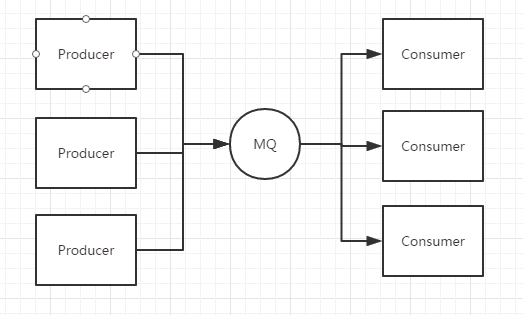
# JMS

JMS，即Java Message Service，通过面向消息中间件（MOM：Message Oriented Middleware）的方式很好的解决了上面的问题。大致的过程是这样的：发送者把消息发送给消息服务器，消息服务器将消息存放在若干队列/主题中，在合适的时候，消息服务器会将消息转发给接受者。在这个过程中，发送和接受是异步的，也就是发送无需等待，而且发送者和接受者的生命周期也没有必然关系；在pub/sub模式下，也可以完成一对多的通信，即让一个消息有多个接受者。



# 专业术语

Provider/MessageProvider：生产者

Consumer/MessageConsumer：消费者

PTP：Point To Point，点对点通信消息模型

Pub/Sub：Publish/Subscribe，发布订阅消息模型

Queue：队列，目标类型之一，和PTP结合

Topic：主题，目标类型之一，和Pub/Sub结合

ConnectionFactory：连接工厂，JMS用它创建连接

Connnection：JMS Client到JMS Provider的连接

Destination：消息目的地，由Session创建

# ActiveMQ QuickStart

ActiveMQ是Apache出品的，非常流行的消息中间件，可以说要掌握消息中间件，需要从ActiveMQ开始，要掌握更加强大的RocketMQ，也需要ActiveMQ的基础，因此我们来搞定它吧。官网地址：http://activemq.apache.org/，目前最新的版本是5.14.4，我这边将以最新版来讲解。这篇文章主要是ActiveMQ的初步，因此我这边暂时用windows版本，后期采用Linux。

ActiveMQ目录结构

|  |
| --- |
| bin下面存放的是ActiveMQ的启动脚本activemq.bat，注意分32、64位  conf里面是配置文件，重点关注的是activemq.xml、jetty.xml、jetty-realm.properties。在登录ActiveMQ Web控制台需要用户名、密码信息；在JMS CLIENT和ActiveMQ进行何种协议的连接、端口是什么等这些信息都在上面的配置文件中可以体现。  data目录下是ActiveMQ进行消息持久化存放的地方，默认采用的是kahadb，当然我们可以采用leveldb，或者采用JDBC存储到MySQL，或者干脆不使用持久化机制。  webapps，注意ActiveMQ自带Jetty提供Web管控台  lib中ActiveMQ为我们提供了分功能的JAR包，当然也提供了activemq-all-5.14.4.jar |

# 发布消息demo

|  |
| --- |
| //1.创建连接工厂ConnectionFactory 需要user，password，url  ConnectionFactory connectionFactory = **new** ActiveMQConnectionFactory(  ActiveMQConnectionFactory.***DEFAULT\_USER***,  ActiveMQConnectionFactory.***DEFAULT\_PASSWORD***,  ActiveMQConnectionFactory.***DEFAULT\_BROKER\_URL***);  //2.创建链接COnnection 注意连接默认是关闭的，需要start开启  Connection connection = connectionFactory.createConnection();  connection.start();  //3.创建session（上下文环境对象）  //创建session可以配置参数，比如事物，自动签收模式  Session session = connection.createSession(Boolean.***FALSE***,Session.***AUTO\_ACKNOWLEDGE***);  //4.创建Destination，在PTP模式下是queue，在pub/sub模式下是topic  Destination destination = session.createQueue("queue\_test");  //5.创建MessageProducer 发送消息的生产者 / 接受消息的消费者  MessageProducer messageProducer = session.createProducer(destination);  //6.设置持久化方式  messageProducer.setDeliveryMode(DeliveryMode.***NON\_PERSISTENT***);  //7.定义消息对象发送  //SteamMessage Java 原始值得数据流  //MapMessage 一套键值对  //TextMessage 一个字符串对象  //ObjectMessage 一个序列化Java对象  //BytesMessage 一个为基数字节的数据流  TextMessage textMessage = session.createTextMessage();  textMessage.setText("hello world2");  messageProducer.send(textMessage);  //8.释放链接资源  **if**(connection != **null**){  connection.close();  } |

# 接受消息demo

|  |
| --- |
| //1.创建连接工厂ConnectionFactory 需要user，password，url  ConnectionFactory connectionFactory = **new** ActiveMQConnectionFactory(  ActiveMQConnectionFactory.***DEFAULT\_USER***,  ActiveMQConnectionFactory.***DEFAULT\_PASSWORD***,  ActiveMQConnectionFactory.***DEFAULT\_BROKER\_URL***);  //2.创建链接COnnection 注意连接默认是关闭的，需要start开启  Connection connection = connectionFactory.createConnection();  connection.start();  //3.创建session（上下文环境对象）  //创建session可以配置参数，比如事物，自动签收模式  //AUTO\_ACKNOWLEDGE：表示在消费者receive消息的时候自动的签收  //CLIENT\_ACKNOWLEDGE：表示消费者receive消息后必须手动的调用acknowledge()方法进行签收  //DUPS\_OK\_ACKNOWLEDGE：签不签收无所谓了，只要消费者能够容忍重复的消息接受，当然这样会降低Session的开销  //实际开发过程中使用CLIENT\_ACKNOWLEDGE。  Session session = connection.createSession(Boolean.***FALSE***,Session.***AUTO\_ACKNOWLEDGE***);  //4.创建Destination，在PTP模式下是queue，在pub/sub模式下是topic  Destination destination = session.createQueue("queue\_test");  //5.创建MessageProducer 发送消息的生产者 / 接受消息的消费者  MessageConsumer messageConsumer = session.createConsumer(destination);  **while** (**true**) {  // 设置接收者接收消息的时间，为了便于测试，这里谁定为10s  TextMessage message = (TextMessage) messageConsumer.receive(10000);  **if**(**null** != message){  System.***out***.println(message.getText());  }  } |