1. 关于 ActiveMQ

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（百度百科给出的概述）。我们可以简单的理解：两个应用程序之间需要进行通信，我们使用一个 JMS 服务，进行中间的转发，通过 JMS 的使用，我们可以解除两个程序之间的耦合。

消息队列中间件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，实现高性能，高可用，可伸缩和最终一致性架构。目前使用较多的消息队列有ActiveMQ，RabbitMQ，ZeroMQ，Kafka，MetaMQ，RocketMQ

1.2. 什么是 ActiveMQ

Apache ActiveMQ 是 Apache 软件基金会所研发的开放源代码消息中间件；由于 ActiveMQ 是一个纯 Java 程序，因此只需要操作系统支持 Java 虚拟机，ActiveMQ 便可执行。

业务场景说明:

消息队列在大型电子商务类网站，如京东、淘宝、去哪儿等网站有着深入的应用，

队列的主要作用是消除高并发访问高峰，加快网站的响应速度。

在不使用消息队列的情况下，用户的请求数据直接写入数据库，在高并发的情况下，会对数据库造成巨大的压力，同时也使得系统响应延迟加剧。

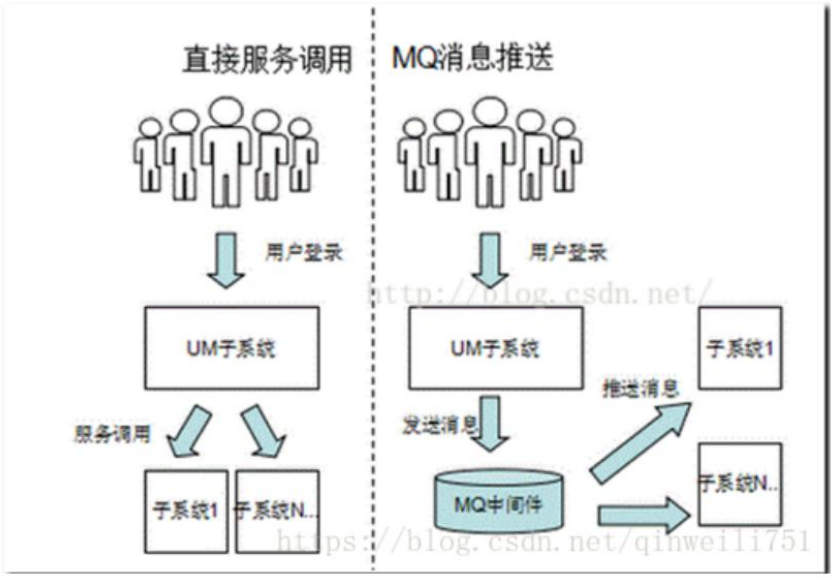
在使用队列后，用户的请求发给队列后立即返回,

（例如: 当然不能直接给用户提示订单提交成功，京东上提示：您“您提交了订单，请等待系统确认”），

再由消息队列的消费者进程从消息队列中获取数据，异步写入数据库。

由于消息队列的服务处理速度远快于数据库，因此用户的响应延迟可得到有效改善。

图解说明:



1.3. 消息队列说明

消息队列中间件是分布式系统中重要的组件，主要解决应用耦合，异步消息，流量削锋等问题。

实现高性能，高可用，可伸缩和最终一致性架构。是大型分布式系统不可缺少的中间件。

目前在生产环境，使用较多的消息队列有ActiveMQ，RabbitMQ，ZeroMQ，Kafka，MetaMQ，RocketMQ等。

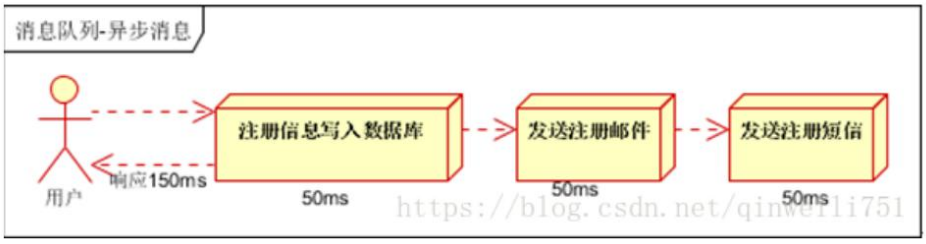
1.4. 消息队列应用场景

消息队列在实际应用中常用的使用场景。异步处理，应用解耦，流量削锋和消息通讯四个场景。

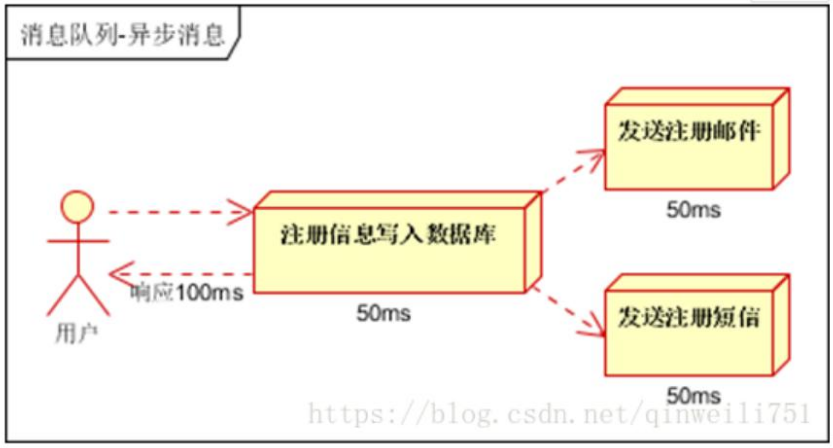
1.4.1. 异步处理

场景说明：用户注册后，需要发注册邮件和注册短信。传统的做法有两种1.串行的方式；2.并行方式。

（1）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。



（2）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。与串行的差别是，并行的方式可以提高处理的时间。



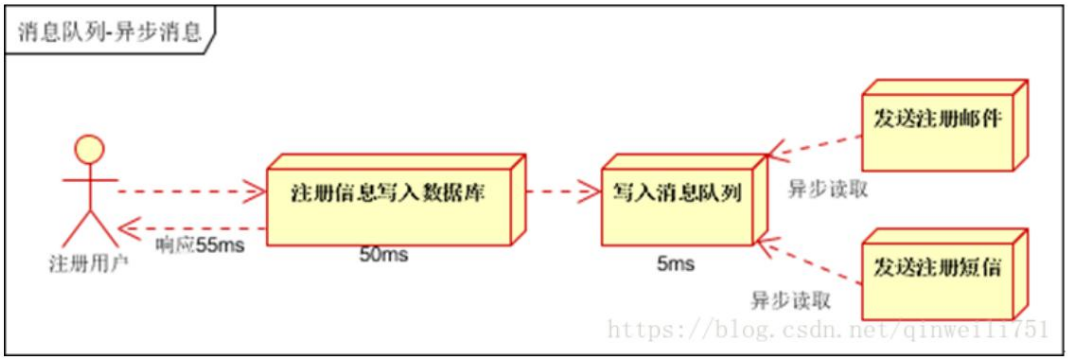
假设三个业务节点每个使用50毫秒钟，不考虑网络等其他开销，则串行方式的时间是150毫秒，并行的时间可能是100毫秒。

因为CPU在单位时间内处理的请求数是一定的，假设CPU1秒内吞吐量是100次。

则串行方式1秒内CPU可处理的请求量是7次（1000/150）。并行方式处理的请求量是10次（1000/100）。

小结：如以上案例描述，传统的方式系统的性能（并发量，吞吐量，响应时间）会有瓶颈。如何解决这个问题呢?

引入消息队列，将不是必须的业务逻辑，异步处理。改造后的架构如下：



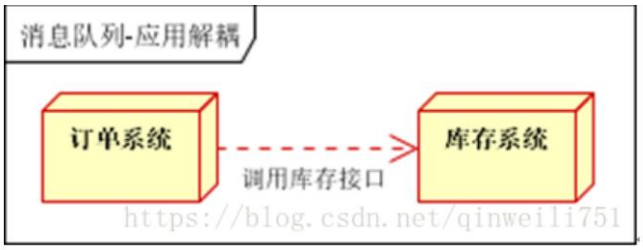
按照以上约定，用户的响应时间相当于是注册信息写入数据库的时间，也就是50毫秒。

注册邮件，发送短信写入消息队列后，直接返回，因此写入消息队列的速度很快，基本可以忽略，

因此用户的响应时间可能是50毫秒。所以基于此架构改变后，系统的吞吐量提高到每秒20 QPS。比串行提高了3倍，比并行提高了两倍。

1.4.2. 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。传统的做法是，订单系统调用库存系统的接口。如下图：

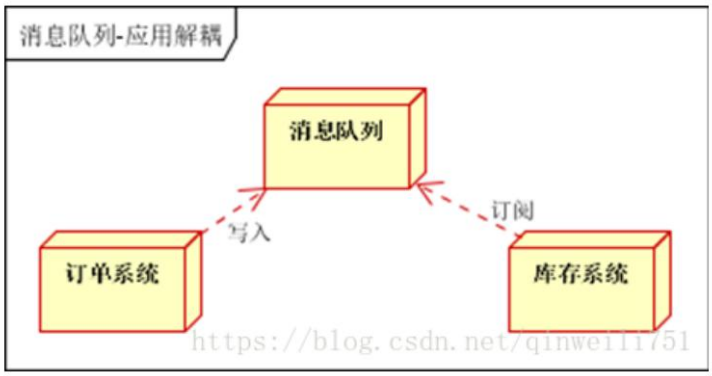


传统模式的缺点：

1） 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；

2） 订单系统与库存系统耦合；

如何解决以上问题呢？引入应用消息队列后的方案，如下图：



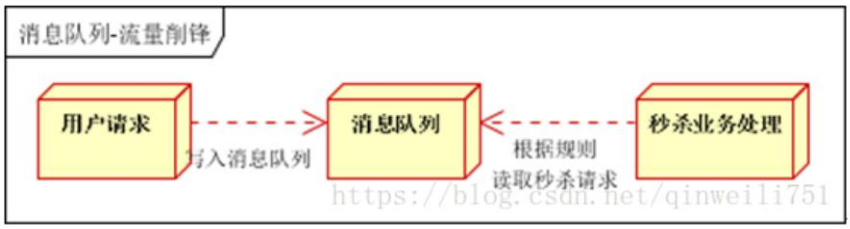
- 1:订单系统：用户下单后，订单系统完成持久化处理，将消息写入消息队列，返回用户订单下单成功,请等待物流配送。
- 2:库存系统：订阅下单的消息，采用拉/推的方式，获取下单信息，库存系统根据下单信息，进行库存操作。
- 3:假如：在下单时库存系统不能正常使用。也不影响正常下单，
- 因为下单后，订单系统写入消息队列就不再关心其他的后续操作了。实现订单系统与库存系统的应用解耦。

1.4.3. 流量削峰

流量削锋也是消息队列中的常用场景，一般在秒杀或团抢活动中使用广泛。

应用场景：秒杀活动，一般会因为流量过大，导致流量暴增，应用容易挂掉。为解决这个问题，一般需要在应用前端加入消息队列。

1. 可以控制活动的人数.
2. 可以缓解短时间内高流量压垮应用;

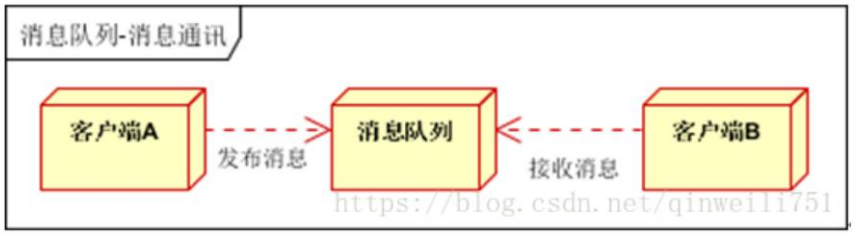


1. 用户的请求，服务器接收后，首先写入消息队列。假如消息队列长度超过最大数量，则直接抛弃用户请求或跳转到错误页面;
2. 秒杀业务根据消息队列中的请求信息，再做后续处理。

1.4.4. 消息通信

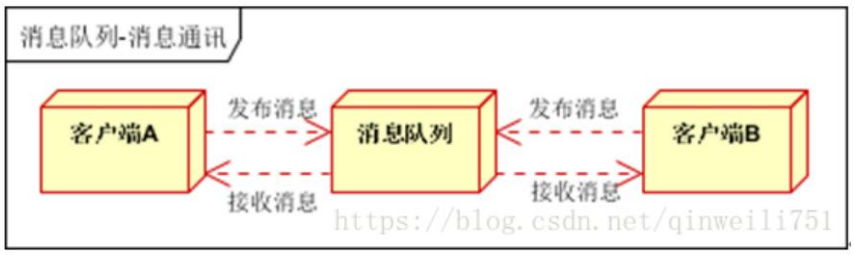
消息通讯是指，消息队列一般都内置了高效的通信机制，因此也可以用在纯的消息通讯。比如实现点对点消息队列，或者聊天室等。

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。

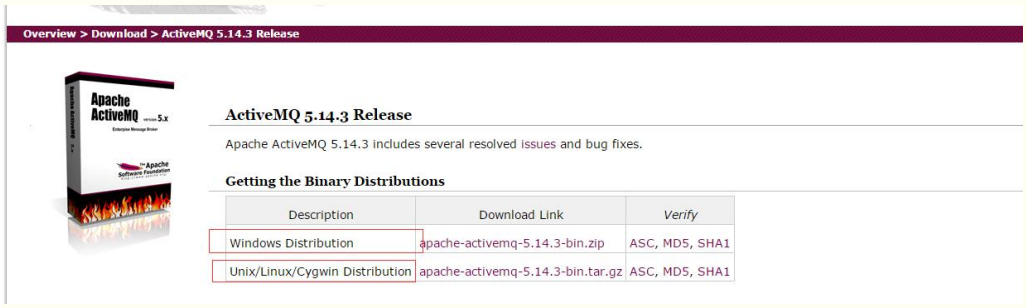
以上实际是消息队列的两种消息模式，点对点或发布订阅模式。

2. 下载安装 ActiveMQ

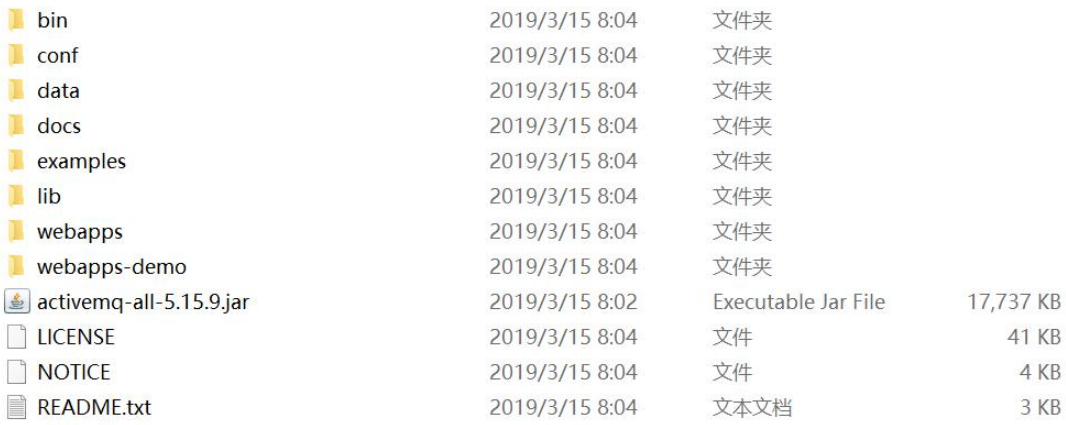
2.1. 下载

ActiveMQ 官网下载地址：http://activemq.apache.org/download.html

ActiveMQ 提供了 Windows 和 Linux、Unix 等几个版本，楼主这里选择了 Linux 版本下进行开发。



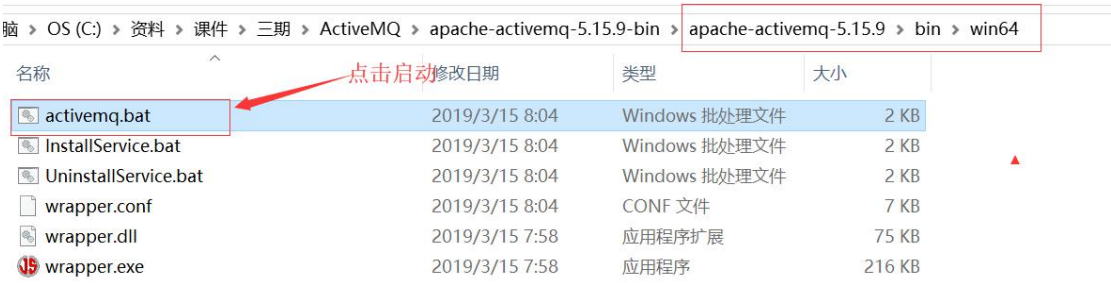
解压



从它的目录来说，还是很简单的：

bin 存放的是脚本文件
conf 存放的是基本配置文件
data 存放的是日志文件
docs 存放的是说明文档
examples 存放的是简单的实例
lib 存放的是 activemq 所需 jar 包
webapps 用于存放项目的目录

2.2. 启动 ActiveMQ



ActiveMQ 默认启动时，启动了内置的 jetty 服务器，提供一个用于监控 ActiveMQ 的 admin 应用。

admin：_

我们在浏览器打开链接之后输入账号密码（这里和 tomcat 服务器类似）

默认账号：admin

密码：admin

3. 消息传送模型

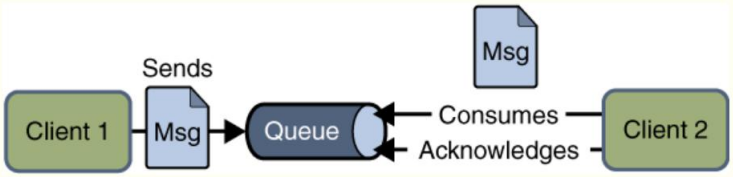
3.1. ActiveMQ 支持两种截然不同的消息传送模型

ActiveMQ 支持两种截然不同的消息传送模型：PTP（即点对点模型）和 Pub/Sub（即发布 /订阅模型），分别称作：PTP Domain 和 Pub/Sub Domain。

一、PTP 消息传送模型

(1)、Point-to-Point Messaging Domain（点对点通信模型）

a、模式图：



b、涉及到的概念：

在点对点通信模式中，应用程序由消息队列，发送方，接收方组成。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。

c、特点：

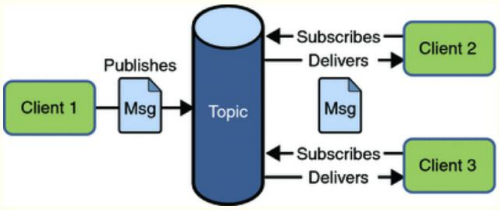
- 每个消息只要一个消费者
- 发送者和接收者在时间上是没有时间的约束，也就是说发送者在发送完消息之后，不管接收者有没有接受消息，都不会影响发送方发送消息到消息队列中。
- 发送方不管是否在发送消息，接收方都可以从消息队列中去到消息（The receiver can fetch message whether it is running or not when the sender sends the message）
- 接收方在接收完消息之后，需要向消息队列应答成功

点对点的模式主要建立在一个队列上面，当连接一个列队的时候，发送端不需要知道接收端是否正在接收，可以直接向 ActiveMQ 发送消息，发送的消息，将会先进入队列中，如果有接收端在监听，则会发向接收端，如果没有接收端接收，则会保存在 activemq 服务器，直到接收端接收消息，点对点的消息模式可以有多个发送端，多个接收端，但是一条消息，只会被一个接收端给接收到，哪个接收端先连上 ActiveMQ，则会先接收到，而后来的接收端则接收不到那条消息

二、Pub/Sub 消息传送模型（发送/订阅模型）----不支持负载均衡

(2)、Publish/Subscribe Messaging Domain（发布/订阅通信模型）

a、模式图：



b、涉及到的概念：

在发布/订阅消息模型中，发布者发布一个消息，该消息通过topic传递给所有的客户端。该模式下，发布者与订阅者都是匿名的，即发布者与订阅者都不知道对方是谁。并且可以动态的发布与订阅Topic。Topic主要用于保存和传递消息，且会一直保存消息直到消息被传递给客户端。

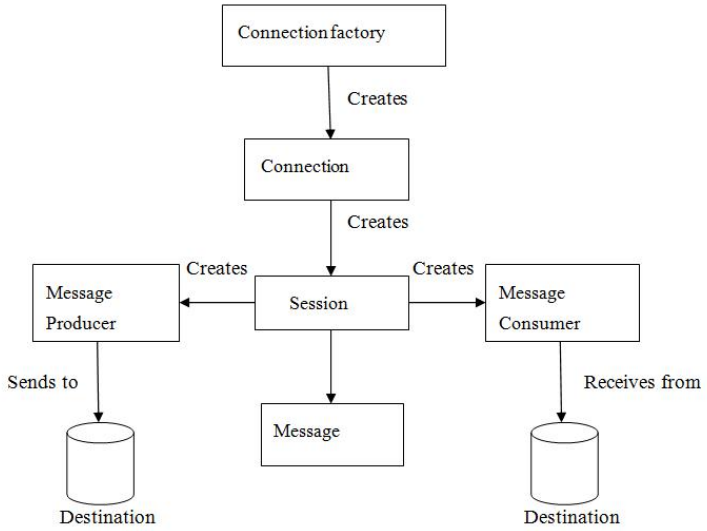
c、特点：

- 一个消息可以传递个多个订阅者（即：一个消息可以有多个接受方）
- 发布者与订阅者具有时间约束，针对某个主题（Topic）的订阅者，它必须创建一个订阅者之后，才能消费发布者的消息，而且为了消费消息，订阅者必须保持运行的状态。
- 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。

订阅/发布模式，同样可以有着多个发送端与多个接收端，但是接收端与发送端存在时间上的依赖，就是如果发送端发送消息的时候，接收端并没有监听消息，那么 ActiveMQ 将不会保存消息，将会认为消息已经发送，换一种说法，就是发送端发送消息的时候，接收端不在线，是接收不到消息的，哪怕以后监听消息，同样也是接收不到的。这个模式还有一个特点，那就是，发送端发送的消息，将会被所有的接收端给接收到，不类似点对点，一条消息只会被一个接收端给接收到。

3.2. 开发 java 消息服务（JMS）的步骤

1. 管理对象（Administered objects）-连接工厂（Connection Factories）和目的地（Destination）
2. 连接对象（Connections）
3. 会话（Sessions）
4. 消息生产者（Message Producers）
5. 消息消费者（Message Consumers）
6. 消息监听者（Message Listeners）



4. 创建一个 ActiveMQ 工程

ActiveMQ 消息队列的使用及应用 - 朱小杰 - 博客园.mhtml

4.1. 点对点模型实现

4.2. 发布订阅模型实现

5. Spring 整合 ActiveMQ

在实际的项目中如果使用原生的 ActiveMQ API 开发会比较麻烦，因为需要创建连接工厂，创建连接等，我们应该使用一个模板来做这些繁琐的事情，Spring 帮我们做了！

5.1. 导入 jar

```
<dependency>
  <groupId>org.apache.activemq</groupId>
```

```
  <artifactId>activemq-core</artifactId>
  <version>5.7.0</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.activemq/activemq-all -->
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-jms</artifactId>
  <version>4.0.0.RELEASE</version>
</dependency>
<dependency>
  <groupId>org.apache.activemq</groupId>
  <artifactId>activemq-pool</artifactId>
  <version>5.7.0</version>
</dependency>

<!-- https://mvnrepository.com/artifact/org.apache.xbean/xbean-spring -->
<dependency>
  <groupId>org.apache.xbean</groupId>
  <artifactId>xbean-spring</artifactId>
  <version>4.4</version>
</dependency>
```

Java消息队列-Spring整合ActiveMq.mhtml