Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!

# 私塾在线《高级软件架构师实战培训阶段一》

## ——跟着cc学架构系列精品教程

私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 本部分课程概览

- 根据实际的应用需要，学习要用到的ActiveMQ的知识，以快速上手、理解并掌握ActiveMQ
- 一： ActiveMQ简介
  包括：是什么、能干什么、特点；消息中间件的功能、特点、应用场景等
- 二： ActiveMQ安装和基本使用
  包括：通过源码安装、基本的配置示例、启动、测试运行、关闭等
- 三：理解和掌握JMS
  包括：基本概念、消息结构、可靠性机制、PTP、Pub/Sub、API结构、JMS应用开发的基本步骤、持久和非持久的Topic等
- 四：用ActiveMQ构建应用
  包括：多种启动Broker的方法、单独应用的开发、结合Spring的开发等
- 五：ActiveMQ的Transport
  包括：多种传输协议的功能、配置和使用



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 本部分课程概览

- 六： ActiveMQ的消息存储
  包括：队列和topic、KahaDB、AMQ、JDBC、MMS等
- 七： ActiveMQ的Network
  包括：在一台服务器启动多个Broker；静态网络连接的功能、配置等；“丢失”消息的处理；容错或可负载均衡的连接；动态网络连接等
- 八：ActiveMQ的集群
  包括： 队列消费者集群、Broker的集群、 Master Slave等
- 九： Destination高级特性
  包括： 通配符、组合队列、配置启动的Destinations、 删除不活动的Destinations、 Destination选项、虚拟Destinations、镜像队列、每个Destination单独策略配置等
- 十： Message Dispatch高级特性
  包括： 消息游标、异步发送、严格分发策略、轮询分发策略、优化批量确认、生产者流量控制等



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 本部分课程概览

- 十一： Message高级特性
  包括：消息属性、 Advisory Message、延迟和定时消息投递、Blob消息、消息转换等
- 十二： Consumer高级特性
  包括：独有消费者、消息异步分发、消息优先级、管理持久化消息、消息分组、消息选择器、消息重递策略、慢消费者处理等
- 十三：杂项技术
  包括：监控和管理Broker、集成ActiveMQ和Tomcat、什么时候使用ActiveMQ等
- 十四： ActiveMQ优化
  包括：影响ActiveMQ性能的因素、常见的优化方式和配置等



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ简介

- ActiveMQ是什么

  ActiveMQ是Apache推出的，一款开源的，完全支持JMS1.1和J2EE 1.4规范的JMS Provider实现的消息中间件（Message Oriented Middleware，MOM）

- ActiveMQ能干什么

  最主要的功能就是：实现JMS Provider，用来帮助实现高可用、高性能、可伸缩、易用和安全的企业级面向消息服务的系统

- ActiveMQ特点

  完全支持JMS1.1和J2EE 1.4规范 （持久化，XA消息，事务)
  支持多种传送协议：in-VM,TCP,SSL,NIO,UDP,JGroups,JXTA
  可插拔的体系结构，可以灵活定制，如：消息存储方式、安全管理等
  很容易和Application Server集成使用
  多种语言和协议编写客户端。语言: Java,C,C++,C#,Ruby,Perl,Python,PHP
  从设计上保证了高性能的集群，客户端-服务器，点对点
  可以很容易的和Spring结合使用
  支持通过JDBC和journal提供高速的消息持久化
  支持与Axis的整合



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 消息中间件

- MOM基本功能：将信息以消息的形式，从一个应用程序传送到另一个或多个应用程序。
- MOM主要特点：

1：消息异步接受，类似手机短信的行为，消息发送者不需要等待消息接受者的响应，减少软件多系统集成的耦合度；

2：消息可靠接收，确保消息在中间件可靠保存，只有接收方收到后才删除消息，多个消息也可以组成原子事务

- 消息中间件的主要应用场景：

在多个系统间进行整合和通讯的时候，通常会要求：

1：可靠传输，数据不能丢失，有的时候，也会要求不能重复传输；

2：异步传输，否则各个系统同步发送接受数据，互相等待，造成系统瓶颈

- 目前比较知名的消息中间件：

IBM MQSeries
BEA WebLogic JMS Server
Oracle AQ
Tibco
SwiftMQ
AcitveMQ：是免费的java实现的消息中间件



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ安装和基本使用

- 下载并安装ActiveMQ服务器端

  1：从http://activemq.apache.org/download.html下载最新的ActiveMQ

  2：直接解压，然后拷贝到你要安装的位置就好了

- 启动运行

  1：普通启动：到ActiveMQ/bin下面，./activemq start

  2：启动并指定日志文件 ./activemq start > /tmp/activemqlog

- 检查是否已经启动

  ActiveMQ默认采用61616端口提供JMS服务，使用8161端口提供管理控制台服务，执行以下命令以便检验是否已经成功启动ActiveMQ服务：

1：比如查看61616端口是否打开： netstat -an | grep 61616

2：也可以直接查看控制台输出或者日志文件

3：还可以直接访问ActiveMQ的管理页面：http://192.168.1.106:8161/admin/
默认的用户名和密码是admin/admin

- 关闭ActiveMQ，可以用./activemq stop
  暴力点的可以用ps -ef | grep activemq 来得到进程号，然后kill掉



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 基本的Queue消息发送-1

- 配置Maven所需的依赖，示例如下：

```
<dependency>
    <groupId>org.apache.activemq</groupId>
    <artifactId>activemq-all</artifactId>
    <version>5.9.0</version>
</dependency>

<dependency>
    <groupId>org.apache.xbean</groupId>
    <artifactId>xbean-spring</artifactId>
    <version>3.16</version>
</dependency>
```

- Queue消息发送的示例代码如下：



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 基本的Queue消息发送-2

```
public class JmsSend {
    public static void main(String[] args) throws Exception {
 ConnectionFactory connectionFactory = new
    ActiveMQConnectionFactory("tcp://192.168.1.106:61616");
        Connection connection = connectionFactory.createConnection();
        connection.start();

      Session session = connection.createSession(Boolean.TRUE, Session.AUTO_ACKNOWLEDGE);
        Destination destination = session.createQueue("my-queue");

        MessageProducer producer = session.createProducer(destination);
        for(int i=0; i<3; i++) {
            TextMessage message = session.createTextMessage("message--"+i);
            Thread.sleep(1000);
            //通过消息生产者发出消息
            producer.send(message);
        }
        session.commit();
        session.close();
        connection.close();
    }
}
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 基本的Queue消息接收

```
public class JmsReceiver {
    public static void main(String[] args) throws Exception {
       ConnectionFactory cf = new ActiveMQConnectionFactory("tcp://192.168.1.106:61616");
        Connection connection = cf.createConnection();
        connection.start();

final Session session = connection.createSession(Boolean.TRUE, Session.AUTO_ACKNOWLEDGE);
        Destination destination = session.createQueue("my-queue");

        MessageConsumer consumer = session.createConsumer(destination);
        int i=0;
        while(i<3) {
            i++;
            TextMessage message = (TextMessage) consumer.receive();
            session.commit();
            System.out.println("收到消 息: " + message.getText());
        }
        session.close();
        connection.close();
    }
}
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖！



# JMS基本概念-1

- JMS是什么

  JMS Java Message Service，Java消息服务，是Java EE中的一个技术。
- JMS规范

  JMS定义了Java 中访问消息中间件的接口，并没有给予实现，实现JMS 接口的消息中间件称为JMS Provider，例如ActiveMQ
- JMS provider：实现JMS接口和规范的消息中间件
- JMS message：JMS的消息，JMS消息由以下三部分组成：

  1：消息头：每个消息头字段都有相应的getter和setter方法

  2：消息属性：如果需要除消息头字段以外的值，那么可以使用消息属性

  3：消息体：封装具体的消息数据
- JMS producer：消息生产者，创建和发送JMS消息的客户端应用
- JMS consumer：消息消费者，接收和处理JMS消息的客户端应用

  消息的消费可以采用以下两种方法之一：

  1：同步消费：通过调用消费者的receive方法从目的地中显式提取消息，receive 方法可以一直阻塞到消息到达。

  2：异步消费：客户可以为消费者注册一个消息监听器，以定义在消息到达时所采取的动作



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



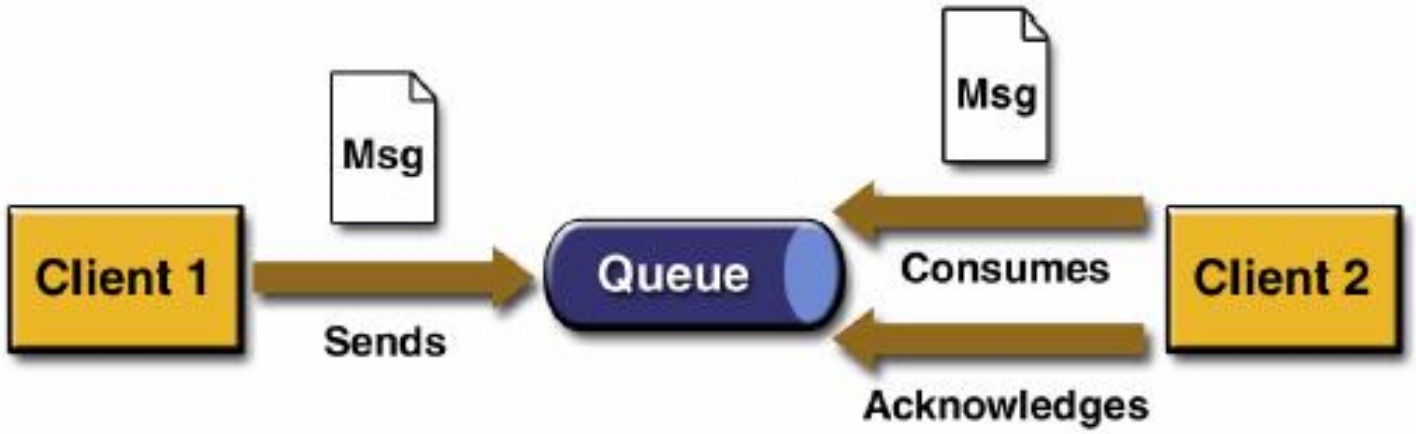
# JMS基本概念-2

- JMS domains：消息传递域，JMS规范中定义了两种消息传递域：点对点 （point-to-point，简写成PTP）消息传递域和发布/订阅消息传递域（publish/subscribe，简写成pub/sub）

1：点对点消息传递域的特点如下：

（1）每个消息只能有一个消费者

（2）消息的生产者和消费者之间没有时间上的相关性。无论消费者在生产者发送消息的时候是否处于运行状态，它都可以提取消息。






私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖！



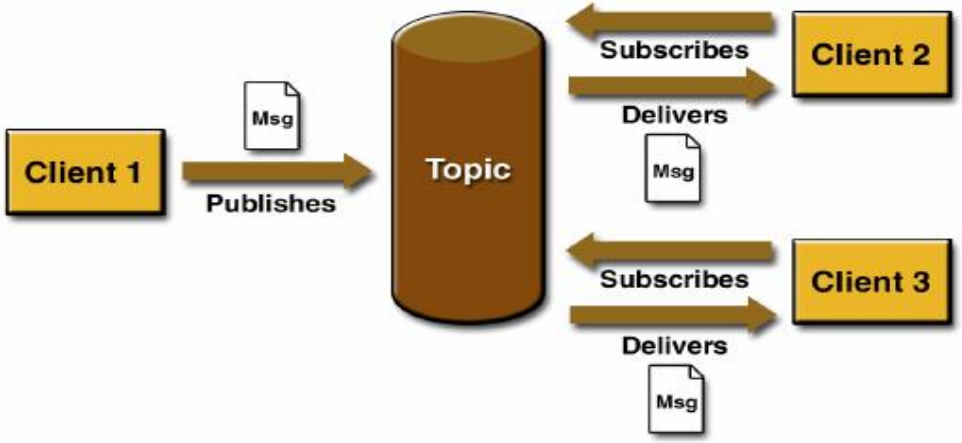
# JMS基本概念-3

2：发布/订阅消息传递域的特点如下：

（1）每个消息可以有多个消费者

（2）生产者和消费者之间有时间上的相关性。订阅一个主题的消费者只能消费自它订阅之后发布的消息。JMS 规范允许客户创建持久订阅，这在一定程度上放松了时间上的相关性要求。持久订阅允许消费者消费它在未处于激活状态时发送的消息。

3：在点对点消息传递域中，目的地被称为队列（queue）；在发布/订阅消息传递域中，目的地被称为主题（topic）






私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# JMS基本概念-4

- Connection factory：连接工厂，用来创建连接对象，以连接到JMS的provider
- JMS Connection：封装了客户与JMS 提供者之间的一个虚拟的连接
- JMS Session：是生产和消费消息的一个单线程上下文

  会话用于创建消息生产者（producer）、消息消费者（consumer）和消息（message）等。会话提供了一个事务性的上下文，在这个上下文中，一组发送和接收被组合到了一个原子操作中。
- Destination：消息发送到的目的地
- Acknowledge：签收
- Transaction：事务
- JMS client：用来收发消息的Java应用
- Non-JMS client：使用JMS provider本地API写的应用，用来替换JMS API实现收发消息的功能，通常会提供其他的一些特性，比如：CORBA、RMI等。
- Administered objects：预定义的JMS对象，通常在provider规范中有定义，提供给JMS客户端来访问，比如： ConnectionFactory和Destination

做最好的在线学习社区
网　址：http://sishuok.com
咨询QQ：2371651507
私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# JMS的消息结构-1

- JMS 消息由以下几部分组成：消息头，属性和消息体
- 消息头包含消息的识别信息和路由信息，消息头包含一些标准的属性如下：

  1：JMSDestination：由send方法设置
  2：JMSDeliveryMode：由send方法设置
  3：JMSExpiration：由send方法设置
  4：JMSPriority：由send方法设置
  5：JMSMessageID：由send方法设置
  6：JMSTimestamp：由客户端设置
  7：JMSCorrelationID ：由客户端设置
  8：JMSReplyTo ：由客户端设置
  9：JMSType ：由客户端设置
  10：JMSRedelivered：由JMS Provider设置
- 标准的 JMS 消息头包含以下属性：

1：JMSDestination：消息发送的目的地：主要是指Queue和Topic，自动分配



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# JMS的消息结构-2

2：JMSDeliveryMode：传送模式。有两种 ：持久模式和非持久模式。一条持久性的消息应该被传送“一次仅仅一次”，这就意味者如果JMS提供者出现故障，该消息并不会丢失，它会在服务器恢复之后再次传递。一条非持久的消息最多会传送一次，这意味这服务器出现故障，该消息将永远丢失。自动分配

3：JMSExpiration：消息过期时间，等于 Destination 的send 方法中的timeToLive值加上发送时刻的GMT 时间值。如果timeToLive 值等于零，则JMSExpiration 被设为零，表示该消息永不过期。如果发送后，在消息过期时间之后消息还没有被发送到目的地，则该消息被清除。自动分配

4：JMSPriority：消息优先级，从 0-9 十个级别，0-4 是普通消息，5-9 是加急消息。JMS 不要求JMS Provider 严格按照这十个优先级发送消息，但必须保证加急消息要先于普通消息到达。默认是4级。自动分配

5：JMSMessageID：唯一识别每个消息的标识，由JMS Provider 产生。自动分配

6：JMSTimestamp：一个JMS Provider在调用send()方法时自动设置的。它是消息被发送和消费者实际接收的时间差。自动分配



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# JMS的消息结构-3

7：JMSCorrelationID：用来连接到另外一个消息，典型的应用是在回复消息中连接到原消息。在大多数情况下，JMSCorrelationID用于将一条消息标记为对JMSMessageID标示的上一条消息的应答，不过，JMSCorrelationID可以是任何值，不仅仅是JMSMessageID。由开发者设置

8：JMSReplyTo：提供本消息回复消息的目的地址。由开发者设置

9：JMSType：消息类型的识别符。由开发者设置

10：JMSRedelivered：如果一个客户端收到一个设置了JMSRedelivered属性的消息，则表示可能客户端曾经在早些时候收到过该消息，但并没有签收(acknowledged)。如果该消息被重新传送，JMSRedelivered=true反之，JMSRedelivered =false。自动设置



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# JMS的消息结构-4

- 消息体，JMS API定义了5种消息体格式，也叫消息类型，可以使用不同形式发送接收数据，并可以兼容现有的消息格式。包括：TextMessage、MapMessage、BytesMessage、StreamMessage和ObjectMessage
- 消息属性，包含以下三种类型的属性：

  1：应用程序设置和添加的属性，比如：

  Message.setStringProperty(“username”,username);

  2：JMS定义的属性

  使用“JMSX”作为属性名的前缀，

  connection.getMetaData().getJMSXPropertyNames()， 方法返回所有连接支持的JMSX 属性的名字。

  3：JMS供应商特定的属性
- JMS定义的属性如下：

1：JMSXUserID：发送消息的用户标识，发送时提供商设置

2：JMSXAppID：发送消息的应用标识，发送时提供商设置



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# JMS的消息结构-5

3：JMSXDeliveryCount：转发消息重试次数,第一次是1，第二次是2，… ，发送时提供商设置

4：JMSXGroupID：消息所在消息组的标识，由客户端设置

5：JMSXGroupSeq：组内消息的序号第一个消息是1，第二个是2，…，由客户端设置

6：JMSXProducerTXID ：产生消息的事务的事务标识，发送时提供商设置

7：JMSXConsumerTXID ：消费消息的事务的事务标识，接收时提供商设置

8：JMSXRcvTimestamp ：JMS 转发消息到消费者的时间，接收时提供商设置

9：JMSXState：假定存在一个消息仓库，它存储了每个消息的单独拷贝，且这些消息从原始消息被发送时开始。每个拷贝的状态有：1（等待），2（准备），3（到期）或4（保留）。由于状态与生产者和消费者无关，所以它不是由它们来提供。它只和在仓库中查找消息相关，因此JMS没有提供这种API。由提供商设置



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# JMS的可靠性机制-1

- 消息接收确认

JMS消息只有在被确认之后，才认为已经被成功地消费了。消息的成功消费通常包含三个阶段：客户接收消息、客户处理消息和消息被确认。

在事务性会话中，当一个事务被提交的时候，确认自动发生。在非事务性会话中，消息何时被确认取决于创建会话时的应答模式（acknowledgement mode）。该参数有以下三个可选值：

Session.AUTO_ACKNOWLEDGE：当客户成功的从receive方法返回的时候，或者从MessageListener.onMessage方法成功返回的时候，会话自动确认客户收到的消息。

Session.CLIENT_ACKNOWLEDGE：客户通过调用消息的acknowledge方法确认消息。需要注意的是，在这种模式中，确认是在会话层上进行，确认一个被消费的消息将自动确认所有已被会话消费的消息。例如，如果一个消息消费者消费了10 个消息，然后确认第5 个消息，那么所有10 个消息都被确认。

Session.DUPS_ACKNOWLEDGE：该选择只是会话迟钝的确认消息的提交。如果JMS provider失败，那么可能会导致一些重复的消息。如果是重复的消息，那么JMS provider 必须把消息头的JMSRedelivered字段设置为true



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# JMS的可靠性机制-2

- 消息持久性，JMS 支持以下两种消息提交模式：

  PERSISTENT：指示JMS provider持久保存消息，以保证消息不会因为JMS provider的失败而丢失

  NON_PERSISTENT：不要求JMS provider持久保存消息

- 消息优先级

  可以使用消息优先级来指示JMS provider首先提交紧急的消息。优先级分10个级别，从0（最低）到9（最高）。如果不指定优先级，默认级别是4。需要注意的是，JMS provider并不一定保证按照优先级的顺序提交消息

- 消息过期

  可以设置消息在一定时间后过期，默认是永不过期

- 消息的临时目的地

  可以通过会话上的createTemporaryQueue 方法和createTemporaryTopic方法来创建临时目的地。它们的存在时间只限于创建它们的连接所保持的时间。只有创建该临时目的地的连接上的消息消费者才能够从临时目的地中提取消息



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# JMS的可靠性机制-3

- 持久订阅

首先消息生产者必须使用PERSISTENT提交消息。客户可以通过会话上的createDurableSubscriber方法来创建一个持久订阅，该方法的第一个参数必须是一个topic。第二个参数是订阅的名称。

JMS provider会存储发布到持久订阅对应的topic上的消息。如果最初创建持久订阅的客户或者任何其它客户，使用相同的连接工厂和连接的客户ID，相同的主题和相同的订阅名，再次调用会话上的createDurableSubscriber方法，那么该持久订阅就会被激活。JMS provider会向客户发送客户处于非激活状态时所发布的消息。

持久订阅在某个时刻只能有一个激活的订阅者。持久订阅在创建之后会一直保留，直到应用程序调用会话上的unsubscribe方法。



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# JMS的可靠性机制-4

- 本地事务

在一个JMS客户端，可以使用本地事务来组合消息的发送和接收。JMS Session接口提供了commit和rollback方法。事务提交意味着生产的所有消息被发送，消费的所有消息被确认；事务回滚意味着生产的所有消息被销毁，消费的所有消息被恢复并重新提交，除非它们已经过期。

事务性的会话总是牵涉到事务处理中，commit或rollback方法一旦被调用，一个事务就结束了，而另一个事务被开始。关闭事务性会话将回滚其中的事务。

需要注意的是，如果使用请求/回复机制，即发送一个消息，同时希望在同一个事务中等待接收该消息的回复，那么程序将被挂起，因为知道事务提交，发送操作才会真正执行。

需要注意的还有一个，消息的生产和消费不能包含在同一个事务中。



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖！



# JMS的PTP模型

- JMS PTP(Point-to-Point)模型定义了客户端如何向队列发送消息，从队列接收消息，以及浏览队列中的消息。

  PTP模型是基于队列的，生产者发消息到队列，消费者从队列接收消息，队列的存在使得消息的异步传输成为可能。和邮件系统中的邮箱一样，队列可以包含各种消息，JMS Provider 提供工具管理队列的创建、删除。

- PTP的一些特点：

1：如果在Session 关闭时，有一些消息已经被收到，但还没有被签收(acknowledged)，那么，当消费者下次连接到相同的队列时，这些消息还会被再次接收

2：如果用户在receive 方法中设定了消息选择条件，那么不符合条件的消息会留在队列中，不会被接收到

3：队列可以长久地保存消息直到消费者收到消息。消费者不需要因为担心消息会丢失而时刻和队列保持激活的连接状态，充分体现了异步传输模式的优势



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# JMS的Pub/Sub模型

- JMS Pub/Sub 模型定义了如何向一个内容节点发布和订阅消息，这些节点被称作topic

  主题可以被认为是消息的传输中介，发布者(publisher)发布消息到主题，订阅者(subscribe) 从主题订阅消息。主题使得消息订阅者和消息发布者保持互相独立，不需要接触即可保证消息的传送。

- Pub/Sub的一些特点：

1：消息订阅分为非持久订阅和持久订阅

非持久订阅只有当客户端处于激活状态，也就是和JMS Provider保持连接状态才能收到发送到某个主题的消息，而当客户端处于离线状态，这个时间段发到主题的消息将会丢失，永远不会收到。

持久订阅时，客户端向JMS 注册一个识别自己身份的ID，当这个客户端处于离线时，JMS Provider会为这个ID 保存所有发送到主题的消息，当客户再次连接到JMS Provider时，会根据自己的ID 得到所有当自己处于离线时发送到主题的消息。

2：如果用户在receive 方法中设定了消息选择条件，那么不符合条件的消息不会被接收

3：非持久订阅状态下，不能恢复或重新派送一个未签收的消息。只有持久订阅才能恢复或重新派送一个未签收的消息。

4：当所有的消息必须被接收，则用持久订阅。当丢失消息能够被容忍，则用非持久订阅

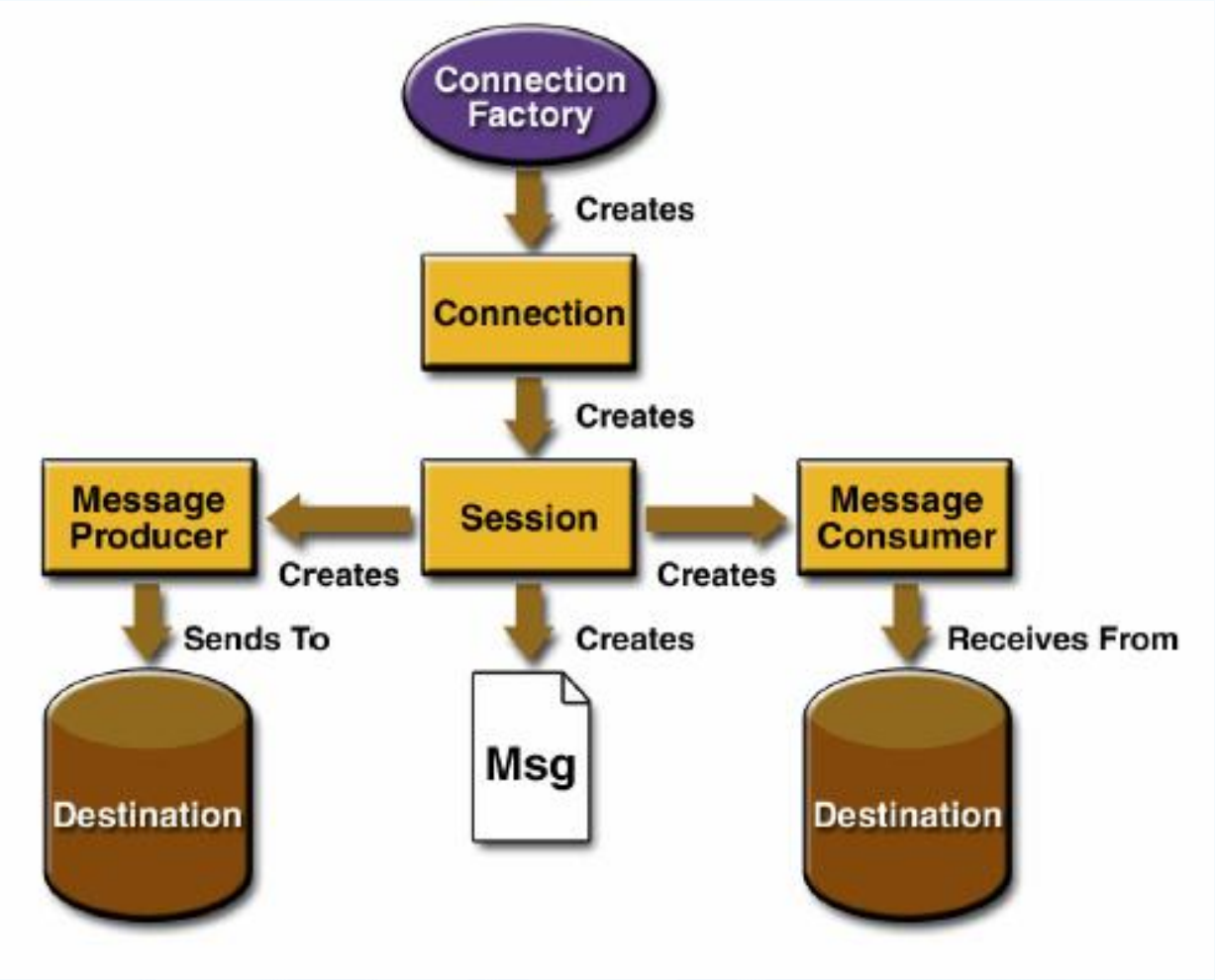


私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# JMS的API结构






私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 一个JMS应用的基本步骤

- JMS开发的基本步骤

1：创建一个JMS connection factory
2：通过connection factory来创建JMS connection
3：启动JMS connection
4：通过connection创建JMS session
5：创建JMS destination
6：创建JMS producer,或者创建JMS message，并设置destination
7：创建JMS consumer，或者是注册一个JMS message listener
8：发送或者接受JMS message(s)
9：关闭所有的JMS资源(connection, session, producer, consumer等)



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 非持久的Topic消息示例

- 对于非持久的Topic消息的发送

基本跟前面发送队列信息是一样的，只是把创建Destination的地方，由创建队列替换成创建Topic，例如：

```
Destination destination = session.createTopic("MyTopic");
```

- 对于非持久的Topic消息的接收

1：必须要接收方在线，然后客户端再发送信息，接收方才能接收到消息

2：同样把创建Destination的地方，由创建队列替换成创建Topic，例如：

```
Destination destination = session.createTopic("MyTopic");
```

3：由于不知道客户端发送多少信息，因此改成while循环的方式了，例如：

```
Message message = consumer.receive();
 while(message!=null) {
      TextMessage txtMsg = (TextMessage)message;
      System.out.println("收到消 息：" + txtMsg.getText());
      message = consumer.receive(1000L);
 }
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 持久的Topic消息示例-1

- 对于持久的Topic消息的发送

```
ConnectionFactory connectionFactory = new
    ActiveMQConnectionFactory("tcp://192.168.1.106:61616");
    Connection connection = connectionFactory.createConnection();
    Session session = connection.createSession(Boolean.TRUE, Session.AUTO_ACKNOWLEDGE);
    Topic destination = session.createTopic("MyTopic");
    MessageProducer producer = session.createProducer(destination);
    producer.setDeliveryMode(DeliveryMode.PERSISTENT);
    connection.start();
    for(int i=0; i<2; i++) {
        TextMessage message = session.createTextMessage("messagedd--"+i);
        Thread.sleep(1000);
        //通过消息生产者发出消息
        producer.send(message);
    }
    session.commit();
    session.close();
    connection.close();
```

1：要用持久化订阅，发送消息者要用 DeliveryMode.PERSISTENT 模式发现，在连接之前设定
2：一定要设置完成后，再start 这个 connection



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 持久的Topic消息示例-2

- 对于持久的Topic消息的接收

```
ConnectionFactory cf = new ActiveMQConnectionFactory("tcp://192.168.1.106:61616");
    Connection connection = cf.createConnection();
    connection.setClientID("cc1");
    final Session session = connection.createSession(Boolean.TRUE, Session.AUTO_ACKNOWLEDGE);
    Topic destination = session.createTopic("MyTopic");
    TopicSubscriber ts = session.createDurableSubscriber(destination, "T1");
    connection.start();
    Message message = ts.receive();
    while(message!=null) {
        TextMessage txtMsg = (TextMessage)message;
        session.commit();
        System.out.println("收到消 息：" + txtMsg.getText());
        message = ts.receive(1000L);
    }
    session.close();
    connection.close();
```

1：需要在连接上设置消费者id，用来识别消费者
2：需要创建TopicSubscriber来订阅
3：要设置好了过后再start 这个 connection
4：一定要先运行一次，等于向消息服务中间件注册这个消费者，然后再运行客户端发送信息，这个时候，无论消费者是否在线，都会接收到，不在线的话，下次连接的时候，会把没有收过的消息都接收下来。

做最好的在线学习社区
网　址：http://sishuok.com
咨询QQ：2371651507
私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 关于持久化和非持久化消息

- 持久化消息

  这是 ActiveMQ 的默认传送模式，此模式保证这些消息只被传送一次和成 功使用一次。对于这些消息，可靠性是优先考虑的因素。可靠性的另一个重要方面是确保持久性消息传送至目标后，消息服务在向消费者传送它们之前不会丢失这些消息。

  这意味着在持久性消息传送至目标时，消息服务将其放入持久性数据存储。如果消息服务由于某种原因导致失败，它可以恢复此消息并将此消息传送至相应的消费者。虽然这样增加了消息传送的开销，但却增加了可靠性。

- 非持久化消息

  保证这些消息最多被传送一次。对于这些消息，可靠性并非主要的考虑因素。 此模式并不要求持久性的数据存储，也不保证消息服务由于某种原因导致失败后消息不会丢失。 有两种方法指定传送模式：

1. 使用setDeliveryMode 方法，这样所有的消息都采用此传送模式； 如：
   `producer.setDeliveryMode(DeliveryMode.NON_PERSISTENT);`
2. 使用send 方法为每一条消息设置传送模式



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 用ActiveMQ构建应用-1

- Broker：相当于一个ActiveMQ服务器实例
- 命令行启动参数示例如下：

1：activemq start ：使用默认的activemq.xml来启动

2：activemq start xbean:file:../conf/activemq-2.xml ：使用指定的配置文件来启动

3：如果不指定file，也就是xbean:activemq-2.xml，那么xml必须在classpath下面

- 用ActiveMQ来构建Java应用

这里主要将用ActiveMQ Broker作为独立的消息服务器来构建JAVA应用。ActiveMQ也支持在vm中通信基于嵌入式的broker，能够无缝的集成其它java应用



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 用ActiveMQ构建应用-2

- 嵌入式Broker启动

1：Broker Service启动broker ，示例如下：

```
BrokerService broker = new BrokerService();
broker.setUseJmx(true);
broker.addConnector("tcp://localhost:61616");
broker.start();
```

2：BrokerFactory启动broker ，示例如下：

```
String Uri = "properties:broker.properties";
BrokerService broker1 = BrokerFactory.createBroker(new URI(Uri));
broker1.addConnector("tcp://localhost:61616");
broker1.start();
```

3：broker.properties的内容示例如下：

```
useJmx=true
persistent=false
brokerName=Cheese
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 用ActiveMQ构建应用-3

n 利用Spring集成Broker，Spring的配置文件如下：

```
<beans>
    <bean id="admins" class="org.apache.activemq.security.AuthenticationUser">
           <constructor-arg index="0" value="admin" />
           <constructor-arg index="1" value="password" />
           <constructor-arg index="2" value="admins,publisher,consumers" />
    </bean>
    <bean id="publishers"        class="org.apache.activemq.security.AuthenticationUser">
           <constructor-arg index="0" value="publisher" />
           <constructor-arg index="1" value="password" />
           <constructor-arg index="2" value="publisher,consumers" />
    </bean>
    <bean id="consumers" class="org.apache.activemq.security.AuthenticationUser">
           <constructor-arg index="0" value="consumer" />
           <constructor-arg index="1" value="password" />
           <constructor-arg index="2" value="consumers" />
    </bean>
    <bean id="guests" class="org.apache.activemq.security.AuthenticationUser">
           <constructor-arg index="0" value="guest" />
           <constructor-arg index="1" value="password" />
           <constructor-arg index="2" value="guests" />
    </bean>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 用ActiveMQ构建应用-4

```
<bean id="simpleAuthPlugin"         class="org.apache.activemq.security.SimpleAuthenticationPlugin">
            <property name="users">
                        <util:list>
                                    <ref bean="admins" />
                                    <ref bean="publishers" />
                                    <ref bean="consumers" />
                                    <ref bean="guests" />
                        </util:list>
            </property>
    </bean>
<bean id="broker" class="org.apache.activemq.broker.BrokerService" init-method="start" destroy-method="stop">
            <property name="brokerName" value="myBroker" />
            <property name="persistent" value="false" />
            <property name="transportConnectorURIs">
                        <list>
                                    <value>tcp://localhost:61616</value>
                        </list>
            </property>
            <property name="plugins">
                        <list>
                                    <ref bean="simpleAuthPlugin"/>
                        </list>
            </property>
    </bean>
</beans>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 用ActiveMQ构建应用-5

- 或者配置BrokerFactoryBean，示例如下：

```
<beans>
    <bean id="broker"        class="org.apache.activemq.xbean.BrokerFactoryBean">
          <property name="config"    value="resources/activemq-simple.xml"/>
          <property name="start" value="true" />
    </bean>
</beans>
```

- ActiveMQ的启动：

1：可以通过在应用程序中以编码的方式启动broker，例如： broker.start();

如果需要启动多个broker，那么需要为broker设置一个名字。例如：

```
BrokerService broker = new BrokerService();
broker.setName("fred");
broker.addConnector("tcp://localhost:61616");
broker.start();
```

2：还可以通过spring来启动，前面已经演示过了



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ结合Spring开发-1

- Spring提供了对JMS的支持，需要添加Spring支持jms的包，如下：

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-jms</artifactId>
    <version>4.0.3.RELEASE</version>
</dependency>
```

- 添加ActiveMQ的pool包

```
<dependency>
    <groupId>org.apache.activemq</groupId>
    <artifactId>activemq-pool</artifactId>
    <version>5.9.0</version>
</dependency>
```

- 然后需要在Spring的配置文件中，配置jmsTemplate，示例如下：



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ结合Spring开发-2

```
<bean id="jmsFactory" class="org.apache.activemq.pool.PooledConnectionFactory"
        destroy-method="stop">
        <property name="connectionFactory">
            <bean class="org.apache.activemq.ActiveMQConnectionFactory">
                <property name="brokerURL">
                    <value>tcp://192.168.1.106:61679</value>
                </property>
            </bean>
        </property>
        <property name="maxConnections" value="100"></property>
    </bean>
    <bean id="jmsTemplate" class="org.springframework.jms.core.JmsTemplate">
        <property name="connectionFactory" ref="jmsFactory" />
        <property name="defaultDestination" ref="destination" />
        <property name="messageConverter">
            <bean class="org.springframework.jms.support.converter.SimpleMessageConverter" />
        </property>
    </bean>

    <bean id="destination" class="org.apache.activemq.command.ActiveMQQueue">
        <constructor-arg index="0" value="spring-queue" />
    </bean>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ结合Spring开发-3

- queue消息发送

```
@Autowired
private JmsTemplate jt = null;
public static void main(String[] args) {
ApplicationContext ctx = new ClassPathXmlApplicationContext("applicationContext.xml");
    SpringJMSClient ct = (SpringJMSClient)ctx.getBean("springJMSClient");
    ct.jt.send(new MessageCreator() {
          public Message createMessage(Session s) throws JMSException {
                TextMessage msg = s.createTextMessage("Spring msg===");
                return msg;
          }
    });
}
```

- queue消息接收

```
@Autowired
private JmsTemplate jt = null;
public static void main(String[] args) throws Exception {
ApplicationContext ctx = new ClassPathXmlApplicationContext("applicationContext.xml");
SpringJMSReceiverClient ct = (SpringJMSReceiverClient)ctx.getBean("springJMSReceiverClient");
    String msg = (String)ct.jt.receiveAndConvert();
    System.out.println("msg==="+msg);
}
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ结合Spring开发-4

- 如果topic的话，首先需要修改spring的配置：

  先添加topic的配置，当然，需要修改jmsTemplate配置里面的defaultDestination，如果不想修改这个配置，那么直接把Destination注入程序，在程序里面选择发送的Destination也可以：

```
<bean id="destinationTopic" class="org.apache.activemq.command.ActiveMQTopic">
    <constructor-arg index="0" value="spring-topic" />
</bean>
```

其他的客户端发送和接收跟队列基本是一样的。

- 如果想要在Spring中配置消费者的话，就不需要再启动接收的客户端了，配置如下：

```
<bean id="jmsContainer"
    class="org.springframework.jms.listener.DefaultMessageListenerContainer">
    <property name="connectionFactory" ref="jmsFactory" />
    <property name="destination" ref="destinationTopic" />
    <property name="messageListener" ref="messageListener" />
</bean>
<bean id="messageListener"
    class="cn.javass.activemq.MyMessageListener">
</bean>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ结合Spring开发-5

- 当然，需要写一个类来实现消息监听，如：

```
public class MyMessageListener implements MessageListener{
    public void onMessage(Message arg0) {
        TextMessage msg = (TextMessage)arg0;
        try {
            System.out.println("receive txt msg==="+msg.getText());
        } catch (JMSException e) {
            e.printStackTrace();
        }
    }
}
```

- 这样测试的时候，只需要启动消息生产者就好了



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ结合Spring开发-6

- ActiveMQ结合Spring开发最佳实践和建议

1：Camel框架支持大量的企业集成模式，可以大大简化集成组件间的大量服务和复杂的消息流。而Spring框架更注重简单性，仅仅支持基本的最佳实践。

2：Spring消息发送的核心架构是JmsTemplate，隔离了像打开、关闭Session和Producer的繁琐操作，因此应用开发人员仅仅需要关注实际的业务逻辑。但是JmsTemplate损害了ActiveMQ的PooledConnectionFactory对session和消息producer的缓存机制而带来的性能提升。

3：新的Spring里面，可以设置org.springframework.jms.connection.CachingConnectionFactory的sessionCacheSize ，或者干脆使用ActiveMQ的PooledConnectionFactory

4：不建议使用JmsTemplate的receive()调用，因为在JmsTemplate上的所有调用都是同步的，这意味着调用线程需要被阻塞，直到方法返回，这对性能影响很大

5：请使用DefaultMessageListenerContainer，它允许异步接收消息并缓存session和消息consumer，而且还可以根据消息数量动态的增加或缩减监听器的数量



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 连接到ActiveMQ

- Connector：ActiveMQ提供的，用来实现连接通讯的功能。包括：client-to-broker、broker-to-broker。 ActiveMQ允许客户端使用多种协议来连接
- 配置Transport Connector，在conf/activemq.xml里面，大致如下：

```
<transportConnectors>
    <transportConnector name="openwire" uri="tcp://localhost:61616"
        discoveryUri="multicast://default"/>
    <transportConnector name="ssl" uri="ssl://localhost:61617"/>
    <transportConnector name="stomp" uri="stomp://localhost:61613"/>
    <transportConnector name="xmpp" uri="xmpp://localhost:61222"/>
</transportConnectors>
```

- ActiveMQ支持的client-broker通讯协议如下：

1：TCP：这个也是缺省使用的协议

2：NIO　　3：UDP　　4：SSL　　5：Http（s）

6：VM：如果客户端和broker在一个虚拟机内的话，通过VM协议通讯在VM内通讯，从而减少网络传输的开销



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ支持的传输协议和配置-1

- Transmission Control Protocol (TCP)

1：这是默认的Broker配置，TCP的Client监听端口是61616。

2：在网络传输数据前，必须要序列化数据，消息是通过一个叫wire protocol的来序列化成字节流。默认情况下，ActiveMQ把wire protocol叫做OpenWire，它的目的是促使网络上的效率和数据快速交互。

3：TCP连接的URI形式：tcp://hostname:port?key=value&key=value，加粗部分是必须的

4：TCP传输的优点：

（1） TCP协议传输可靠性高，稳定性强

（2）高效性：字节流方式传递，效率很高

（3）有效性、可用性：应用广泛，支持任何平台

5：所有关于Transport协议的可配置参数，可以参见：

http://activemq.apache.org/configuring-version-5-transports.html

- New I/O API Protocol（NIO）

1：NIO协议和TCP协议类似，但NIO更侧重于底层的访问操作。它允许开发人员对同一资源可有更多的client调用和服务端有更多的负载。

做最好的在线学习社区
网 址：http://sishuok.com
咨询QQ：2371651507
私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ支持的传输协议和配置-2

2：适合使用NIO协议的场景：

（1）可能有大量的Client去链接到Broker上

一般情况下，大量的Client去链接Broker是被操作系统的线程数所限制的。因此，NIO的实现比TCP需要更少的线程去运行，所以建议使用NIO协议

（2）可能对于Broker有一个很迟钝的网络传输

NIO比TCP提供更好的性能

3：NIO连接的URI形式：nio://hostname:port?key=value

4：Transport Connector配置示例：

```
<transportConnectors>
    <transportConnector
        name="tcp"
        uri="tcp://localhost:61616?trace=true" />
    <transportConnector
        name="nio"
        uri="nio://localhost:61618?trace=true" />
</transportConnectors>
```

上面的配置，示范了一个TCP协议监听61616端口，一个NIO协议监听61618端口



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ支持的传输协议和配置-3

- User Datagram Protocol（UDP)

1：UDP和TCP的区别

（1）TCP是一个原始流的传递协议，意味着数据包是有保证的，换句话说，数据包是不会被复制和丢失的。UDP，另一方面，它是不会保证数据包的传递的

（2）TCP也是一个稳定可靠的数据包传递协议，意味着数据在传递的过程中不会被丢失。这样确保了在发送和接收之间能够可靠的传递。相反，UDP仅仅是一个链接协议，所以它没有可靠性之说

2：从上面可以得出：TCP是被用在稳定可靠的场景中使用的；UDP通常用在快速数据传递和不怕数据丢失的场景中，还有ActiveMQ通过防火墙时，只能用UDP

3：UDP连接的URI形式：udp://hostname:port?key=value

4：Transport Connector配置示例：

```
<transportConnectors>
    <transportConnector
          name="udp"
          uri="udp://localhost:61618?trace=true" />
</transportConnectors>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ支持的传输协议和配置-4

- Secure Sockets Layer Protocol (SSL)

1：连接的URI形式：ssl://hostname:port?key=value

2：Transport Connector配置示例：

```
<transportConnectors>
   <transportConnector name="ssl" uri="ssl://localhost:61617?trace=true"
   />
</transportConnectors>
```

- Hypertext Transfer Protocol (HTTP/HTTPS)

1：像web和email等服务需要通过防火墙来访问的，Http可以使用这种场合

2：连接的URI形式：http://hostname:port?key=value或者 https://hostname:port?key=value

3：Transport Connector配置示例：

```
<transportConnectors>
   <transportConnector name="http"
   uri="http://localhost:8080?trace=true" />
</transportConnectors>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ支持的传输协议和配置-5

- VM Protocol（VM）

1：VM transport允许在VM内部通信，从而避免了网络传输的开销。这时候采用的连接不是socket连接，而是直接的方法调用。

2：第一个创建VM连接的客户会启动一个embed VM broker，接下来所有使用相同的broker name的VM连接都会使用这个broker。当这个broker上所有的连接都关闭的时候，这个broker也会自动关闭。

3：连接的URI形式：vm://brokerName?key=value

4：Java中嵌入的方式：

vm:broker:(tcp://localhost:6000)?brokerName=embeddedbroker&persistent=false ， 定义了一个嵌入的broker名称为embededbroker以及配置了一个tcptransprotconnector在监听端口6000上

5：使用一个加载一个配置文件来启动broker

vm://localhost?brokerConfig=xbean:activemq.xml



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



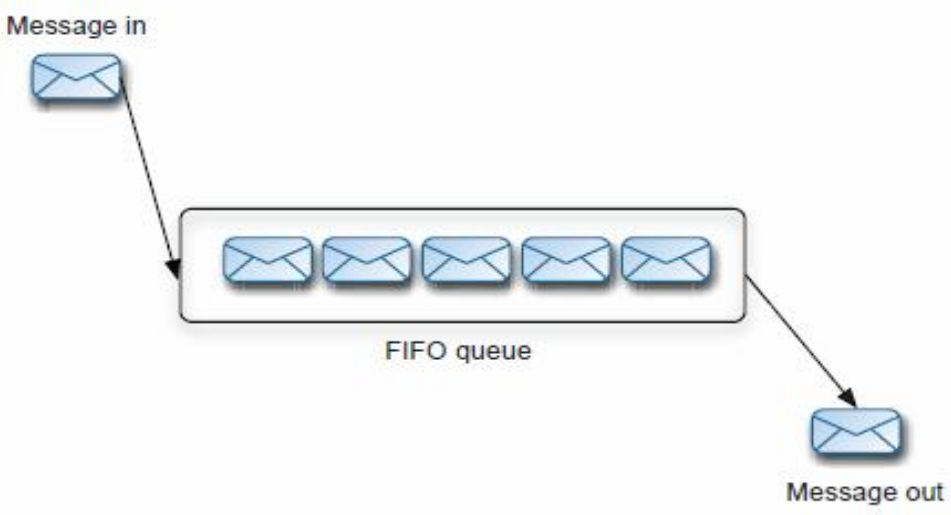
# ActiveMQ的消息存储持久化-1

- 概述

  ActiveMQ不仅支持persistent和non-persistent两种方式，还支持消息的恢复（ recovery ）方式。

- PTP

  Queue的存储是很简单的，就是一个FIFO的Queue






私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

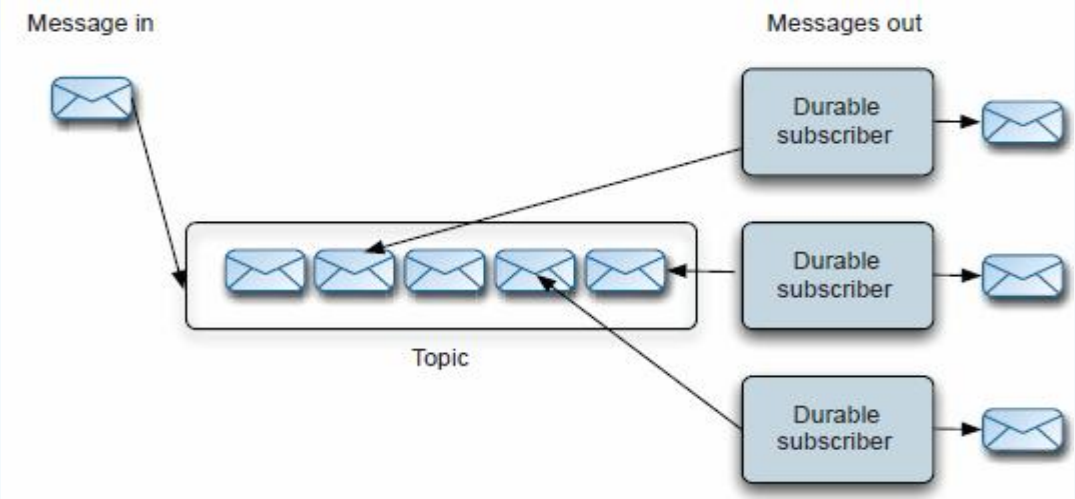
Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖！



# ActiveMQ的消息存储持久化-2

- PUB/SUB

  对于持久化订阅主题，每一个消费者将获得一个消息的复制。




- 有效的消息存储

  ActiveMQ提供了一个插件式的消息存储，类似于消息的多点传播，主要实现了如下几种：

1：AMQ消息存储-基于文件的存储方式，是以前的默认消息存储

2：KahaDB消息存储-提供了容量的提升和恢复能力，是现在的默认存储方式

3：JDBC消息存储-消息基于JDBC存储的

4：Memory 消息存储-基于内存的消息存储



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的消息存储持久化-3

- KahaDB Message Store概述

  KahaDB是目前默认的存储方式，可用于任何场景，提高了性能和恢复能力。消息存储使用一个事务日志和仅仅用一个索引文件来存储它所有的地址。

  KahaDB是一个专门针对消息持久化的解决方案，它对典型的消息使用模式进行了优化。在Kaha中，数据被追加到data logs中。当不再需要log文件中的数据的时候，log文件会被丢弃。

- KahaDB基本配置例子

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

可用的属性有：

1：director：KahaDB存放的路径，默认值activemq-data

2：indexWriteBatchSize：  批量写入磁盘的索引page数量，默认值1000

3：indexCacheSize：内存中缓存索引page的数量，默认值10000

4：enableIndexWriteAsync：是否异步写出索引，默认false

5：journalMaxFileLength：设置每个消息data log的大小，默认是32MB

6：enableJournalDiskSyncs：设置是否保证每个没有事务的内容，被同步写入磁盘，JMS持久化的时候需要，默认为true

7：cleanupInterval：在检查到不再使用的消息后，在具体删除消息前的时间，默认30000



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的消息存储持久化-4

8：checkpointInterval：checkpoint的间隔时间，默认5000

9：ignoreMissingJournalfiles：是否忽略丢失的消息日志文件，默认false

10：checkForCorruptJournalFiles：在启动的时候，将会验证消息文件是否损坏，默认false

11：checksumJournalFiles：是否为每个消息日志文件提供checksum，默认false

12：archiveDataLogs： 是否移动文件到特定的路径，而不是删除它们，默认false

13：directoryArchive：定义消息已经被消费过后，移动data log到的路径，默认null

14：databaseLockedWaitDelay：获得数据库锁的等待时间 (used by shared master/slave)，默认10000

15：maxAsyncJobs：设置最大的可以存储的异步消息队列，默认值10000，可以和concurrent MessageProducers 设置成一样的值

16：concurrentStoreAndDispatchTransactions：是否分发消息到客户端，同时事务存储消息，默认true

17：concurrentStoreAndDispatchTopics：是否分发Topic消息到客户端，同时进行存储，默认true

18：concurrentStoreAndDispatchQueues：是否分发queue消息到客户端，同时进行存储，默认true

- 在Java中内嵌使用Broker，使用KahaDB的例子



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的消息存储持久化-5

```
public class EmbeddedBrokerUsingKahaDBStoreExample {
    BrokerService createEmbeddedBroker() throws Exception {
          BrokerService broker = new BrokerService();
          File dataFileDir = new File("target/amq-in-action/kahadb");
          KahaDBStore kaha = new KahaDBStore();
          kaha.setDirectory(dataFileDir);
          // Using a bigger journal file
          kaha.setJournalMaxFileLength(1024*100);
          // small batch means more frequent and smaller writes
          kaha.setIndexWriteBatchSize(100);
          // do the index write in a separate thread
          kaha.setEnableIndexWriteAsync(true);
          broker.setPersistenceAdapter(kaha);
          //create a transport connector
          broker.addConnector("tcp://localhost:61616");
          //start the broker
          broker.start();
          return broker;
    }
}
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的消息存储持久化-6

- AMQ Message Store概述

  AMQ Message Store是ActiveMQ5.0缺省的持久化存储，它是一个基于文件、事务存储设计为快速消息存储的一个结构，该结构是以流的形式来进行消息交互的。

  这种方式中，Messages被保存到data logs中，同时被reference store进行索引以提高存取速度。Date logs由一些单独的data log文件组成，缺省的文件大小是32M，如果某个消息的大小超过了data log文件的大小，那么可以修改配置以增加data log文件的大小。如果某个data log文件中所有的消息都被成功消费了，那么这个data log文件将会被标记，以便在下一轮的清理中被删除或者归档。

- AMQ Message Store配置示例

```
<broker brokerName="broker" persistent="true" useShutdownHook="false">
    <persistenceAdapter>
        <amqPersistenceAdapter directory="${activemq.base}/data"
                                                maxFileLength="32mb"/>
    </persistenceAdapter>
</broker>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖！



# ActiveMQ的消息存储持久化-7

- 使用JDBC来持久化消息

ActiveMQ支持使用JDBC来持久化消息，预定义的表如下：

1：消息表，缺省表名为ACTIVEMQ_MSGS，queue和topic都存在里面，结构如下：

| Column name | Default type | Description |
|---|---|---|
| ID | INTEGER | The sequence ID used to retrieve the message. |
| CONTAINER | VARCHAR(250) | The destination of the message. |
| MSGID_PROD | VARCHAR(250) | The ID of the message producer. |
| MSGID_SEQ | INTEGER | The producer sequence number for the message. This together with the MSGID_PROD is equivalent to the JMS-MessageID. |
| EXPIRATION | BIGINT | The time in milliseconds when the message will expire. |
| MSG | BLOB | The serialized message itself. |

2：ACTIVEMQ_ACKS表存储持久订阅的信息和最后一个持久订阅接收的消息ID，结构如下：



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖！



# ActiveMQ的消息存储持久化-8

| Column name | Default type | Description |
|---|---|---|
| CONTAINER | VARCHAR(250) | The destination of the message |
| SUB_DEST | VARCHAR(250) | The destination of the durable subscriber (can be different from the container if using wildcards) |
| CLIENT_ID | VARCHAR(250) | The client ID of the durable subscriber |
| SUB_NAME | VARCHAR(250) | The subscriber name of the durable subscriber |
| SELECTOR | VARCHAR(250) | The selector of the durable subscriber |
| LAST_ACKED_ID | Integer | The sequence ID of last message received by this subscriber |

3：锁定表，缺省表名为ACTIVEMQ_LOCK，用来确保在某一时刻，只能有一个ActiveMQ broker实例来访问数据库 ，结构如下：

| Column name | Default type | Description |
|---|---|---|
| ID | INTEGER | A unique ID for the lock |
| Broker Name | VARCHAR(250) | The name of the ActiveMQ broker that has the lock |

做最好的在线学习社区
网　址：http://sishuok.com
咨询QQ：2371651507
私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的消息存储持久化-9

- 使用JDBC来持久化消息的配置示例

```
<beans>
    <broker brokerName="test-broker" persistent=true
    xmlns="http://activemq.apache.org/schema/core">
            <persistenceAdapter>
                    <jdbcPersistenceAdapter dataSource= “#mysql-ds"/>
            </persistenceAdapter>
    </broker>
<bean name="mysql-ds" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
            <property name="driverClassName"><value>org.gjt.mm.mysql.Driver</value></property>
            <property
    name="url"><value>jdbc:mysql://192.168.1.100:3306/test?useUnicode=true&amp;characterEncodi
    ng=UTF-8</value></property>
            <property name="username"> <value>root</value> </property>
            <property name="password" value="cc"/>
    </bean>
```

- JDBC Message Store with ActiveMQ Journal

  这种方式克服了JDBC Store的不足，使用快速的缓存写入技术，大大提高了性能。 配置示例如下：



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的消息存储持久化-10

```
<beans>
    <broker brokerName="test-broker" xmlns="http://activemq.apache.org/schema/core">
          <persistenceFactory>
                    <journalPersistenceAdapterFactory
                    journalLogFiles="4"
                    journalLogFileSize="32768"
                    useJournal="true"
                    useQuickJournal="true"
                    dataSource="#derby-ds"
                    dataDirectory="activemq-data" />
          </persistenceFactory>
    </broker>
</beans>
```

- JDBC Store和JDBC Message Store with ActiveMQ Journal的区别

1：Jdbc with journal的性能优于jdbc

2：Jdbc用于master/slave模式的数据库分享

3：Jdbc with journal不能用于master/slave模式

4：一般情况下，推荐使用jdbc with journal



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的消息存储持久化-11

- Memory Message Store

  内存消息存储主要是存储所有的持久化的消息在内存中。这里没有动态的缓存存在，所以你必须注意设置你的broker所在的JVM和内存限制

- Memory Message Store配置示例

```
<beans>
    <broker brokerName="test-broker" persistent="false"
    xmlns="http://activemq.apache.org/schema/core">
          <transportConnectors>
                  <transportConnector uri="tcp://localhost:61635"/>
          </transportConnectors>
    </broker>
</beans>
```

- 在Java中内嵌使用Broker，使用Memory的例子

```
public void createEmbeddedBroker() throws Exception {
    BrokerService broker = new BrokerService();
    broker.setPersistent(false);
    broker.addConnector("tcp://localhost:61616");
    broker.start();
}
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 在一台服务器上启动多个Broker

- 步骤如下：

1：把整个conf文件夹复制一份，比如叫做conf2

2：修改里面的activemq.xml文件

（1）里面的brokerName 不能跟原来的重复

（2）数据存放的文件名称不能重复，比如：

```
<kahaDB directory="${activemq.data}/kahadb_2"/>
```

（3）所有涉及的transportConnectors 的端口，都要跟前面的不一样

3：修改jetty.xml，主要就是修改端口，比如：

```
<property name=“port” value=“8181” />
```
端口必须和前面的不一样

4：到bin下面，复制一个activemq，比如叫做activemq2：

（1）修改程序的id，不能和前面的重复

```
ACTIVEMQ_PIDFILE="$ACTIVEMQ_DATA/activemq2-`hostname`.pid"
```

（2）修改配置文件路径

```
ACTIVEMQ_CONF="$ACTIVEMQ_BASE/conf2"
```

（3）修改端口，里面有个tcp的61616的端口，要改成不一样的，最好跟activemq.xml里面的tcp的端口一致

（4）然后就可以执行了，如果执行没有权限的话，就授权：chmod 751 activemq2



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!

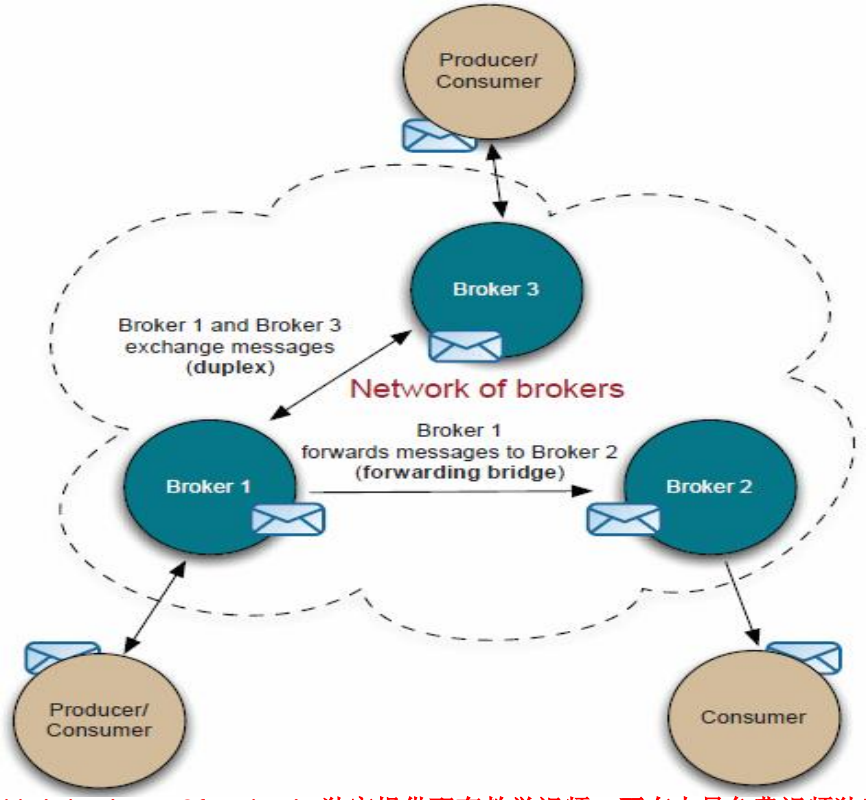


# ActiveMQ的静态网络链接-1

- ActiveMQ的networkConnector是什么

在某些场景下，需要多个ActiveMQ的Broker做集群，那么就涉及到Broker到Broker的通信，这个被称为ActiveMQ的networkConnector。

ActiveMQ的networkConnector默认是单向的，一个Broker在一端发送消息，另一Broker在另一端接收消息。这就是所谓的“桥接”。 ActiveMQ也支持双向链接，创建一个双向的通道对于两个Broker，不仅发送消息而且也能从相同的通道来接收消息，通常作为duplex connector来映射，如下：






私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的静态网络链接-2

- “discovery“的概念

  一般情况下，discovery是被用来发现远程的服务，客户端通常想去发现所有可利用的brokers；另一层意思，它是基于现有的网络Broker去发现其他可用的Brokers。

  有两种配置Client到Broker的链接方式，一种方式：Client通过Statically配置的方式去连接Broker，一种方式：Client通过discovery agents来dynamically的发现Brokers

- Static networks

  Static networkConnector是用于创建一个静态的配置对于网络中的多个Broker。这种协议用于复合url，一个复合url包括多个url地址。格式如下：

  static:(uri1,uri2,uri3,...)?key=value

1：配置示例如下：

```
<networkConnectors>
    <networkConnector name="local network"
    uri="static://(tcp://remotehost1:61616,tcp://remotehost2:61616)"/>
</networkConnectors>
```



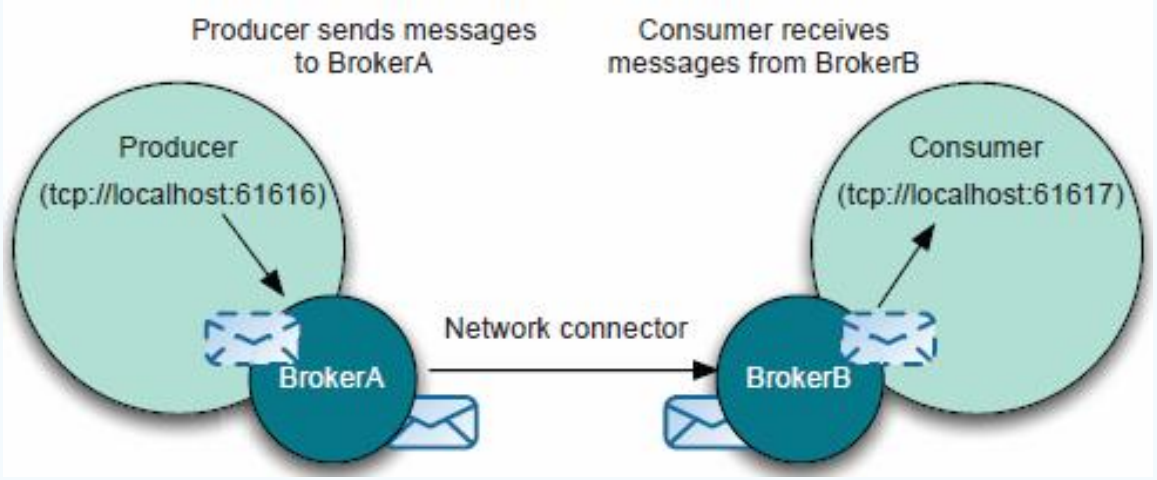
私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖！



# ActiveMQ的静态网络链接-3

- Static networkConnector的基本原理示意图：




上图中，两个Brokers是通过一个static的协议来网络链接的。一个Consumer链接到brokerB的一个地址上 ， 当Producer在brokerA上以相同的地址发送消息时，此时它将被转移到brokerB上。也就是，BrokerA会转发消息到BrokerB上。



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的静态网络链接-4

n networkConnector配置的可用属性：

1：name：默认是bridge

2：dynamicOnly：默认是false，如果为true，持久订阅被激活时才创建对应的网路持久订阅。默认是启动时激活

3：decreaseNetworkConsumerPriority：默认是false。设定消费者优先权，如果为true，网络的消费者优先级降低为-5。如果为false，则默认跟本地消费者一样为0

4：networkTTL ：默认是1 ，网络中用于消息和订阅消费的broker数量

5：messageTTL ：默认是1 ，网络中用于消息的broker数量

6：consumerTTL：默认是1 ，网络中用于消费的broker数量

7：conduitSubscriptions ：默认true，是否把同一个broker的多个consumer当做一个来处理

8：dynamicallyIncludedDestinations ：默认为空，要包括的动态消息地址，类似于excludedDestinations，如：

```
<dynamicallyIncludedDestinations>
    <queue physicalName="include.test.foo"/>
    <topic physicalName="include.test.bar"/>
</dynamicallyIncludedDestinations>
```

9：staticallyIncludedDestinations ：默认为空，要包括的静态消息地址。类似于excludedDestinations，如：

```
<staticallyIncludedDestinations>
    <queue physicalName="always.include.queue"/>
</staticallyIncludedDestinations>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的静态网络链接-5

10：excludedDestinations ：默认为空，指定排除的地址，示例如下：

```
<networkConnectors>
    <networkConnector uri="static:///(tcp://localhost:61617)"
            name="bridge" dynamicOnly="false" conduitSubscriptions="true"
            decreaseNetworkConsumerPriority="false">
            <excludedDestinations>
                    <queue physicalName="exclude.test.foo"/>
                    <topic physicalName="exclude.test.bar"/>
            </excludedDestinations>
            <dynamicallyIncludedDestinations>
                    <queue physicalName="include.test.foo"/>
                    <topic physicalName="include.test.bar"/>
            </dynamicallyIncludedDestinations>
            <staticallyIncludedDestinations>
                    <queue physicalName="always.include.queue"/>
                    <topic physicalName="always.include.topic"/>
            </staticallyIncludedDestinations>
    </networkConnector>
</networkConnectors>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的静态网络链接-6

11：duplex ：默认false，设置是否能双向通信

12：prefetchSize ：默认是1000，持有的未确认的最大消息数量，必须大于0，因为网络消费者不能自己轮询消息

13：suppressDuplicateQueueSubscriptions：默认false，如果为true，重复的订阅关系一产生即被阻止

14：bridgeTempDestinations ：默认true，是否广播advisory messages来创建临时destination

15：alwaysSyncSend ：默认false，如果为true，非持久化消息也将使用request/reply方式代替oneway方式发送到远程broker。

16：staticBridge ：默认false，如果为true，只有staticallyIncludedDestinations中配置的destination可以被处理。



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的静态网络链接-7

n “丢失”的消息

有这样的场景，broker1和broker2通过networkConnector连接，一些consumers连接到broker1，消费broker2上的消息。消息先被broker1从broker2上消费掉，然后转发给这些consumers。不幸的是转发部分消息的时候broker1重启了，这些consumers发现broker1连接失败，通过failover连接到broker2上去了，但是有一部分他们还没有消费的消息被broker2已经分发到了broker1上去了。这些消息，就好像是消失了，除非有消费者重新连接到broker1上来消费。怎么办呢？

从5.6版起，在destinationPolicy上新增的选项replayWhenNoConsumers。这个选项使得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做重复消息而不被分发，示例如下：

```
<destinationPolicy>
    <policyMap>
          <policyEntries>
                    <policyEntry queue=">" enableAudit="false">
                              <networkBridgeFilterFactory>
          <conditionalNetworkBridgeFilterFactory replayWhenNoConsumers="true"/>
                              </networkBridgeFilterFactory>
                    </policyEntry>
          </policyEntries>
    </policyMap>
</destinationPolicy>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 容错的链接-1

- Failover Protocol

  前面讲述的都是Client配置链接到指定的broker上。但是，如果Broker的链接失败怎么办呢？此时，Client有两个选项：要么立刻死掉，要么去连接到其它的broker上。

  Failover协议实现了自动重新链接的逻辑。这里有两种方式提供了稳定的brokers列表对于Client链接。第一种方式：提供一个static的可用的Brokers列表。第二种方式：提供一个dynamic 发现的可用Brokers。

- Failover Protocol 的配置方式

  failover:(uri1,...,uriN)?key=value　　或者 failover:uri1,...,uriN

- Failover Protocol 的默认配置

  默认情况下，这种协议用于随机的去选择一个链接去链接，如果链接失败了，那么会链接到其他的Broker上。默认的配置定义了延迟重新链接，意味着传输将会在10秒后自动的去重新链接可用的broker。当然所有的重新链接参数都可以根据应用的需要而配置。

- Failover Protocol 的使用示例，在客户端程序里面：

```
ConnectionFactory connectionFactory = new
    ActiveMQConnectionFactory("failover:(tcp://192.168.1.106:61679,tcp://192.168.1.106:61819)
    ?randomize=false");
```

- Failover Protocol 可用的配置参数：



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 容错的链接-2

1：initialReconnectDelay：在第一次尝试重连之前等待的时间长度（毫秒），默认10
2：maxReconnectDelay：最长重连的时间间隔（毫秒），默认30000
3：useExponentialBackOff：重连时间间隔是否以指数形式增长，默认true
4：backOffMultiplier：递增倍数，默认2.0
5：maxReconnectAttempts： 默认-1|0，自版本5.6起：-1为默认值，代表不限重试次数；0代表从不重试（只尝试连接一次，并不重连），5.6以前的版本：0为默认值，代表不限重试次数所有版本：如果设置为大于0的数，代表最大重试次数
6：startupMaxReconnectAttempts：初始化时的最大重连次数。一旦连接上，将使用maxReconnectAttempts的配置，默认0
7：randomize：使用随机链接，以达到负载均衡的目的，默认true
8：backup：提前初始化一个未使用连接，以便进行快速失败转移，默认false
9：timeout：设置发送操作的超时时间（毫秒），默认-1
10：trackMessages：设置是否缓存[故障发生时]尚未传送完成的消息，当broker一旦重新连接成功，便将这些缓存中的消息刷新到新连接的代理中，使得消息可以在broker切换前后顺利传送，默认false
11：maxCacheSize：当trackMessages启用时，缓存的最大字节，默认为128*1024bytes
12：updateURIsSupported：设定是否可以动态修改broker uri（自版本5.4起），默认true



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的动态网络链接-1

- 多播协议multicast

  ActiveMQ使用Multicast 协议将一个Service和其他的Broker的Service连接起来。IP multicast是一个被用于网络中传输数据到其它一组接收者的技术。Ip multicast传统的概念称为组地址。组地址是ip地址在224.0.0.0到239.255.255.255之间的ip地址。 ActiveMQ broker使用multicast协议去建立服务与远程的broker的服务的网络链接。

- 基本的格式配置

  multicast://ipadaddress:port?transportOptions

  transportOptions如下：

1：group：表示唯一的组名称，缺省值default
2：minmumWireFormatVersion：被允许的最小的wireformat版本，缺省为0
3：trace：是否追踪记录日志，默认false
4：useLocalHost：表示本地机器的名称是否为localhost，默认true
5：datagramSize：特定的数据大小，默认值4 * 1024
6：timeToLive：消息的生命周期，默认值-1
7：loopBackMode：是否启用loopback模式，默认false
8：wireFormat：默认用wireFormat命名
9：wireFormat.*：前缀是wireFormat



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的动态网络链接-2

- 配置示例

1：默认配置，请注意，默认情况下是不可靠的多播，数据包可能会丢失

multicast://default

2：特定的ip和端口

multicast://224.1.2.3:6255

3：特定的ip和端口以及组名

multicast://224.1.2.3:6255?group=mygroupname

- Activemq使用multicast协议的配置格式如下

```
<broker xmlns="http://activemq.apache.org/schema/core" brokerName="multicast"
    dataDirectory="${activemq.base}/data">
    <networkConnectors>
          <networkConnector name="default-nc" uri="multicast://default"/>
    </networkConnectors>
    <transportConnectors>
          <transportConnector name="openwire" uri="tcp://localhost:61616"
          discoveryUri="multicast://default"/>
    </transportConnectors>
</broker>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的动态网络链接-3

- 上面的配置说明

1：uri=“multicast://default”中的default是activemq默认的ip，默认动态的寻找地址

2：“discoveryUri”是指在transport中用multicast的default的地址传递

3：“uri”指动态寻找可利用的地址

4：如何防止自动的寻找地址？

（1）名称为openwire的transport，移除discoveryUri=”multicast://default”即可。传输链接用默认的名称openwire来配置broker的tcp多点链接，这将允许其它broker能够自动发现和链接到可用的broker中。

（2）名称为“default-nc”的networkConnector，注释掉或者删除即可。

ActiveMQ默认的networkConnector基于multicast协议的链接的默认名称是default-nc，而且自动的去发现其他broker。去停止这种行为，只需要注销或者删除掉default-nc网络链接。

（3）使brokerName的名字唯一，可以唯一识别Broker的实例，默认是localhost

- Multicast 协议和普通的tcp协议

它们是差不多的，不同的是Multicast能够自动的发现其他broker，从而替代了使用static功能列表brokers。用multicast协议可以在网络中频繁的添加和删除ip不会有影响。

multicast协议的好处是：能够适应动态变化的地址。

缺点：自动的链接地址和过度的销耗网络资源。



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的动态网络链接-4

- Discovery协议

  Discovery是在multicast协议的功能上定义的。功能类似与failover功能。它将动态的发现multicast 协议的broker的链接并且随机的链接其中一个broker。

- 基本配置格式如下：

discovery:(discoveryAgentURI)?transportOptions

transportOptions如下：

1：reconnectDelay：再次寻址等待时间，缺省值10

2：initialReconnectDelay：初始化设定再次寻址等待时间，缺省值10

3：maxReconnectDelay：最大寻址等待时间， 缺省值30000

4：useExponentialBackOff：是否尝试BackOff重链接，默认是true

5：backOffMultiplier：尝试Backoff的次数，默认是2

6：maxReconnectAttempts：如果异常，最大的重新链接个数，默认是0

7：group：组唯一的地址，默认是default

示例：

discovery:(multicast://default)?initialReconnectDelay=100



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的动态网络链接-5

- Discovery协议的配置示例

```
<broker name="foo">
    <transportConnectors>
        <transportConnector uri="tcp://localhost:0"
    discoveryUri="multicast://default"/>
    </transportConnectors>
</broker>
```

- Peer协议

ActiveMQ提出了peer transport connector 以让你更加容易的去嵌入broker中网络中。它将创建一个优于vm链接的p2p网络链接。 默认格式如下：

peer://peergroup/brokerName?key=value

- Peer协议基本使用

当我们启动了用peer协议时，应用将自动的启动内嵌broker，也将会自动的去配置其它broker来建立链接，当然了，前提是必须属于一个组。 配置如下：

peer://groupa/broker1?persistent=false

另外，生产者和消费者都各自链接到嵌入到自己应用的broker，并且在在本地的同一个组名中相互访问数据。



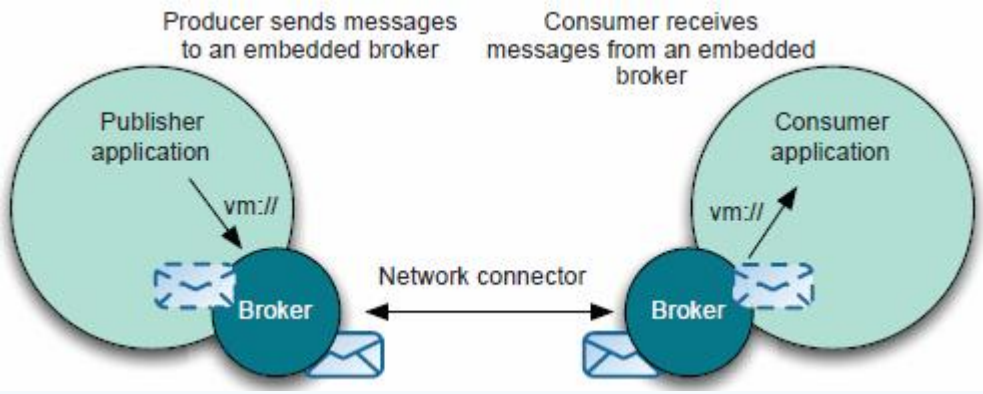
私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的动态网络链接-6

- Peer协议的基本原理示意图




在本地机器断网的情况下，本地的client访问本地brokerA将任然正常。在断网的情况下发送消息到本地brokerA，然后网路链接正常后，所有的消息将重新发送并链接到brokerB

- Fanout协议

Fanout协议是同时链接多个broker，默认的格式如下：

fanout:(fanoutURI)?key=value

示例：fanout:(static:(tcp://host1:61616,tcp://host2:61616,tcp://host3:61616))

表示client将试图链接到三个static列表中定义的三个URI



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的动态网络链接-7

- Fanout协议的配置方式如下：

  fanout:(discoveryURI)?transportOptions

  transportOptions如下：

1：initialReconnectDelay：重新链接的等待时间，默认是10

2：maxReconnectDelay：最大重新链接的等待时间，默认是30000

3：useExponentialBackOff：是否尝试BackOff重链接，默认是true

4：backOffMultiplier：尝试Backoff的次数，默认是2

5：maxReconnectAttempts：如果异常，最大的重新链接个数，默认是0

6：fanOutQueues：是否将topic消息转换queue消息，默认false

7：minAckCount：Broker链接的最小数，默认是2

配置示例：

```
fanout:(static:(tcp://localhost:61616,tcp://remotehost:61616))?initialReconne
    ctDelay=100
```

- 特别提醒

  Activemq不推荐使Consumer使用fanout协议。当Provider发送消息到多个broker中，测试Consumer可能收到重复的消息



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的集群-1

- Queue consumer clusters

  ActiveMQ支持Consumer对消息高可靠性的负载平衡消费，如果一个Consumer死掉，该消息会转发到其它的Consumer消费的Queue上。如果一个Consumer获得消息比其它Consumer快，那么他将获得更多的消息。因此推荐ActiveMQ的Broker和Client使用failover：//transport的方式来配置链接。

- Broker clusters

  大部情况下是使用一系列的Broker和Client链接到一起。如果一个Broker死掉了，Client可以自动链接到其它Broker上。实现以上行为需要用failover协议作为Client。

  如果启动了多个Broker，Client可以使用static discover或者 Dynamic discovery容易的从一个broker到另一个broker直接链接。

  这样当一个broker上没有Consumer的话，那么它的消息不会被消费的，然而该broker会通过存储和转发的策略来把该消息发到其它broker上。

  **特别注意：**ActiveMQ默认的两个broker，static链接后是单方向的，broker-A可以访问消费broker-B的消息，如果要支持双向通信，需要在netWorkConnector配置的时候，设置duplex=true 就可以了。



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的集群-2

- Master Slave

在5.9的版本里面，废除了Pure Master Slave的方式，目前支持：

1：Shared File System Master Slave：基于共享储存的Master-Slave：多个broker实例使用一个存储文件，谁拿到文件锁就是master，其他处于待启动状态，如果master挂掉了，某个抢到文件锁的slave变成master

2：JDBC Master Slave：基于JDBC的Master-Slave：使用同一个数据库，拿到LOCK表的写锁的broker成为master

3：Replicated LevelDB Store：基于ZooKeeper复制LevelDB存储的Master-Slave机制，这个是5.9新加的

具体的可以到官方察看： http://activemq.apache.org/masterslave.html

- JDBC Master Slave的方式

利用数据库作为数据源，采用Master/Slave模式，其中在启动的时候Master首先获得独有锁，其它Slaves Broker则等待获取独有锁。

推荐客户端使用Failover来链接Brokers。

具体如下图所示：

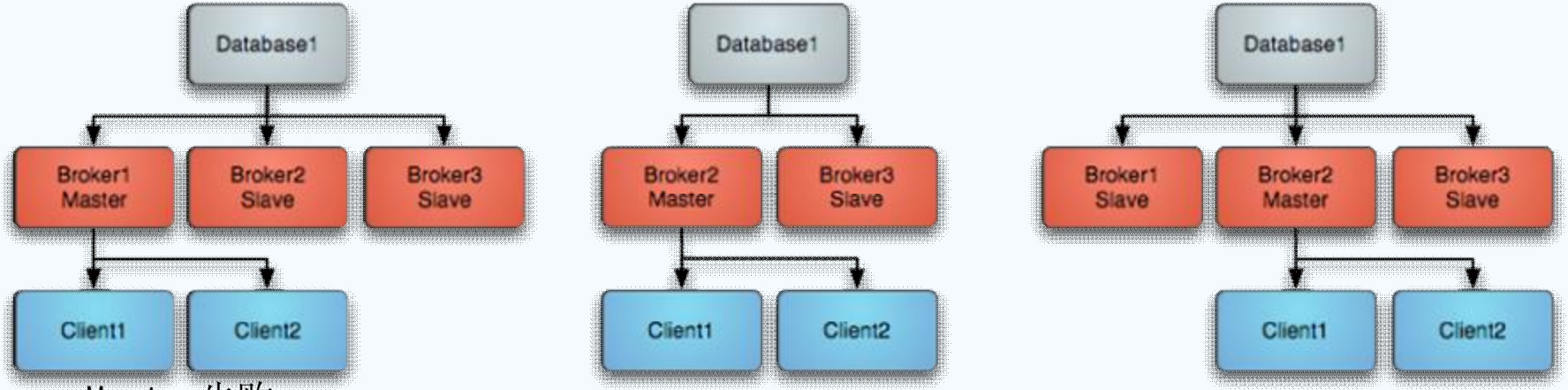


私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ的集群-3




- Master失败

  如果Master失败，则它释放独有锁，其他Slaver则获取独有锁，其它Slaver立即获得独有锁后此时它将变成Master，并且启动所有的传输链接。同时，Client将停止链接之前的Master和将会轮询链接到其他可以利用的Broker即新Master。如上中图所示

- Master重启

  任何时候去启动新的Broker，即作为新的Slave来加入集群，如上右图所示

- JDBC Master Slave的配置

  使用<jdbcPersistenceAdapter/>来配置消息的持久化，自动就会使用JDBC Master Slave的方式。



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Destination高级特性之Wildcards

- Wildcards用来支持名字分层体系，它不是JMS规范的一部分，是ActiveMQ的扩展。
  ActiveMQ支持以下三种wildcards：

1：“.” 用于作为路径上名字间的分隔符
2：“*” 用于匹配路径上的任何名字
3：">" 用于递归地匹配任何以这个名字开始的destination

- 示例，设想你有如下两个destinations
  PRICE.STOCK.NASDAQ.IBM （IBM在NASDAQ的股价）
  PRICE.STOCK.NYSE.SUNW （SUN在纽约证券交易所的股价）
  那么：

1：PRICE.> ：匹配任何产品的价格变动
2：PRICE.STOCK.> ：匹配任何产品的股票价格变动
3：PRICE.STOCK.NASDAQ.* ：匹配任何在NASDAQ下面的产品的股票价格变动
4：PRICE.STOCK.*.IBM：匹配任何IBM的产品的股票价格变动

- 客户化路径分隔符，比如你想要用 “/” 来替换 “.”

```
<plugins>
    <destinationPathSeparatorPlugin/>
</plugins>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Destination高级特性之Composite Destinations-1

**n** 组合队列 Composite Destinations

组合队列允许用一个虚拟的destination代表多个destinations。这样就可以通过composite destinations在一个操作中同时向多个queue发送消息。

1：客户端实现的方式

在composite destinations中，多个destination之间采用“,”分割。例如：

```
Queue queue = new ActiveMQQueue("FOO.A,FOO.B,FOO.C");
```

如果你希望使用不同类型的destination，那么需要加上前缀如queue:// 或topic://，例如：

```
Queue queue = new ActiveMQQueue("FOO.A,topic://NOTIFY.FOO.A");
```

2：在xml配置实现的方式

```
<destinationInterceptors>
    <virtualDestinationInterceptor>
        <virtualDestinations>
            <compositeQueue name="MY.QUEUE">
                <forwardTo>
                    <queue physicalName="my-queue" />
                    <queue physicalName="my-queue2" />
                </forwardTo>
            </compositeQueue>
        </virtualDestinations>
    </virtualDestinationInterceptor>
</destinationInterceptors>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Destination高级特性之Composite Destinations-2

3：使用filtered destinations，在xml配置实现的方式

```
<destinationInterceptors>
    <virtualDestinationInterceptor>
        <virtualDestinations>
            <compositeQueue name="MY.QUEUE">
                <forwardTo>
        <filteredDestination selector="odd = 'yes'" queue="FOO"/>
        <filteredDestination selector="i = 5" topic="BAR"/>
                </forwardTo>
            </compositeQueue>
        </virtualDestinations>
    </virtualDestinationInterceptor>
</destinationInterceptors>
```

4：避免在network连接broker中，出现重复消息

```
<networkConnectors>
    <networkConnector uri="static://(tcp://localhost:61617)">
        <excludedDestinations>
            <queue physicalName="Consumer.*.VirtualTopic.>"/>
        </excludedDestinations>
    </networkConnector>
</networkConnectors>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Destination高级特性之Configure Startup Destinations

- Configure Startup Destinations

  如果需要在ActiveMQ启动的时候，创建Destination的话，可以如下配置：

```
<broker xmlns="http://activemq.apache.org/schema/core">
    <destinations>
        <queue physicalName="FOO.BAR" />
        <topic physicalName="SOME.TOPIC" />
    </destinations>
</broker>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Destination高级特性之Delete Inactive Destinations

- Delete Inactive Destinations

一般情况下，ActiveMQ的queue在不使用之后，可以通过web控制台或是JMX方式来删除掉。当然，也可以通过配置，使得broker可以自动探测到无用的队列（一定时间内为空的队列）并删除掉，回收响应资源。可以如下配置：

```
<broker xmlns="http://activemq.apache.org/schema/core"
        schedulePeriodForDestinationPurge="10000">
    <destinationPolicy>
        <policyMap>
            <policyEntries>
                <policyEntry queue=">" gcInactiveDestinations="true"
    inactiveTimoutBeforeGC="30000"/>
            </policyEntries>
        </policyMap>
    </destinationPolicy>
</broker>
```

说明：

schedulePeriodForDestinationPurge：设置多长时间检查一次，这里是10秒，默认为0

inactiveTimoutBeforeGC：设置当Destination为空后，多长时间被删除，这里是30秒，默认为60

gcInactiveDestinations： 设置删除掉不活动队列，默认为false



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Destination高级特性之Destination Options

- Destination Options

队列选项是给consumer在JMS规范之外添加的功能特性，通过在队列名称后面使用类似URL的语法添加多个选项。包括：

1：consumer.prefetchSize，consumer持有的未确认最大消息数量，默认值 variable

2：consumer.maximumPendingMessageLimit：用来控制非持久化的topic在存在慢消费者的情况下，丢弃的数量，默认0

3：consumer.noLocal ：默认false

4：consumer.dispatchAsync ：是否异步分发 ，默认true

5：consumer.retroactive：是否为回溯消费者 ，默认false

6：consumer.selector：Jms的Selector，默认null

7：consumer.exclusive：是否为独占消费者 ，默认false

8：consumer.priority：设置消费者的优先级，默认0

使用示例：

```
queue = new
    ActiveMQQueue("TEST.QUEUE?consumer.dispatchAsync=false&consumer.prefetchSize=10");co
    nsumer = session.createConsumer(queue);
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Destination高级特性之Visual Destinations-1

- 概述

虚拟Destinations用来创建逻辑Destinations，客户端可以通过它来生产和消费消息，它会把消息映射到物理Destinations。ActiveMQ支持两种方式：

1：虚拟主题（Virtual Topics）
2：组合 Destinations（Composite Destinations）

- 为何使用虚拟主题

ActiveMQ中，topic只有在持久订阅下才是持久化的。持久订阅时，每个持久订阅者，都相当于一个queue的客户端，它会收取所有消息。这种情况下存在两个问题：

1：同一应用内consumer端负载均衡的问题：也即是同一个应用上的一个持久订阅不能使用多个consumer来共同承担消息处理功能。因为每个consumer都会获取所有消息。

queue模式可以解决这个问题，但broker端又不能将消息发送到多个应用端。所以，既要发布订阅，又要让消费者分组，这个功能JMS规范本身是没有的。

2：同一应用内consumer端failover的问题：由于只能使用单个的持久订阅者，如果这个订阅者出错，则应用就无法处理消息了，系统的健壮性不高

为了解决这两个问题，ActiveMQ中实现了虚拟Topic的功能



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Destination高级特性之Visual Destinations-2

- 如何使用虚拟主题

1：对于消息发布者来说，就是一个正常的Topic，名称以VirtualTopic.开头。例如VirtualTopic.Orders，代码示例如下：

```
Topic destination = session.createTopic("VirtualTopic.Orders");
```

2：对于消息接收端来说，是个队列，不同应用里使用不同的前缀作为队列的名称，即可表明自己的身份即可实现消费端应用分组。

例如Consumer.A.VirtualTopic.Orders，说明它是名称为A的消费端，同理Consumer.B.VirtualTopic.Orders说明是一个名称为B的客户端。可以在同一个应用里使用多个consumer消费此queue，则可以实现上面两个功能。

又因为不同应用使用的queue名称不同（前缀不同），所以不同的应用中都可以接收到全部的消息。每个客户端相当于一个持久订阅者，而且这个客户端可以使用多个消费者共同来承担消费任务。

代码示例如下：

```
Destination destination = session.createQueue("Consumer.A.VirtualTopic.Orders");
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Destination高级特性之Visual Destinations-3

3：默认虚拟主题的前缀是：VirtualTopic.>

自定义消费虚拟地址默认格式：Consumer.*.VirtualTopic.>

自定义消费虚拟地址可以改，比如下面的配置就把它修改了。

xml配置示例如下：

```
<broker xmlns="http://activemq.apache.org/schema/core">
    <destinationInterceptors>
        <virtualDestinationInterceptor>
            <virtualDestinations>
<virtualTopic name=">" prefix="VirtualTopicConsumers.*." selectorAware="false"/>
            </virtualDestinations>
        </virtualDestinationInterceptor>
    </destinationInterceptors>
</broker>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Destination高级特性之Mirrored Queues

- 概述

  ActiveMQ中每个queue中的消息只能被一个consumer消费。然而，有时候你可能希望能够监视生产者和消费者之间的消息流。你可以通过使用Virtual Destinations 来建立一个virtual queue 来把消息转发到多个queues中。但是 为系统中每个queue都进行如此的配置可能会很麻烦。

- 使用

  ActiveMQ支持Mirrored Queues。Broker会把发送到某个queue的所有消息转发到一个名称类似的topic，因此监控程序只需要订阅这个mirrored queue topic。为了启用Mirrored Queues，首先要将BrokerService的useMirroredQueues属性设置成true，然后可以通过destinationInterceptors设置其它属性，如mirror topic的前缀，缺省是“VirtualTopic.Mirror.”。

  比如修改后缀的配置示例如下：

```
<destinationInterceptors>
   <mirroredQueue copyMessage="true" postfix=".qmirror" prefix=""/>
</destinationInterceptors>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Destination高级特性之Per Destination Policies

- ActiveMQ支持多种不同的策略，来单独配置每一个Destination
  它的属性很多，可以参见官方文档：
  http://activemq.apache.org/per-destination-policies.html

  官方文档最后还给了一个例子做参考。



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖！



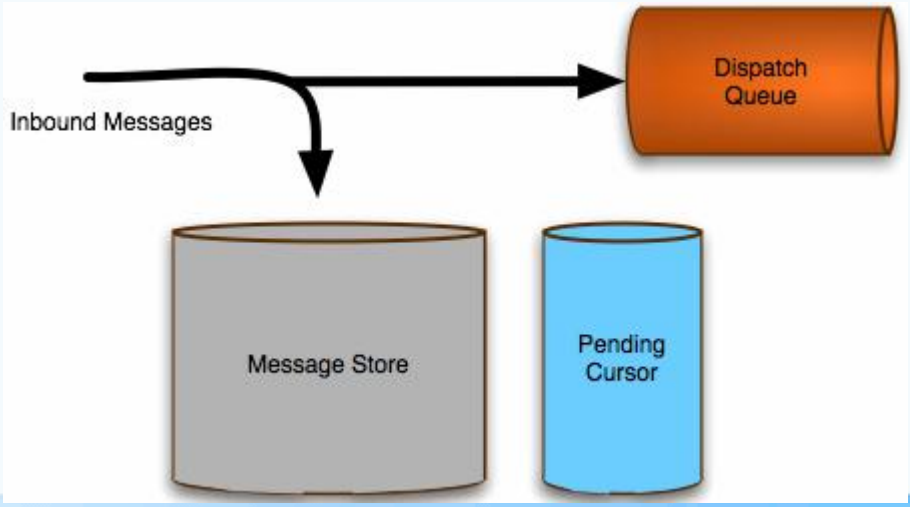
# Message Dispatch高级特性之Message Cursors-1

- 概述

  ActiveMQ发送持久消息的典型处理方式是：当消息的消费者准备就绪时，消息发送系统把存储的消息按批次发送给消费者，在发送完一个批次的消息后，指针的标记位置指向下一批次待发送消息的位置，进行后续的发送操作。这是一种比较健壮和灵活的消息发送方式，但大多数情况下，消息的消费者不是一直处于这种理想的活跃状态。

  因此，从ActiveMQ5.0.0版本开始，消息发送系统采用一种混合型的发送模式，当消息消费者处理活跃状态时，允许消息发送系统直接把持久消息发送给消费者，当消费者处于不活跃状态下，切换使用Cursors来处理消息的发送。

- 当消息消费者处于活跃状态并且处理能力比较强时，被持久存储的消息直接被发送到与消费者关联的发送队列，见下图





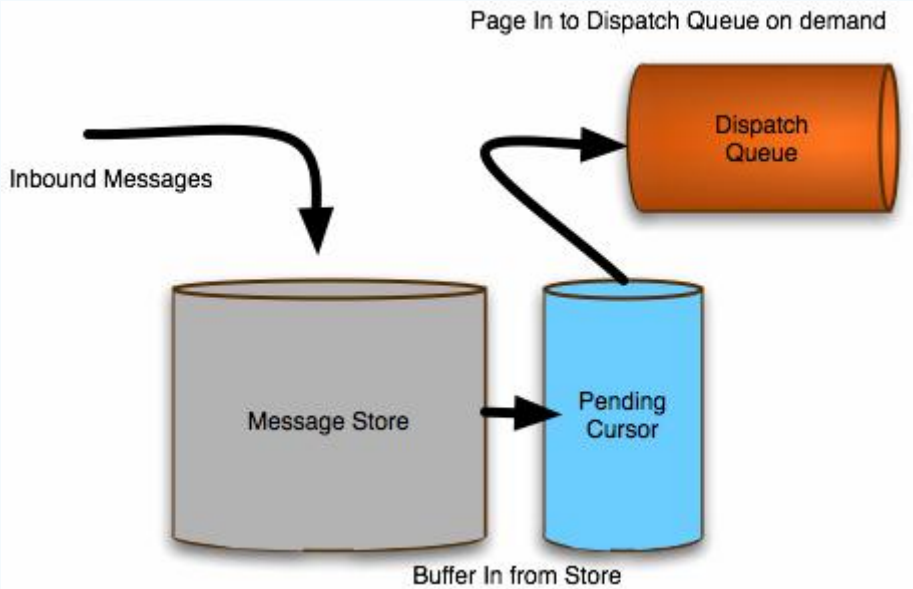

私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message Dispatch高级特性之Message Cursors-2

- 当消息已经出现积压，消费者再开始活跃；或者消费者的消费速度比消息的发送速度慢时，消息将从Pending Cursor中提取，并发送与消费者关联的发送队列。见下图




- Message Cursors分成三种类型

1：Store-based

2：VM

3：File-based



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

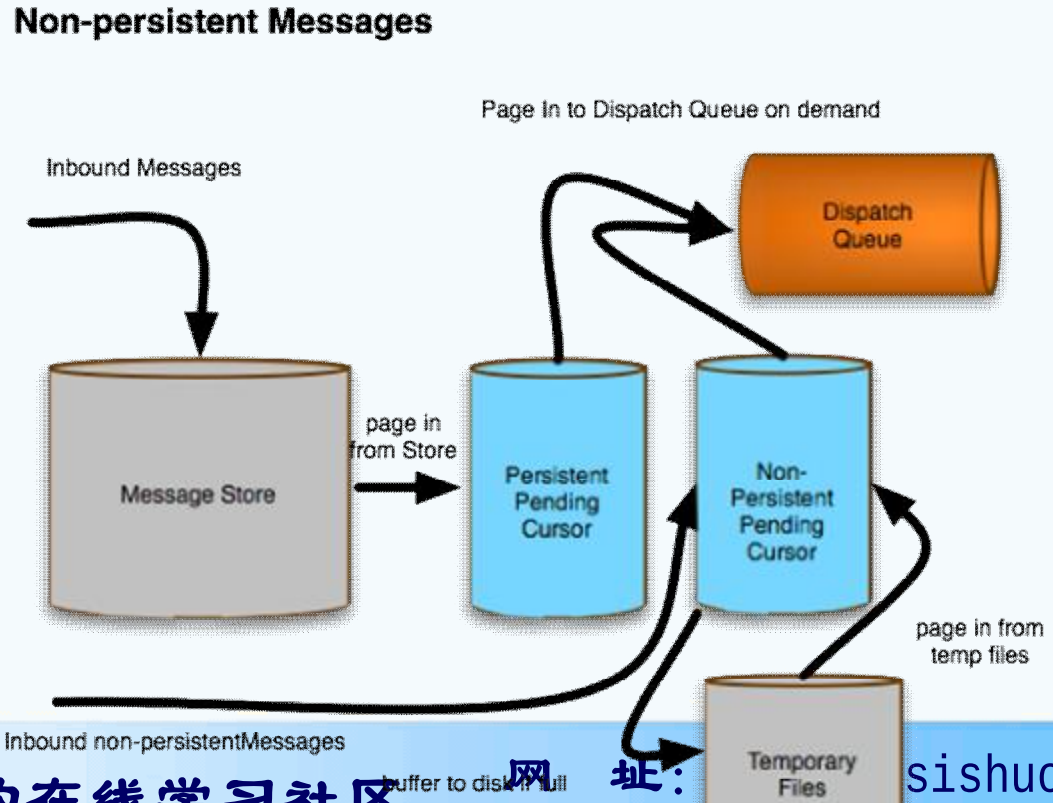
Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message Dispatch高级特性之Message Cursors-3

- Store-based

从activemq5.0开始，默认使用此种类型的cursor，其能够满足大多数场景的使用要求。同时支持非持久消息的处理，Store-based内嵌了File-based的模式，非持久消息直接被Non-Persistent Pending Cursor所处理。工作模式见下图




做最好的在线学习社区
网 址: www.sishuok.com
咨询QQ: 2371651507
私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

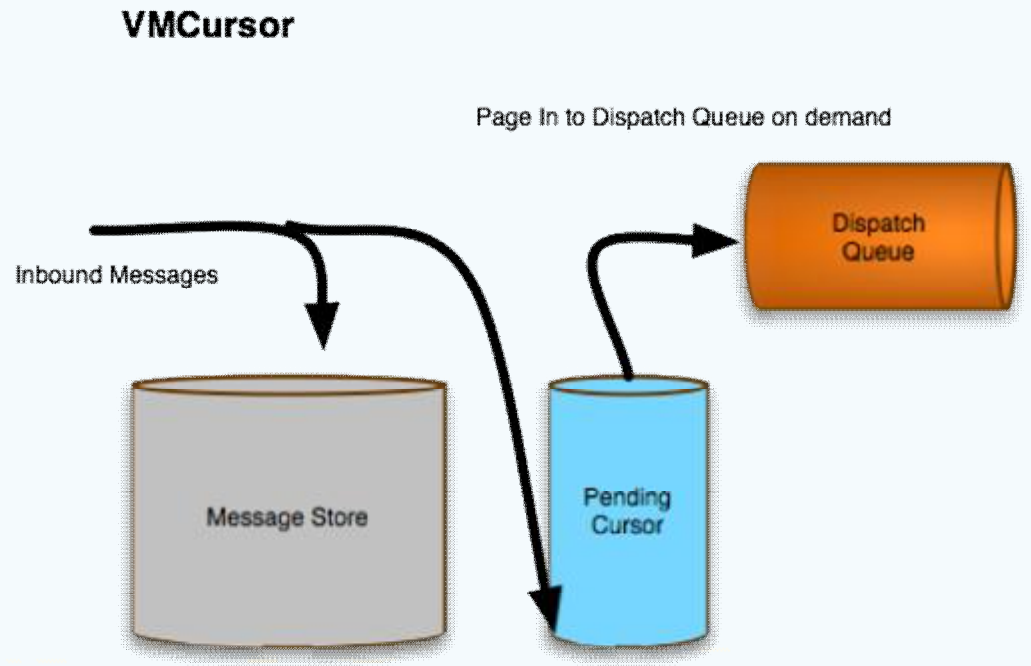
Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message Dispatch高级特性之Message Cursors-4

- VM

相关的消息引用存储在内存中，当满足条件时，消息直接被发送到消费者与之相关的发送队列，处理速度非常快，但出现慢消费者或者消费者长时间处于不活跃状态的情况下，无法适应。工作模式见下图：






私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

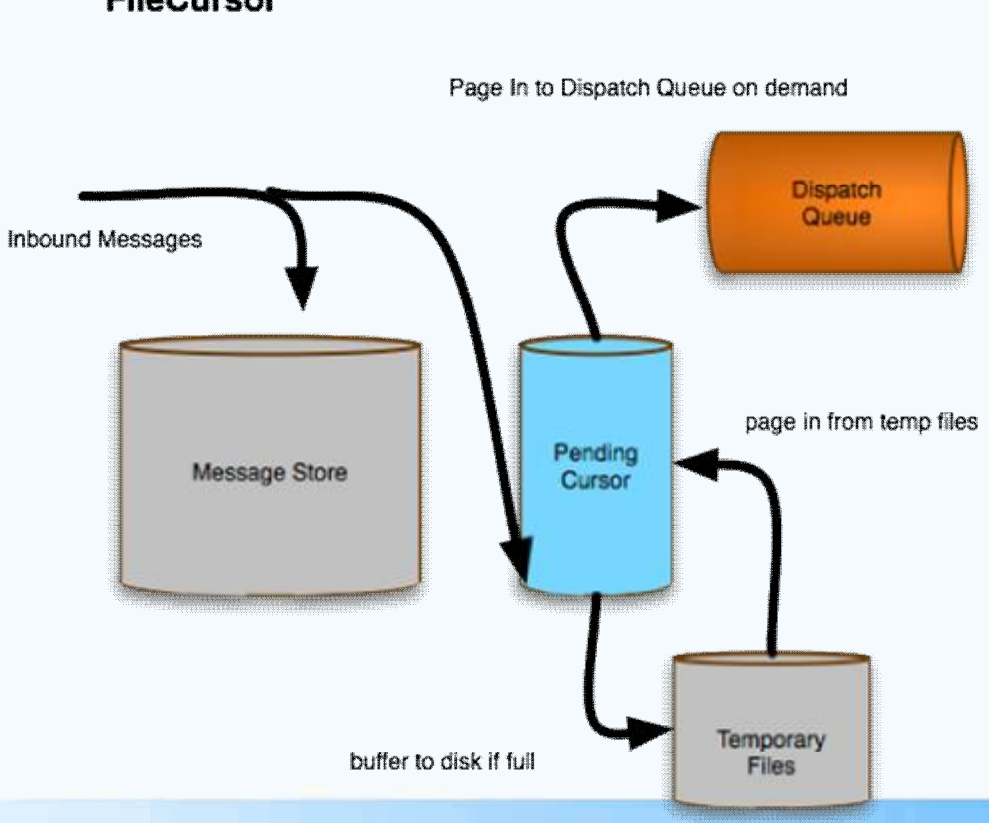
Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message Dispatch高级特性之Message Cursors-5

- File-based

当内存设置达到设置的限制，消息被存储到磁盘中的临时文件中。工作模式见下图：






私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message Dispatch高级特性之Message Cursors-6

- 配置使用

在缺省情况下，ActiveMQ会根据使用的Message Store来决定使用何种类型的Message Cursors，但是你可以根据destination来配置Message Cursors，例如：

1：对Topic subscribers

```
<destinationPolicy>
    <policyMap>
            <policyEntries>
            <policyEntry topic="org.apache.>" producerFlowControl="false" memoryLimit="1mb">
                                <dispatchPolicy>
                                        <strictOrderDispatchPolicy />
                                </dispatchPolicy>
                                <deadLetterStrategy>
                                <individualDeadLetterStrategy  topicPrefix="Test.DLQ." />
                                </deadLetterStrategy>
                                <pendingSubscriberPolicy>
                                        <vmCursor />
                                </pendingSubscriberPolicy>
                                <pendingDurableSubscriberPolicy>
                                        <vmDurableCursor/>
                                </pendingDurableSubscriberPolicy>
                        </policyEntry>
            </policyEntries>
    </policyMap>
</destinationPolicy>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message Dispatch高级特性之Message Cursors-7

- 配置说明：有效的Subscriber类型是vmCursor和fileCursor，缺省是store based cursor。有效的持久化Subscriber的cursor types是storeDurableSubscriberCursor，vmDurableCursor 和 fileDurableSubscriberCursor，缺省是store based cursor。
- 对于Queues的配置

```
<destinationPolicy>
    <policyMap>
          <policyEntries>
                    <policyEntry queue="org.apache.>">
                              <deadLetterStrategy>
                    <individualDeadLetterStrategy queuePrefix="Test.DLQ."/>
                              </deadLetterStrategy>
                              <pendingQueuePolicy>
                                        <vmQueueCursor />
                              </pendingQueuePolicy>
                    </policyEntry>
          </policyEntries>
    </policyMap>
</destinationPolicy>
```

配置说明：有效的类型是storeCursor，vmQueueCursor 和 fileQueueCursor



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message Dispatch高级特性之Async Sends

- Async Sends

  AciveMQ支持异步和同步发送消息，是可以配置的。通常对于快的消费者，是直接把消息同步发送过去，但对于一个Slow Consumer，你使用同步发送消息可能出现Producer堵塞等现象，慢消费者适合使用异步发送。

- 配置使用

1：ActiveMQ默认设置dispatcheAsync=true是最好的性能设置。如果你处理的是Fast Consumer则使用dispatcheAsync=false

2：在Connection URI级别来配置使用Async Send

```
cf = new ActiveMQConnectionFactory("tcp://locahost:61616?jms.useAsyncSend=true");
```

3：在ConnectionFactory级别来配置使用Async Send

```
((ActiveMQConnectionFactory)connectionFactory).setUseAsyncSend(true);
```

4：在Connection级别来配置使用Async Send

```
((ActiveMQConnection)connection).setUseAsyncSend(true);
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message Dispatch高级特性之Dispatch Policies-1

- 严格顺序分发策略（Strict Order Dispatch Policy）

通常ActiveMQ会保证topic consumer以相同的顺序接收来自同一个producer的消息，但有时候也需要保证不同的topic consumer以相同的顺序接收消息，然而，由于多线程和异步处理，不同的topic consumer可能会以不同的顺序接收来自不同producer的消息。

Strict order dispatch policy 会保证每个topic consumer会以相同的顺序接收消息，代价是性能上的损失。以下是一个配置例子：

```
<policyEntry topic="ORDERS.>">
    <dispatchPolicy>
        <strictOrderDispatchPolicy />
    </dispatchPolicy>
</policyEntry>
```

对于Queue的配置为：

```
<policyEntry queue=">" strictOrderDispatch="false" />
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message Dispatch高级特性之Dispatch Policies-2

- 轮询分发策略（Round Robin Dispatch Policy）

  ActiveMQ的prefetch缺省参数是针对处理大量消息时的高性能和高吞吐量而设置的，所以缺省的prefetch参数比较大。而且缺省的dispatch policies会尝试尽可能快的填满prefetch缓冲。

  然而在有些情况下，例如只有少量的消息而且单个消息的处理时间比较长，那么在缺省的prefetch和dispatch policies下，这些少量的消息总是倾向于被分发到个别的consumer上。这样就会因为负载的不均衡分配而导致处理时间的增加。

  Round robin dispatch policy会尝试平均分发消息，以下是一个例子：

```
<policyEntry topic="ORDERS.>">
    <dispatchPolicy>
        <roundRobinDispatchPolicy/>
    </dispatchPolicy>
</policyEntry>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message Dispatch高级特性之Optimized Acknowledgement

- ActiveMQ缺省支持批量确认消息，由于批量确认会提高性能。如果希望在应用程序中禁止经过优化的确认方式，那么可以采用如下方法：

1：在Connection URI 上启用Optimized Acknowledgements

```
cf = new
    ActiveMQConnectionFactory("tcp://locahost:61616?jms.optimizeAcknowledge=true");
```

2：在ConnectionFactory 上启用Optimized Acknowledgements

```
((ActiveMQConnectionFactory)connectionFactory).setOptimizeAcknowledge(true);
```

3：在Connection上启用Optimized Acknowledgements

```
((ActiveMQConnection)connection).setOptimizeAcknowledge(true);
```

4：5.6以后的版本，还可以在Connection URI上设置setOptimizeAcknowledgeTimeOut参数，默认值为300ms，你可以设置自己要用的值，0表示禁用。



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message Dispatch高级特性之Producer Flow Control-1

- 生产者流量控制（Producer Flow Control）

  流量控制的含义：当生产者产生消息过快，超过流量限制的时候，生产者将会被阻塞直到资源可以继续使用，或者抛出一个JMSException，可以通过<systemUsage>来配置。
- 同步发送消息的producer会自动使用producer flow control ；对于异步发送消息的producer，要使用producer flow control，你先要为connection配置一个ProducerWindowSize参数，如下：

  ((ActiveMQConnectionFactory)cf).setProducerWindowSize(1024000);

  ProducerWindowSize是producer在发送消息的过程中，收到broker对于之前发送消息的确认之前， 能够发送消息的最大字节数
- 可以禁用producer flow control，以下是ActiveMQ配置文件的一个例子

```
<destinationPolicy>
    <policyMap>
          <policyEntries>
          <policyEntry topic="FOO.>" producerFlowControl="false"/>
          </policyEntries>
    </policyMap>
</destinationPolicy>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message Dispatch高级特性之Producer Flow Control-2

- 注意，自从ActiveMQ 5.x中引入新的消息游标之后，非持久化消息被分流到了临时文件存储中，以此来减少非持久化消息传送使用的内存总量。

  结果就是，你可能会发现一个队列的内存限制永远达不到，因为游标不需要使用太多的内存。如果你真的想把所有的非持久化消息存放在内存中，并在达到内存限制的时候停掉生产者，你需要配置<vmQueueCursor>，示例如下：

```
<policyEntry queue=">" producerFlowControl="true" memoryLimit="1mb">
  <pendingQueuePolicy>
    <vmQueueCursor/>
  </pendingQueuePolicy>
</policyEntry>
```

上面的配置可以保证所有的非持久化队列消息都保存在内存中，每一个队列的内存限制为1Mb

- 配置客户端的异常

  为了应对代理空间不足，而导致不确定的阻塞send()方法的一种替代方案，就是将其配置成客户端抛出的一个异常。通过将sendFailIfNoSpace属性设置为true，代理将会引起send()方法失败，并抛出javax.jms.ResourceAllocationException异常，传播到客户端。下面是一个配置的示例：



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message Dispatch高级特性之Producer Flow Control-3

```
<systemUsage>
 <systemUsage sendFailIfNoSpace="true">
   <memoryUsage>
     <memoryUsage limit="20 mb"/>
   </memoryUsage>
 </systemUsage>
</systemUsage>
```

这么配置的好处是，客户端可以捕获javax.jms.ResourceAllocationException异常，稍等一下，并重试send()操作，而不是无限期地傻等下去。

- 从5.3.1版本之后，sendFailIfNoSpaceAfterTimeout属性被加了进来。这个属性同样导致send()方法失败，并在客户端抛出异常，但仅当等待了指定时间之后才触发。如果在配置的等待时间过去之后，代理上的空间仍然没有被释放，仅当这个时候send()方法才会失败，并且在客户端抛出异常。示例：

```
<systemUsage>
 <systemUsage sendFailIfNoSpaceAfterTimeout="3000">
   <memoryUsage>
     <memoryUsage limit="20 mb"/>
   </memoryUsage>
 </systemUsage>
</systemUsage>
```

定义超时的单位是毫秒



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message Dispatch高级特性之Producer Flow Control-4

- System usage

可以通过<systemUsage>元素的一些属性来减慢生产者，如下例子：

```
<systemUsage>
  <systemUsage>
    <memoryUsage>
      <memoryUsage limit="64 mb" />
    </memoryUsage>
    <storeUsage>
      <storeUsage limit="100 gb" />
    </storeUsage>
    <tempUsage>
      <tempUsage limit="10 gb" />
    </tempUsage>
  </systemUsage>
</systemUsage>
```

你可以为非持久化的消息设置内存限制，为持久化消息设置磁盘空间，以及为临时消息设置总的空间，broker将在减慢生产者之前使用这些空间。使用上述的默认设置，broker将会一直阻塞send()方法的调用，直至一些消息被消费，有了可用的空间。

做最好的在线学习社区
网　址：http://sishuok.com
咨询QQ：2371651507
私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message高级特性之Message Properties

- ActiveMQ支持很多消息属性，具体可以参见
  http://activemq.apache.org/activemq-message-properties.html
- 常见的一些属性说明

1：Queue的消息默认是持久化的
2：消息的优先级默认是4
3：消息发送时设置了时间戳
4：消息的过期时间默认是永不过期，过期的消息进入DLQ，可以配置DLQ及其处理策略
5：如果消息时重发的，将会标记出来
6：JMSReplyTo标识响应消息发送到哪个Queue
7：JMSCorelationID标识此消息相关联的消息id，可以用这个标识把多个消息连接起来
8：JMS同时也记录了消息重发的次数，默认是6次
9：如果有一组关联的消息需要处理，可以分组：只需要设置消息组的名字和这个消息是第几个消息
10：如果消息中一个事务环境，则TXID将被设置
11：此外ActiveMQ在服务器端额外设置了消息入列和出列的时间戳
12：ActiveMQ里消息属性的值，不仅可以用基本类型，还可以用List或Map类型



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message高级特性之Advisory Message-1

- Advisory Message是ActiveMQ自身的系统消息地址，可以监听该地址来获取activemq的系统信息。目前支持获取如下信息：

1：consumers，producers 和 connections的启动和停止
2：创建和销毁temporary destinations
3：topics 和 queues的消息过期
4：brokers 发送消息给 destinations，但是没有consumers
5：connections 启动和停止

- 几点说明：

1：所有Advisory的topic，前缀是：ActiveMQ.Advisory
2：所有Advisory的消息类型是：‘Advisory’ ，所有的Advisory都有的消息属性有：originBrokerId、originBrokerName、originBrokerURL
3：具体支持的topic和queue，请参看http://activemq.apache.org/advisory-message.html

- 打开Advisories ， 默认Advisory的功能是关闭的

```
<destinationPolicy>
    <policyMap>
          <policyEntries>
                <policyEntry topic=">" advisoryForConsumed="true" />
          </policyEntries>
    </policyMap>
</destinationPolicy>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message高级特性之Advisory Message-2

- 关闭Advisories ， 有好几种方法

1：<broker advisorySupport="false">

2：也可在Java中写：

```
BrokerService broker = new BrokerService();
broker.setAdvisorySupport(false);
...
broker.start();
```

3：也可以在ActiveMQConnectionFactory上设置‘watchTopicAdvisories’ 属性

```
ActiveMQConnectionFactory factory = new ActiveMQConnectionFactory();
factory.setWatchTopicAdvisories(false);
```

4：也可在ConnectionURI上写：

```
"tcp://localhost:61616?jms.watchTopicAdvisories=false"
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message高级特性之Advisory Message-3

n 使用的方法和步骤：

1：要在配置文件里面开启Advisories

2：消息发送端没有变化

3：消息接收端：

（1）根据你要接收的信息类型，来设置不同的topic，当然也可以使用AdvisorySupport这个类来辅助创建，比如你想要得到消息生产者的信息，你可以：

```
Topic d=session.createTopic("ActiveMQ.Advisory.Producer.Topic.MyTopic");
```

也可以使用：

```
Topic d = session.createTopic("MyTopic");
Destination d2 = AdvisorySupport.getProducerAdvisoryTopic(destination);
```

（2）由于这个topic默认不是持久化的，所以应该先开启接收端，然后再发送topic信息

（3）接收消息的时候，接收到的消息类型是ActiveMQMessage，所以类型转换的时候，要转换成ActiveMQMessage，然后再通过getDataStructure方法来得到具体的信息对象，如：

```
if (message instanceof ActiveMQMessage) {
    try {
          ActiveMQMessage aMsg = (ActiveMQMessage) message;
          ProducerInfo prod = (ProducerInfo) aMsg.getDataStructure();
          System.out.println("count==="+aMsg.getProperty("producerCount"));
          System.out.println(" prodd==="+prod.getProducerId());
    } catch (Exception e) {              e.printStackTrace(); }
}
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message高级特性之延迟和定时消息投递-1

- 延迟和定时消息投递（Delay and Schedule Message Delivery）

  有时候我们不希望消息马上被broker投递出去，而是想要消息60秒以后发给消费者，或者我们想让消息没隔一定时间投递一次，一共投递指定的次数。。。类似这种需求，ActiveMQ提供了一种broker端消息定时调度机制。

  我们只需要把几个描述消息定时调度方式的参数作为属性添加到消息，broker端的调度器就会按照我们想要的行为去处理消息。当然需要在xml中配置schedulerSupport属性为true

- 一共有4个属性

1：AMQ_SCHEDULED_DELAY ：延迟投递的时间

2：AMQ_SCHEDULED_PERIOD ：重复投递的时间间隔

3：AMQ_SCHEDULED_REPEAT：重复投递次数

4：AMQ_SCHEDULED_CRON：Cron表达式

- ActiveMQ也提供了一个封装的消息类型：org.apache.activemq.ScheduledMessage，可以使用这个类来辅助设置，使用例子如：延迟60秒

```
MessageProducer producer = session.createProducer(destination);
TextMessage message = session.createTextMessage("test msg");
long time = 60 * 1000;
message.setLongProperty(ScheduledMessage.AMQ_SCHEDULED_DELAY, time);
producer.send(message);
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message高级特性之延迟和定时消息投递-2

- 例子：延迟30秒，投递10次，间隔10秒：

```
TextMessage message = session.createTextMessage("test msg");
long delay = 30 * 1000;
long period = 10 * 1000;
int repeat = 9;
message.setLongProperty(ScheduledMessage.AMQ_SCHEDULED_DELAY, delay);
message.setLongProperty(ScheduledMessage.AMQ_SCHEDULED_PERIOD, period);
message.setIntProperty(ScheduledMessage.AMQ_SCHEDULED_REPEAT, repeat);
```

- 例子：使用 CRON 表达式，每个小时发送一次

```
TextMessage message = session.createTextMessage("test msg");
message.setStringProperty(ScheduledMessage.AMQ_SCHEDULED_CRON, "0 * * * *");
```

CRON表达式的优先级高于另外三个参数，如果在设置了CRON的同时，也有repeat和period参数，则会在每次CRON执行的时候，重复投递repeat次，每次间隔为period。就是说设置是叠加的效果。例如每小时都会发生消息被投递10次，延迟1秒开始，每次间隔1秒：

```
TextMessage message = session.createTextMessage("test msg");
message.setStringProperty(ScheduledMessage.AMQ_SCHEDULED_CRON, "0 * * * *");
message.setLongProperty(ScheduledMessage.AMQ_SCHEDULED_DELAY, 1000);
message.setLongProperty(ScheduledMessage.AMQ_SCHEDULED_PERIOD, 1000);
message.setIntProperty(ScheduledMessage.AMQ_SCHEDULED_REPEAT, 9);
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message高级特性之Blob Messages-1

- 有些时候，我们需要传递Blob（Binary Large OBjects）消息，可以按照如下方式：
- 配置BLOB Transfer Policy，可以在发送方的连接URI上设置，如：

"tcp://192.168.1.106:61679?jms.blobTransferPolicy.uploadUrl=http://192.168.1.106:8171/fileserver/"

- Sending BlobMessages，有几种方式：

1：如果你发送到的文件或者URL存在，比如发给共享文件系统或者是Web server上的web应用，那么你可以使用如下方式：

```
BlobMessage message = session.createBlobMessage(new
URL("http://some.shared.site.com");
producer.send(message);
```

2：也可以在客户端动态的创建文件流，如下：

```
BlobMessage message = session.createBlobMessage(new File("/foo/bar");
```

或者：

```
InputStream in = ...;
BlobMessage message = session.createBlobMessage(in);
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message高级特性之Blob Messages-2

- Receiving BlobMessages示例：

```
if (message instanceof BlobMessage) {
    BlobMessage blobMessage = (BlobMessage) message;
    InputStream in = blobMessage.getInputStream();
    // process the stream...
}
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Message高级特性之Message Transformation

- 有时候需要在JMS provider内部进行message的转换。从4.2版本起，ActiveMQ提供了一个MessageTransformer 接口用于进行消息转换，可以在如下对象上调用：

  ActiveMQConnectionFactory、ActiveMQConnection、ActiveMQSession、ActiveMQMessageConsumer、ActiveMQMessageProducer
- 在消息被发送到JMS provider的消息总线前进行转换。通过producerTransform方法
- 在消息到达消息总线后，但是在consumer接收到消息前进行转换。通过consumerTransform方法
- 当然MessageTransformer 接口的实现，需要你自己来提供



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Consumer高级特性之Exclusive Consumer

- 独有消费者（Exclusive Consumer）

Queue中的消息是按照顺序被分发到consumers的。然而，当你有多个consumers同时从相同的queue中提取消息时，你将失去这个保证。因为这些消息是被多个线程并发的处理。有的时候，保证消息按照顺序处理是很重要的。例如，你可能不希望在插入订单操作结束之前执行更新这个订单的操作。

ActiveMQ从4.x版本起开始支持Exclusive Consumer。 Broker会从多个consumers中挑选一个consumer来处理queue中所有的消息，从而保证了消息的有序处理。如果这个consumer失效，那么broker会自动切换到其它的consumer。 可以通过Destination Options 来创建一个Exclusive Consumer，如下：

```
queue = new ActiveMQQueue("TEST.QUEUE?consumer.exclusive=true");
consumer = session.createConsumer(queue);
```

还可以给consumer设置优先级，以便针对网络情况进行优化，如下：

```
queue = new
ActiveMQQueue("TEST.QUEUE?consumer.exclusive=true&consumer.priority=10");
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Consumer高级特性之Consumer Dispatche Async

- 在activemq4.0以后，你可以选择broker同步或异步的把消息分发给消费者。可以设置dispatchAsync 属性，默认是true，通常情况下这是最佳的。
- 你也可以修改，可以通过如下几种方式

1：在ConnectionFactory层设置

```
((ActiveMQConnectionFactory)connectionFactory).setDispatchAsync(false);
```

2：在Connection上设置

这个设置将会覆盖ConnectionFactory上的设置

```
((ActiveMQConnection)connection).setDispatchAsync(false);
```

3：在Consumer来设置

```
queue = new ActiveMQQueue("TEST.QUEUE?consumer.dispatchAsync=false");
consumer = session.createConsumer(queue);
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Consumer高级特性之Consumer Priority

- JMS JMSPriority 定义了十个消息优先级值，0 是最低的优先级，9 是最高的优先级。另外，客户端应当将0 - 4 看作普通优先级，5 - 9 看作加急优先级。

- 如何定义Consumer Priority的优先级呢？ 配置如下：

```
queue = new ActiveMQQueue("TEST.QUEUE?consumer.priority=10");
consumer = session.createConsumer(queue);
```

- Consumer的Priority的划分为0~127个级别，127是最高的级别，0是最低的也是ActiveMQ默认的。 这种配置可以让Broker根据Consumer的优先级来发送消息先到较高的优先级的Consumer上，如果某个较高的Consumer的消息装载慢，则Broker会把消息发送到仅次于它优先级的Consumer上。



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Consumer高级特性之Manage Durable Subscribers

- 消息的持久化，保证了消费者离线后，再次进入系统，不会错过消息，但是这也会消耗很多的资源。从5.6开始，可以对持久化消息进行如下管理：

- Removing inactive subscribers
  我们还可能希望删除那些不活动的订阅者，如下：

```
<broker name="localhost" offlineDurableSubscriberTimeout="86400000"
offlineDurableSubscriberTaskSchedule="3600000">
```

1： offlineDurableSubscriberTimeout：离线多长时间就过期删除，缺省是-1，就是不删除
2： offlineDurableSubscriberTaskSchedule：多长时间检查一次，缺省300000，单位毫秒



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Consumer高级特性之Message Groups-1

- Message Groups就是对消息分组，它是Exclusive Consumer功能的增强：

  逻辑上，Message Groups 可以看成是一种并发的Exclusive Consumer。跟所有的消息都由唯一的consumer处理不同，JMS 消息属性JMSXGroupID 被用来区分message group。Message Groups特性保证所有具有相同JMSXGroupID 的消息会被分发到相同的consumer（只要这个consumer保持active）。

- 另外一方面，Message Groups特性也是一种负载均衡的机制。 在一个消息被分发到consumer之前，broker首先检查消息JMSXGroupID属性。如果存在，那么broker 会检查是否有某个consumer拥有这个message group。如果没有，那么broker会选择一个consumer，并将它关联到这个message group。此后，这个consumer会接收这个message group的所有消息，直到：

  1：Consumer被关闭

  2：Message group被关闭，通过发送一个消息，并设置这个消息的JMSXGroupSeq为-1

- 创建一个Message Groups，只需要在message对象上设置属性即可，如下：

```
message.setStringProperty("JMSXGroupID","GroupA");
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Consumer高级特性之Message Groups-2

- 关闭一个Message Groups，只需要在message对象上设置属性即可，如下：

```
message.setStringProperty("JMSXGroupID","GroupA");
message.setIntProperty("JMSXGroupSeq", -1);
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Consumer高级特性之Message Selectors

- JMS Selectors用在获取消息的时候，可以基于消息属性和Xpath语法对消息进行过滤。JMS Selectors由SQL92语义定义。以下是个Selectors的例子：

```
consumer = session.createConsumer(destination, "JMSType = 'car' AND weight > 2500");
```

1：JMS Selectors表达式中，可以使用IN、NOT IN、LIKE等

2：需要注意的是，JMS Selectors表达式中的日期和时间需要使用标准的long型毫秒值

3：表达式中的属性不会自动进行类型转换，例如：

```
myMessage.setStringProperty("NumberOfOrders", "2");
```

那么此时“NumberOfOrders > 1” 求值结果会是false

4：Message Groups虽然可以保证具有相同message group的消息被唯一的consumer顺序处理，但是却不能确定被哪个consumer处理。在某些情况下，Message Groups可以和JMS Selector一起工作，

例如：设想有三个consumers分别是A、B和C。你可以在producer中为消息设置三个message groups分别是“A”、“B”和“C”。然后令consumer A使用“JMXGroupID = ‘A’”作为selector。B和C也同理。这样就可以保证message group A的消息只被consumer A处理。需要注意的是，这种做法有以下缺点：

（1）producer必须知道当前正在运行的consumers，也就是说producer和consumer被耦合到一起。

（2）如果某个consumer失效，那么应该被这个consumer消费的消息将会一直被积压在broker上。



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Consumer高级特性之Redelivery Policy-1

- ActiveMQ在接收消息的Client有以下几种操作的时候，需要重新传递消息：

1：Client用了transactions，且在session中调用了rollback()
2：Client用了transactions，且在调用commit()之前关闭
3：Client在CLIENT_ACKNOWLEDGE的传递模式下，在session中调用了recover()

- 可以通过设置ActiveMQConnectionFactory和ActiveMQConnection来定制想要的再次传送策略，可用的Redelivery属性如下：

1：collisionAvoidanceFactor：设置防止冲突范围的正负百分比，只有启用useCollisionAvoidance参数时才生效。也就是在延迟时间上再加一个时间波动范围。默认值为0.15
2：maximumRedeliveries：最大重传次数，达到最大重连次数后抛出异常。为-1时不限制次数，为0时表示不进行重传。默认值为6。
3：maximumRedeliveryDelay：最大传送延迟，只在useExponentialBackOff为true时有效（V5.5），假设首次重连间隔为10ms，倍数为2，那么第二次重连时间间隔为 20ms，第三次重连时间间隔为40ms，当重连时间间隔大的最大重连时间间隔时，以后每次重连时间间隔都为最大重连时间间隔。默认为-1。
4：initialRedeliveryDelay：初始重发延迟时间，默认1000L
5：redeliveryDelay：重发延迟时间，当initialRedeliveryDelay=0时生效，默认1000L
6：useCollisionAvoidance：启用防止冲突功能，默认false
7：useExponentialBackOff：启用指数倍数递增的方式增加延迟时间，默认false
8：backOffMultiplier：重连时间间隔递增倍数，只有值大于1和启用useExponentialBackOff参数时才生效。默认是5



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Consumer高级特性之Redelivery Policy-2

- 在接受的Client可以如下设置：

```
ActiveMQConnectionFactory cf = new ActiveMQConnectionFactory(
"failover:(tcp://192.168.1.106:61679,tcp://192.168.1.106:61819)?randomize=false");

RedeliveryPolicy policy = new RedeliveryPolicy();
policy.setMaximumRedeliveries(3);
cf.setRedeliveryPolicy(policy);
```

- 当消息试图被传递的次数超过配置中maximumRedeliveries属性的值时，那么，broker会认定该消息是一个死消息，并被把该消息发送到死队列中。 默认，aciaveMQ中死队列被声明为“ActivemMQ.DLQ”，所有不能消费的消息都被传递到该死队列中。 你可以在acivemq.xml中配置individualDeadLetterStrategy属性，示例如下：

```
<policyEntry queue= "> " >
    <deadLetterStrategy>
          <individualDeadLetterStrategy queuePrefix= "DLQ."
    useQueueForQueueMessages= "true" />
    </deadLetterStrategy>
</policyEntry>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Consumer高级特性之Redelivery Policy-3

- 自动删除过期消息

  有时需要直接删除过期的消息而不需要发送到死队列中，可以使用属性processExpired=false来设置，示例如下：

```
<policyEntry queue= "> " >
    <deadLetterStrategy>
        <sharedDeadLetterStrategy processExpired= "false" />
    </deadLetterStrategy>
</policyEntry>
```

- 存放非持久消息到死队列中

  默认情况下，Activemq不会把非持久的死消息发送到死队列中。非持久性如果你想把非持久的消息发送到死队列中，需要设置属性processNonPersistent=“true”，示例如下：

```
<policyEntry queue= "> " >
    <deadLetterStrategy>
        <sharedDeadLetterStrategy processNonPersistent= "true" />
    </deadLetterStrategy>
</policyEntry>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Consumer高级特性之Redelivery Policy-4

- Redelivery Policy per Destination

在V5.7之后，你可以为每一个Destination配置一个Redelivery Policy。示例如：

```
ActiveMQConnection connection ...  // Create a connection
RedeliveryPolicy queuePolicy = new RedeliveryPolicy();
queuePolicy.setInitialRedeliveryDelay(0);
queuePolicy.setRedeliveryDelay(1000);
queuePolicy.setUseExponentialBackOff(false);
queuePolicy.setMaximumRedeliveries(2);

RedeliveryPolicy topicPolicy = new RedeliveryPolicy();
topicPolicy.setInitialRedeliveryDelay(0);
topicPolicy.setRedeliveryDelay(1000);
topicPolicy.setUseExponentialBackOff(false);
topicPolicy.setMaximumRedeliveries(3);
// Receive a message with the JMS API
RedeliveryPolicyMap map = connection.getRedeliveryPolicyMap();
map.put(new ActiveMQTopic(">"), topicPolicy);
map.put(new ActiveMQQueue(">"), queuePolicy);
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Consumer高级特性之Slow Consumer Handling-1

- Prefetch机制

ActiveMQ通过Prefetch机制来提高性能，方式是在客户端的内存里可能会缓存一定数量的消息。缓存消息的数量由prefetch limit来控制。当某个consumer的prefetch buffer已经达到上限，那么broker不会再向consumer分发消息，直到consumer向broker发送消息的确认，确认后的消息将会从缓存中去掉。

可以通过在ActiveMQConnectionFactory或者ActiveMQConnection上设置ActiveMQPrefetchPolicy对象来配置prefetch policy。也可以通过connection options或者destination options来配置。例如：

```
tcp://localhost:61616?jms.prefetchPolicy.all=50
tcp://localhost:61616?jms.prefetchPolicy.queuePrefetch=1
queue = new ActiveMQQueue("TEST.QUEUE?consumer.prefetchSize=10");
```

- prefetch size的缺省值如下：

1：persistent queues (default value: 1000)
2：non-persistent queues (default value: 1000)
3：persistent topics (default value: 100)
4：non-persistent topics (default value: Short.MAX_VALUE -1)



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Consumer高级特性之Slow Consumer Handling-2

- 慢Consumer处理

  慢消费者会在非持久的topics上导致问题：一旦消息积压起来，会导致broker把大量消息保存在内存中，broker也会因此而变慢。目前ActiveMQ使用Pending Message Limit Strategy来解决这个问题。除了prefetch buffer之外，你还要配置缓存消息的上限，超过这个上限后，新消息到来时会丢弃旧消息。

  通过在配置文件的destination map中配置PendingMessageLimitStrategy，可以为不用的topic namespace配置不同的策略。

- Pending Message Limit Strategy（等待消息限制策略）目前有以下两种：

1： Constant Pending Message Limit Strategy

Limit可以设置0、>0、-1三种方式： 0表示：不额外的增加其预存大小。 >0表示：再额外的增加其预存大小。 -1表示：不增加预存也不丢弃旧的消息。 这个策略使用常量限制，配置如下：

```
<constantPendingMessageLimitStrategy limit="50"/>
```

2：Prefetch Rate Pending Message Limit Strategy

这种策略是利用Consumer的之前的预存的大小乘以其倍数等于现在的预存大小。比如：

```
<prefetchRatePendingMessageLimitStrategy multiplier="2.5"/>
```

3：说明：在以上两种方式中，如果设置0意味着除了prefetch之外不再缓存消息；如果设置-1意味着禁止丢弃消息。



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Consumer高级特性之Slow Consumer Handling-3

- 配置消息的丢弃策略，目前有三种方式：

1：oldestMessageEvictionStrategy：这个策略丢弃最旧的消息。

2：oldestMessageWithLowestPriorityEvictionStrategy：这个策略丢弃最旧的，而且具有最低优先级的消息。

3：uniquePropertyMessageEvictionStrategy：从5.6开始，可以根据自定义的属性来进行抛弃，比如<uniquePropertyMessageEvictionStrategy propertyName=“STOCK” />，这就表示抛弃属性名称为Stock的消息

- 配置示例：

```
<destinationPolicy>
    <policyMap>
          <policyEntries>
                <policyEntry topic="FOO.>">
                      <dispatchPolicy>
                            <roundRobinDispatchPolicy />
                      </dispatchPolicy>
                </policyEntry>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# Consumer高级特性之Slow Consumer Handling-4

```
                <policyEntry topic="ORDERS.>">
                        <dispatchPolicy>
                                <strictOrderDispatchPolicy />
                        </dispatchPolicy>
                </policyEntry>
                <policyEntry topic="PRICES.>">
        <!-- lets force old messages to be discarded for slow consumers -->
                        <pendingMessageLimitStrategy>
                        <constantPendingMessageLimitStrategy limit="10"/>
                        </pendingMessageLimitStrategy>
                </policyEntry>
                <policyEntry tempTopic="true" advisoryForConsumed="true" />
                <policyEntry tempQueue="true" advisoryForConsumed="true" />
        </policyEntries>
    </policyMap>
</destinationPolicy>
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 监控和管理Broker

- Web Console方式

  直接访问ActiveMQ的管理页面：http://192.168.1.106:8161/admin/，默认的用户名和密码是admin/admin。具体配置在conf/jetty.xml里面

- Hawtio-web Management Console方式

  默认的用户名密码是admin/admin

- JMX方式



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 集成ActiveMQ和Tomcat-1

- ActiveMQ和Tomcat可以很自如的集成到一起使用，而不需要使用JNDI的方式，启动Tomcat的时候就可以启动ActiveMQ，方式如下：

1：修改web.xml

```
<context-param>
        <param-name>brokerURI</param-name>
        <param-value>/WEB-INF/activemq.xml</param-value>
    </context-param>
    <listener>
    <listener-class>org.apache.activemq.web.SpringBrokerContextListener</listener-class>
    </listener>
```

2：增加WEB-INF/activemq.xml，这里给个最简单的

```
<beans
  xmlns="http://www.springframework.org/schema/beans"
  xmlns:amq="http://activemq.apache.org/schema/core"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
  http://activemq.apache.org/schema/core http://activemq.apache.org/schema/core/activemq-core-
    5.2.0.xsd
  http://activemq.apache.org/camel/schema/spring
    http://activemq.apache.org/camel/schema/spring/camel-spring.xsd">
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 集成ActiveMQ和Tomcat-2

```
<broker xmlns="http://activemq.apache.org/schema/core" brokerName="localhost">
    <persistenceAdapter>
        <kahaDB directory="/usr/common/testdata/kahadb"/>
    </persistenceAdapter>
         <transportConnectors>
        <transportConnector name="openwire" uri="tcp://192.168.1.106:61616"/>
    </transportConnectors>
</broker>
</beans>
```

3：在web应用下拷入activemq的jar包，在ActiveMQ下面的lib包，例如：

cp -r  lib /usr/…目的地址lib

4：在lib下面传入spring的包，就从前面的arch1web应用下面的lib找spring的包就可以了

5：还需要xbean，这是apache的，可以从maven依赖的仓库里面找到

6：然后就可以启动tomcat，进行测试了



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# 什么时候使用ActiveMQ

- 异步调用
- 一对多通信
- 做多个系统的集成，同构、异构
- 作为RPC的替代
- 多个应用相互解耦
- 作为事件驱动架构的幕后支撑
- 为了提高系统的可伸缩性



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ优化-1

- ActiveMQ的性能依赖于很多因素，比如：
  1：网络拓扑结构，比如：嵌入、主从复制、网络连接
  2：transport协议
  3：service的质量，比如topic还是queue，是否持久化，是否需要重新投递，消息超时等
  4：硬件、网络、JVM和操作系统等
  5：生产者的数量，消费者的数量
  6：消息分发要经过的destination数量，以及消息的大小等
- 非持久化消息比持久化消息更快，原因如下：
  1：非持久化发送消息是异步的，Producer不需要等待Consumer的receipt消息
  2：而持久化是要把消息先存储起来，然后再传递
- 尽量使用异步投递消息，示例如：
  `cf.setUseAsyncSend(true);`
- Transaction比Non-transaction更快
- 可以考虑内嵌启动broker，这样应用和Broker之间可以使用VM协议通讯，速度快
- 尽量使用基于文件的消息存储方案，比如使用KahaDB的方式



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ优化-2

- 调整Prefetch Limit，ActiveMQ默认的prefetch大小不同的：

  1：Queue Consumer 默认1000

  2：Queue Browser Consumer默认500

  3：Persistent Topic Consumer默认1000

  4：Non-persistent Topic Consumer默认32767

Prefecth policy设置示例如下：

```
ActiveMQConnectionFactory cf = new ActiveMQConnectionFactory();
Properties props = new Properties();
props.setProperty("prefetchPolicy.queuePrefetch", "1000");
props.setProperty("prefetchPolicy.queueBrowserPrefetch", "500");
props.setProperty("prefetchPolicy.durableTopicPrefetch", "1000");
props.setProperty("prefetchPolicy.topicPrefetch", "32767");
cf.setProperties(props);
```

也可以在创建Destination的时候设置prefetch size，示例如下：

```
Queue queue = new ActiveMQQueue("TEST.QUEUE?consumer.prefetchSize=10");
MessageConsumer consumer = session.createConsumer(queue);
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ优化-3

- 可以考虑生产者流量控制，可以通过xml配置，代码开启方式如下：

```
cf.setProducerWindowSize(1024000);
```

- 可以考虑关闭消息的复制功能，也能部分提高心能，在连接工厂上设置，如下：

```
ActiveMQConnectionFaction cf = …..
cf.setCopyMessageOnSend(false);
```

- 调整TCP协议

TCP协议是ActiveMQ中最常使用的协议，常见有如下配置会影响协议性能：

1：socketBufferSize：socket的缓存大小，默认是65536

2：tcpNoDelay：默认是false

示例如：

```
String url = "failover://(tcp://localhost:61616?tcpNoDelay=true)";
ActiveMQConnectionFactory cf = new ActiveMQConnectionFactory(url);
```

- 消息投递和消息确认

官方建议使用自动确认的模式，同时还可以开启优化确认的选项，如下：

```
ActiveMQConnectionFactory cf = new ActiveMQConnectionFactory();
cf.setOptimizeAcknowledge(true);
```



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送

Java私塾-最专业的Java就业培训专家，因为专业，所以出色！值得你的信赖!



# ActiveMQ优化-4

- 在消费者这边，session会在一个单独的线程中分发消息给消费者，如果你使用的自动确认模式，为了增加吞吐量，你可以直接通过session传递消息给消费者，示例如下：

```
ActiveMQConnectionFactory cf = new ActiveMQConnectionFactory();
cf.setAlwaysSessionAsync(false);
```

- 如果使用KahaDB进行消息存储的话，可以调整如下选项来优化性能：

1：indexCacheSize：默认为10000，用来设定缓存页的个数，默认情况一页是4KB，一般来说缓存的大小尽可能的设置大一些，以避免内存不足时频繁的交换。

2：indexWriteBatchSize：默认1000，用来设置脏索引（脏索引就是cache中的index和消息存储中的index状态不一样）达到多少之后，就需要把索引存储起来。如果你想最大化broker的速度，那么就把这个值设置的尽可能的大一些，这样的话，仅会在到达checkpoint的时候，索引才会被存储起来。但是这样会增大系统出错的时候，丢失大量的元数据的风险。

3：journalMaxFileLength：缺省32mb，当broker的吞吐量特别大的时候，日志文件会很快被写满，这样会因为频繁的关闭文件，打开文件而导致性能低下。你可以通过调整文件的size，减少文件切换的频率，从而获得轻微的性能改善。

4：enableJournalDiskSyncs：缺省为true，通常，broker会在给producer确认之前，把消息同步到磁盘上（并且确保消息物化到磁盘上）。你可以通过设置这个选项为false，从而获得本质的性能改善。但是这样的话，多少会降低broker的可靠性。



私塾在线http://sishuok.com?frombook 独家提供配套教学视频，更有大量免费视频独家大放送