# 传统RPC中间件技术的缺点

像CORBA，DCOM，RMI等RPC（Remote Produre call 远程程序调用）中间件技术（A->B的通信）的缺点：

1. 同步通信，客户端发起调用后，必须等待服务端返回结果后，客户端才能继续执行。
2. 必须保证客户端和服务端都正常运行，其中一个挂掉客户端都会异常。
3. 只能一个客户端给一个服务端发起请求。（点对点通信）

# 面向消息的中间件（MOM）的优点

MOM（Message Oriented Middleware 面向消息的中间件）：发送者将消息发送给消息服务器，消息服务器将消息放在多个队列中，在合适的时候将消息转发给接收者（A->消息服务器->B的通信）。

优点：

1. 异步通信，客户端（发送者）无须等待。
2. 不需要同时保证客户端和服务端都正常，如果服务端挂了，会在服务端调起后转发消息给服务端。
3. 一个消息可以有多个接收者。（一对多通信）
4. A和B两个程序实现解耦，特别是A（Java程序），B（C++程序）不需要实现彼此的接口，A也不需要调用B的接口，而是实现消息中间件的接口即可；或者当A需要推送消息给B，C，D多个服务接口，A中要调B，C，D接口，这样耦合度很高，如果用中间件，只要把这一份消息放到中间件即可。

# Java消息服务（JMS）

JMS是一个接口，定义了用Java访问消息中间件的接口，但没有实现。

（注意：JMS是一个API，它与具体的平台无关）

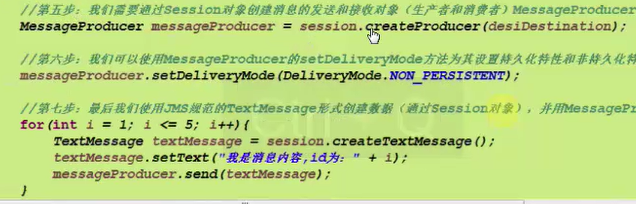
## 3.1 JMS Provider

实现JMS接口的消息中间件称为JMS Provider。

## 3.2 生产者（Message Provider）

消息的产生者。

图1



## 3.3 消费者（Message Consumer）

消息的接收者。

图1



## 3.4 点对点模型（PTP）（Point to Point）

点对点的通信：生产者将消息放到队列中，消费者从队列中取消息。

（注意：一个队列支持多个消费者，但队列中的一个消息只能被一个消费者消费，换句话说，消息1如果被消费者A消费了，消费者B不能消费消息1）

图1



## 3.5 发布/订阅模型（Pub/Sub）

发布/订阅模式，生产者发布消息，消息保存在一个主题（TOPIC）中，订阅该主题的消费者可以从主题中获取消息。

（注意：一个主题可以有多个消费者订阅，主题中的一个消息可以同时被多个消费者消费）

图1



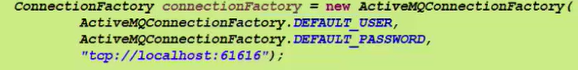
## 3.6 队列（Queue）

Queue是消息队列，发送者把消息发给消息服务器后，消息服务器把消息放在队列（Queue）中。

## 3.7 连接工厂（ConnectionFactory接口）

连接工厂可以配置连接中间件的账号，密码，URL。

图1



## 3.8 连接（Connection接口）

使用ConnectionFactory可以创建一个连接实例。

（注意：连接默认是关闭的）

图1



## 3.9 目的地（Destincation接口）

消息目标是指消息发送到哪里或者接收的消息从哪来，可以是队列（Queue，点对点模式中），也可以是主题（TOPIC，在发布/订阅模式中）。

（quque1是队列名称）

图1

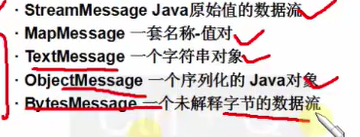


## 3.10 消息（Message接口）

消息包括：

1. 消息头（必须）：包含消息的路由等信息。
2. 一组消息属性（可选）：包含额外的属性。
3. 一个消息体（可选）：有五种类型的消息，如图1。

图1



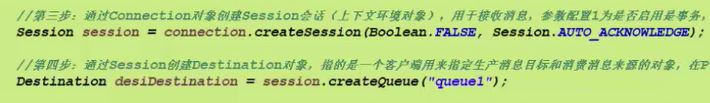
## 3.11 会话（Session接口）

Session是一个单线程的上下文，用来发送和接收消息。

（注意：因为Session是单线程的，所以接收是按照发送的顺序一个一个接收的）

Session支持事务，在事务提交前，都可以回滚来取消消息。

图1



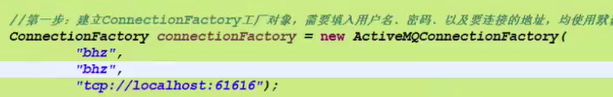
## 3.12 安全认证

必须设置账号密码才能建立ConnectionFactory接口的正确实现。

图1



图2



## 3.13 签收模式

签收模式表示接收者收到队列传送过来的消息后，是否告诉Session我（接收者）已经收到了。

AUTO\_ACKNOWLEDGE：接收者收到消息后，自动告诉Session我已经收到消息了。

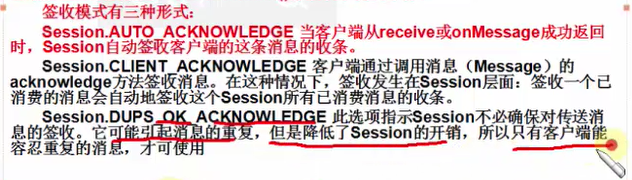
Client\_ACKNOWLEDGE：接收者收到消息后，需要接收者调用方法才能告诉Session我收到消息了。（注意：如果不调用方法告诉Session已经收到了，Session会认为这条消息没有被消费，则队列中的消息不会减少）

图1



DUPS\_OK\_ACKNOWLEDGE：在多个接收者存在时，不确保签收会导致重复发送消息。

图1



### 3.13.1 手工签收（Client\_ACKNOWLEDGE）的优点

因为一般接收消息后，都会进行一系列业务操作，手工签收可以当后续业务操作失败时不签收，这样这条消息就仍然保存在队列中。

## 3.14 使用事务

（注意：事务要提交，否则队列中的消息不会减少）。

图1



## 3.15 消费模式

### 3.15.1 同步消费（Synchronous）

消费者（订阅者或接收方）通过receive()来接收消息。

（注意：receive()会一直阻塞直到消息到达，如果没有消息会一直阻塞在这个方法上）

### 3.15.2 异步消费（Asynchronous）

消费者（订阅者或接收方）需要注册一个消息监听器，只要消息到达，通过监听器的onMessage()方法接收消息。

## 3.16 MQ和JMS的关系

JMS是一个规范，一个接口，MQ是具体实现JMS规范的产品，具体有ActiveMQ，RocketMQ，RabbitMQ，HornetMQ等。

# 消息

## 4.1 消息优先级

优先级总共有0-9十个级别，0-4是普通消息，5-9是加急消息。如图1的第4个参数i是优先级。

（注意：ActiveMQ的消息不能百分之百保证加急消息比全部普通消息先到达目的地）

图1



## 4.2 消息过期

消息在MQ服务不关闭的前提下，默认是永不过期的。

### 4.2.1 设置过期时间

因为消息默认永不过期，可以设置超时时间，超过这个时间消息过期。如图1是2分钟后失效。

图1



## 4.3 消息持久化

消息默认不持久化，关闭MQ服务后，消息会丢失。

# 安全策略

如果不加安全策略，只要别人知道我的MQ的IP和端口，都可以向里面添加和编辑消息，这是很危险的。

最简单的安全策略是设置连接MQ的账号和密码。

（注意：图1是连接MQ的账号密码，不是MQ管控台（8161端口的web）的登录账号密码）

图1

