一、ActiveMQ简介

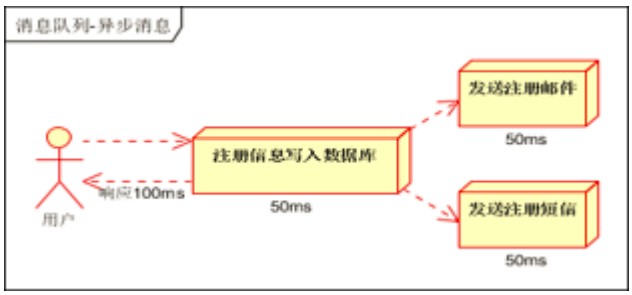
概念

ActiveMQ是Apache开源基金会研发的消息中间件。是完全支持JMS1.1和J2EE1.4规范的 JMS provider实现

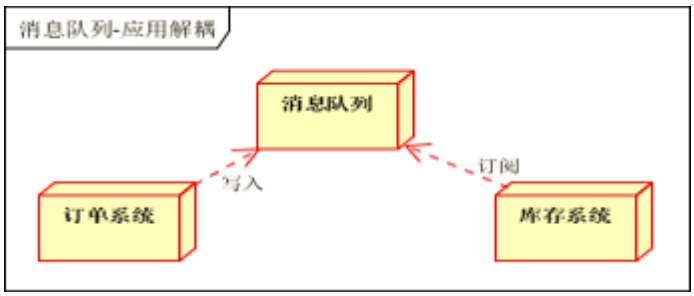
ActiveMQ主要应用在分布式系统架构中，帮助构建高可用、高性能、可伸缩的企业级面向消息服务的系统

使用场景

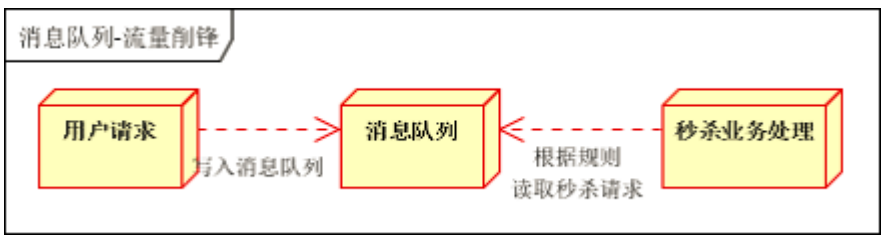
- 消息队列-异步消息：举我们简单的登录功能来说，一般来说用户登录需要将注册信息写入到数据库、发送注册短信、发送注册邮件，假设每个步骤响应时间是50ms,如果是同步的方式，那么整个注册过程需要消耗150ms；如果使用MQ来异步发送，实际上整个过程只需50ms。



- 消息队列-应用解耦：在传统的方式当中，比如订单系统写入数据、从库存里面读取数据，在调用的第三方或多或少会存在API侵入的问题，如果订单系统或者库存系统挂了，会影响本身系统的使用；如果使用消息中间件将所有的消息数据通过MQ来交换，都从MQ里面去读取数据，即使订单系统或者库存系统挂了，也不会影响本身系统的功能，只是查询不到新数据而已。



- 消息队列-流量削峰：



二、JMS规范

JMS基本概念

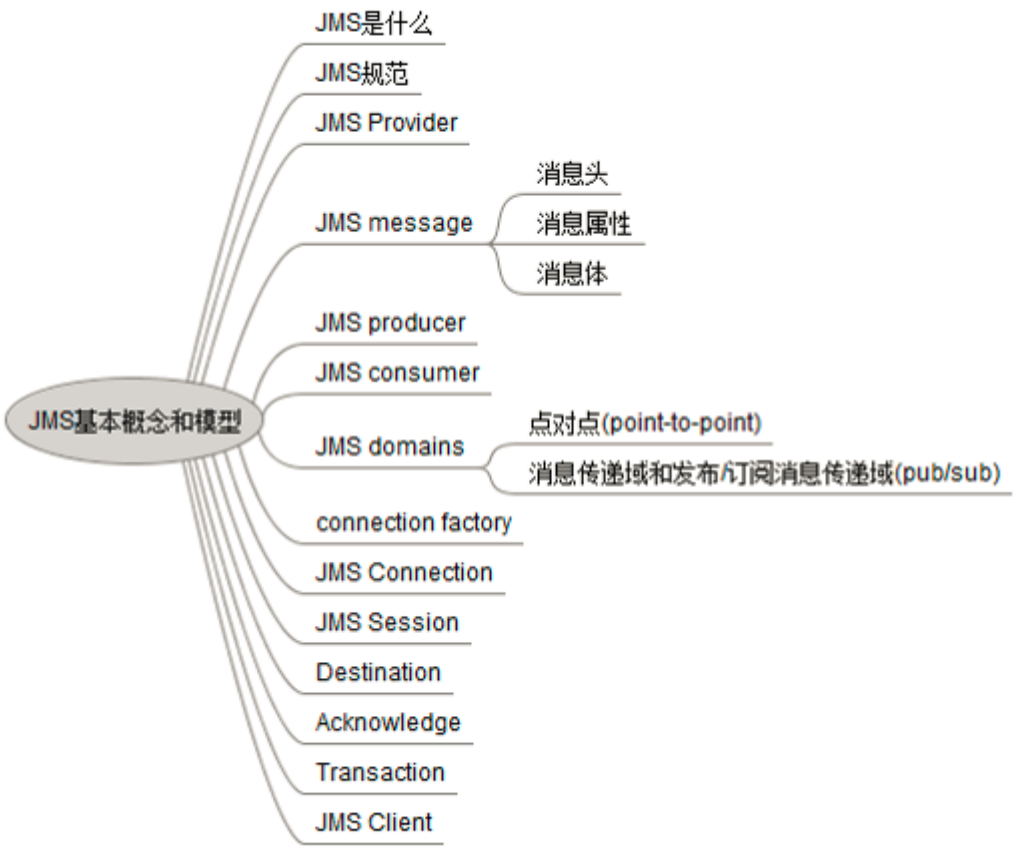
java消息服务（Java MessageService）是java平台中关于面向消息中间件的API，用于在两个应用程序之间，或者分布式系统中发送消息，进行异步通信

JMS是一个与具体平台无关的API，绝大多数MOM（MessageOriented Middleware）（面向消息中间件）提供商都对JMS提供了支持

其他开源的JMS提供商

JbossMQ(jboss4)、jbossmessaging(jboss5)、joram、ubermq、mantamq、openjms

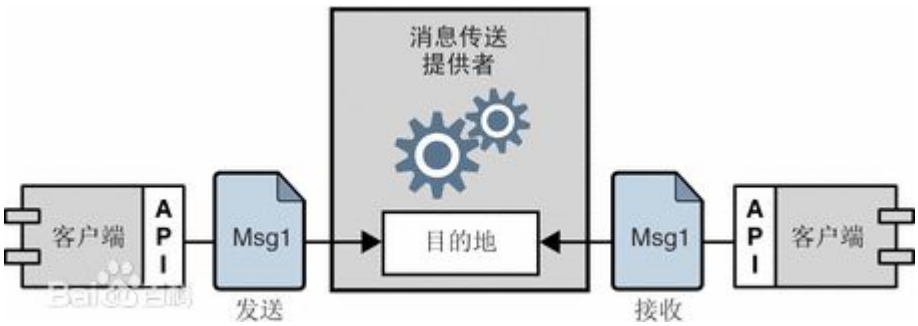
JMS模型



什么是MOM

面向消息的中间件，使用消息传送提供者来协调消息传输操作。 MOM需要提供API和管理工具。 客户端调用api。把消息发送到消息传送提供者指定的目的地

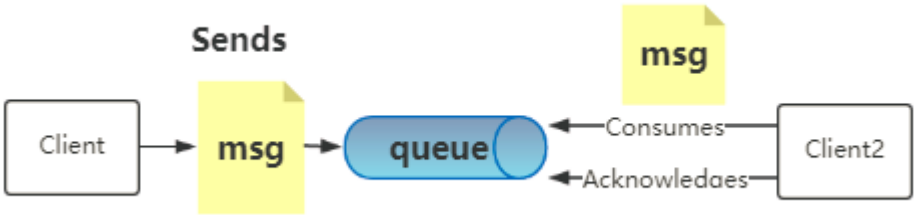
在消息发送之后，客户端会继续执行其他的工作。并且在接收方收到这个消息确认之前。提供者一直保留该消息



消息传递域

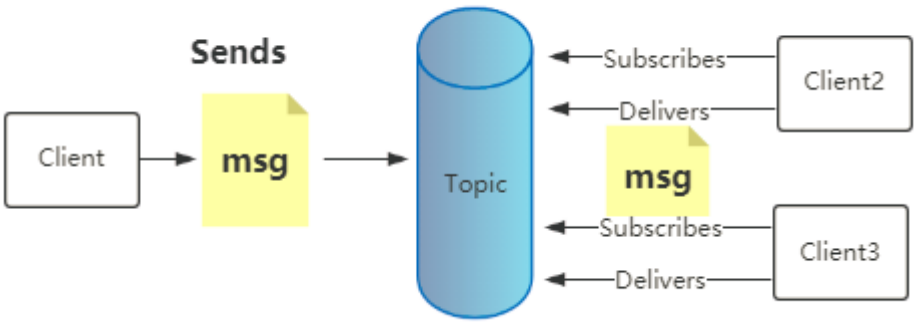
点对点(p2p)

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，都可以提取消息



发布订阅(pub/sub)

1.每个消息可以有多个消费者 2.消息的生产者和消费者之间存在时间上的相关性，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规范允许提供客户端创建持久订阅



消息的组成

消息头

包含消息的识别信息和路由信息

JMS消息头预定义了若干字段用于客户端与JMS提供者之间识别和发送消息，预编译头如下：

– JMSDestination – JMSDeliveryMode – JMSMessageID – JMSTimestamp – JMSCorrelationID – JMSReplyTo – JMSRedelivered – JMSType – JMSExpiration – JMSPriority

消息体

TextMessage MapMessage BytesMessage StreamMessage 输入输出流 ObjectMessage 可序列化对象

属性

我们可以给消息设置自定义属性，这些属性主要是提供给应用程序的。对于实现消息过滤功能，消息属性非常有用，JMS API定义了一些标准属性，JMS服务提供者可以选择性的提供部分标准属性。

JMS API

ConnectionFactory 连接工厂 Connection 封装客户端与JMS provider之间的一个虚拟的连接 Session 生产和消费消息的一个单线程上下文； 用于创建producer、consumer、message、queue..\ Destination 消息发送或者消息接收的目的地 MessageProducer/consumer 消息生产者/消费者

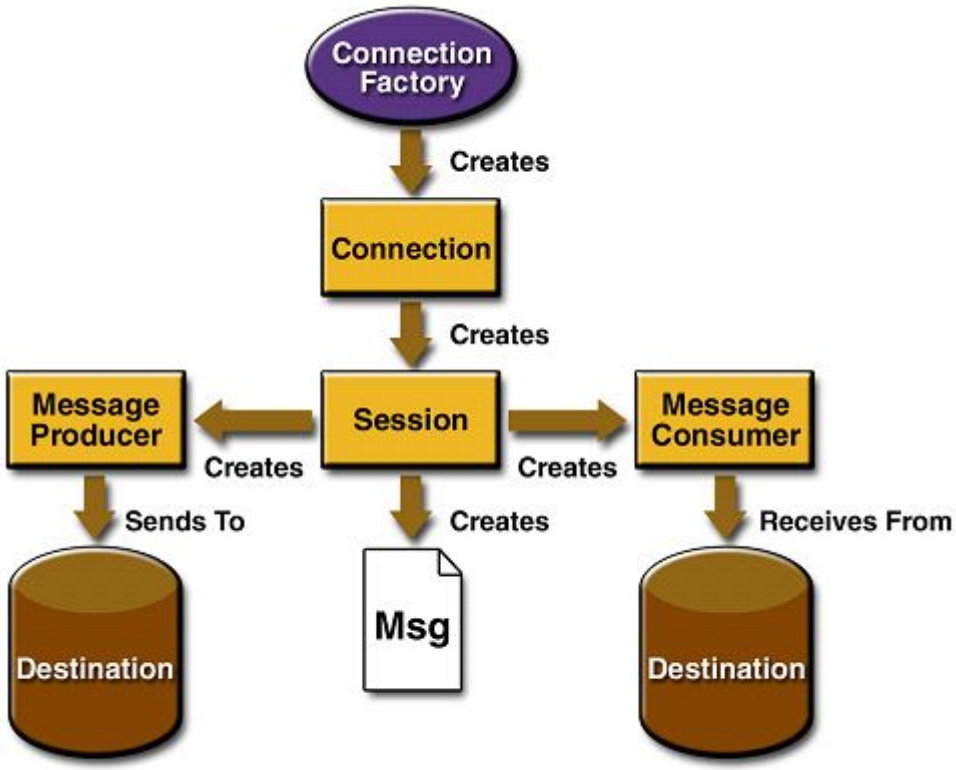


Figure 33–5 The JMS API Programming Model

JMS的可靠性机制

JMS消息之后被确认后，才会认为是被成功消费。消息的消费包含三个阶段： 客户端接收消息、客户端处理消息、消息被确认

事务性会话

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。改参数有三个可选值

① 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。

②Session.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。

③Session.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

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

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session.createTemporaryQueue("queue");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

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。如果最初创建的持久订阅的客户或者任何其他客户，使用相同的连接工厂和链接的客户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会向客户发送客户出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。持久订阅在床加你之后会一直保留，知道应用程序挑用会话上的unsubscribe（取消订阅）方法。

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。

JMS Provider会缓存每个生产者当前生产的所有消息，直到commit或者rollback，commit操作将会导致事务中所有的消息被持久存储；rollback意味着JMS Provider将会清除此事务下所有的消息记录。在事务未提交之前，消息是不会被持久化存储的，也不会被消费者消费

事务提交意味着生产的所有消息都被发送。消费的所有消息都被确认；

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，也就是下次仍然能够接收到发送端的消息，除非消息已经过期了

消息的持久订阅和非持久订阅（**pub/sub**模型）

非持久订阅只有当客户端出于激活状态，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，而当客户端出于离线 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，客户端向JMS注册一个识别自己身份的ID，当这个客户端出于离线状态时，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得到所有当自己出于离线时发送到主题的消息。

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

非持久订阅状态下，不能恢复和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

当所有消息必须被接收则用持久订阅，当丢失消息能够被容忍，则使用非持久订阅

JMS （P2P）模型

如果session关闭时，有一些消息已经收到，但还没有被签收，那么当消费者下次连接到相同的队列时，消息还会被签收 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 队列可以长久保存消息直到消息被消费者签收。消费者不需要担心因为消息丢失而时刻与jms provider保持连接状态

JMS消息的发送策略

持久化消息

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

可以设置消息发送端发送持久化消息的异步方式

connectionFactory.setUseAsyncSend(**true**);

回执窗口大小设置

```
connectionFactory.setProducerWindowSize();
```

非持久化消息

```
textMessage.setJMSDeliveryMode(DeliveryMode.NON_PERSISTENCE);
```

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

```
connectionFactory.setAlwaysSyncSend();
```

consumer获取消息是pull还是（broker的主动 push）

默认情况下，mq服务器（broker）采用异步方式向客户端主动推送消息(push)。也就是说broker在向某个消费者会话推送消息后，不会等待消费者响应消息，直到消费者处理完消息以后，主动向broker返回处理结果

prefetchsize“预取消息数量“

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

Queue

持久化消息 prefetchSize=1000

非持久化消息 1000

topic

持久化消息 100

非持久化消息 32766

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

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

消表示已经被consumer接收但未确认的消息。

消息确认

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

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

REDELIVERED_ACK_TYPE (broker会重新发送该消息) 重发侧策略

DELIVERED_ACK_TYPE 消息已经接收，但是尚未处理结束

STANDARD_ACK_TYPE

表示消息处理成功