[**rabbitmq学习1:hello world**](http://wubin850219.iteye.com/blog/1002932)

 rabbitMQ是一个在AMQP基础上完整的，可服用的企业消息系统。他遵循**Mozilla Public License** 开源协议。

  关于amqp可参考<http://www.oschina.net/p/rabbitmq/>

 rabbitmq是一个消费的代理；通过生产者客户端生产一个信息，转送给消费者客户端;在这个传输过程中，根据你的需要可以经过路由、缓冲、持久化来得到这个消息。

  先通过一个例子开始：通过rabbitmq输出"Hello World!"

http://dl.iteye.com/upload/attachment/0074/4725/b1eadeac-9491-3612-87db-175f208d7d4c.png

其中P代表生产者、C表示消费者、中间红色部分代表消息队列

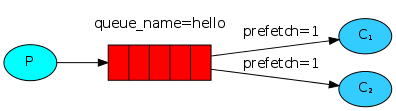
参照：<http://wubin850219.iteye.com/blog/1002932>

[**rabbitmq学习2:Work Queues**](http://wubin850219.iteye.com/blog/1003840)

在前面的已经提到了一对一的情况;现在一个生产者与多个消费者的情况（Work Queues）。  
Work Queues的示意图如下：



对于上图的模型中对于c端的worker来说。RabbitMQ服务器可能一直发送多个消息给一个worker，而另一个可能几乎不做任何事情。这样就会导致一个worker很忙，而另一个却很空闲。这种情况可能都不想出现。如何解决这个问题呢。当然最理想的情况是均匀分配消息给每个worker。我们可能通过channel . basicQos(1)方法（ prefetchCount = 1 ）来设置同一时间每次发给一个消息给一个worker。示意图如下：

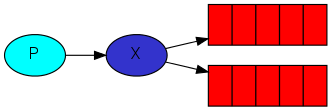


参照：<http://wubin850219.iteye.com/blog/1003840>

[**rabbitmq学习3:Publish/Subscribe**](http://wubin850219.iteye.com/blog/1004921)

在前面的Work Queue中的消息是均匀分配消息给消费者；如果我想把消息分发给所有的消费者呢？那应当怎么操作呢？这就是要下面提到的Publish/Subscribe（分布/订阅）。让我们开始Publish/Subscribe之旅吧!

Publish/Subscribe的工作示意图如下：



在上图中的X表示Exchange(交换区);Exchange的类型有：direct , topic , headers 和 fanout

Publish/Subscribe的Exchang的类型为fanout;声明Publish/Subscribe的Exchang代码如下：

**Java代码  [收藏代码](javascript:void())**

1. channel.exchangeDeclare("logs", "fanout");

对于Work Queue中提到的发布消息的代码如下：

**Java代码  [收藏代码](javascript:void())**

1. channel.basicPublish("", queueName,   **null**, message.getBytes());

 但对于Publish/Subscribe中发布消息中的Queue的使用的是默认的；代码如下：

**Java代码  [收藏代码](javascript:void())**

1. channel.basicPublish( "logs", "", **null**, message.getBytes());

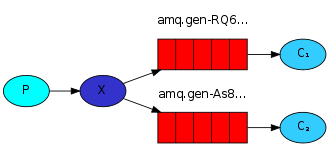
Exchange和各Queue之间是如何通信的呢？主要是通过把Exchange和各Queue绑定（binding）；示意代码如下:

**Java代码  [收藏代码](javascript:void())**

1. channel.queueBind(queueName, exchangeName, "");

Publish/Subscribe加入绑定的工作示意图如下：

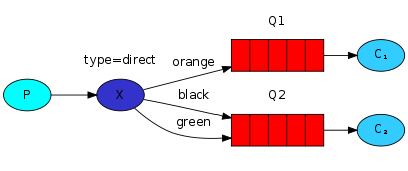




<http://wubin850219.iteye.com/blog/1004921>

[**rabbitmq学习4:Routing**](http://wubin850219.iteye.com/blog/1004948)

    在《[rabbitmq学习3:Publish/Subscribe](http://wubin850219.iteye.com/blog/1004921) 》中已经学习了发送一个消息，所有消费者端都能收到。那现在这节准备介绍通过路由规则来接受生产者端所发送的消费。Routing的工作示意图如下：

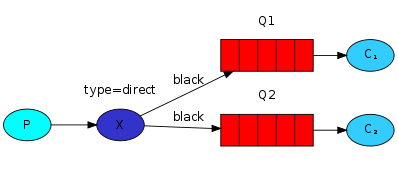


对于Routing的示意图与Publish/Subscribe中的示意图区别：

第一：Publish/Subscribe的Exchange的类型为“fanout”,而Routing的类型为“direct”

第二：Publish/Subscribe的路由为默认的，而Routing的路由是自定义的。

可能从上图的示意图如可以发现可以把Routing的模式也可以转化Publish/Subscribe的模式，如示意图



我们也可能把所有的数据发送到一个Queue中去，示意图如下：



<http://wubin850219.iteye.com/blog/1004948>

[**rabbitmq学习5:Topics**](http://wubin850219.iteye.com/blog/1004973)

   在前面的《[rabbitmq学习4:Routing](http://wubin850219.iteye.com/blog/1004948) 》中使用一般的名字的路由，现在想通过一些路由规则让消费者来接受符合规则的消息？那应当怎么样呢？那就要用到类型为topic的Exchange了。

Topics的工作示意图如下：



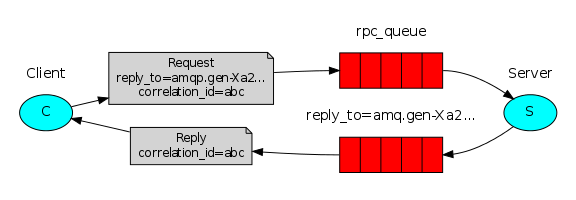
我们可能从图中看到有\*和#两个通配符。\*表示通配一个词；#表示通配0个或多个词。

<http://wubin850219.iteye.com/blog/1004973>

[**rabbitmq学习6:RPC**](http://wubin850219.iteye.com/blog/1006300)

   在《[rabbitmq学习2:Work Queues](http://wubin850219.iteye.com/blog/1003840) 》中我们已经知道了在多个worker如何分配耗时的任务。如果我现在要在远程的机器上运行然后得到结果，那应当怎么做呢？那就要用到RPC（*Remote Procedure Call* or *RPC* ）了！

   现在来看看来看看Rabbitmq中RPC吧！RPC的工作示意图如下：



   上图中的C代表客户端，S表示服务器端；Rabbitmq中的RPC流程如下：

1、首先客户端发送一个reply\_to和corrention\_id的请求,发布到RPC队列中；

2、服务器端处理这个请求，并把处理结果发布到一个回调Queue,此Queue的名称应当与reply\_to的名称一致

3、客户端从回调Queue中得到先前corrention\_id设定的值的处理结果。如果碰到和先前不一样的corrention\_id的值，将会忽略而不是抛出异常。

  对于上面所提到的回调Queue中的消费处理使用的是BasicProperties类；而消息 属性在AMQP的协议中规定有14个；而很多大部分我们没有用到。常用的几个属性有：

delivery\_mode : 标记消息是持久性消息还是瞬态信息。在前面的“Work Queue”中我们已经提到过；

  content\_type : 用来描述MIME的类型。如把其类型设定为JSON；

  reply\_to : 用于命名一个回调Queue；

  correlation\_id : 用于与相关联的请求的RPC响应.

<http://wubin850219.iteye.com/blog/1006300>