# 答疑

## 消息的发送策略

持久化消息

默认情况下，生产者发送的消息是持久化的。消息发送到broker以后，producer会等待broker对这条消息的处理情况的反馈

可以设置消息发送端发送持久化消息的异步方式

connectionFactory.setUseAsyncSend(**true**);

回执窗口大小设置

connectionFactory.setProducerWindowSize();

非持久化消息

textMessage.setJMSDeliveryMode(DeliveryMode.NON\_PERSISTENCE);

非持久化消息模式下，默认就是异步发送过程，如果需要对非持久化消息的每次发送的消息都获得broker的回执的话

connectionFactory.setAlwaysSyncSend();

## consumer获取消息是pull还是（broker的主动 push）

默认情况下，mq服务器（broker）采用异步方式向客户端主动推送消息(push)。也就是说broker在向某个消费者会话推送消息后，不会

等待消费者响应消息，直到消费者处理完消息以后，主动向broker返回处理结果

prefetchsize

“预取消息数量“

broker端一旦有消息，就主动按照默认设置的规则推送给当前活动的消费者。 每次推送都有一定的数量限制，而这个数量就是prefetchSize

Queue

持久化消息 prefetchSize=1000

非持久化消息 1000

topic

持久化消息 100

非持久化消息 32766

假如prefetchSize=0 . 此时对于consumer来说，就是一个pull模式

## 关于acknowledge为什么能够在第5次主动执行ack以后，把前面的消息都确认掉



消表示已经被consumer接收但未确认的消息。

### 消息确认

ACK\_TYPE，消费端和broker交换ack指令的时候，还需要告知broker ACK\_TYPE。

ACK\_TYPE表示确认指令的类型，broker可以根据不同的ACK\_TYPE去针对当前消息做不同的应对策略

REDELIVERED\_ACK\_TYPE (broker会重新发送该消息) 重发侧策略

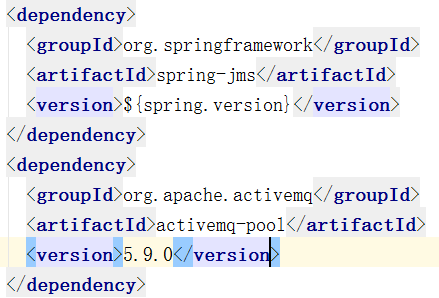
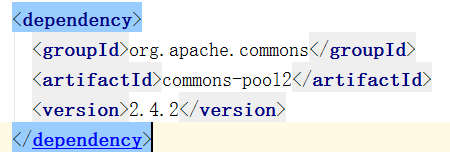
DELIVERED\_ACK\_TYPE 消息已经接收，但是尚未处理结束

STANDARD\_ACK\_TYPE 表示消息处理成功

# ActiveMQ结合spring开发

Spring提供了对JMS的支持，需要添加Spring 支持JMS的包

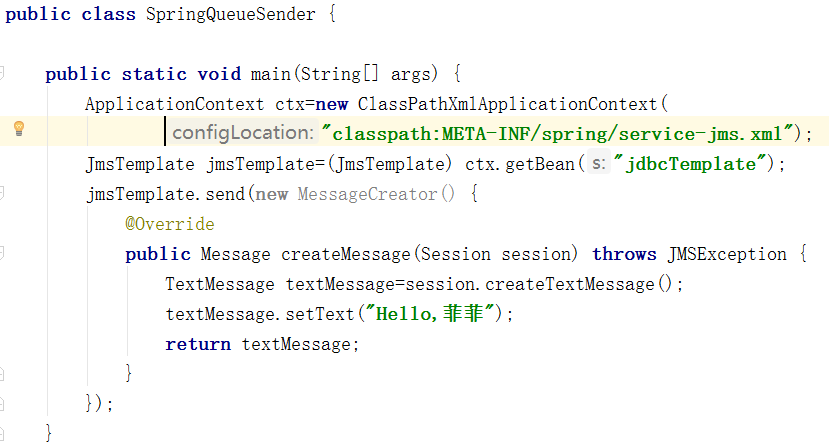
## 添加jar依赖

## 配置spring文件



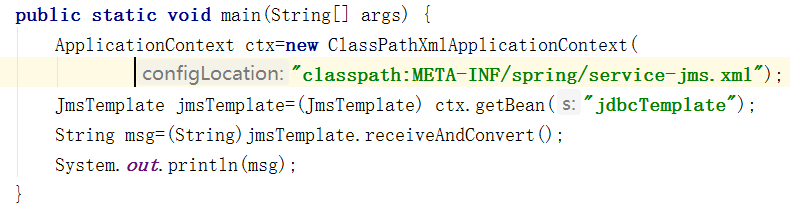
## 编写发送端代码



## 配置接收端spring文件

直接copy发送端的文件

## 编写接收端代码

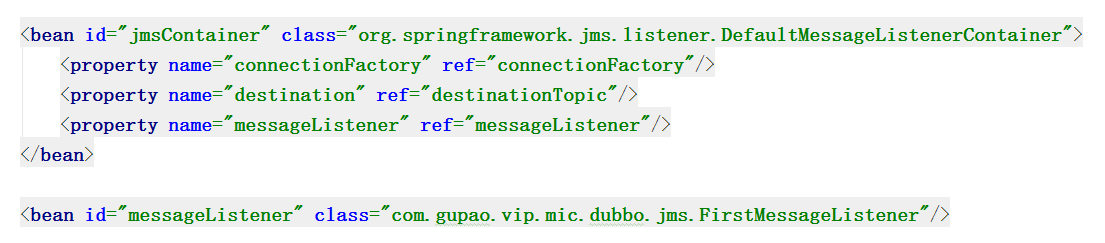


## spring的发布订阅模式配置

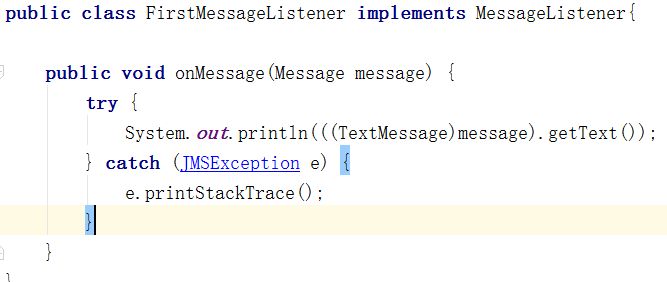
|  |  |
| --- | --- |
|  |  |

## 以事件通知方式来配置消费者

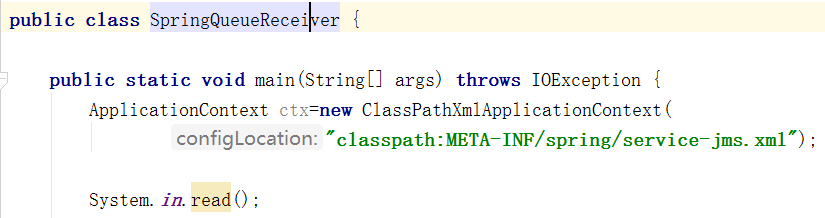
### 更改消费端的配置



### 增加FirstMessageListener监听类



### 启动spring容器



# ActiveMQ支持的传输协议

client端和broker端的通讯协议

**TCP、**UDP 、NIO、SSL、Http（s）、vm

# ActiveMQ持久化存储



1. kahaDB 默认的存储方式

<persistenceAdapter>

<kahaDB directory="${activemq.data}/kahadb"/>

</persistenceAdapter>

1. AMQ 基于文件的存储方式

写入速度很快，容易恢复。

文件默认大小是32M

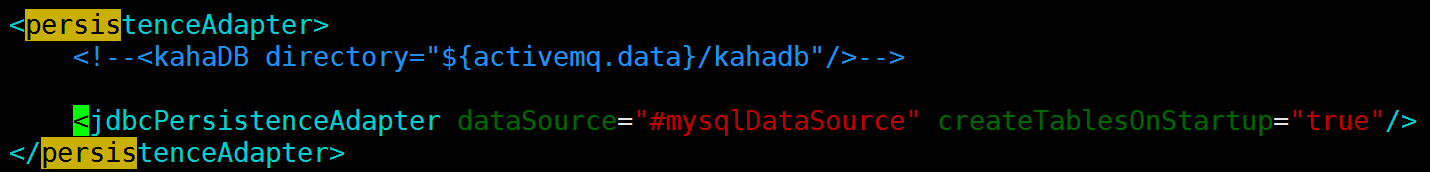
1. JDBC 基于数据库的存储

ACTIVEMQ\_ACKS ： 存储持久订阅的信息

ACTIVEMQ\_LOCK ： 锁表（用来做集群的时候，实现master选举的表）

ACTIVEMQ\_MSGS ： 消息表

## 第一步



## 第二步



## 第三步

添加jar包依赖

JDBC Message store with activeMQ journal

1. 引入了快速缓存机制，缓存到Log文件中
2. 性能会比jdbc store要好
3. JDBC Message store with activeMQ journal 不能应用于master/slave模式
4. Memory 基于内存的存储

# LevelDB

5.8以后引入的持久化策略。通常用于集群配置

# ActiveMQ的网络连接

activeMQ如果要实现扩展性和高可用性的要求的话，就需要用用到网络连接模式

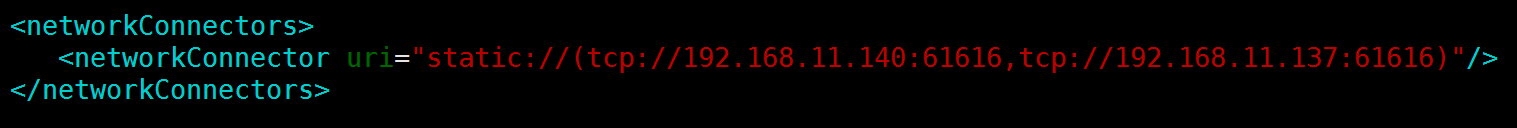
## NetworkConnector

主要用来配置broker与broker之间的通信连接

|  |  |
| --- | --- |
| http://dl2.iteye.com/upload/attachment/0101/1747/10190bfe-30d7-3021-8bbe-9d7882530083.png | 如上图所示，服务器S1和S2通过NewworkConnector相连，则生产者P1发送消息，消费者C3和C4都可以接收到，而生产者P3发送的消息，消费者C1和C2同样也可以接收到 |

### 静态网络连接

修改activemq.xml，增加如下内容



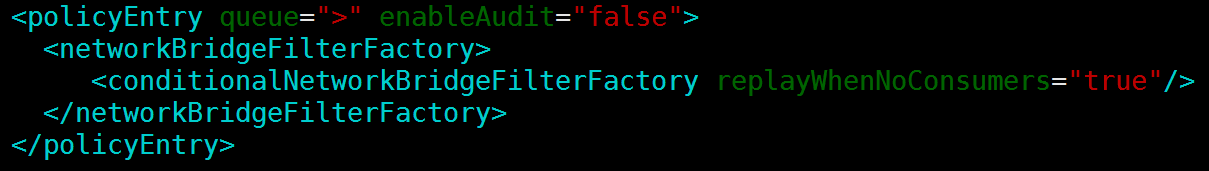
|  |  |
| --- | --- |
|  | 两个Brokers通过一个staic的协议来进行网络连接。一个Consumer连接到BrokerB的一个地址上，当Producer在BrokerA上以相同的地址发送消息是，此时消息会被转移到BrokerB上，也就是说BrokerA会转发消息到BrokerB上 |

## 丢失的消息

一些consumer连接到broker1、消费broker2上的消息。消息先被broker1从broker2消费掉，然后转发给这些consumers。假设，转发消息的时候broker1重启了，这些consumers发现brokers1连接失败，通过failover连接到broker2.但是因为有一部分没有消费的消息被broker2已经分发到broker1上去了，这些消息就好像消失了。除非有消费者重新连接到broker1上来消费

从5.6版本开始，在destinationPolicy上新增了一个选项replayWhenNoConsumers属性，这个属性可以用来解决当broker1上有需要转发的消息但是没有消费者时，把消息回流到它原始的broker。同时把enableAudit设置为false，为了防止消息回流后被当作重复消息而不被分发

通过如下配置，在activeMQ.xml中。 分别在两台服务器都配置。即可完成消息回流处理



### 动态网络连接