# Rabbitmq 消息队列

## 消息队列使用场景&好处

消息队列一般是在项目中,将一些无需及时返回且耗时的操作提取出来,进行了异步处理,而这种异步处理的方式大大的节省了服务器的请求相应时间,从而提高了系统的吞吐量.

在项目启动之初来预测将来项目会碰到什么需求,是极其困难的. 消息队列在处理过程中插入了一个隐含的,基于数据的接口层,两边的处理过程都要实现这一接口.这允许你独立的扩展和修改两边的处理过程,只要确保他们遵守同样的借口约束. 消息队列可以解决这样的问题,也就是解耦性. 解耦性伴随的好处就是降低冗余,灵活,易于扩展.

峰值处理能力:

当你的应用上了hacker news的首页,你将发现访问流量攀升到一个不同寻常的水平.在访问量剧增的情况下, 你的应用仍然要继续发挥作用, 但是这样的突发流量并不常见;

如果以为能处理这类峰值访问为标准来投入资源随时待命无疑是巨大的浪费.使用消息队列能够使关键组件顶住增长的访问压力. 而不是因为超出符合的请求和崩溃.

## 二. Rabbitmq的来源

Rabbitmq是用Erlang实现的一个高并发高可靠AMQP消息队列服务器.

显然Rabbitmq与Erlang和AMQP有关,下面简单介绍一下Erlang和AMQP

### Erlang

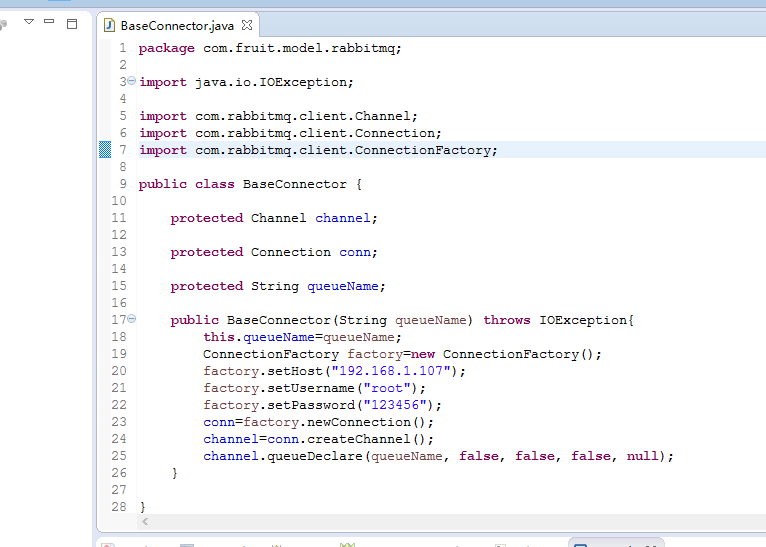
Erlang是一门

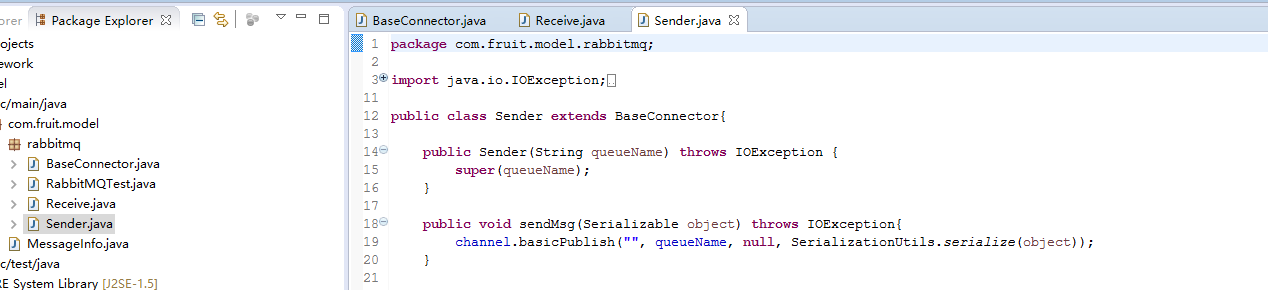
动态类型的函数式编程语言,它也是一门解释型语言,由Erlang虚拟机解释执行. 从语言模型上说,Erlang是基于Actor模型的实现. 在Actor模型里面. 万物皆Actor, 每个Actor都封装着内部状态, Actor相互之间只能通过消息传递这一方式来进行通信, 对应到Erlang里,每个Actor对应一个Erlang进程, 进程之间通过消息进行通信. 相比共享内存, 进程间通过消息传递来通信带来的直接好处就是消除了直接的锁开销(不考虑Erlang虚拟机底层实现中的锁应用).

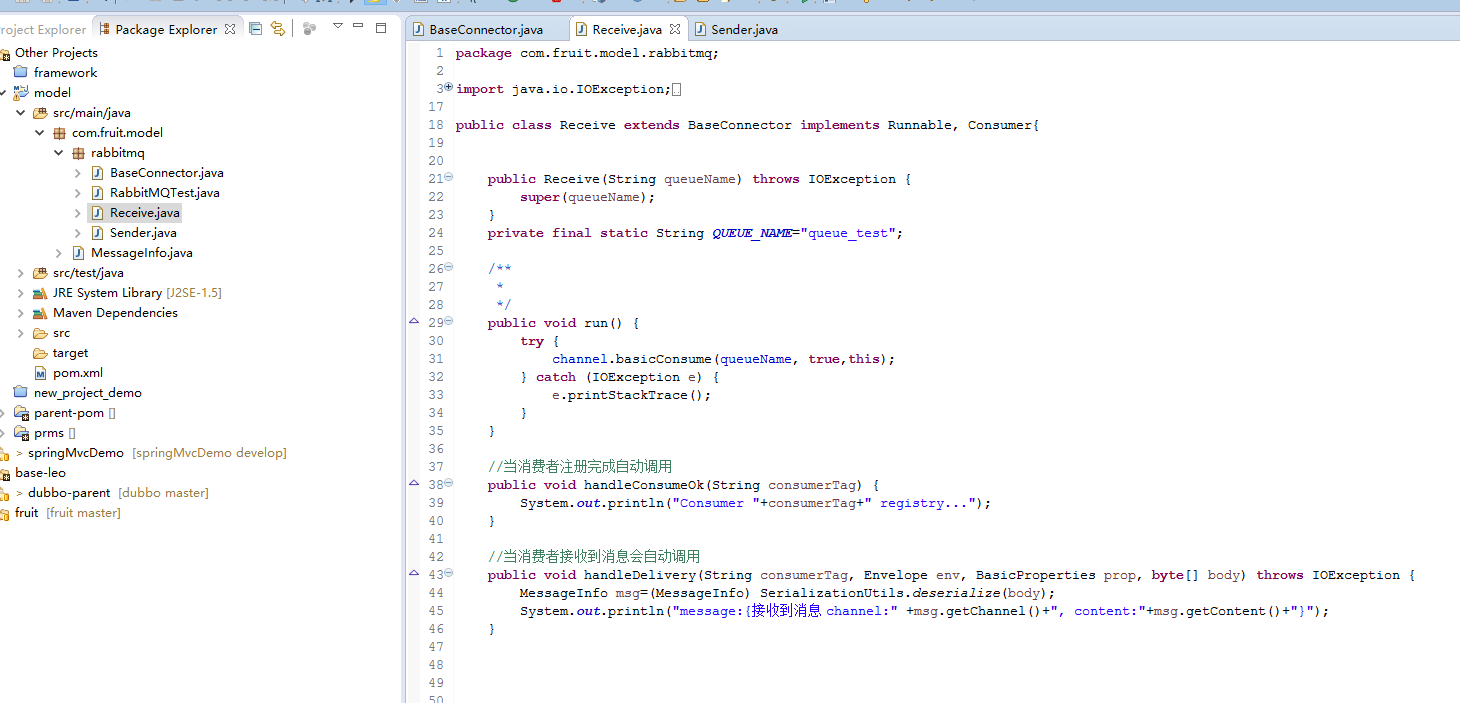
### AMQP

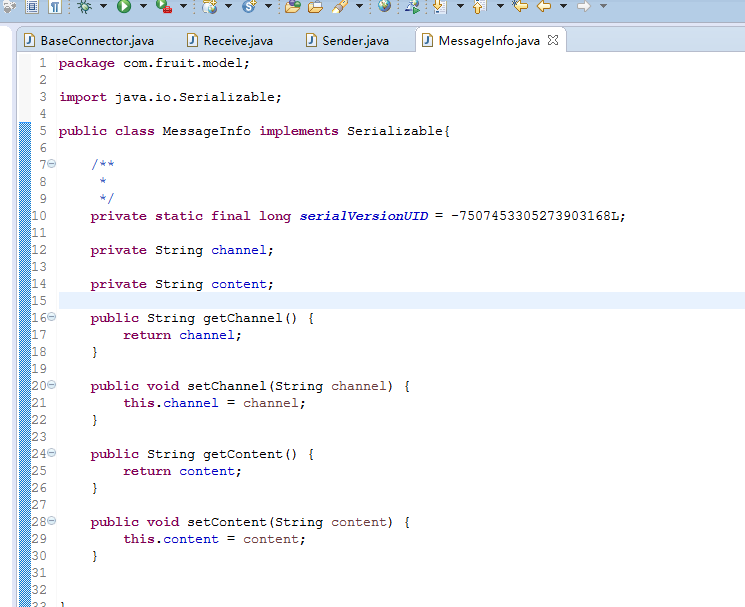
AMQP(Advanced Message Queue Protocol)定义了一种消息系统规范,这个规范描述了在一个分布式系统中各个子系统如何通过消息交互. 而Rabbitmq则是AMQP的一种基于Erlang的实现. AMQP将分布式系统中各个子系统隔离开来, 各系统之间不再有依赖. 子系统仅依赖于消息. 子系统不关系消息的发送者,也不关心消息的接受者.

## Rabbitmq 实例(JAVA)









## RabbitMQ使用的道具具体介绍

RabbitMQ是用Erlang实现的高并发,高可靠 消息队列服务器

Erlang是RabbitMQ的一个依赖环境,没什么好说的,我们更加关注它的一身表演技巧从哪里来的.这里就看AMQP吧(Advanced Message Queue Protocol),看完AMQP估计会对Rabbitmq有更深入的理解.

### AMQP

AMQP中有四个概念非常重要: 虚拟主机(virtual host),交换机(exchange),队列(queue),和绑定(binding) .一个虚拟机持有一组 交换机,队列和绑定.

为什么需要多个虚拟机呢?

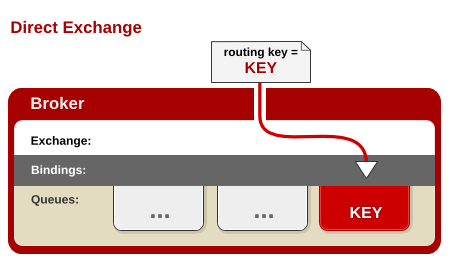
原因很简单,Rabbitmq当中,用户只能在虚拟机的粒度控制权限. 因此 如果需要禁止A组访问B组的交换机/队列/绑定, 必须为A和B分别创建一个虚拟机. 每个Rabbitmq都有一个默认的虚拟机 “/”. 如果这就够了,那现在就可以开始了.

#### QUEUE

队列(queue)是你的消息(message)的终点,可以理解为装消息的容器. 消息就一直在里面. 直到有客户端也就是消费者(consumer)连接到这个队列且将其取走为止,

#### EXCHANGE

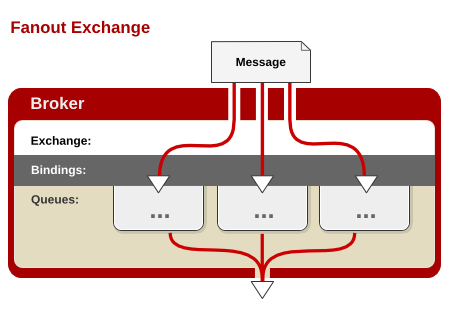
##### Direct Exchange



任何发送到Direct Exchange的消息都会被转发到RouteKey中指定的Queue.

1. 一般情况可以使用rabbitMQ自带的Exchange: “” (该Exchange的名字为空字符串,下文称其为default Exchange)
2. 这种模式下不需要将Exchange进行任何绑定(binding)操作.
3. 消息传递时需要一个”RouteKey”, 可以简单的理解为要发送到的队列名字.
4. 如果vhost中不存在RouteKey中指定的队列名,则该消息会被抛弃.

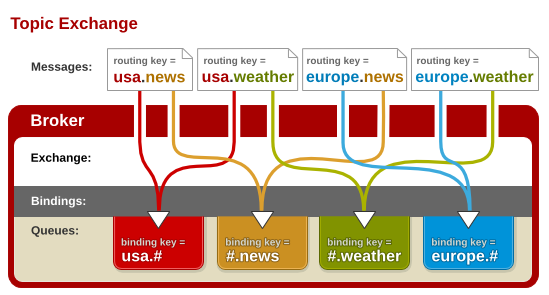
##### Fanout Exchange



任何发送到Fanout Exchange的消息都会被转发到与该Exchange绑定的(binding)所有Queue上

1. 可以理解为路由麦的模式.
2. 这种模式不需要RouteKey
3. 这种模式需要提前将Exchange于Queue进行绑定,一个Exchange可以绑定多个Queue,一个Queue可以同多个Exchange进行绑定.
4. 如果接受到消息的Exchange没有与任何Queue绑定,则消息会被抛弃.

##### Topic Exchange



任何发送到Topic Exchange的消息都会被转发到所有关心RouteKey中指定话题的Queue上

1. 这种模式较为复杂,简单来说,就是每个队列都有其关心的主题,所有的消息都带有一个”标题”(RouteKey),Exchange会将消息转发到所有关注主题 能与RouteKey模糊匹配的队列.
2. 这种模式需要RouteKey, 也需要提前绑定Exchange与Queue.
3. 在进行绑定时, 需要提供一个该队列关心的主题, 如 “#.log.#” 表示该队列关心所有涉及 log 的消息 (一个RouteKey 为”MQ.log.error”)的消息会被转发到该队列.
4. “#”表示0个或若干个关键字，“\*”表示一个关键字。如“log.\*”能与“log.warn”匹配，无法与“log.warn.timeout”匹配；但是“log.#”能与上述两者匹配。
5. 同样，如果Exchange没有发现能够与RouteKey匹配的Queue，则会抛弃此消息。