# RabbitMQ简介

* Message

消息由消息头和消息体组成。消息体是不透明的，而消息头则由一系列的可选属性组成，这些属性包括routing-key（路由键）、priority（相对于其他消息的优先权）、delivery-

该消息可能需要持久性存储）等。

* Publisher

消息的生产者，也是一个向交换器发布消息的客户端应用程序。

* Exchange

交换器，用来接收生产者发送的消息并将这些消息路由给服务器中的队列。

Exchange有4种类型：direct(默认)，fanout, topic, 和headers，不同类型的Exchange转发消息的策略有所区别

* Queue

消息队列，用来保存消息直到发送给消费者。它是消息的容器，也是消息的终点。一个消息可投入一个或多个队列。消息一直在队列里面，等待消费者连接到这个队列将其取走。

* Binding

绑定，用于消息队列和交换器之间的关联。一个绑定就是基于路由键将交换器和消息队列连接起来的路由规则，所以可以将交换器理解成一个由绑定构成的路由表。Exchange 和Queue的绑定可以是多对多的关系。

* Connection

网络连接，比如一个TCP连接。

* Channel

信道，多路复用连接中的一条独立的双向数据流通道。信道是建立在真实的TCP连接内的虚拟连接，AMQP 命令都是通过信道发出去的，不管是发布消息、订阅队列还是接收消息，这些动作都是通过信道完成。因为对于操作系统来说建立和销毁 TCP 都是非常昂贵的开销，所以引入了信道的概念，以复用一条 TCP 连接

* Consumer

消息的消费者，表示一个从消息队列中取得消息的客户端应用程序。

* Virtual Host

虚拟主机，表示一批交换器、消息队列和相关对象。虚拟主机是共享相同的身份认证和加

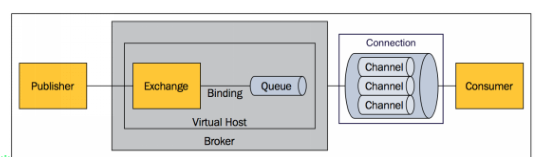
密环境的独立服务器域。每个 vhost 本质上就是一个 mini 版的 RabbitMQ 服务器，拥有

自己的队列、交换器、绑定和权限机制。vhost 是 AMQP 概念的基础，必须在连接时指定，

RabbitMQ 默认的 vhost 是 / 。

* Broker

表示消息队列服务器实体

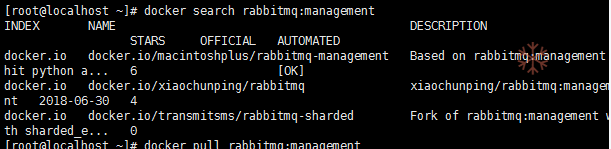


# rabbitmq安装

## 基于docker

1.查询rabbitMQ镜像：management版本，不指定默认为最新版本latest

docker search rabbitmq:management

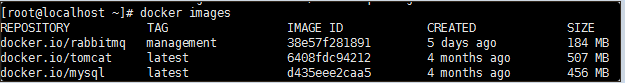


2.拉取RabbitMQ镜像

docker pull rabbitmq:management

3.查看docker镜像列表

docker images



4.启动RabbitMQ：做下端口隐射

docker run -d -p 15672:15672 -p 5672:5672 -e RABBITMQ\_DEFAULT\_USER=guest -e RABBITMQ\_DEFAULT\_PASS=guest --name rabbitmq --hostname=rabbitmqhostone rabbitmq:management

参数介绍:

-d 后台运行

-p 隐射端口

--name 指定rabbitMQ名称

RABBITMQ\_DEFAULT\_USER 指定用户账号

RABBITMQ\_DEFAULT\_PASS 指定账号密码

5.执行如上命令后访问：http://ip:15672/

输入默认账号密码：guest/guest

## 管理界面

### 登录

http://ip:15672/

输入默认账号密码：guest/guest



### Admin新增用户

用户管理和权限管理都在Admin页签里



1、超级管理员(administrator)

可登陆管理控制台，可查看所有的信息，并且可以对用户，策略(policy)进行操作。

2、监控者(monitoring)

可登陆管理控制台，同时可以查看rabbitmq节点的相关信息(进程数，内存使用情况，磁盘使用情况等)

3、策略制定者(policymaker)

可登陆管理控制台, 同时可以对policy进行管理。但无法查看节点的相关信息

4、普通管理者(management)

仅可登陆管理控制台，无法看到节点信息，也无法对策略进行管理。

5、其他

无法登陆管理控制台，通常就是普通的生产者和消费者。

# springBoot集成RabbitMQ

SpringBoot集成rabbitMQ还是比较简单的，因为springBoot使用RabbitTemplate对常用操作进行了封装。

## 导入依赖

接下来我们来看一下集成过程。首先导入依赖。

<!-- 无需在parent的配置文件中添加 -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-amqp</artifactId>

</dependency>

## yml配置

spring:

rabbitmq:

host: 172.15.33.52

port: 5672

username: guest

password: guest

## 三种交换机示例

### direct

首先是配置类，在配置类中我们需要声明交换机，队列和绑定关系。

这里我们创建了一个叫directExchange的交换机，绑定了directQueue和directQueue2两个队列，路由键是direct.



消息的生产者，我们通过一个Controller来进行模拟,直接引用rabbitTemplate

当我在浏览器访问对应连接的时候，就会生产一条消息发送到directExchange交换机，路由key为：direct, 消息内容为：发送一条测试消息：direct



消息的消费者。

由于我们又两个队列都绑定了交换机，且routeKey一样，所以会打印两条。要注意direct只有routeKey完全匹配的时候才能被消费，同时每个队列中的消息只会 被消费一次。

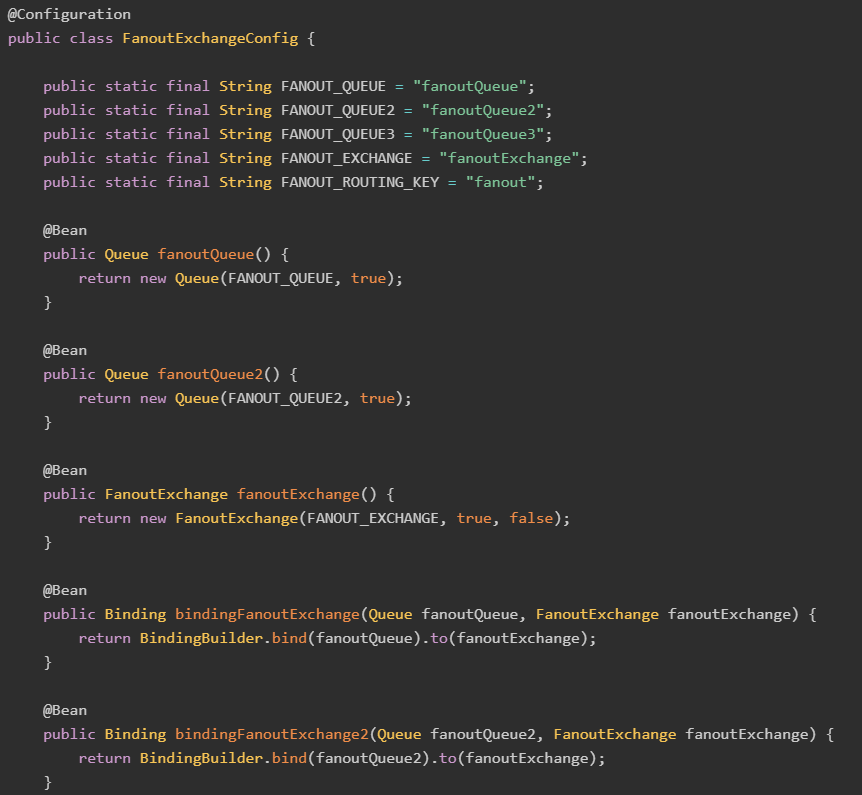


### fanout

配置类

这里也是用一个Fanout类型的交换机绑定了两个队列，要注意在这种模式下，是不需要指定routing-Key的，因为所有绑定的队列都会收到消息。

Fanout模式是发布订阅模式，不需要绑定路由键，this.rabbitTemplate.convertAndSend("amq.fanout","",content,correlationData);，只要和fanout exchange绑定就可以，只要队列绑定了fanout exchange，发送者发消息后，exchange都会将消息发给对应消费者队列



生产者



消费者

fanout交换机： 扇型交换机，这个交换机没有路由键概念，就算你绑了路由键也是无视的。 这个交换机在接收到消息后，会直接转发到绑定到它上面的所有队列

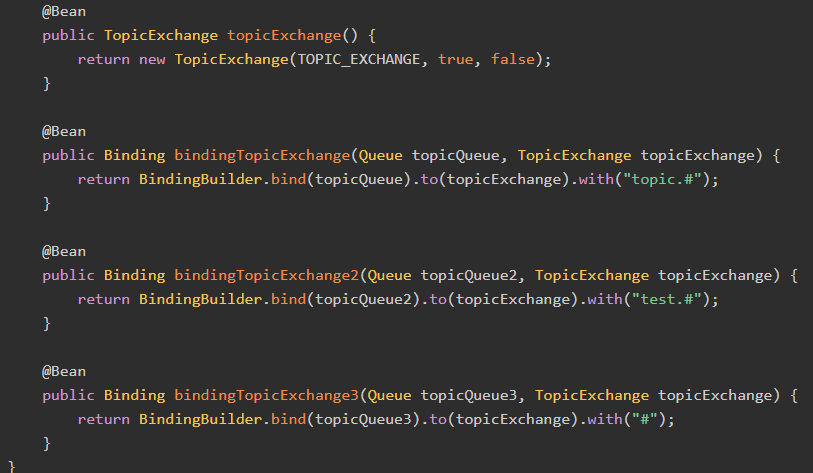


### topic

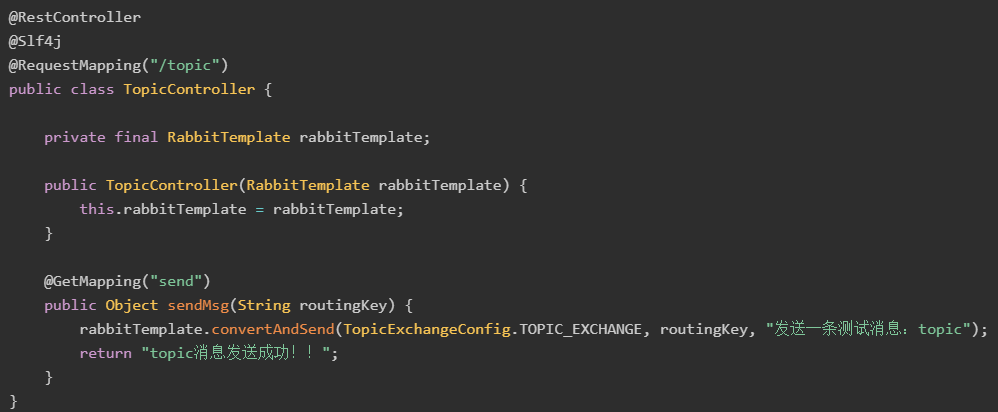
主题交换机，会根据routing-Key的匹配规则，将消息发送到符合规则的队列中。

配置类





生产者



消费者

