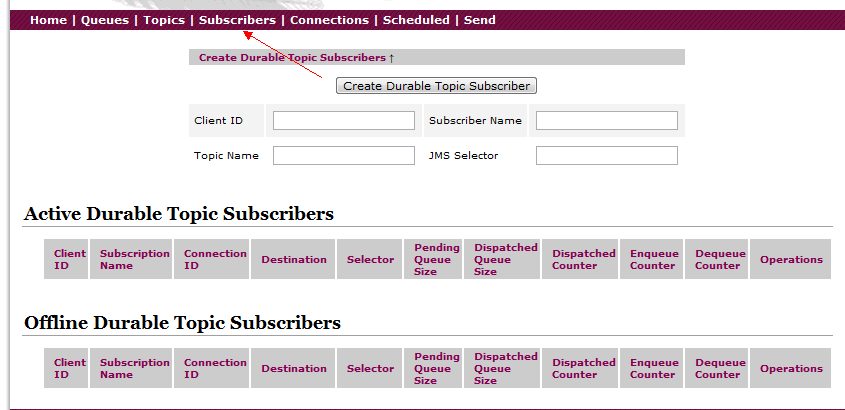
队列管理



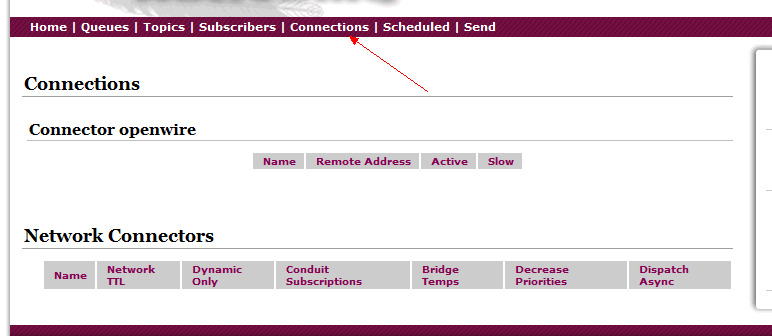
话题管理



订阅者管理



连接管理



发送JMS消息

