5-2 RabbitMQ入门

一、选用RabbitMQ

在消息队列产品的选择上，我选用了RabbitMQ。估计有很多同学质疑，为什么不选择Kafka呢？这里我需要解释几句。

消息队列产品有很多，比如说常见的有RocketMQ、RabbitMQ、ActiveMQ和Kafka。其中Kafka的性能是最好的，并发量比较大，而且消息收发的速度也非常快。但是消息收发的可靠性上，Kafka不如RabbitMQ，而且技术选型的时候执行速度并不是唯一标准。比速度的话，汇编语言碾压一切高级语言，但是现在我们写程序几乎不会选用汇编语言，而是要兼顾开发效率、易用性和生态圈。RabbitMQ还有另外一个杀手锏，那就是既支持消息异步收发，又支持同步收发，这个太牛了。虽然我们现在大部分的场景对应的是消息异步收发，但是有的场合要支持消息的同步收发，这时候RabbitMQ能适应各种业务场景的优点就显现出来了。所以在项目立项的时候，选择RabbitMQ是最稳妥的方案。

安装RabbitMQ

因为RabbitMQ依赖Erlang环境，其实在Windows上面安装起来并不复杂，但是在MacOS或者Linux系统上面安装Erlang和RabbitMQ比较麻烦，而且最后我们开发的Emos项目要发布到云主机上面，所以这里我们借助Docker环境，拉取一个RabbitMQ镜像，然后创建容器就可以使用RabbitMQ了，非常简单方便。

把git项目中的`rabbitmq.tar.gz`文件上传到CentOS系统，把镜像导入到Docker环境。

```
1. docker load < rabbitmq.tar.gz
```

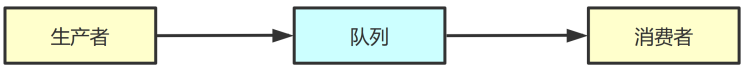
执行命令，创建`rabbitmq`容器，然后就可以使用`RabbitMQ`消息队列了。

```
1. docker run -it -d --name mq  -p 15672:15672 -p 5672:5672 rabbitmq
```

二、RabbitMQ的五种队列模式

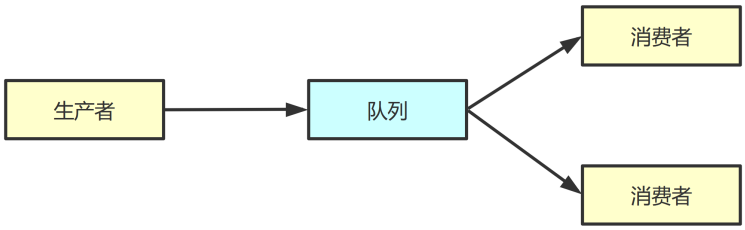
1. 简单模式

一个生产者（发送方）对应一个消费者（接收方）



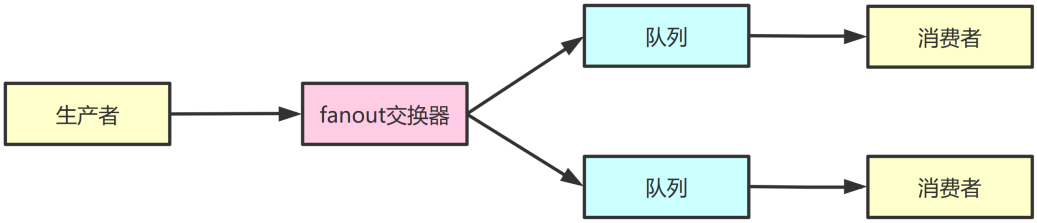
2. Work模式

一个生产者对应多个消费者，但是只能有一个消费者获得消息（排他）



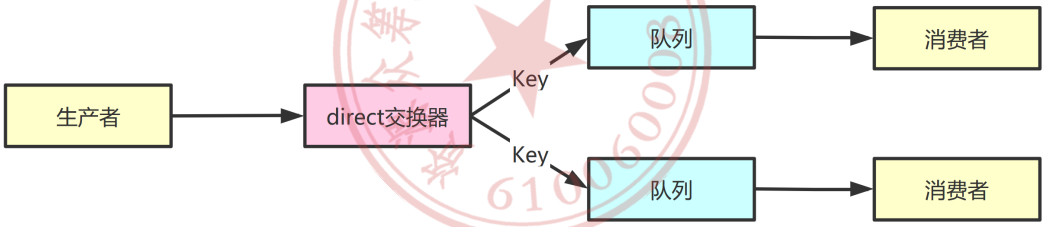
3. 发布/订阅模式

一个消费者将消息首先发送到fanout交换器，交换器绑定到多个队列，然后与之对应的所有消费者都能接收到消息（不排他）



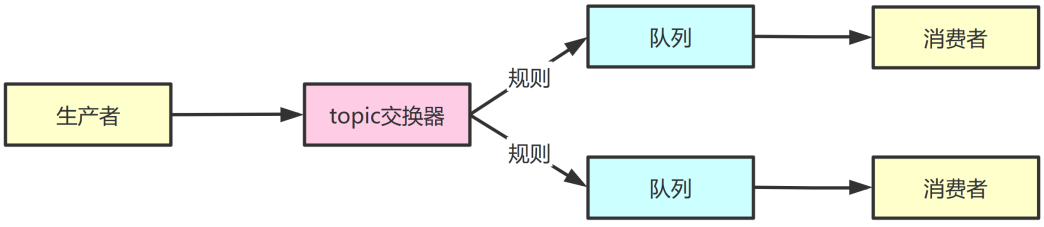
4. 路由模式

生产者将消息发送到direct交换器，交换器按照关键字（Key），把消息路由到某个队列



5. 主题模式

生产者将消息发送到Topic交换器，交换器按照复杂的规则，把消息路由到某个队列



三、消息持久化

消息的可靠性是RabbitMQ的一大特色，那么RabbitMQ是如何保证消息可靠性的呢？答案就是消息持久化。持久化可以防止在异常情况下丢失数据。除了消息持久化之外，甚至交换器和队列都能持久化。

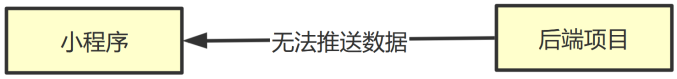
四、消息过期时间

默认情况下，消息是无限期存储在RabbitMQ上面的，但是我们可以设置消息过期时间，到期之后无论该消息是否已经被接收，都会被RabbitMQ删除。

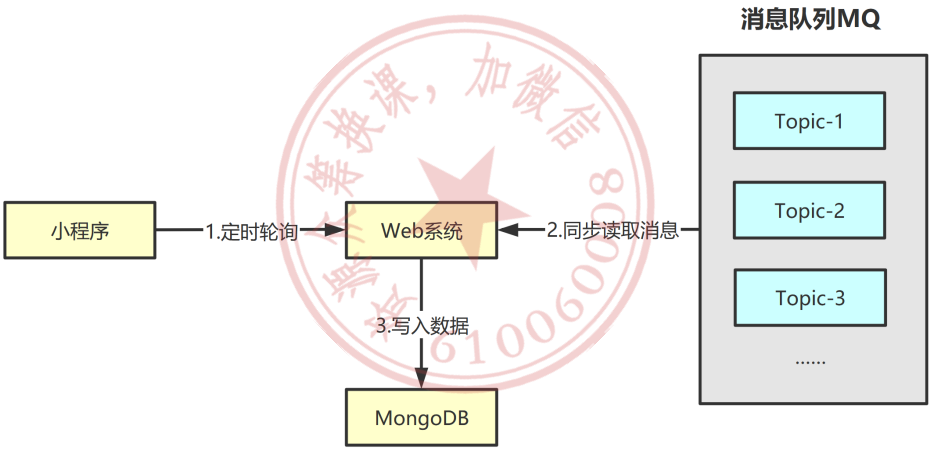
五、Ack应答

消费者接收消息之后，必须返回一个Ack应答，那么RabbitMQ才会认为这条消息接收成功。如果想要删除这条消息，消费者发送Ack应答的时候，附带一个 `deliveryTag` 标志位就可以了。

六、同步接收和异步接收



很多网上的RabbitMQ代码案例，消费者采用的是异步的接收消息。的确这种方式，消耗的系统资源较少。但是小程序和后端项目之间并不是长连接，所以后端项目异步方式接收到队列中的消息，也无法推送给移动端的小程序。也许有人想到了小程序自带的消息推送机制，但是这个推送功能是有严格限制的，有效期和推送的次数都做了非常严格的限定。



我们可以让后端的Java项目采用同步的方式接收队列中的消息。在移动端，我们创建定时器，然后向后端Java项目发出轮询请求。后端Java项目接收到轮询请求之后，用同步方式接收队列中的消息，然后把消息存储在MongoDB上面，最后向小程序返回接收了多少条新消息，移动端则弹出提示框告知用户有信息的消息通知。