# ActiveMQ--5.监控

# 介绍下ActiveMQ都需要监控的内容

**监控硬件物理空间是否充足：**

ActiveMQ有3个重要的参数，存储空间百分比，内存空间百分比和临时空间百分比。这三个参数的意义很明显，如果值到了100，则表明硬件空间已满，Broker不能再接受任何的消息了，除非有消息消费并且删除，Broker才可以再接收消息。

如果这些值长时间都比较高，接近阀值，则表示硬件的配置不能满足要求，建议更换硬件，或者给予ActiveMQ的环境配置比较小，建议给予更多的资源给ActiveMQ。

如果平时保持在一个稳定值，有一段时间突然增高，则有两种可能。一种可能是用户量大增（当然大家都希望是这样），另一种可能是某个或者某几个消息消费者死机了。需要人工干预重启。

还有一种可能是平时保持在一个稳定值，但是一段时间内突然降低了，则表示消息的生产者可能出现问题了。

**监控队列：**

如果ActiveMQ使用队列，需要监控队列的未消费消息数量，消费者数量，消息入队和出队的数量。

未消费消息数量的含义和监控硬件的内存和硬盘空间差不多，过多的消息堆积可能是有消息消费者死机。

消费者数量应该是一个相对固定的值，这个值减少，就直接表示有的消费者已经死机。

 消息入队和出队数量如果增长缓慢或者不增长，则表示消息发送者已经死机。一般是通过增量监控。

**监控主题：**

如果ActiveMQ使用主题，需要监控的内容和队列相似，但是没有未消费的消息数量。这里指的注意的是，ActiveMQ提供Advisory Message，用于帮助使用统计一些额外信息，详细情况在后面介绍。

**监控订阅者：**

同样，如果ActiveMQ使用主题，订阅者的信息就十分重要。需要监控已经下线的订阅者。

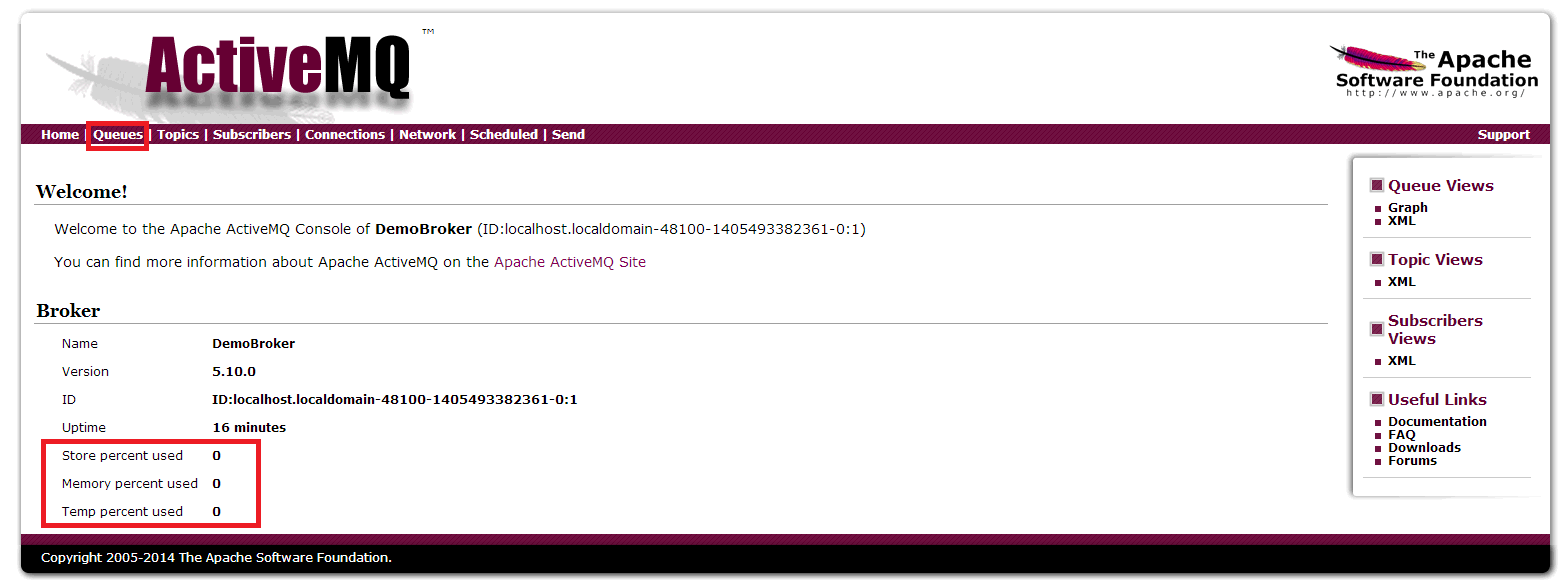
介绍完需要监控的内容，下面介绍一下ActiveMQ提供的几种监控方法。

**Web Console监控：**

之所以先介绍这个，是因为Web Console是最简单的，不需要任何配置就能使用的监控方式。同时也是最直观的监控方式。

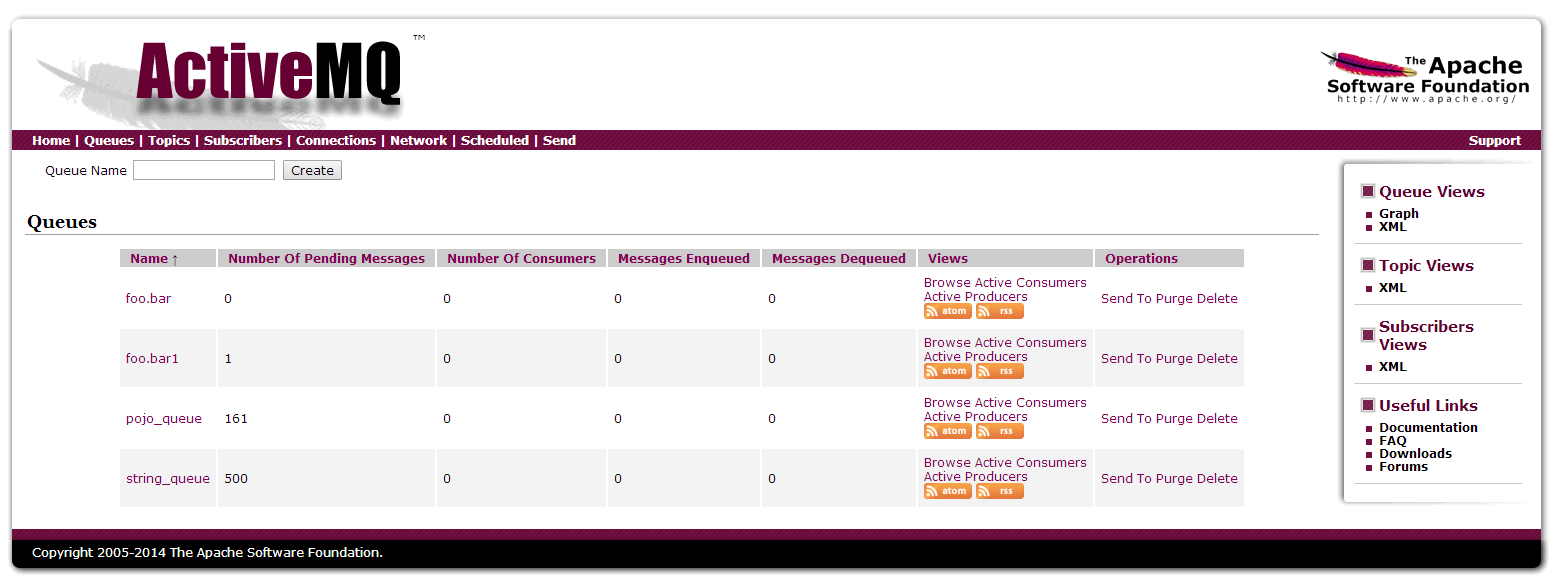
直接在浏览器访问http://activemq-host:8161/admin/。把localhost换成你activemq启动机器的ip，端口号默认是8161，可以在${ACTIVEMQ\_HOME}/conf/jetty.xml中修改。默认的用户名和密码都是admin。

进入页面首先看到



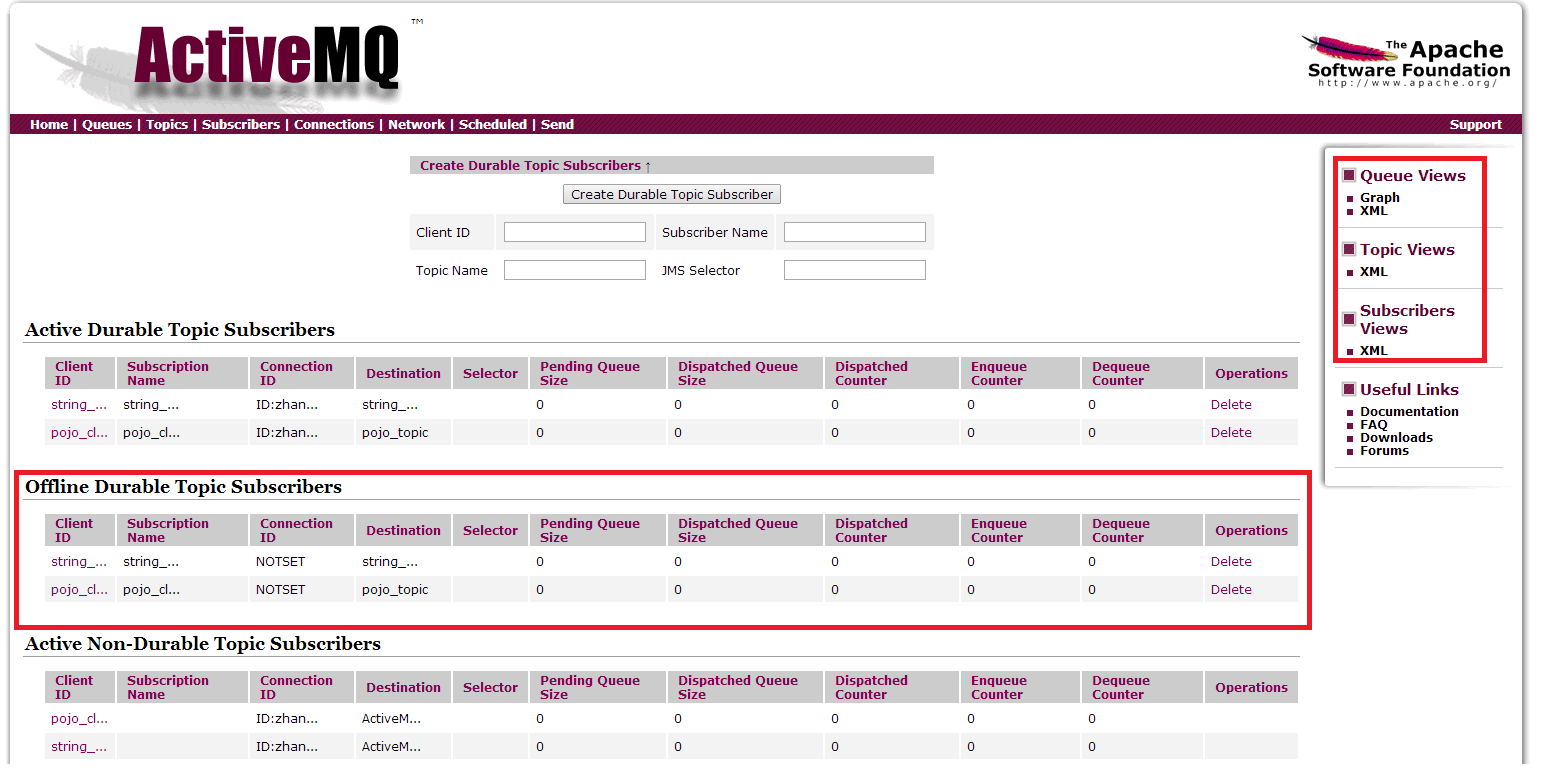
其中下面的红色框中标示出的就是硬件物理空间。

然后点击上面红色框可以进入队列的监控页面。



可以看到上面所提到的队列的一些重要数据。点击傍边的Topics可以进入主题监控页面，大致差不多，就不截图了。

下面看一下订阅者监控界面。



需要重点关注的是中间红色框圈住的部分，是离线的订阅者。

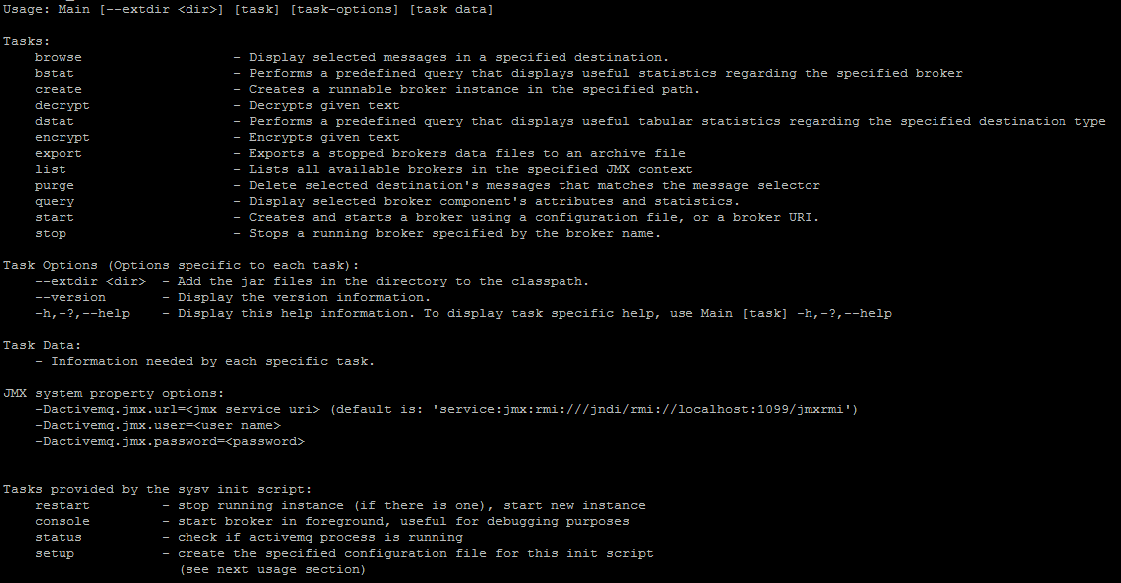
这样的页面一目了然，非常适合人类观看。但是缺点是，不适合程序直接抓数据，写监控脚本，自动报警等。这时你一定发现了这张图右边红色框部分。分别提供了队列，主题和订阅者的xml数据，下图是队列的xml数据。



这样就便于程序去访问这些数据。但是缺点是数据不够全，且xml也不如json容易解析，用shell脚本解析的话比较麻烦。shell解析的话，命令行的方式才是最舒服的。

**命令行监控：**

ActiveMQ当然也提供了命令行监控的方式。执行${ACTIVEMQ\_HOME}/bin/activemq，会列出命令行的使用帮助。请注意虽然Windows平台也可以使用这个命令行，但是仅仅限于启动ActiveMQ实例，其他命令都不能很好的支持，所以这里所说的命令行方式，都是在Linux平台上的。



限于篇幅，这里只是截取了重要的部分。可以看到命令行提供了16个命令，包括start，stop，restart，console等启动停止实例相关的命令。还有一些控制ActiveMQ Broker的命令，如create，purge等。这里主要介绍几个监控可能用得到的命令

activemq status，显示当前的ActiveMQ是否运行正常，并且能显示pid。



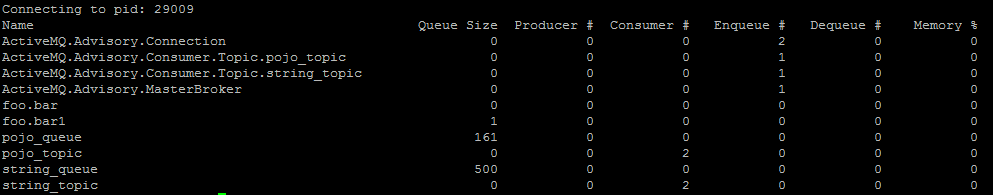
activemq list，列出当前Broker名字。

activemq bstat，显示当前Broker的统计信息。

activemq query，根据筛选条件显示Broker的统计信息。如：activemq query -QQueue=string\_queue，是只显示string\_queue这个队列的统计信息。更多使用方法可以使用activemq query --help显示帮助。

activemq browse，可以查询当前Broker为被消费的消息，会显示消息的详细信息，如消息头，消息内容，优先级等。类似于数据库的查询功能。有自己的一套查询语法但是不是很复杂，同样可以使用activemq browse --help显示帮助。需要注意的是，只有Queue才可以查询，Topic是不可以的，所以这个功能虽然强大，但是有点鸡肋。

activemq dstat，比较有用的一个功能，可以用来查询队列的关键数值，如队列大小，生产者消费者数量，消息出队入队统计等。还可以支持通过类别查询，如只查询队列或者只查询主题。



个人认为，activemq status和activemq dstat是比较常用的两个监控命令，可以使用shell，然后grep/awk解析。命令行虽然方便，但是明显缺失必要的信息，如硬件使用百分比，订阅者下线信息等。至少目前不能完成监控的全部工作，希望以后可以持续完善功能。

**JMX监控：**

由Java开发的程序，一般都是支持JMX监控的，ActiveMQ也不例外。JMX监控是最全的，任何细节都可以通过JMX获取。如果远程连接JMX监控需要一些额外的配置。本机直接连接不需要，但是由于ActiveMQ是部署在linux上，所以应该很少有人会直接连接本地的JMX，除非是本地开发调试阶段。

1. 修改/etc/hosts 检查hosts文件设置，不用127.0.0.1，用实际IP地址。

2. 修改${ACTIVEMQ\_HOME}/bin/active文件。找到下面几行，是连续的。应该都是注释掉的，解除注释，开启远程连接JMX，端口号默认是11099。

[?](http://my.oschina.net/u/719192/blog/291657)

|  |  |
| --- | --- |
| 1  2  3  4  5 | ACTIVEMQ\_SUNJMX\_START= "-Dcom.sun.management.jmxremote.port=11099 "  ACTIVEMQ\_SUNJMX\_START= "$ACTIVEMQ\_SUNJMX\_START -Dcom.sun.management.jmxremote.password.file=${ACTIVEMQ\_CONF}/jmx.password"  ACTIVEMQ\_SUNJMX\_START= "$ACTIVEMQ\_SUNJMX\_START -Dcom.sun.management.jmxremote.access.file=${ACTIVEMQ\_CONF}/jmx.access"  ACTIVEMQ\_SUNJMX\_START= "$ACTIVEMQ\_SUNJMX\_START -Dcom.sun.management.jmxremote.ssl=false"  ACTIVEMQ\_SUNJMX\_START= "$ACTIVEMQ\_SUNJMX\_START -Dcom.sun.management.jmxremote" |

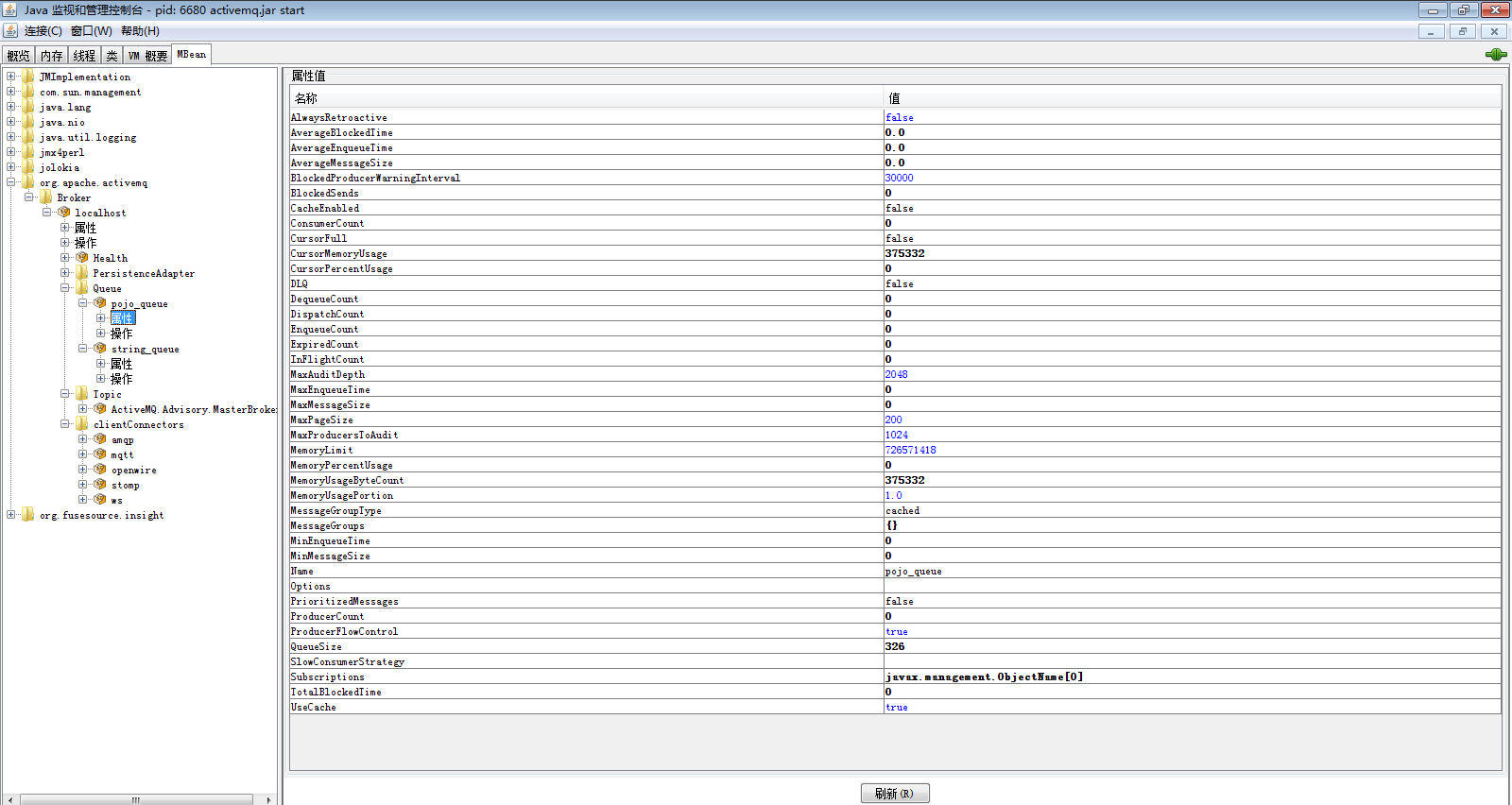
3. 修改JMX用户和密码文件，必须是当前用户只读，别的用户无权限。否则启动时会报异常。

[?](http://my.oschina.net/u/719192/blog/291657)

|  |  |
| --- | --- |
| 1 | chmod  400 ${ACTIVEMQ\_HOME} /conf/jmx .\* |

 其中jmx.access是用户权限配置文件，jmx.password是用户密码配置文件。

配置到此，就可以使用jconsole等客户端连接了。启动jconsle，选择远程进程，填写remotehost:11099。其中remotehost是ActiveMQ所在机器的ip地址，11099是配置文件中开启的监听端口号。默认用户名是admin,密码是activemq。默认用户名和密码可以在jmx.access和jmx.password修改。连接之后如下图所示：



可以看到，从左侧树形菜单可以找到ActiveMQ的mbean，然后可以查看任何公开的JMX属性。

如果你配置完了前3步，仍然不能远程连接JMX，请继续下面的配置。

4. 查看${ACTIVEMQ\_HOME}/conf/activemq.xml中的broker节点中useJmx="true"属性。这个属性可以没有或者为true，但是不可以是false。

5. 修改${ACTIVEMQ\_HOME}/conf/activemq.xml中的broker节点中找到managementContext节点，修改如下：

[?](http://my.oschina.net/u/719192/blog/291657)

|  |  |
| --- | --- |
| 1  2  3 | < managementContext >       < managementContext  createConnector = "true"  connectorPort = "11099"  />  </ managementContext > |

 其中connectorPort是你想发布的JMX端口号。注意端口号必须和之前${ACTIVEMQ\_HOME}/bin/activemq里发布的端口号保持一致。

JMX可以使用java开发程序，进行监测。但是对其他语言的支持有限。读到这里，你是不是觉得，每种监控方式各有优缺点，但是很难找出一个都很完美的方式，既可以看到全部的信息，又可以轻松的用程序实现监控。如果你有这方面的需求，请你继续看下一个监控方式。

**JMX REST API：**

可能你还记得，之前提到的Web Console也有xml的数据展现形式，但是解析困难且数据不全。而命令行模式数据更加不全。所以，ActiveMQ提供把JMX导出为REST API。这样就最大限度的满足了各种需求，既有完整的数据，又可以方便编写程序监控。当然如果以后命令行的方式能提供更多的信息，那么这个JMX REST API就可以被替代了。但是至少现在来说，他是不可替代的。

ActiveMQ使用一个开源的组件Jolokia来将JMX自动发布成为REST API。如果使用这个方式，你不需要配置那些上一节提到的繁琐远程JMX配置。他的原理是使用本地的JMX，然后通过HTTP发布出来，并不是远程访问JMX。

在浏览器访问http://activemq-host:8161/api/jolokia/。默认的用户名和密码同样都是admin。可以看到浏览器会返回json信息来描述当前系统。

[?](http://my.oschina.net/u/719192/blog/291657)

|  |  |
| --- | --- |
| 1 | { "request" :{ "type" : "version" }, "value" :{ "agent" : "1.2.1" , "protocol" : "7.1" , "config" :{ "discoveryEnabled" : "false" , "agentId" : "xxx.xxx.xxx.xxx-29009-71a9b4c7-servlet" , "agentType" : "servlet" , "agentDescription" : "Apache ActiveMQ" }, "info" :{ "product" : "activemq" , "vendor" : "Apache" , "version" : "5.10.0" }}, "timestamp" :1405499528, "status" :200} |

通过Jolokia API可以访问JMX发布的任何mbean。

<http://activemq-host:8161/api/jolokia/list>  显示当前所有的mbean。

<http://activemq-host:8161/api/jolokia/read/><mbean name >/<attribute name> 显示某个mbean的某个属性

还有很多其他的操作，如<http://activemq-host:8161/api/jolokia/write/><mbean name >/<attribute name>可以动态执行方法。由于本篇文章只涉及监控，所以其他的操作先不深入。

我们重点关注http://activemq-host:8161/api/jolokia/read/。

先说一下Mbean名字的规则。一般的MBean名字类似于：org.apache.[activemq:type=Broker,brokerName=DemoBroker](http://activemqtype=Broker,brokerName=DemoBroker)

org.apache.activemq:是ActiveMQ在这个Mbean的Domain名，可以修改managementContext的jmxDomainName中配置名称。

[?](http://my.oschina.net/u/719192/blog/291657)

|  |  |
| --- | --- |
| 1  2  3 | < managementContext >       < managementContext  createConnector = "true"  connectorPort = "11099"  jmxDomainName = "MyBroker"  />  </ managementContext > |

 type=Broker是固定的。

brokerName=DemoBroker，其中DemoBroker是在配置文件中Broker所配置的名字。

后面会有一些自定义属性。如：destinationType=Queue,destinationName=pojo\_queue。

看到这里你可能会比较迷惑，下面给出例子，请慢慢体会。

<http://10.2>activemq-host:8161/api/jolokia/read/org.apache.[activemq:type=Broker,brokerName=DemoBroker/TotalEnqueueCount](http://activemqtype=Broker,brokerName=DemoBroker)

获取入队的消息总数，返回结果如下：

[?](http://my.oschina.net/u/719192/blog/291657)

|  |  |
| --- | --- |
| 1 | { "request" :{ "mbean" : "org.apache.[activemq:brokerName=DemoBroker,type=Broker](http://activemqbrokerName=DemoBroker,type=Broker)" , "attribute" : "TotalEnqueueCount" , "type" : "read" }, "value" :5672, "timestamp" :1405501289, "status" :200} |

关键的数据是value的值5672，表示有5672个消息入队。

如果使用shell编写监控脚本，可以使用命令：

[?](http://my.oschina.net/u/719192/blog/291657)

|  |  |
| --- | --- |
| 1 | wget --user admin --password admin --auth-no-challenge http: //activemq-host :8161 /api/jolokia/read/org .apache.activemq: type =Broker,brokerName=localhost /TotalEnqueueCount |

下面列出几个关键的监控属性。

<base\_url>/read/org.apache.[activemq:type=Broker,brokerName=localhost,service=Health/CurrentStatus](http://activemqtype=Broker,brokerName=localhost,service=Health)

获取当前系统状态

<base\_url>/read/org.apache.[activemq:type=Broker,brokerName=](http://activemqtype=Broker,brokerName=)localhost/MemoryPercentUsage

获取内存使用量

<base\_url>/read/org.apache.[activemq:type=Broker,brokerName=](http://activemqtype=Broker,brokerName=)localhost/StorePercentUsage

获取硬盘空间使用量

<base\_url>/read/org.apache.[activemq:type=Broker,brokerName=](http://activemqtype=Broker,brokerName=)localhost/TempPercentUsage

获取临时文件硬盘空间使用量

<base\_url>/read/org.apache.[activemq:type=Broker,brokerName=](http://activemqtype=Broker,brokerName=)localhost,destinationType=Queue,destinationName=pojo\_queue/QueueSize

获取队列pojo\_queue的未消费消息数量

<base\_url>/read/org.apache.[activemq:type=Broker,brokerName=localhost,destinationType=Queue,destinationName=pojo\_queue/ProducerCount](http://activemqtype=Broker,brokerName=localhost,destinationType=Queue,destinationName=pojo_queue)

获取队列pojo\_queue的生产者数量

<base\_url>/read/org.apache.[activemq:type=Broker,brokerName=localhost,destinationType=Queue,destinationName=pojo\_queue/ConsumerCount](http://activemqtype=Broker,brokerName=localhost,destinationType=Queue,destinationName=pojo_queue)

获取队列pojo\_queue的消费者数量

<base\_url>/read/org.apache.[activemq:type=Broker,brokerName=localhost,destinationType=Queue,destinationName=pojo\_queue/EnqueueCount](http://activemqtype=Broker,brokerName=localhost,destinationType=Queue,destinationName=pojo_queue)

获取队列pojo\_queue的已消费消息数量

<base\_url>/read/org.apache.[activemq:type=Broker,brokerName=localhost,destinationType=Queue,destinationName=pojo\_queue/DequeueCount](http://activemqtype=Broker,brokerName=localhost,destinationType=Queue,destinationName=pojo_queue)

获取队列pojo\_queue的已接收消息数量

<base\_url>/read/org.apache.[activemq:type=Broker,brokerName=](http://activemqtype=Broker,brokerName=)localhost,destinationType=Topic,destinationName=pojo\_topic,endpoint=Consumer,clientId=pojo\_client\_1,consumerId=Durable(pojo\_client\_1\_pojo\_client\_1)/Active

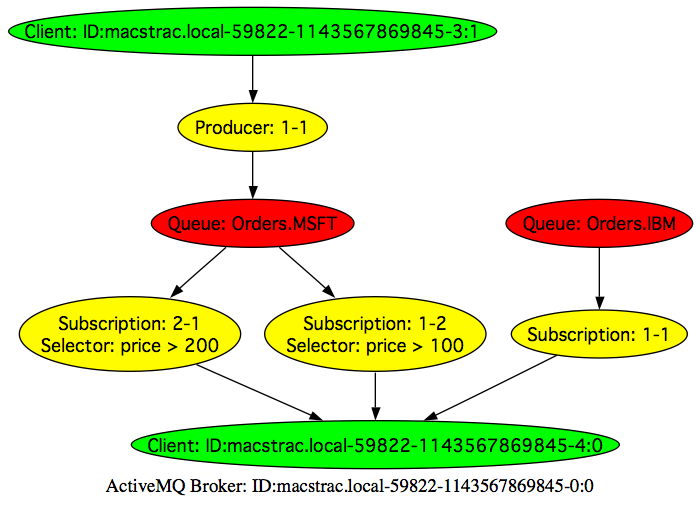
获取主题pojo\_topic的client id为pojo\_client\_1的订阅者是否已经下线

**Advisory Message监控：**

ActiveMQ提供了一些额外的信息帮助用户获取系统状态，这些额外信息都存在主题中，以**ActiveMQ.Advisory.**开头。这些信息可以通过前面说的任何一种方式查询。这些消息包括Queue/Topic创建销毁的次数，消费比较慢的消费者统计等等。可以参考[官方文档](http://activemq.apache.org/advisory-message.html)查询每个Advisory Message的意义。

**插件监控：**

ActiveMQ还提供了一些额外的插件来监控系统，主要有可视化插件监控和统计插件监控，需要在activemq.xml中进行配置，最主要的有<connectionDotFilePlugin/>，<destinationDotFilePlugin/>和<statisticsBrokerPlugin/>。对于一些生成图标的监控系统比较有用。下面是connectionDotFilePlugin的使用截图。



**第三方监控：**

最后一种监控方式就是第三方提供的监控，这种监控主要包括ActiveMQ Monitor，Apache ActiveMQBrowser ，HermesJMS等。由于是第三方提供的监控平台，所以一旦activemq升级，监控平台不升级，可能会带来兼容问题，所以请谨慎使用。

另外不得不提的是hawtio，也是一个第三方提供的监控平台，提供了HTML5的绚丽界面。已经在ActiveMQ 5.9的版本中内嵌入发布包。但是由于Apache内部的意见分歧，在ActiveMQ 5.9.1的版本中又将其删除。如果想用，只能自己下载hawtio部署到ActiveMQ的环境中。