# ActiveMQ特征

# (http://activemq.apache.org/features.html)

## 消费者特征（ [Consumer Features](http://activemq.apache.org/consumer-features.html)）

### 消费者异步分发（[Consumer Dispatch Async](http://activemq.apache.org/consumer-dispatch-async.html)）

在ActiveMQ 4及其后面的版本，ActiveMQ提供者实行同步或异步分发消息给消费者变得更加可配置。现在它可以通过对每一个消费者的Destination自定义配置数据配置在连接的URL上、Connection和ConnectionFactory而不是先前仅仅只能配置在服务端连接器（transport）上。

这样使得如果你想将消息异步分发给处理慢的消费者，但同步发送给处理更快的消费者（避免同步和添加其他镇静队列（seda queue）上下文切换的开销）更有意义。使用同步消息分发的缺点如果将消息分发给慢消费者则生产者更容易阻塞。

注意，如果ActiveMQ中的消费者采用监听器（MessageListener）来监听接收消息，且设置了同步分发和AUTO\_ACKNOWLEDGE确认模式，则消费者会运行完onMessage(Message message)方法后才给ActiveMQ提供者确认 。

例：通过ConnectionFactory来配置消息的异步分发:

((ActiveMQConnectionFactory)connectionFactory).setDispatchAsync(false);

例：通过Connection来配置消息的异步分发（这里设置会覆盖掉ConnectionFactory中的配置）:

((ActiveMQConnection)connection).setDispatchAsync(false);

例：通过Destination URL来配置消息的异步分发（这里设置会覆盖掉Connection中的配置）:

queue = new ActiveMQQueue("TEST.QUEUE?consumer.dispatchAsync=false");

consumer = session.createConsumer(queue);

### 独占消费者（[Exclusive Consumer](http://activemq.apache.org/exclusive-consumer.html)）

一般情况下，ActiveMQ在队列中会维持消息的顺序排列并按顺序的将它们分发到消费者。但是，如果你有多个JMS会话（Session）并且消费者都从同一个队列消费消息（即使是在不同的JVM上），你将失去按顺序处理消息的担保，因为此时消息被不同的线程并发处理。

有时候保证消息按照顺序被处理是非常重要的，所以我们在J2EE集群环境中通常指定集群中的一个JVM来在队列上只提供一个消费者来避免丢失顺序。但问题是如果这个指定的JVM宕机了，则在队列上再也没有其他的处理过程。

在ActiveMQ 4.X中有一个新的特征叫做独占消费者（Exclusive Consumer）或者独占队列（Exclusive Queues），提供者将挑选一个消费者来按顺序获得一个队列的所有消息，如果这个消费者挂了，则提供者将自动故障切换（failover）来选择其他的消费者来代替它。效果是这样的，J2EE集群环境中的每个JVM都有相同设置和配置；提供者选择一个消费者来作为主消费者，并把所有消息都按照顺序发送给它直到它死；这是你将立即故障转移来选择另一个消费者来作为主消费者。这对于那些挣扎于在J2EE集群环境中怎么做高可用性分布式JMS服务是非常有用的。

例：通过Destination设置独占消费者

queue = new ActiveMQQueue("TEST.QUEUE?consumer.exclusive=true");

consumer = session.createConsumer(queue);

### 消息组（Message Groups ）

为了增强独占消费者（Exclusive Consumer），ActiveMQ提供了如下几个特性：

1. 在一个队列中保证相关消息的处理顺序。
2. 多个消费者处理消息的负载均衡。
3. 如果一个JVM故障，高可用性/自动故障转移（auto-failover）到其他消费者。

所以消息组（Message Groups）逻辑上有挺像一个平行的独占消费者。标志的JMS消息头中的JMSXGroupID就是被用来定义消息属于哪个消息组。消息组的特性能保证所有组里的消息将被发送到相同的消费者，即使该消费者死了，另外一个消费者将被选择。

### 消息分发

那么消息是怎么样分配给消费者的呢？假设一个队列queue1（队列上设置预读为1（如ConsumerReceiveTestQueue?consumer.prefetchSize=1））下有5个消费者c1， c2，c3，c4和c5。不同的是c1，c2，c3，c4都由session1创建，c5由session2创建，session1和session2可以不在一个JVM中。

假设这5个消费者都采用异步接收消息，即都采用setMessageListener来设置消息监听器来接收消息（这时你的队列设置的预读prefetchSize必须大于等于1，否则会报错）。如果queue1中有500条消息，则你会发现c5可以接收到将近250条消息，而其他的250条消息会被分配给session1下的4个消费者（不一定是平均分配）。

假如这5个消费者都采用同步接收消息，即都采用receive()来接收消息，则你可以发现，即使这5个消费者不属于同一个Session，他们接收到的消息数量几乎都是500、5=100条。

所以，最高效的创建异步接收消息的消费者是给每个Session都只创建一个消费者，也就是说，如果你需要5个消费者并行的处理消息，则你需要通过连接来创建5个Session，然后给每个Session创建一个消费者，而不是所有消费者都在同一个Session下创建。可能因为Session是单线程的原因。如果所有消费者都由同一个Session来创建，则会有如下的性能问题：假设session下面有3个消费者c1，c2和c3，这三个消费者都采用异步接收消息，并且都在不同的线程中接收，则刚开始预期的效果并不是这3个消费者都各自接收到一条消息，而是c1接收到一条消息并且消费完应答MQ服务器后，服务器才会分一条消息到这个session，这个session再把消息分给c2，等c2消费完应答后session再把新的消息分配给c3。所以，这样的话我干嘛还要创建三个消费者呢？还不如创建一个消费者，这就是Session单线程的一个弊端。不过如果一个Session被多个线程同时使用，会增加消息服务器的开发难度。

这也是为什么当我们在Spring中使用ActiveMQ时，在创建连接工厂Bean对象（比如org.apache.activemq.spring.ActiveMQConnectionFactory）时，需要把Session异步设置为true的原因（<property name="alwaysSessionAsync" value="true" />）。