Atitit 程序调用方式 api rpc rest mq 模式

目录

[回调接口模式事件驱动 1](#_Toc17516)

[bus总线 1](#_Toc28920)

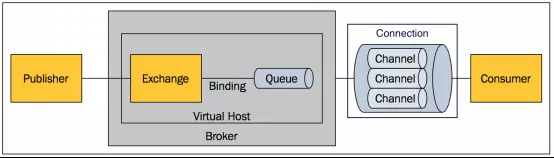
## 回调接口模式事件驱动

不一定使用mq实现

## bus总线

Message Queue的需求由来已久，80年代最早在金融交易中，高盛等公司采用Teknekron公司的产品，当时的Message queuing软件叫做：the information bus（TIB）

于是2004年，摩根大通和iMatrix开始着手Advanced Message Queuing Protocol （AMQP）开放标准的开发。2006年，AMQP规范发布。2007年，Rabbit技术公司基于AMQP标准开发的RabbitMQ 1.0 发布。

**2. AMQP messaging 中的基本概念**  


* **Broker**: 接收和分发消息的应用，RabbitMQ Server就是Message Broker。
* **Virtual host**: 出于多租户和安全因素设计的，把AMQP的基本组件划分到一个虚拟的分组中，类似于网络中的namespace概念。当多个不同的用户使用同一个RabbitMQ server提供的服务时，可以划分出多个vhost，每个用户在自己的vhost创建exchange／queue等。
* **Connection**: publisher／consumer和broker之间的TCP连接。断开连接的操作只会在client端进行，Broker不会断开连接，除非出现网络故障或broker服务出现问题。
* **Channel**: 如果每一次访问RabbitMQ都建立一个Connection，在消息量大的时候建立TCP Connection的开销将是巨大的，效率也较低。Channel是在connection内部建立的逻辑连接，如果应用程序支持多线程，通常每个thread创建单独的channel进行通讯，AMQP method包含了channel id帮助客户端和message broker识别channel，所以channel之间是完全隔离的。Channel作为轻量级的Connection极大减少了操作系统建立TCP connection的开销。
* **Exchange**: message到达broker的第一站，根据分发规则，匹配查询表中的routing key，分发消息到queue中去。常用的类型有：direct (point-to-point), topic (publish-subscribe) and fanout (multicast)。
* **Queue**: 消息最终被送到这里等待consumer取走。一个message可以被同时拷贝到多个queue中。
* **Binding**: exchange和queue之间的虚拟连接，binding中可以包含routing key。Binding信息被保存到exchange中的查询表中，用于message的分发依据

****那么谁应该负责创建这个queue呢？是Consumer，还是Producer？****

如果queue不存在，当然Consumer不会得到任何的Message。但是如果queue不存在，那么Producer Publish的Message会被丢弃。所以，还是为了数据不丢失，Consumer和Producer都try to create the queue！反正不管怎么样，这个接口都不会出问题。

   queue对load balance的处理是完美的。对于多个Consumer来说，RabbitMQ 使用循环的方式（round-robin）的方式均衡的发送给不同的Consumer。

*****2）ACK - 消息确认*****

默认情况下，如果Message 已经被某个Consumer正确的接收到了，那么该Message就会被从queue中移除。当然也可以让同一个Message发送到很多的Consumer。

    如果一个queue没被任何的Consumer Subscribe（订阅），那么，如果这个queue有数据到达，那么这个数据会被cache，不会被丢弃。当有Consumer时，这个数据会被立即发送到这个Consumer，这个数据被Consumer正确收到时，这个数据就被从queue中删除。

     那么什么是正确收到呢？通过ack。每个Message都要被****acknowledged****（确认，ack）。我们可以显示的在程序中去ack，也可以自动的ack。如果有数据没有被ack，那么：

     RabbitMQ Server会把这个信息发送到下一个Consumer。

    如果这个app有bug，忘记了ack，那么RabbitMQ Server不会再发送数据给它，因为Server认为这个Consumer处理能力有限。

   而且ack的机制可以起到限流的作用（****Benefitto throttling****）：在Consumer处理完成数据后发送ack，甚至在额外的延时后发送ack，将有效的balance Consumer的load。

   当然对于实际的例子，比如我们可能会对某些数据进行merge，比如merge 4s内的数据，然后sleep 4s后再获取数据。特别是在监听系统的state，我们不希望所有的state实时的传递上去，而是希望有一定的延时。这样可以减少某些IO，而且终端用户也不会感觉到。

*****2、核心概念*****

*****1）Exchange和Binding*****

交换机Exchange拿到一个消息之后会将它路由给队列。Exchange使用哪种方式路由是由Binding规则决定的。

****a）直连交换机****

根据消息携带的路由键（routing key）将消息投递给对应队列。直连交换机用来处理消息的单播路由。

Message中的“routing key”如果和Binding中的“binding key”一致， Direct exchange则将message发到对应的queue中。

****b）主题交换机****

通过对消息的路由键和队列到交换机的绑定模式之间的匹配，将消息路由给一个或多个队列。主题交换机用来实现消息的多播路由。

****c）扇形交换机****

将消息路由给绑定到它身上的所有队列，且不理会路由键。扇形交换机用来处理消息的广播路由。