Atitit 消息系统 mq 的艺术 attilax总结 v5 t88.docx

Atitit 消息系统 mq 之道 attilax总结

[1. 概念 broker topic producer consumer 1](#_Toc19168)

[2. 常用的消息模式 2](#_Toc6684)

[2.1. 消息通道（Message Channel）模式 2](#_Toc31886)

[2.2. 发布者-订阅者（Publisher-Subscriber）模式 2](#_Toc7863)

[2.3. 2 拉模型 2](#_Toc28143)

[2.4. 图3 推模型 2](#_Toc26781)

[2.5. 图4 Publisher-Subscriber模式（图片来自eaipatterns ） 3](#_Toc25412)

[2.6. 消息路由（Message Router）模式 3](#_Toc2092)

[3. 扩展与标准化 3](#_Toc7178)

[3.1. 拓展:JMS规范Java消息服务（Java Message Service） 3](#_Toc21795)

[3.2. 消息中间件（MOM） 4](#_Toc5742)

[4. 一：JMQ的两种消息模式 1.1:点对点的消息模式 1.2：订阅模式 4](#_Toc27831)

[4.1.1. 1.1:点对点的消息模式 4](#_Toc15245)

[4.1.2. 1.2：订阅模式 4](#_Toc14696)

[5. 常见api 5](#_Toc8816)

[5.1. 3. 发送范例 23.1. 接受 63.2. 查询 10 5](#_Toc6780)

[6. 常见的产品 5](#_Toc18466)

[6.1. MSMQ、IBM MQ、JBoss MQ 5](#_Toc12353)

[6.2. 开源的RabbitMQ、 5](#_Toc19804)

[6.3. Apache ActiveMQ 5](#_Toc15249)

[6.4. Kafka  复杂 5](#_Toc3731)

[7. ref 5](#_Toc12833)

# ****概念 broker topic producer consumer****

****这是张我在Kafka官网上截的图，我大概可以把Kafka的主要结构分为以下几点：****

****producer：生产者，就是生产馒头(老妈)****

****consumer：消费者，就是吃馒头的(你)****

****broker：篮子****

****topic：主题，给馒头带一个标签，topica的馒头是给你吃的，topicb的馒头是给你弟弟吃****

****---------------------****

****作者：疯兔子大叔****

****来源：CSDN****

****原文：https://blog.csdn.net/wing\_93/article/details/78513782****

****版权声明：本文为博主原创文章，转载请附上博文链接！****

# ****常用的消息模式****

## ****消****息通道（Message Channel）模式

我们常常运用的消息模式是Message Channel（消息通道）模式，如图1所示。

消息通道作为在客户端（消费者，Consumer）与服务（生产者，Producer）之间引入的间接层，可以有效地解除二者之间的耦合。只要实现规定双方需要通信的消息格式，以及处理消息的机制与时机，就可以做到消费者对生产者的“无知”。事实上，该模式可以支持多个生产者与消费者。例如，我们可以让多个生产者向消息通道发送消息，因为消费者对生产者的无知性，它不必考虑究竟是哪个生产者发来的消息

## ****发布者-订阅者（Publisher-Subscriber）模式****

一旦消息通道需要支持多个消费者时，就可能面临两种模型的选择：拉模型与推模型。拉模型是由消息的消费者发起的，主动权把握在消费者手中，它会根据自己的情况对生产者发起调用。如图2所示

## ****2 拉模型****

拉模型的另一种体现则由生产者在状态发生变更时，通知消费者其状态发生了改变。但得到通知的消费者却会以回调方式，通过调用传递过来的消费者对象获取更多细节消息。

在基于消息的

## ****图3 推模型****

对于推模型而言，消费者无需了解生产者。在生产者通知消费者时，传递的往往是消息（或事件），而非生产者自身。同时，生产者还可以根据不同的情况，注册不同的消费者，又或者在封装的通知逻辑中，根据不同的状态变化，通知不同的消费者。

两种模型各有优势。拉模型的好处在于可以进一步解除消费者对通道的依赖，通过后台任务去定期访问消息通道。坏处是需要引入一个单独的服务进程，以Schedule形式执行。而对于推模型而言，消息通道事实上会作为消费者观察的主体，一旦发现消息进入，就会通知消费者执行对消息的处理。无论推模型，拉模型，对于消息对象而言，都可能采用类似Observer模式的机制，实现消费者对生产者的订阅，因此这种机制通常又被称为Publisher-Subscriber模式，如图4所示

## ****图4 Publisher-Subscriber模式（图片来自eaipatterns ）****

通常情况下，发布者和订阅者都会被注册到用于传播变更的基础设施（即消息通道）上。发布者会主动地了解消息通道，使其能够将消息发送到通道中；消息通道一旦接收到消息，会主动地调用注册在通道中的订阅者，进而完成对消息内容的消费。

## ****消息路由（Message Router）模式****

无论是Message Channel模式，还是Publisher-Subscriber模式，队列在其中都扮演了举足轻重的角色。然而，在企业应用系统中，当系统变得越来越复杂时，对性能的要求也会越来越高，此时对于系统而言，可能就需要支持同时部署多个队列，并可能要求分布式部署不同的队列。这些队列可以根据定义接收不同的消息，例如订单处理的消息，日志信息，查询任务消息等。这时，对于消息的生产者和消费者而言，并不适宜承担决定消息传递路径的职责。事实上，根据S单一职责原则，这种职责分配也是不合理的，它既不利于业务逻辑的重用，也会造成生产者、消费者与消息队列之间的耦合，从而影响系统的扩展。

既然这三种对象（组件）都不宜承担这样的职责，就有必要引入一个新的对象专门负责传递路径选择的功能，这就是所谓的Message Router模式，如图5所示：

# 扩展与标准化

ActiveMQ 是一个完全支持JMS1.1和J2EE 1.4规范的 JMS Provider实现，尽管JMS规范出台已经是很久的事情了，但是JMS在当今的J2EE应用中间仍然扮演着特殊的地位。

## 拓展:JMS规范Java消息服务（Java Message Service）

JMS:(http://baike.baidu.com/subview/157103/12665866.htm)

JMS即Java消息服务（Java Message Service）应用程序接口，是一个Java平台中关于面向消息中间件（MOM）的API，用于在两个应用程序之间，或分布式系统中发送消息，进行异步通信。Java消息服务是一个与具体平台无关的API，绝大多数MOM提供商都对JMS提供支持。

---------------------

## 消息中间件（MOM）

# 一：JMQ的两种消息模式 1.1:点对点的消息模式 1.2：订阅模式

消息列队有两种消息模式，一种是点对点的消息模式，还有一种就是订阅的模式.

### **1.1:点对点的消息模式**

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

### **1.2：订阅模式**

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

# 常见api

## 3. 发送范例 23.1. 接受 63.2. 查询 10

# 常见的产品

## MSMQ、IBM MQ、JBoss MQ

## 开源的[RabbitMQ](http://www.rabbitmq.com/)、

## [Apache ActiveMQ](http://activemq.apache.org/)

消息通道通常以队列的形式存在，这种先进先出的数据结构无疑最为适合这种处理消息的场景。微软的MSMQ、IBM MQ、JBoss MQ以及开源的[RabbitMQ](http://www.rabbitmq.com/)、[Apache ActiveMQ](http://activemq.apache.org/)都通过队列实现了Message Channel模式。

背景RabbitMQ是一个由erlang开发的AMQP(Advanved Message Queue)的开源实现

## Kafka  复杂

# ref

为什么会需要消息队列(MQ)？ - xuyatao - 博客园.html

ActiveMQ消息队列的使用及应用 - 朱小杰 - 博客园.html

Atiit rabbit mq 读取发送范例

Atitit Kafka 使用总结.docx