# MQ消息队列（中间件）

# 消息队列概述

消息队列是分布式系统中重要的组件，主要解决应用解耦，异步消息，流量削锋等问题，从而实现高性能，高可用，可伸缩和最终一致性架构。

目前消息队列有ActiveMQ、RabbitMQ、ZeroMQ、Kafka、MetaMQ、RocketMQ

# 消息队列应用场景

异步处理、应用解耦、流量削锋和消息通讯四个场景。

## 异步处理

**场景说明：**用户注册后，需要发注册邮件和注册短信。

* **串行方式：**

将注册信息写入数据库成功后，发送注册邮件，再发送注册短信。以上三个任务全部完成后，返回给客户端。



* **并行方式：**

将注册信息写入数据库成功后，发送注册邮件的同时，发送注册短信。以上三个任务完成后，返回给客户端。



* **引入消息队列：**

将不是必须的业务逻辑，异步处理。



## 应用解耦

**场景说明：**用户下单后，订单系统需要通知库存系统。

* **传统的做法：**

订单系统调用库存系统的接口



**缺点：**

假如库存系统无法访问，则订单减库存将失败，从而导致订单失败；

订单系统与库存系统耦合。

* **引入消息队列：**



**订单系统：**用户下单后，订单系统完成持久化处理，将消息写入消息队列，返回用户订单下单成功。

**库存系统：**订阅下单的消息，采用拉/推的方式，获取下单信息，库存系统根据下单信息，进行库存操作。

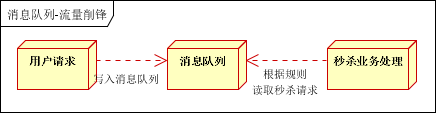
**例如：**下单时库存系统不能正常使用，但不影响正常下单，因为下单后，订单系统将订单写入消息队列就不再关心其他的后续操作。实现订单系统与库存系统的应用解耦。

## 流量削锋

**应用场景：**秒杀活动，流量过大，导致应用挂掉。解决这个问题，一般需要在应用前端加入消息队列。

* **消息队列：**

用户的请求，服务器接收后，首先写入消息队列（假如消息队列长度超过最大数量，则直接抛弃用户请求或跳转到错误页面）。秒杀业务根据消息队列中的请求信息，再做后续处理。



**好处：**

可以控制活动的人数；

可以缓解短时间内高流量压垮应用。

## 日志处理

日志处理是指将消息队列用在日志处理中，比如Kafka的应用，解决大量日志传输的问题。

* **消息队列：**

日志采集客户端，负责日志数据采集，定时写受写入Kafka队列；Kafka消息队列，负责日志数据的接收，存储和转发；日志处理应用：订阅并消费kafka队列中的日志数据。

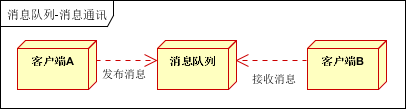


## 消息通讯

消息队列一般都内置了高效的通信机制，因此可以用在的消息通讯。比如，点对点消息队列、聊天室等。

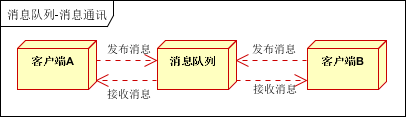
* **点对点通讯：**

客户端A和客户端B使用同一队列，进行消息通讯。



* **聊天室通讯：**

客户端A，客户端B，客户端N订阅同一主题，进行消息发布和接收。实现类似聊天室效果。

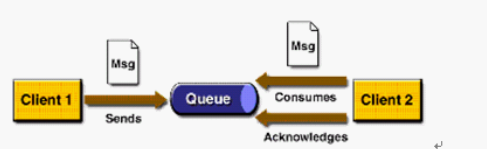


# JMS消息服务（Java Message Service）

## 消息模型

### P2P模式（Point to Point）点对点模式

P2P包含三个角色：消息队列（Queue），发送者(Sender)，接收者(Receiver)。每个消息都被发送到一个特定的队列，接收者从队列中获取消息。队列保留着消息，直到他们被消费或超时。



* **P2P的特点**

每个消息只有一个消费者（Consumer），一旦被消费，消息就不在消息队列中；

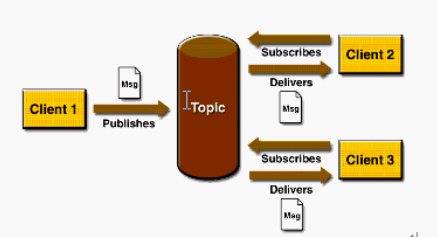
发送者和接收者之间没有时间依赖性，当发送者发送消息之后，不管接收者有没有运行，不会影响到消息被发送到队列；

接收者在成功接收消息之后需向队列应答成功 ；

**如果希望发送的每个消息都会被成功处理，使用P2P模式。**

### Pub/Sub模式（Publish/Subscribe）发布订阅模式

Pub/Sub模式包含三个角色：主题（Topic），发布者（Publisher），订阅者（Subscriber）。多个发布者将消息发送到Topic，系统将这些消息传递给多个订阅者。



* **Pub/Sub的特点**

每个消息可以有多个消费者；

发布者和订阅者之间有时间依赖性。针对某个主题（Topic）的订阅者，必须创建一个订阅者之后，才能消费发布者的消息；

为了消费消息，订阅者必须保持运行的状态。

**为了缓和严格的时间的相关性，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），也能接收到发布者的消息。**

**如果希望发送的消息可以不被做任何处理、或只被一个消息者处理、或者可以被多个消费者处理，可以采用Pub/Sub模式。**

## 消息消费

JMS是异步通信，发送方（生产者）发送消息后可以继续其它业务，不用阻塞等待接收方（消费者）响应。但接收方（消费者）在接收消息上有两种模式：同步接收消息和异步接收消息

* **同步**

订阅者（接收者）通过receive方法来接收消息，receive方法在接收到消息之前（或超时之前）将一直阻塞；

* **异步**

订阅者（接收者）注册一个消息监听器。当消息到达之后，系统自动调用监听器的onMessage方法。

* **其他（不理解 20190702）**

**JNDI：**Java命名和目录接口,是一种标准的Java命名系统接口。可以在网络上查找和访问服务。通过指定一个资源名称，该名称对应于数据库或命名服务中的一个记录，同时返回资源连接建立所必须的信息。

JNDI在JMS中起到查找和访问发送目标或消息来源的作用。

# ActiveMQ的使用（bootdo）

## 传统方式：

### 消息发送者（P2P模式）





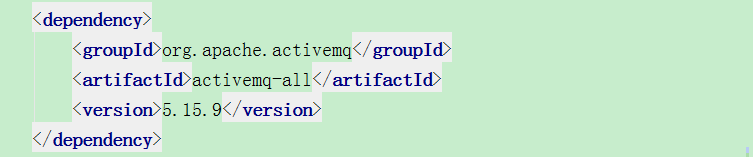
### 消息接收者（P2P模式）





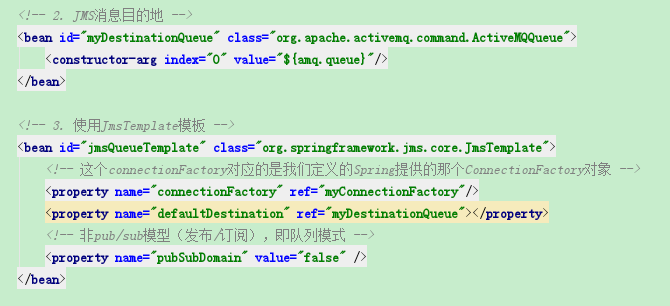
## Spring XML配置方式：

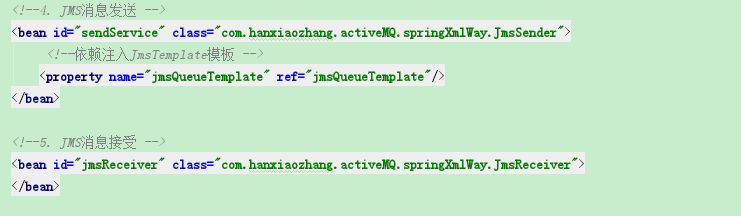
### 引入架包：

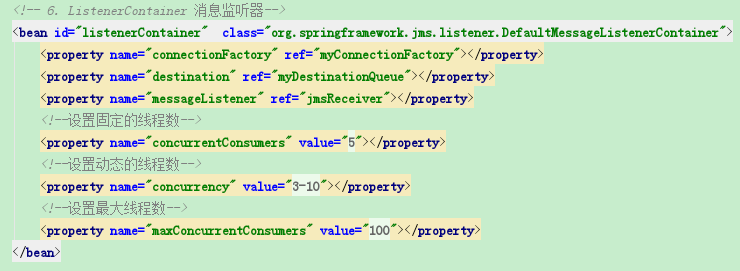


### 配置XML：

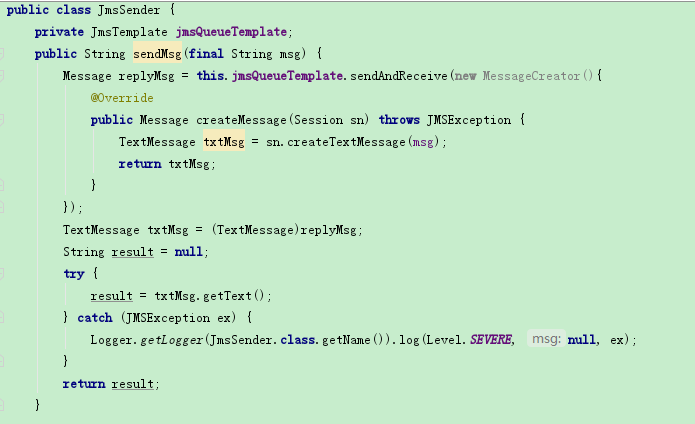




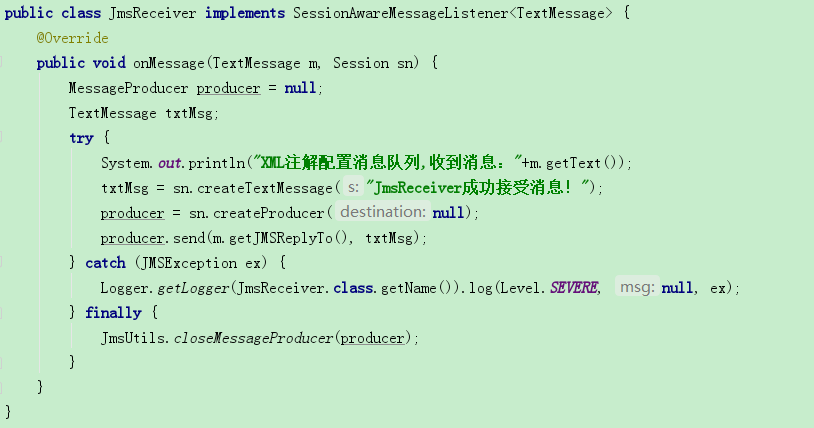




### 消息发送者（P2P模式）：

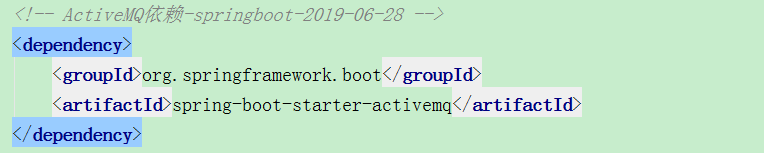


### 消息发送者（P2P模式）：

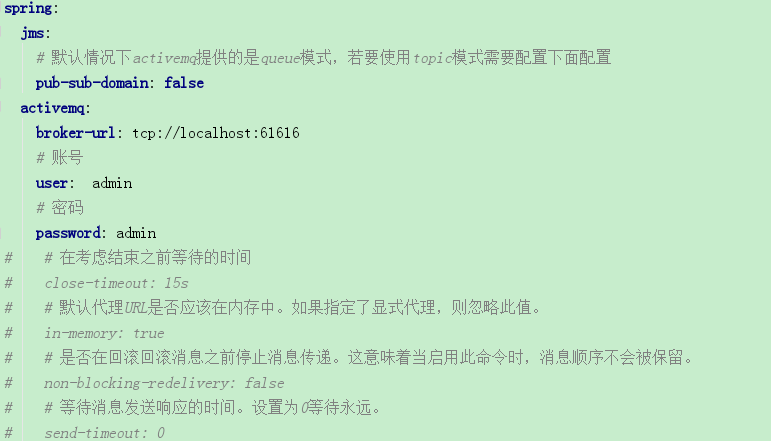


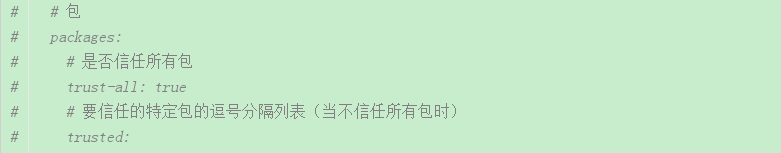
## Springboot 配置方式：

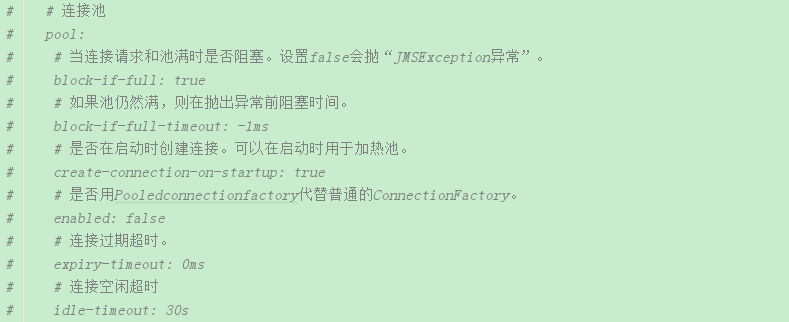
### 引入架包：

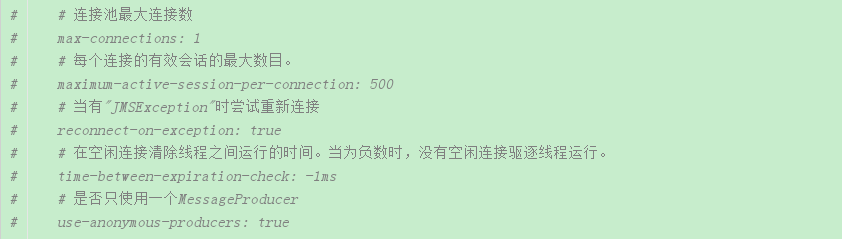


### 配置YML：



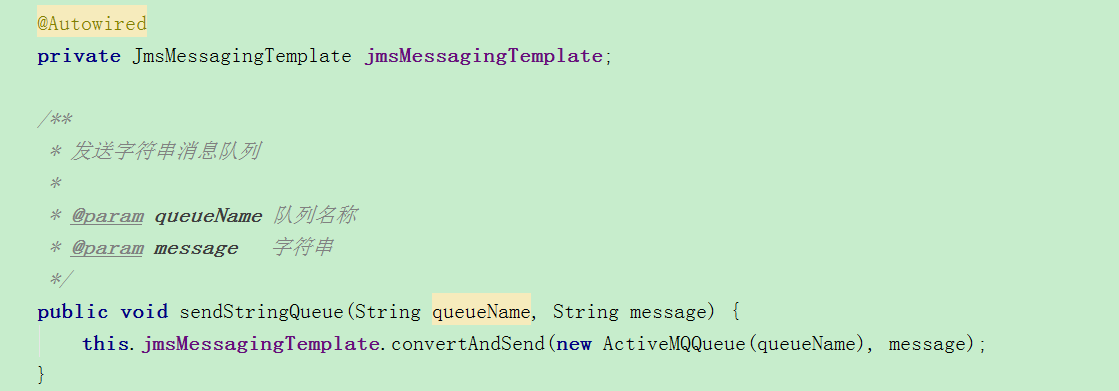


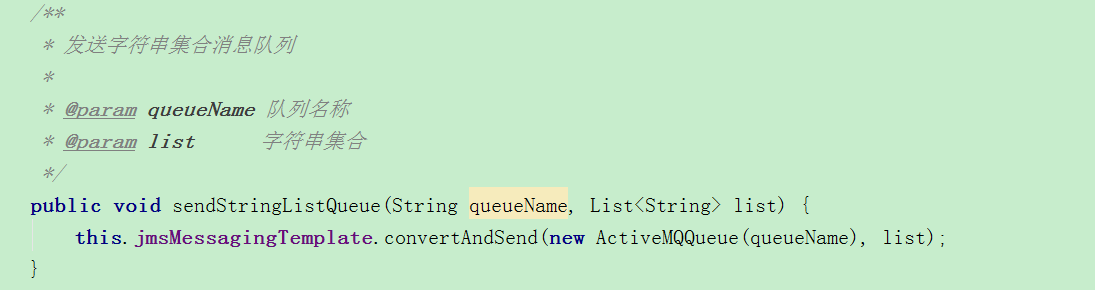




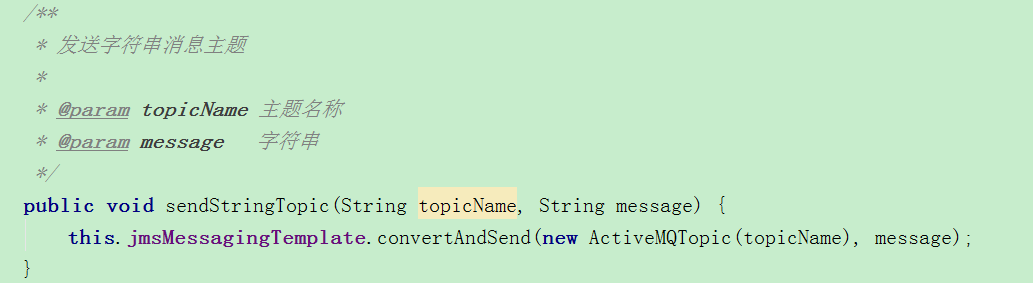
### 消息发送者：

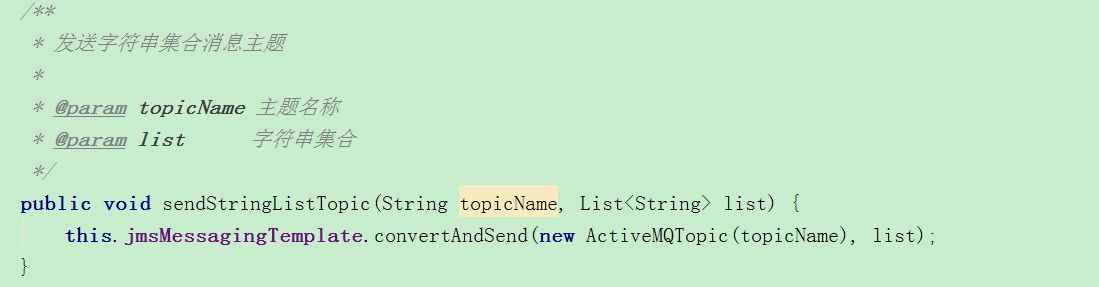
* P2P模式：



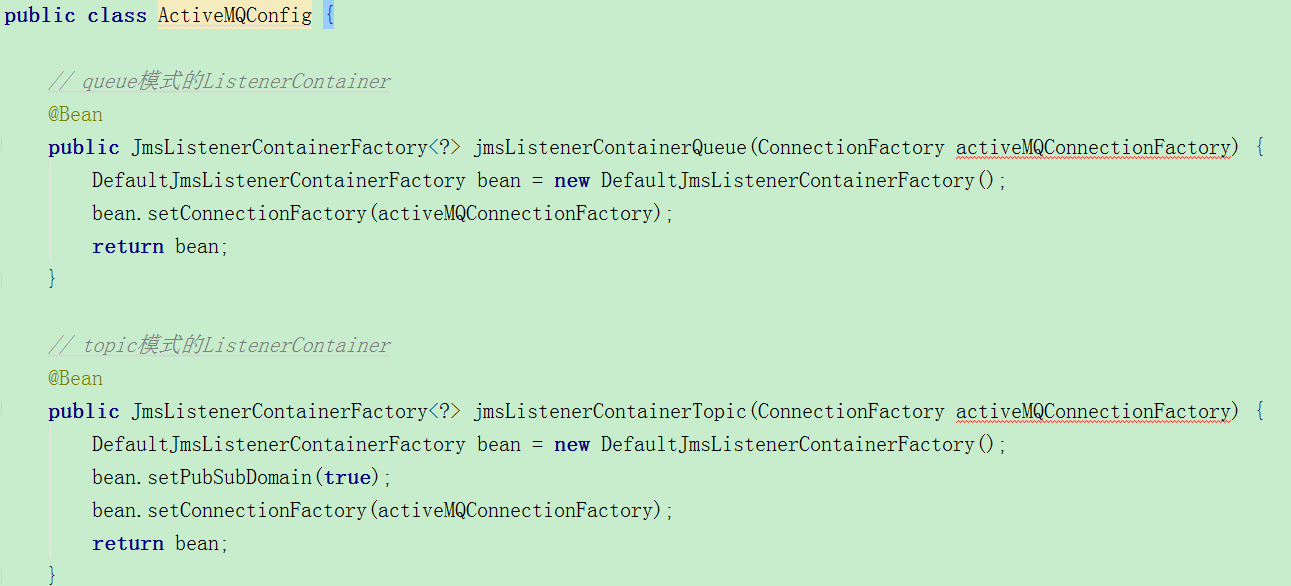


* Pub/Sub模式：



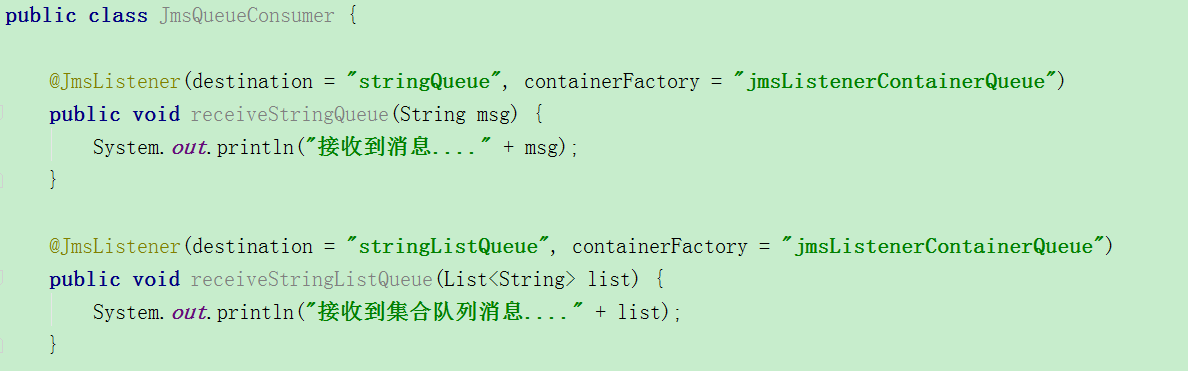


### 配置监听器：

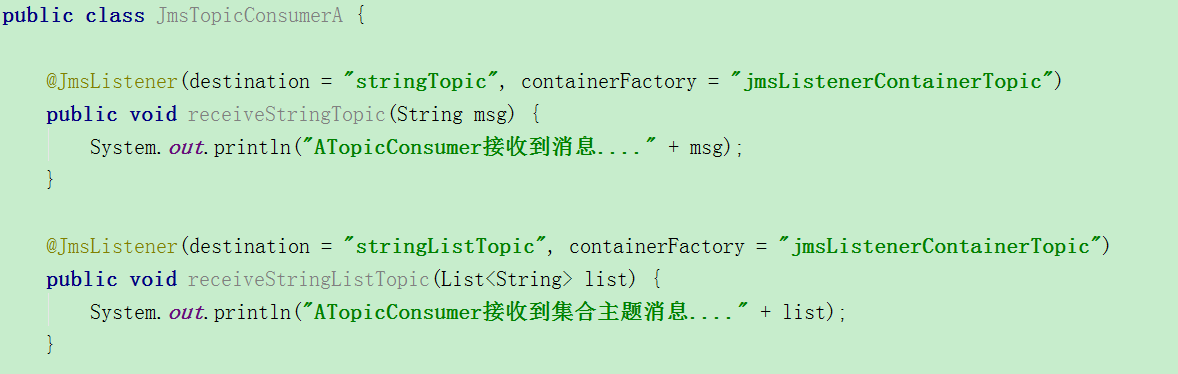


### 消息接收者：

* P2P模式：

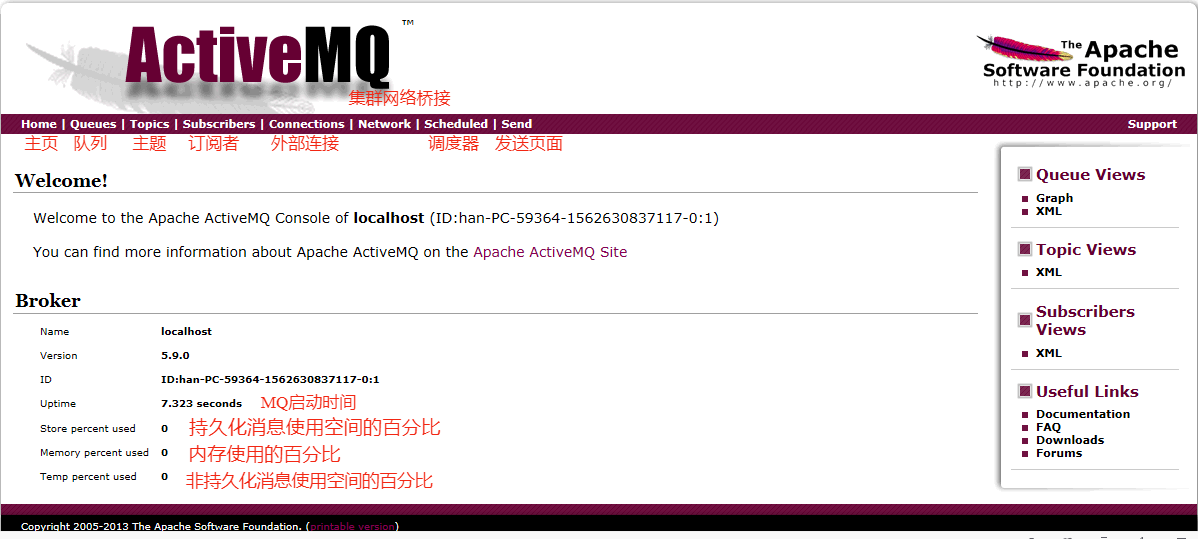


* Pub/Sub模式：



# ActiveMQ管理界面

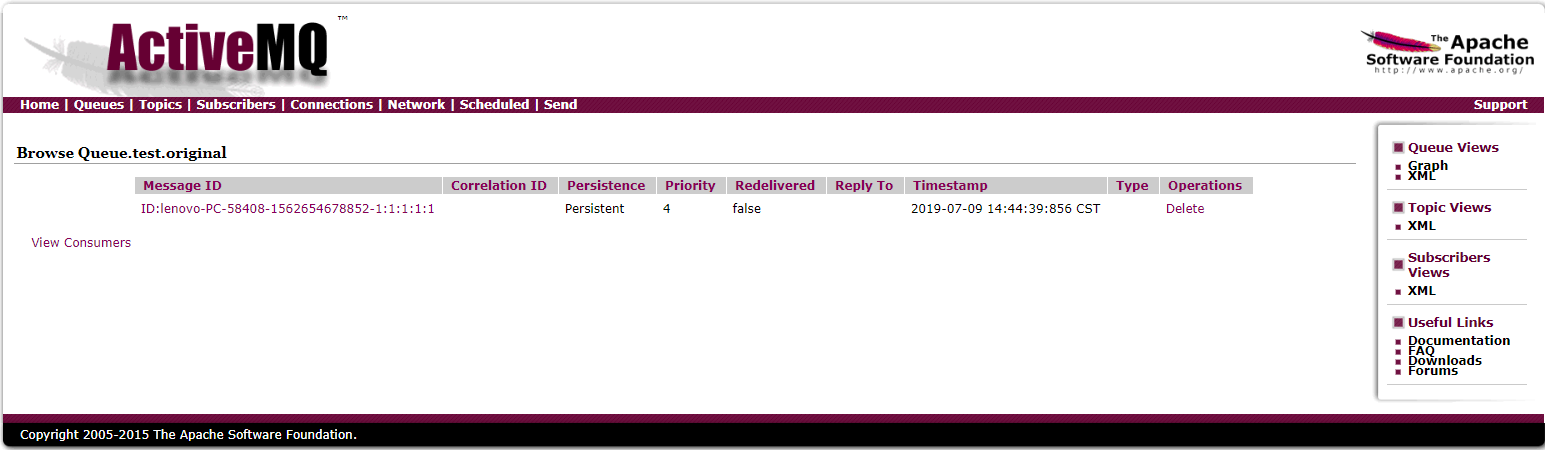
## Home



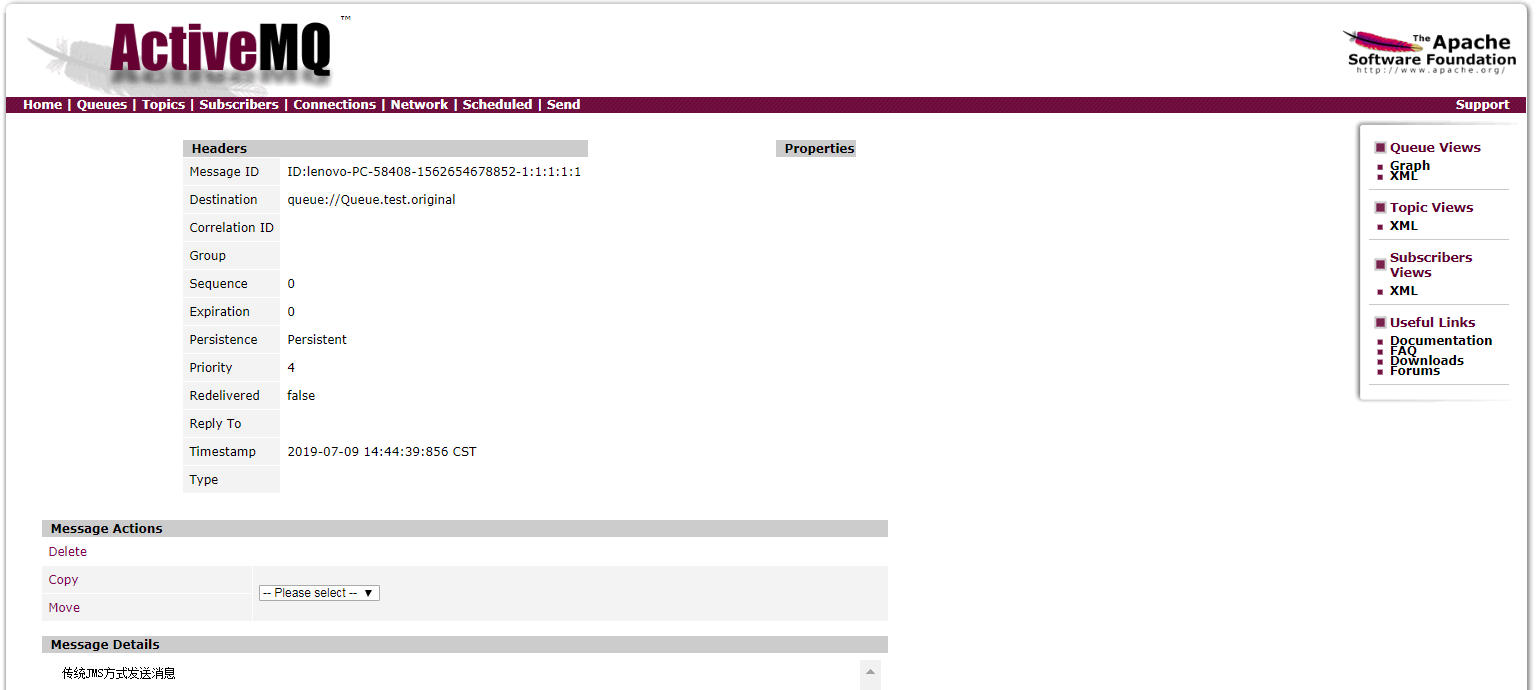
## Queues



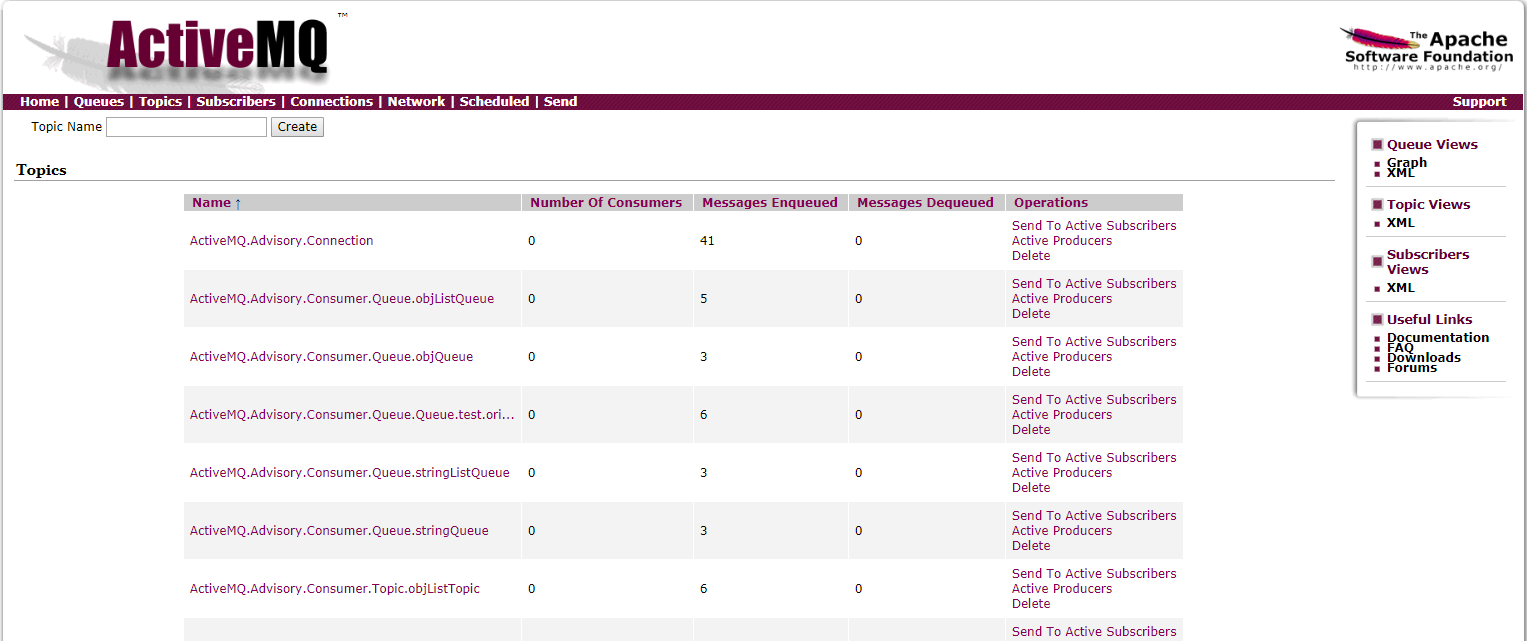
* 点击Browse：



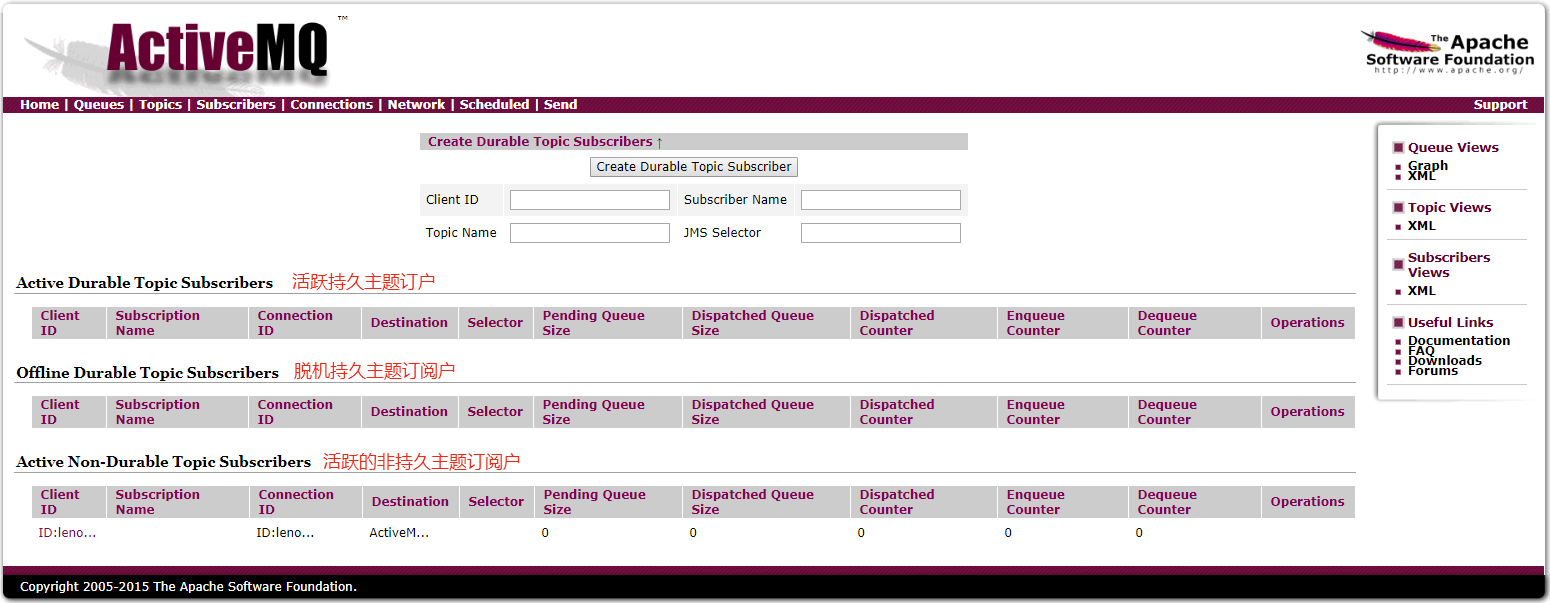
* 点击Message ID ：



## Topics



## SuBseribers



## Connections（连接）

### 作用：

约定ActiveMQ客户端与服务端之间的通信协议；

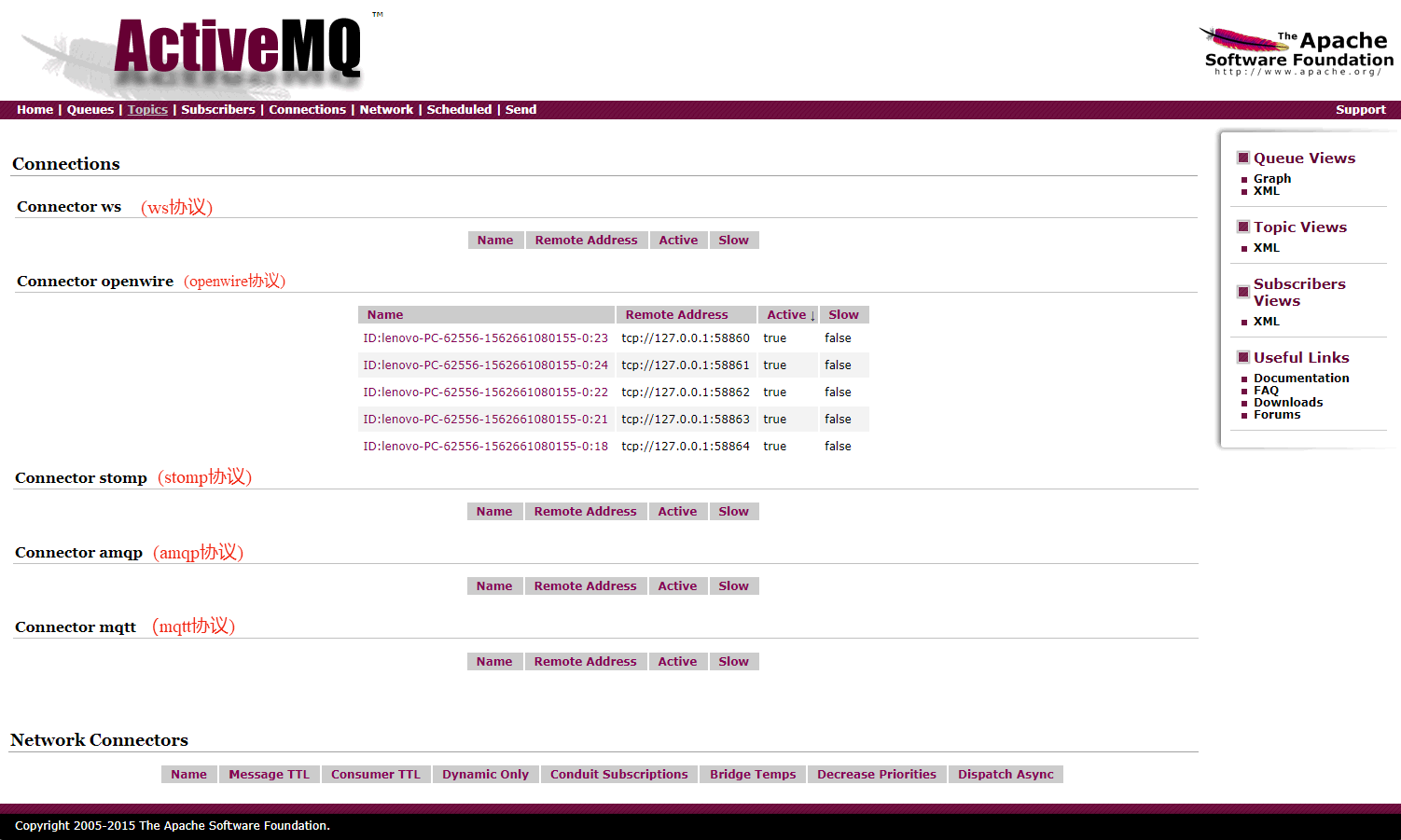
### 分类：

**传输连接器(transport connector)：**用于客户端和服务端之间(client-to-broker)的通信；

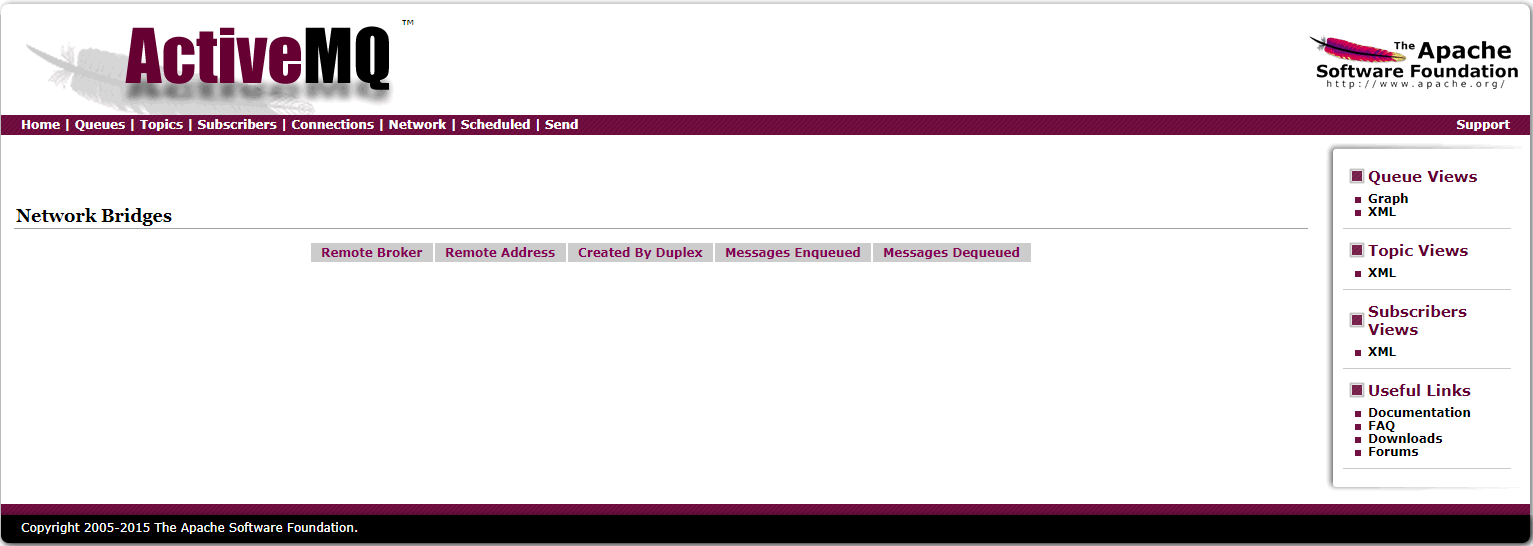
**网络连接器(network connector)：**用户集群中多个服务端之间(broker-to-broker)的通信。

### Tips：

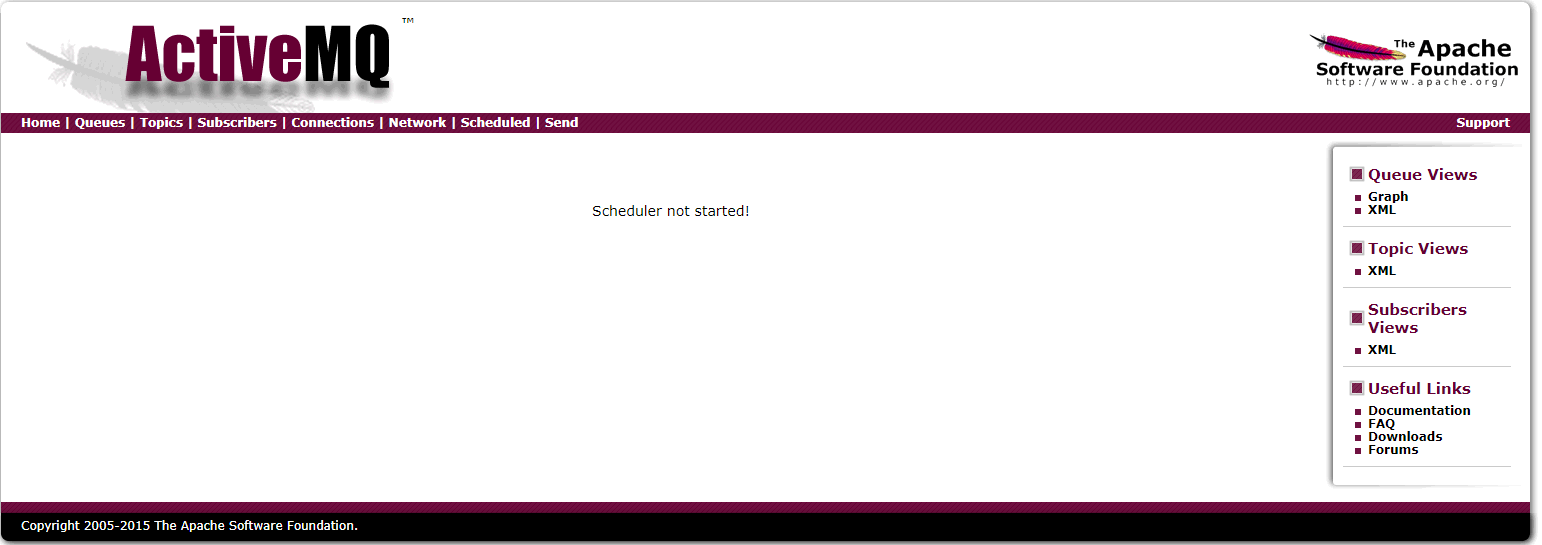
**查询消费者是否监听相关队列：**首先在“Remote Address栏”搜索消费者的IP，如果存在，点击“Name栏”的“连接ID”进入连接页面，查看“Consumers”可以看到队列名称。



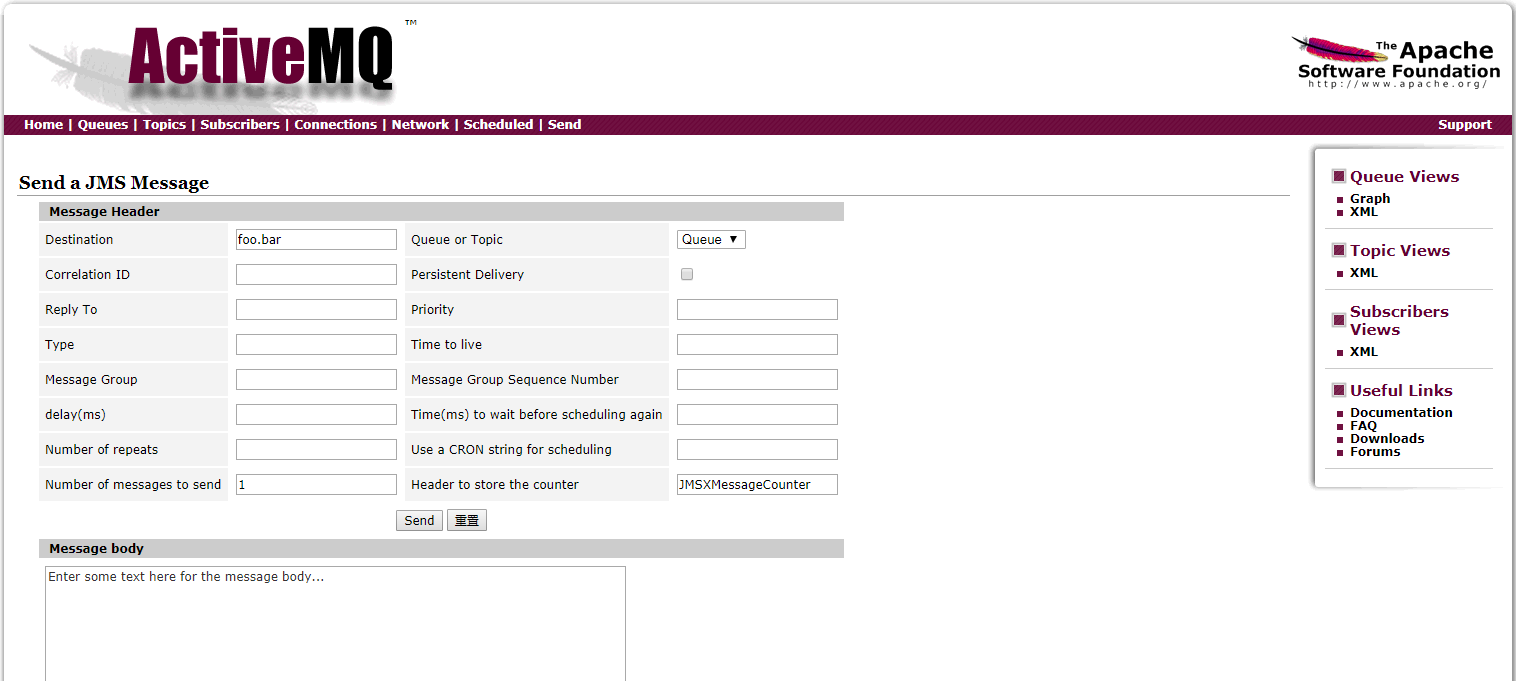
## NetWork



## Scheduled



## Send



<https://www.bilibili.com/video/av17795523/?p=2>

<https://blog.csdn.net/w05980598/article/details/78842061>