# 1》JMS

## 1.1》什么是JMS

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。

## 1.2》为什么要学JMS

在JAVA中，如果两个应用程序之间对各自都不了解，甚至这两个程序可能部署在不同的大洲上，那么它们之间如何发送消息呢？举个例子，一个应用程序A部署在印度，另一个应用程序部署在美国，然后每当A触发某件事后，B想从A获取一些更新信息。当然，也有可能不止一个B对A的更新信息感兴趣，可能会有N个类似B的应用程序想从A中获取更新的信息。在这种情况下，JAVA提供了最佳的解决方案-JMS，完美解决了上面讨论的问题。

JMS与RMI不同，发送消息的时候，接收者不需要在线。服务器发送了消息，然后就不管了；等到客户端上线的时候，能保证接收到服务器发送的消息。这是一个很强大的解决方案，能处理当今世界很多普遍问题。

## 1.3》JMS有什么优势

异步：JMS天生就是异步的，客户端获取消息的时候，不需要主动发送请求，消息会自动发送给可用的客户端。

可靠：JMS保证消息只会递送一次。大家都遇到过重复创建消息问题，而JMS能帮你避免该问题，只是避免而不是杜绝，所以在一些糟糕的环境下还是有可能会出现重复。

## 1.4》JMS数据交互的两种方式

### 点对点消息模型

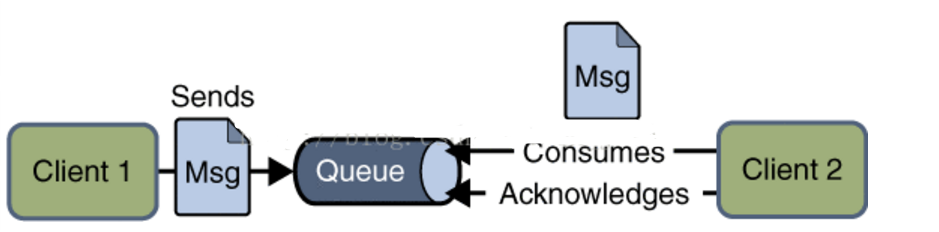
（1）、每个消息只有一个接受者（自己测试了一下，可以有多个接受者，但是当有多个接收者时，每个接收者只能获取随机的几条信息）

（2）、消息发送者和消息接受者并没有时间依赖性。

（3）、当消息发送者发送消息的时候，无论接收者程序在不在运行，都能获取到消息；

（4）、当接收者收到消息的时候，会发送确认收到通知（acknowledgement）。

1. 、点对点消息模型图：



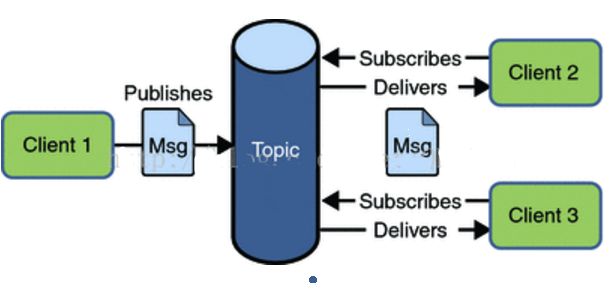
### 发布/订阅消息模型

（1）、一个消息可以传递给多个订阅者

（2）、发布者和订阅者有时间依赖性，只有当客户端创建订阅后才能接受消息，且订阅者需一直保持活动状态以接收消息。

（3）、为了缓和这样严格的时间相关性，JMS允许订阅者创建一个可持久化的订阅。这样，即使订阅者没有被激活（运行），它也能接收到发布者的消息。

（4）、发布/订阅消息模型图：



# 2》ActiveMQ

## 2.1》什么是ActiveMQ

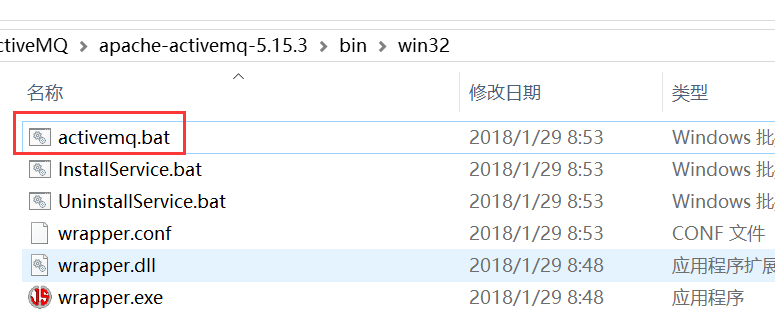
ActiveMQ 是Apache出品，最流行的，能力强劲的开源消息总线。ActiveMQ 是一个完全支持JMS1.1和J2EE 1.4规范的 JMS Provider实现，尽管JMS规范出台已经是很久的事情了，但是JMS在当今的J2EE应用中间仍然扮演着特殊的地位。

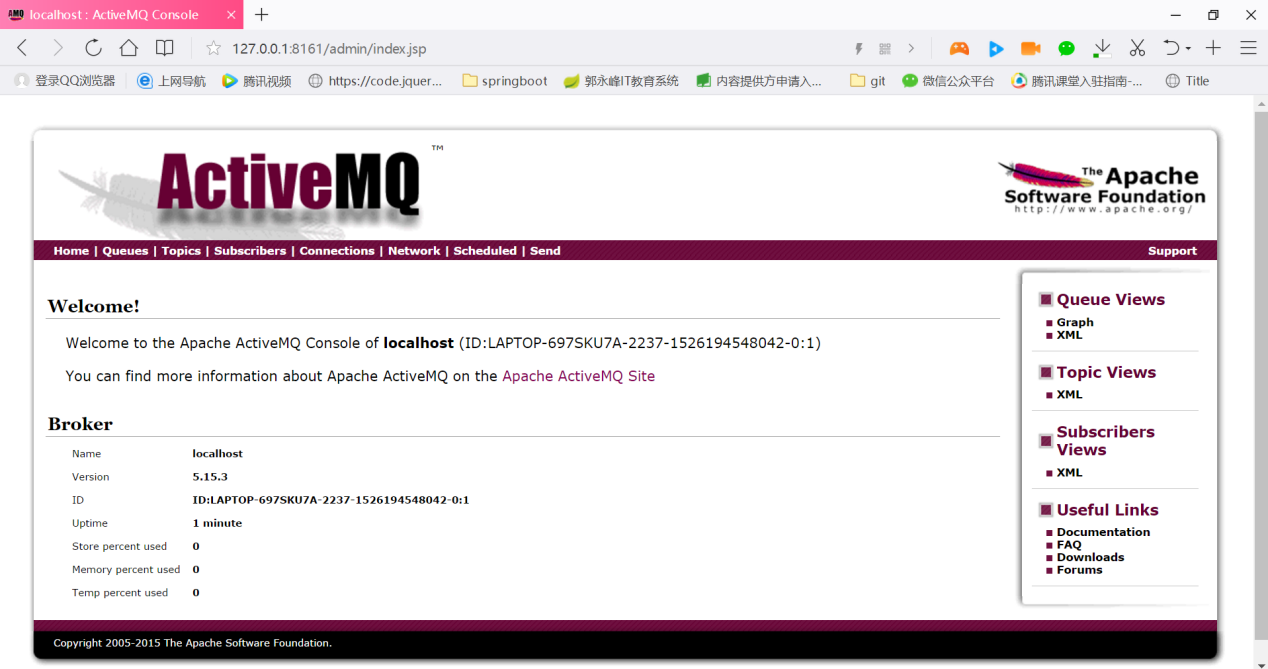
## 2.2》ActiveMQ的下载

主页：<http://activemq.apache.org/> 目前最新版本：5.15.3

开发包及源码下载地址：http://activemq.apache.org/activemq-5111-release.html

ActiveMQ 服务启动地址：http://127.0.0.1:8161/admin/ 用户名/密码 admin/admin





## 2.3》ActiveMQ的点对点使用

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。

Session.CLIENT\_ACKNOWLEDGE。 客户通过消息的 acknowledge 方法确认消息。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\_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。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。

### 消息生产者

|  |
| --- |
| **public class** JMSProducer {   **private static final** String ***USERNAME***= ActiveMQConnection.***DEFAULT\_USER***; *// 默认的连接用户名* **private static final** String ***PASSWORD***=ActiveMQConnection.***DEFAULT\_PASSWORD***; *// 默认的连接密码* **private static final** String ***BROKEURL***=ActiveMQConnection.***DEFAULT\_BROKER\_URL***; *// 默认的连接地址* **public static void** main(String[] args) {    ConnectionFactory connectionFactory;*//连接工厂* Connection connection = **null**;*//连接* Session session;*//会话* Destination destination;*//消息目的地* MessageProducer messageProducer;*//消息生产者* connectionFactory = **new** ActiveMQConnectionFactory(***USERNAME***,***PASSWORD***,***BROKEURL***);   **try** {  connection = connectionFactory.createConnection();*//通过工厂获取连接* connection.start();*//启动连接* session = connection.createSession(**true**,Session.***AUTO\_ACKNOWLEDGE***);*//创建session* destination = session.createQueue(**"新闻队列"**);  messageProducer = session.createProducer(destination);*//创建消息生产者   //发送消息* **for**(**int** i=0;i<10;i++){  TextMessage msg = session.createTextMessage(**"郭永峰IT工作室客服"** + (i + 1) +**"号"**);  messageProducer.send(destination,msg);  }   session.commit();   }**catch** (Exception e){  e.printStackTrace();  }**finally** {  **if**(connection!=**null**){  **try** {  connection.close();  } **catch** (JMSException e) {  *//* ***TODO Auto-generated catch block*** e.printStackTrace();  }  }  }  } } |

### 消息消费者

|  |
| --- |
| **public class** JMSConsumer {   **private static final** String ***USERNAME***= ActiveMQConnection.***DEFAULT\_USER***; *// 默认的连接用户名* **private static final** String ***PASSWORD***=ActiveMQConnection.***DEFAULT\_PASSWORD***; *// 默认的连接密码* **private static final** String ***BROKEURL***=ActiveMQConnection.***DEFAULT\_BROKER\_URL***; *// 默认的连接地址* **public static void** main(String[] args) {    ConnectionFactory connectionFactory;*//连接工厂* Connection connection = **null**;*//连接* Session session;*//会话* Destination destination;*//消息目的地* MessageConsumer consumer;*//消息消费者* connectionFactory = **new** ActiveMQConnectionFactory(***USERNAME***,***PASSWORD***,***BROKEURL***);   **try** {  connection = connectionFactory.createConnection();*//通过工厂获取连接* connection.start();*//启动连接* session = connection.createSession(**true**,Session.***AUTO\_ACKNOWLEDGE***);*//创建session* destination = session.createQueue(**"新闻队列"**);  consumer = session.createConsumer(destination);*//创建消息消费者   //发送消息* **while**(**true**){  TextMessage message = (TextMessage) consumer.receive(10000);  **if**(message != **null**){  System.***out***.println(message.getText());*//获取消息* }  }    }**catch** (Exception e){  e.printStackTrace();  } } |



### 接收消息监听器

|  |
| --- |
|  |
|  |

## 2.4》ActiveMQ的发布和订阅

对前面的项目进行更改就可以了

|  |
| --- |
|  |
|  |

# 3》RabbitMQ

RabbitMQ是一个由erlang开发的AMQP(Advanved Message Queue)的开源实现。

https://blog.csdn.net/whoamiyang/article/details/54954780