1. JMS消息传送模型：

a.点对点：

在点对点消息传送模型中，应用程序由消息队列，发送者，接收者组成。每一个消息发送给一个特殊的消息队列，该队列保存了所有发送给它的消息(除了被接收者消费掉的和过期的消息)。特性如下：

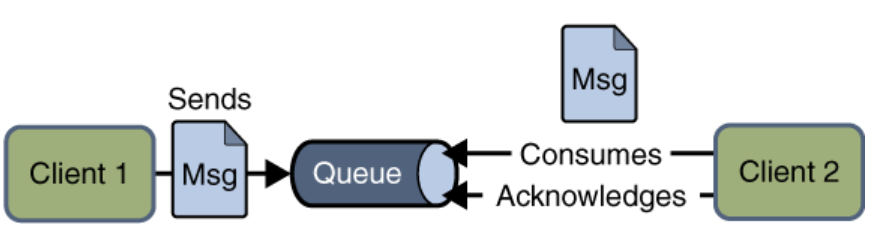
i.在点对点消息传送模型中，应用程序由消息队列，发送者，接收者组成。每一个消息发送给一个特殊的消息队列，该队列保存了所有发送给它的消息(除了被接收者消费掉的和过期的消息)。点对点消息模型有一些特性，如下：

ii.每个消息只有一个接收者；

iii.消息发送者和接收者并没有时间依赖性；

iv.当消息发送者发送消息的时候，无论接收者程序在不在运行，都能获取到消息；

v.当接收者收到消息的时候，会发送确认收到通知（acknowledgement）



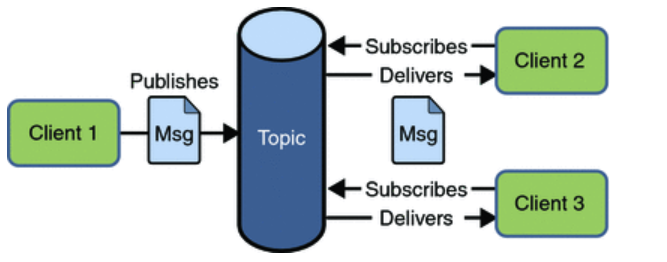
b.发布/订阅:

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。在这种模型中，发布者和订阅者彼此不知道对方，是匿名的且可以动态发布和订阅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。特性如下：

i.一个消息可以传递给多个订阅者

ii.发布者和订阅者有时间依赖性，只有当客户端创建订阅后才能接受消息，且订阅者需一直保持活动状态以接收消息。

iii.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。



1. JMS接收消息方式：
2. 同步：

使用同步方式接收消息的话，消息订阅者调用receive()方法。在receive()中，消息未到达或在到达指定时间之前，方法会阻塞，直到消息可用。

1. 异步：

使用异步方式接收消息的话，消息订阅者需注册一个消息监听者，类似于事件监听器，只要消息到达，JMS服务提供者会通过调用监听器的onMessage()递送消息。

1. JMS应用程序由如下基本模块组成：

a.管理对象（Administered objects）-连接工厂（Connection Factories）和目的地（Destination）

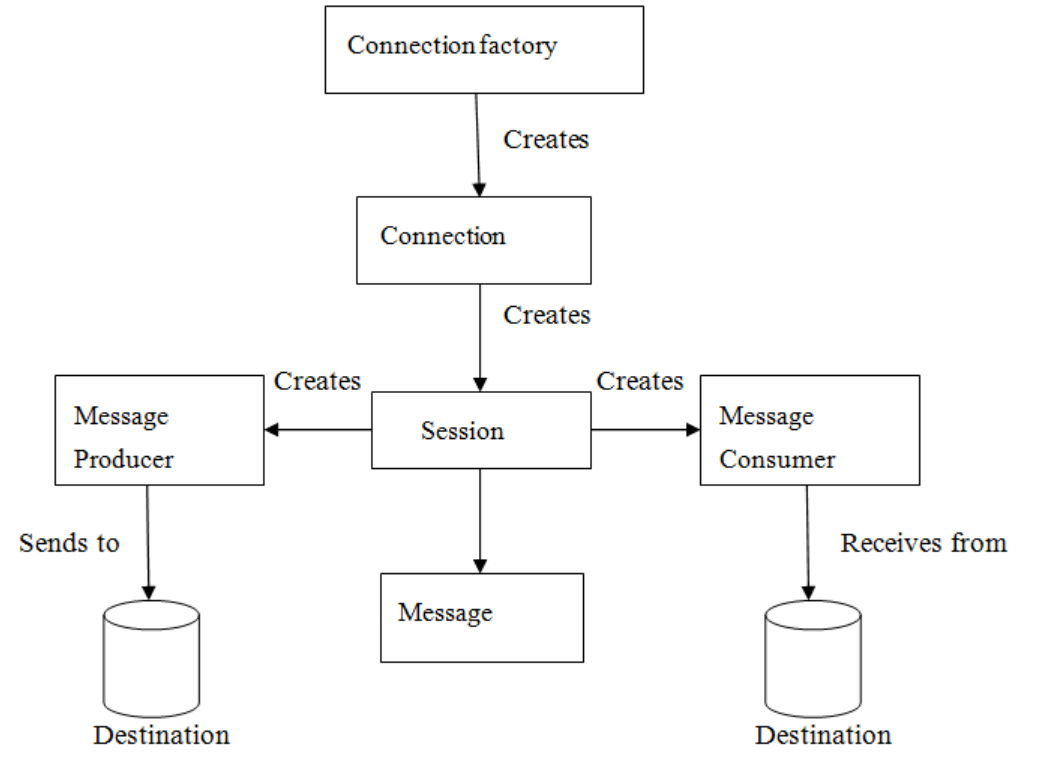
b.连接对象（Connections）

c.会话（Sessions）

d.消息生产者（Message Producers）

e.消息消费者（Message Consumers）

f.消息监听者（Message Listeners）



1. JMS管理对象：管理对象（Administered objects）是预先配置的JMS对象，由系统管理员为使用JMS的客户端创建，主要有两个被管理的对象：连接工厂（ConnectionFactory）、目的地（Destination）。这两个管理对象由JMS系统管理员通过使用Application Server管理控制台创建，存储在应用程序服务器的JNDI名字空间或JNDI注册表。
2. 连接工厂（ConnectionFactory）：客户端使用一个连接工厂对象连接到JMS服务提供者，它创建了JMS服务提供者和客户端之间的连接。JMS客户端（如发送者或接受者）会在JNDI名字空间中搜索并获取该连接。使用该连接，客户端能够与目的地通讯，往队列或话题发送/接收消息
3. 目的地（Destination）：目的地指明消息被发送的目的地以及客户端接收消息的来源。JMS使用两种目的地，队列和话题。
4. JMS连接：连接对象封装了与JMS提供者之间的虚拟连接，如果我们有一个ConnectionFactory对象，可以使用它来创建一个连接。
5. JMS会话（session）: Session是一个单线程上下文，用于生产和消费消息，可以创建出消息生产者和消息消费者。Session对象实现了Session接口，在创建完连接后，我们可以使用它创建Session。
6. JMS消息生产者：消息生产者由Session创建，用于往目的地发送消息。生产者实现MessageProducer接口，我们可以为目的地、队列或话题创建生产者；
7. JMS消息消费者：消息消费者由Session创建，用于接受目的地发送的消息。消费者实现MessageConsumer接口，，我们可以为目的地、队列或话题创建消费者。
8. JMS消息监听器：JMS消息监听器是消息的默认事件处理者，他实现了MessageListener接口，该接口包含一个onMessage方法，在该方法中需要定义消息达到后的具体动作。通过调用setMessageListener方法我们给指定消费者定义了消息监听器

具体可参考：<https://www.cnblogs.com/chenpi/p/5559349.html（JMS>编程接口部分）

1. JMS消息结构：
2. 消息头: JMS消息头预定义了若干字段用于客户端与JMS提供者之间识别和发送消息
3. 消息属性: 我们可以给消息设置自定义属性，这些属性主要是提供给应用程序的。对于实现消息过滤功能，消息属性非常有用，JMS API定义了一些标准属性，JMS服务提供者可以选择性的提供部分标准属性。
4. 消息体: 在消息体中，JMS API定义了五种类型的消息格式，让我们可以以不同的形式发送和接受消息，具体如下：

Text message : javax.jms.TextMessage，表示一个文本对象。

Object message : javax.jms.ObjectMessage，表示一个JAVA对象。

Bytes message : javax.jms.BytesMessage，表示字节数据。

Stream message :javax.jms.StreamMessage，表示java原始值数据流。

Map message : javax.jms.MapMessage，表示键值对。

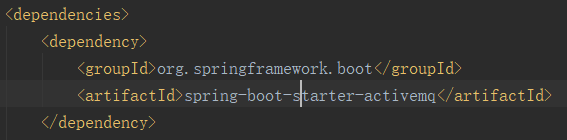
1. JmsTemplate:

JmsTemplate类是JMS核心包中的中心类。它简化了 JMS 的使用，因为在发送或同步接收消息时它帮我们处理了资源的创建和释放。使用JmsTemplate的代码只需要实现规范中定义的回调接口。在JmsTemplate中通过调用代码让MessageCreator回调接口用所提供的会话(Session)创建消息。然而，为了顾及更复杂的 JMS API 应用，回调接口SessionCallback将 JMS 会话提供给用户，回调接口ProducerCallback则公开了Session和MessageProducer的组合。

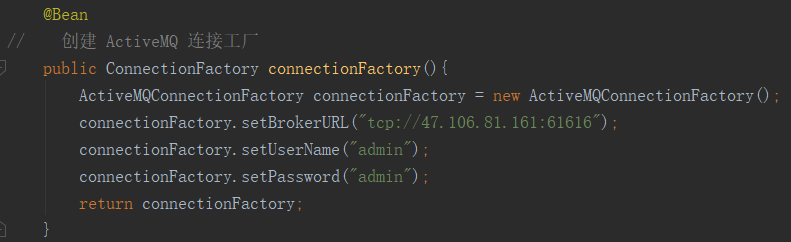
JMS API 公开了发送方法的两种类型，一种接受交付模式、优先级和存活时间作为服务质量（QOS）参数，另一种则使用缺省值作为 QOS 参数（无需参数）方式。由于 JmsTemplate 中有很多发送方法，QOS 参数用 bean 属性进行暴露设置，从而避免在一系列发送方法中的重复。同样地，使用setReceiveTimeout属性设置用于同步接收调用的超时值。

一些 JMS 提供者通过配置ConnectionFactory，管理方式上允许默认的 QOS值 的设置。MessageProducer的发送方法send(Destination destination, Message message)在那些专有的 JMS 中将会使用不一样的 QOS 默认值。 所以，为了提供对 QOS 值域、的统一管理，JmsTemplate必须通过设置布尔值属性isExplicitQosEnabled为true，使它能够使用自己的QOS值。(优先级设置解决方法)

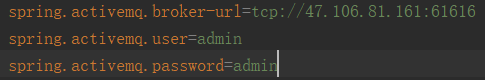
1. JMS简单使用（ActiveMQ为例）：
2. 添加maven依赖：



1. 创建ActiveMQ连接工厂：

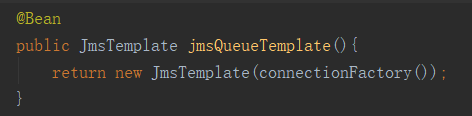


Springboot中可直接进行配置：

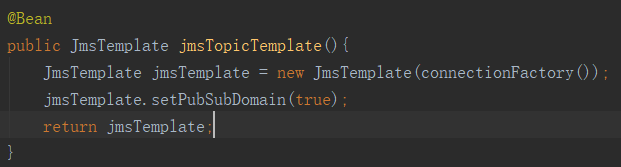


1. 关于消息的发送，是通过 JMS 核心包中的JmsTemplate类来实现的，它简化了 JMS 的使用，因为在发送或同步接收消息时它帮我们处理了资源的创建和释放。它需要引用我们上面创建的连接工厂，具体代码如下：（该部分不需手动编写）：

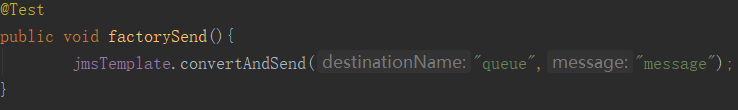
Queue:

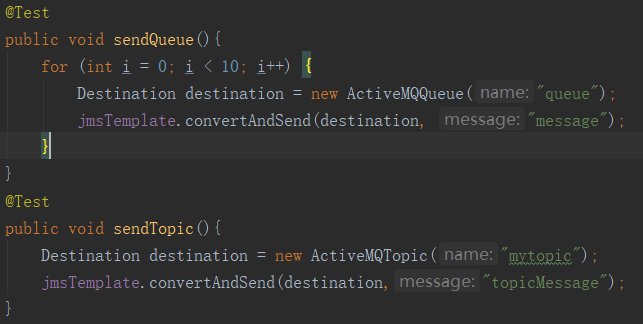


Topic：



1. 发送消息：

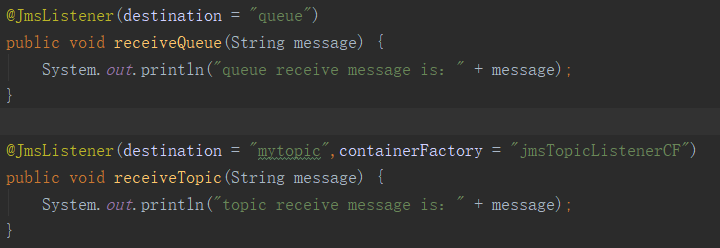




1. 配置监听容器工厂：



1. 采用监听者的方式来监听指定地点的消息，采用注解@JmsListener来设置监听方法。



（默认为接收Queue消息）

1. 队列是不需要我们提前定义好的，它和 RabbitMQ 不一样，它会在我们需要的时候动态的创建的。
2. Topic、Queue两种消息传输方式对比：



1. ActiveMQ消息确认机制（ack机制）：

ACK模式描述了consumer与broker确认消息的方式(时机),比如当消息被consumer接收之后,consumer将在何时确认消息。对于broker而言，只有接收到ACK指令,才会认为消息被正确的接收或者处理成功了,通过ACK，可以在consumer（/producer）与Broker之间建立一种简单的“担保”机制。Ack机制有：

AUTO\_ACKNOWLEDGE = 1    自动确认

CLIENT\_ACKNOWLEDGE = 2    客户端手动确认

DUPS\_OK\_ACKNOWLEDGE = 3    自动批量确认

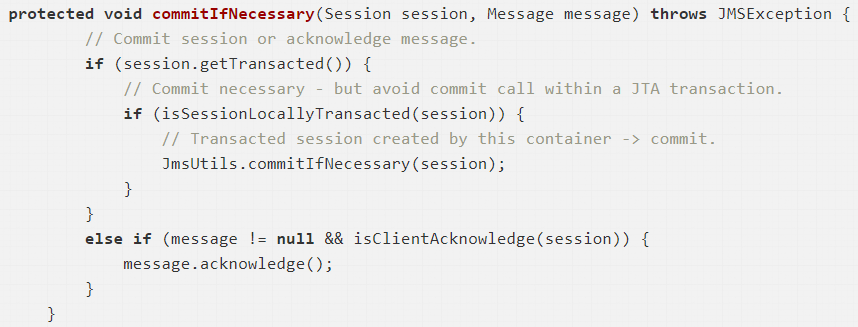
SESSION\_TRANSACTED = 0    事务提交并确认

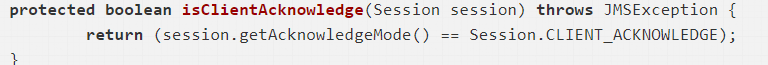
INDIVIDUAL\_ACKNOWLEDGE = 4    单条消息确认 （activemq 独有）

用法：



（ps: 关于Spring下ActiveMQ的消息确认机制有一个问题，发现AcknowledgeMode设置成Session.CLIENT\_ACKNOWLEDGE并没有什么用，还是会自动确认。源码如下，当设置sessionAcknowledgeMode为2时，虽然是客户端手动确认，但是却被spring自动确认了，造成设置无效。这时只需要把sessionAcknowledgeMode的值设置成activemq自定义的类型INDIVIDUAL\_ACKNOWLEDGE = 4即可）





1. 消息预取：

预取限制(prefetch limit)：表示在某个时间段内，可能向消费者传输的最大消息量，如果达到该上限，那么停止发送，直到ActiveMQ收到消费者的acknowledgements(确认，表示已经处理了该消息)。prefetch limit可以针对每个不同的consumer来设置。

为了获取更高的性能，prefetch limit当然是越大越好，只要consumer有足够大的消息缓冲区(messagevolume)。如果消息的总量非常少，而且每个消息的处理时间非常的长，那么，可以将prefetch设置为1，这样，每次向consumer发送一个消息，等其确认已经处理完毕后，再发送第二个。

Ps:如果prefetch设置为0，表示consumer每次　主动向activeMQ要求传输最大的数据量，而不是被动地接收消息。

设置prefetchSize大小方法：

a. 所有的目的地每次最大取50条消息。在连接工厂连接URL里设置:

tcp://localhost:61616?jms.prefetchPolicy.all=50

b.队列每次取1条消息，在连接工厂连接URL里设置

tcp://localhost:61616?jms.prefetchPolicy.queuePrefetch=1

c.特定队列每次预取10条消息。在创建队列时或监听方法上注解设置。

queue = new ActiveMQQueue("TEST.QUEUE?consumer.prefetchSize=10");

or

@JmsListener(destination="queue?consumer.prefetchSize=100")

1. 消息的优先级：

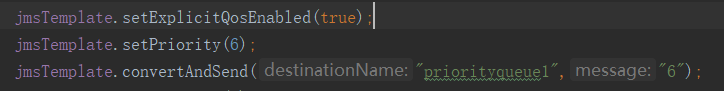
问题：



以上写法设置优先级无效。由于JMS provider将查看消息上的消息优先级属性，并在将消息放入队列之前调用setJMSPriority方法设置默认优先级。默认消息优先级为4（普通优先级），因此若使用setJMSPriority方法设置消息优先级，消息优先级不会是开发人员最初预期设置的优先级。

解决方法：

JMS API有两种发送方法，一种采用发送模式、优先级和存活时间作为服务质量（QOS）参数，另一种使用无需QOS参数的缺省值方法。由于在 JmsTemplate 中有许多种发送方法，QOS参数通过bean的属性方式进行设置，从而避免在多种发送方法中重复。同样，使用 setReceiveTimeout 属性值来设置同步接收调用的超时值。某些JMS供应者允许通过ConnectionFactory的配置来设置缺省的QOS值。这样在调用 MessageProducer 的发送方法 send(Destination destination, Message message) 时会使用那些不同的QOS缺省值，而不是JMS规范中定义的值。所以，为了提供对QOS值的一致管理，JmsTemplate必须通过设置布尔值属性 isExplicitQosEnabled 为true，使它能够使用自己的QOS值。具体代码如下:



存在问题：

接收消息时并未按照优先级顺序接收。

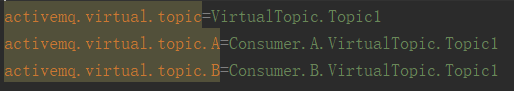
1. Virtual topic：

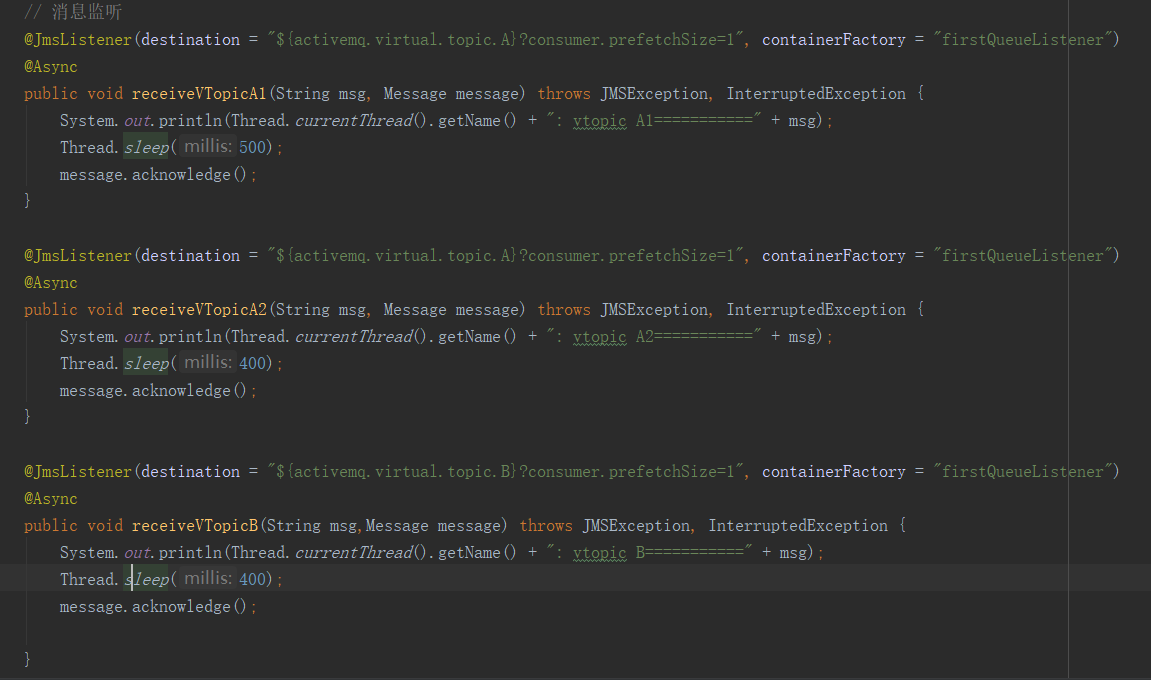
a．使用场景：

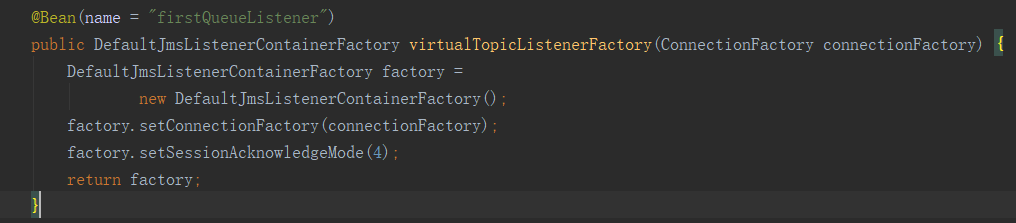
有一个application, 订阅一个Topic消息, 假设这个application部署到2台server上时，2台server会同时监听这个Topic, 这样会导致一个消息重复消费2次，这不符合一些特定的应用场景。

如果设置成Queue，能确保一个消息只被消费一次，但就不能有其他的application同时消费这个消息，也不符合一些特定的应该场景。为了达到在分布式部署的情况下，一个消息， 被同一个application只消费一次，同时也能被其他的application消费，就引入Virtual Topic. 这种方式可以说：在Topic消息层面上，实现负载均衡。

b.代码实现：







c．virtual topic命名规范：

Topic命名： VirtualTopic.xxx

消费者命名： Consumer.yyy.VirtualTopic.xxx

参考：1.JMS操作使用实现：<https://segmentfault.com/a/1190000009682775#articleHeader9>

2.activemq ack确认机制问题：<https://segmentfault.com/a/1190000008707181>

<https://www.jianshu.com/p/a924c30554ca>

3.优先级解决方法（19.2.1）：<http://www.shouce.ren/api/spring2.5/ch19s02.html>

4.virtualTopic：<https://www.jianshu.com/p/a924c30554ca>

5.《Spring 5 官方文档》26. JMS：http://ifeve.com/spring-5-26-jms-java-message-service/