# Windows 下安装RabbitMQ服务器

RabbitMQ是一个在AMQP协议标准基础上完整的，可复用的企业消息系统。它遵循Mozilla Public License开源协议，采用 Erlang 实现的工业级的消息队列(MQ)服务器，Rabbit MQ 是建立在Erlang OTP平台上。

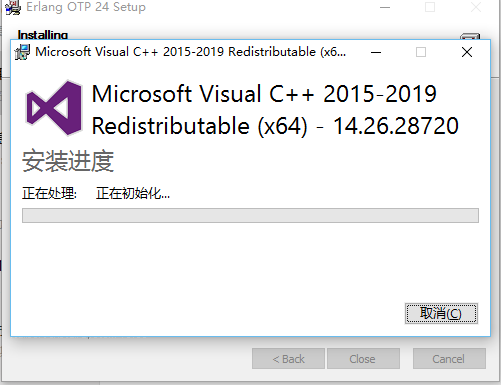
安装RabbitMQ服务器必须首先安装 Erlang 运行环境。

## 安装Erlang

安装Erlang 时要注意安装的RabbityMQ 所依赖的Erlang版本，根据RabbitMQ的要求选择一个版本，这里我要安装的RabbitMQ的最新版本是 24.2 ，他依赖的Erlang版本范

Erlang下载地址：<https://www.erlang.org/downloads>



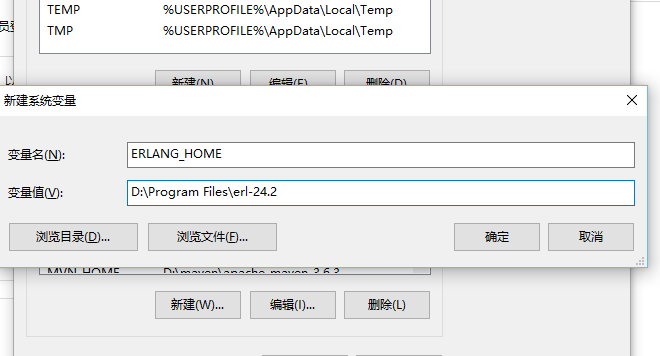


安装完成关闭

## 设置Erlang环境变量

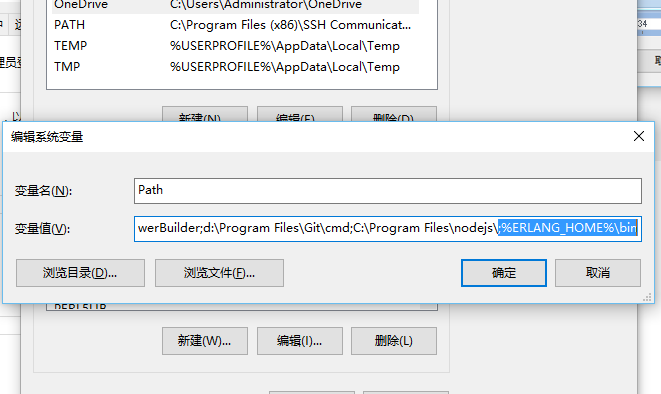
我这里是默认安装所以Erlang的安装路径在：D:\Program Files\erl-24.2

1）新建环境变量 ERLANG\_HOME



2）在系统变量PATH中增加一行

%ERLANG\_HOME%\bin;

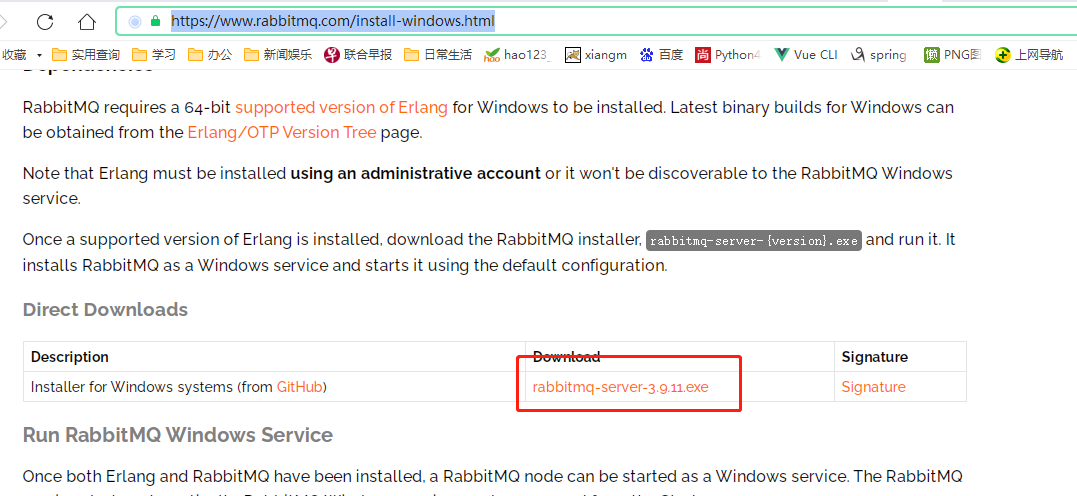


# 安装RabbitMQ服务器

## 下载RabbitMQ服务器安装包

下载地址：<https://www.rabbitmq.com/install-windows.html>

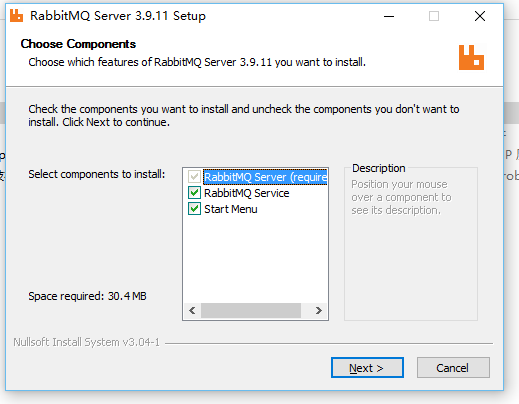
选择与操作系统匹配的版本，建议windows installer。

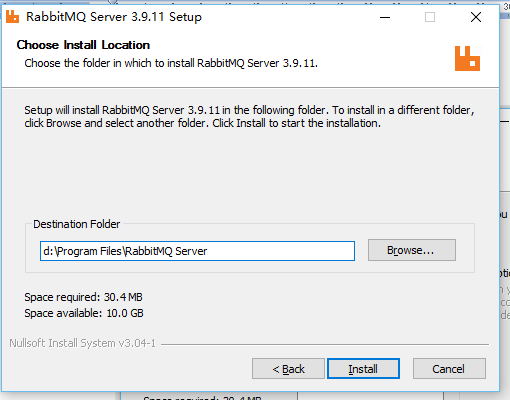


下载文件如下：



1. 安装。安装过程中无配置，下一步确定即可。



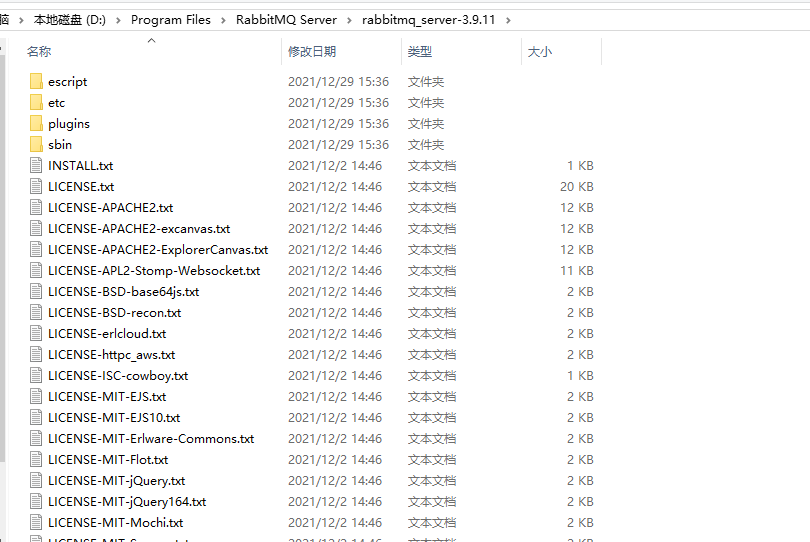


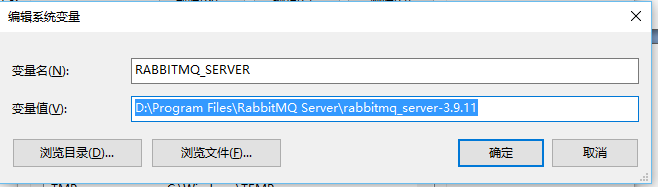
直到安装结束

## 设置RabbitMQ环境变量

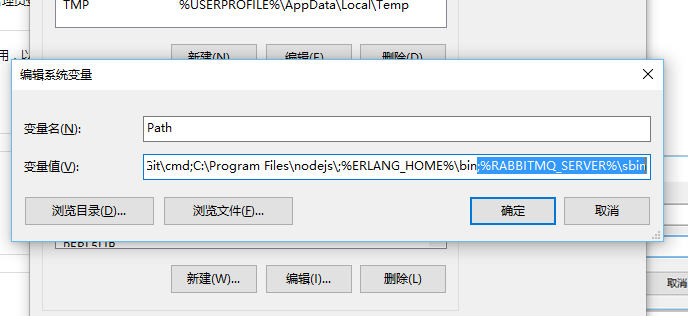
添加一个以RABBITMQ\_SERVER为变量名的环境变量，我们的RabbitMQ 安装在目录

D:\Program Files\RabbitMQ Server\rabbitmq\_server-3.9.11\





然后在系统的path变量中配置如下：;%RABBITMQ\_SERVER%\sbin

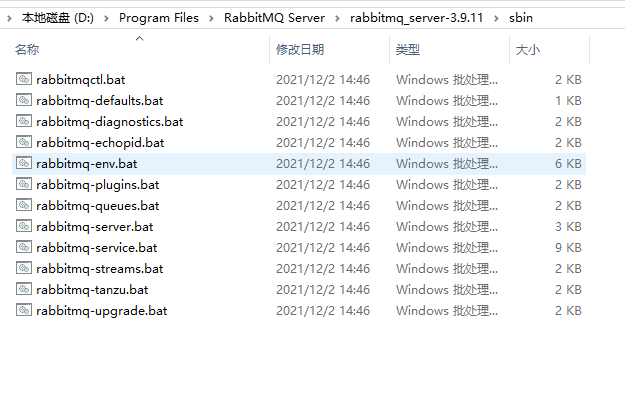


## 配置RabbitMQ服务器

1）以管理员打开cmd，使用CD命令进入RabbitMQ的安装目录的sbin，

我们的安装目录是：

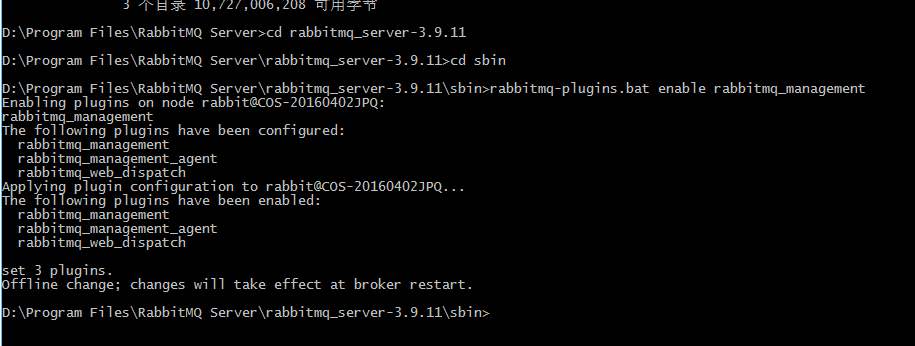
D:\Program Files\RabbitMQ Server\rabbitmq\_server-3.9.11\sbin



执行如下命令开启web管理页面：

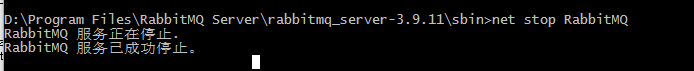
rabbitmq-plugins.bat enable rabbitmq\_management



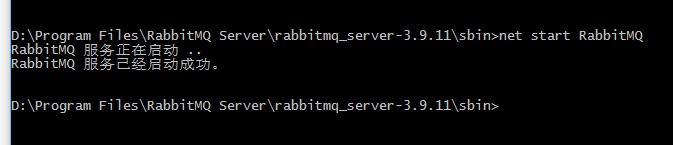


依次使用如下命令重启服务：

net stop RabbitMQ



net start RabbitMQ



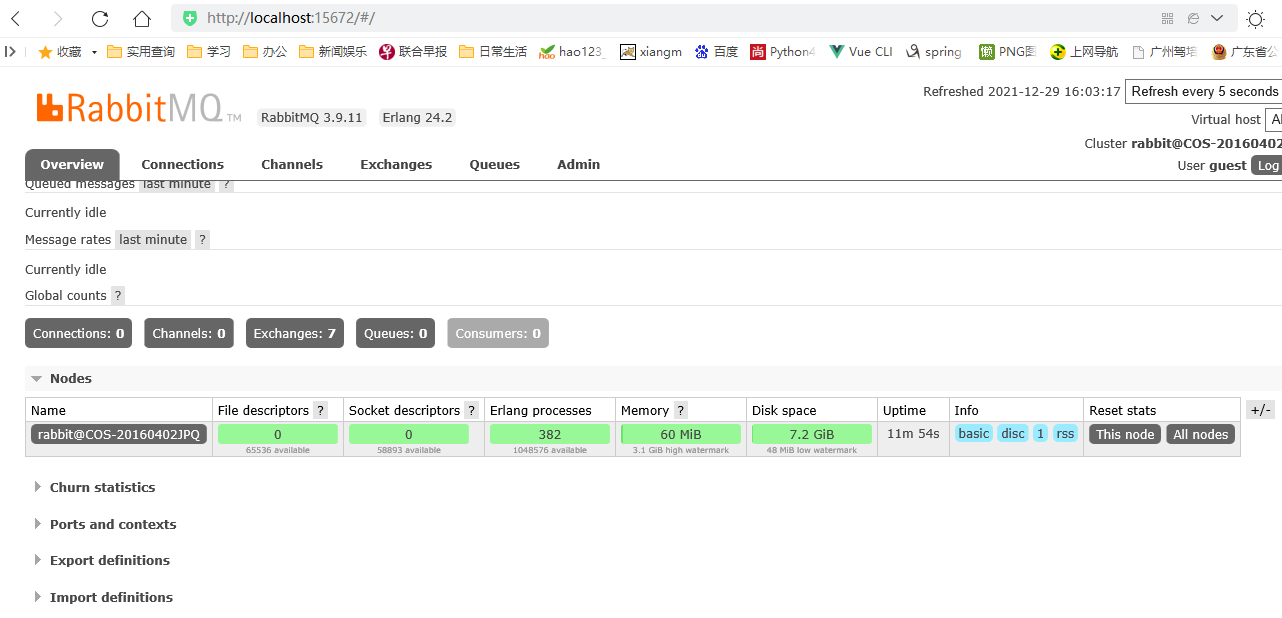
注：上述两条命令为全局命令，必须以管理员命令提示符

## 访问管理页面

[http://localhost:15672/](https://link.zhihu.com/?target=http%3A//localhost%3A15672/)

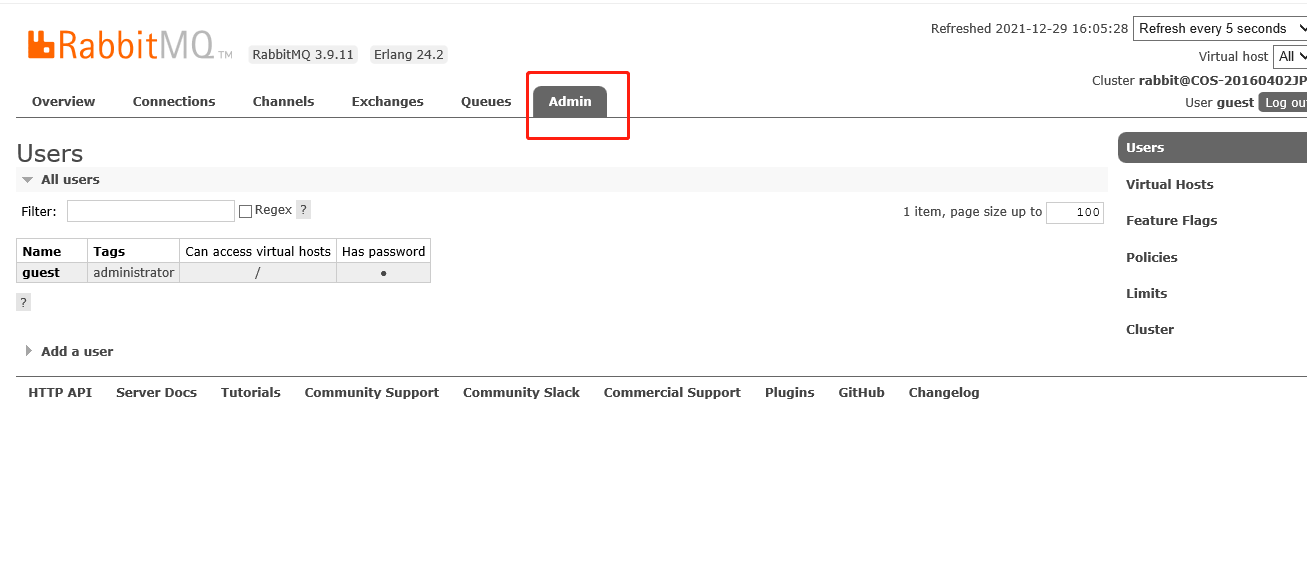
用户名：guest

密码：guest



设置用户：

RabbitMQ服务器安装后，默认仅启用guest账户，只能通过localhost访问。登录后，首先进行用户配置：



然后点击 Add a user

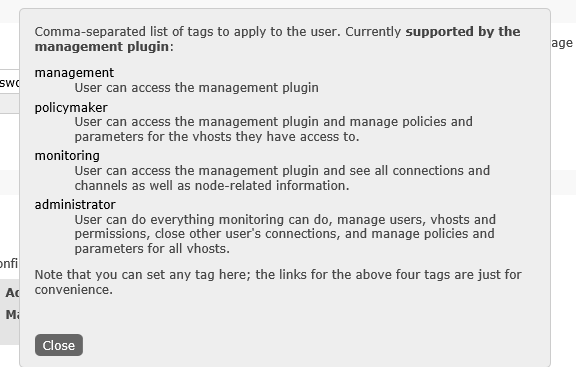


添加一个用户

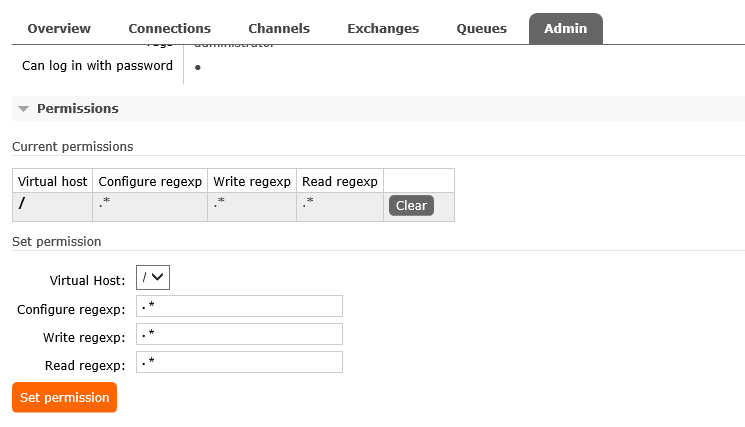
Username：admin Password：123456

Tags 选择 Admin

Tags为该用户的权限组，首个用户建议分配管理员权限，能够使用RabbitMQ服务器的全部功能：



添加完用户后，需要给用户设置主机访问权限，点击用户名，进入用户设置界面：



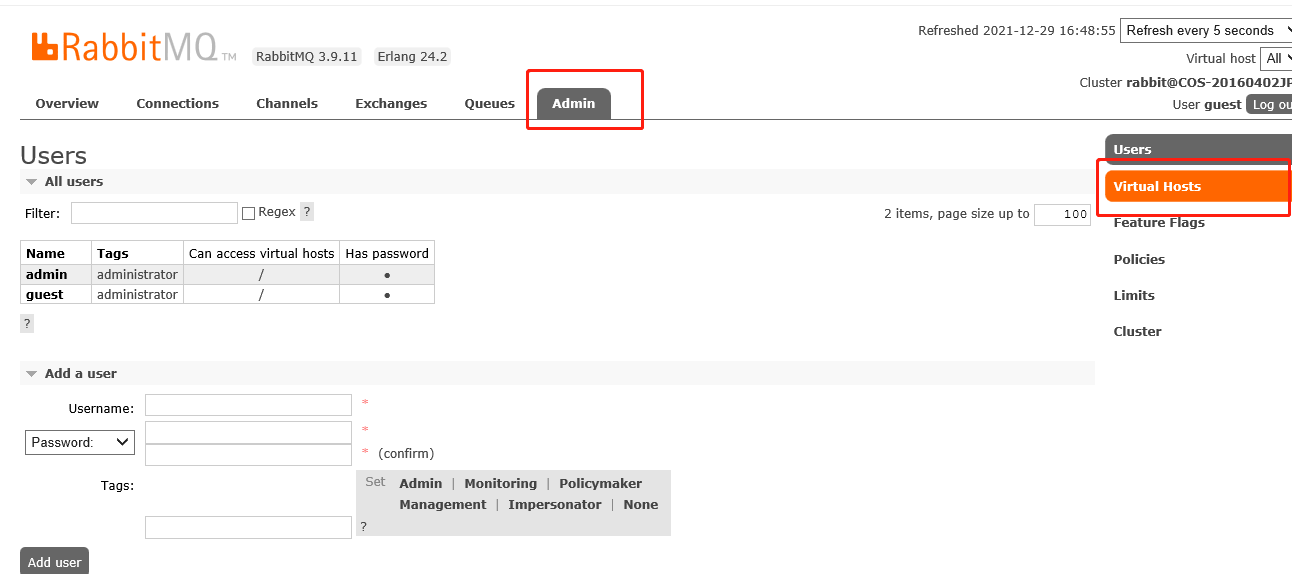
点击“Set permission”按钮，为该用户赋予访问所有虚拟主机的权限，即可在任意网络连通的设备上使用服务器的IP地址访问：

至此，RabbitMQ服务器搭建完毕，可以使用RabbitMQ客户端进行通讯，通讯端口默认为5672。

## 添加VirtualHost

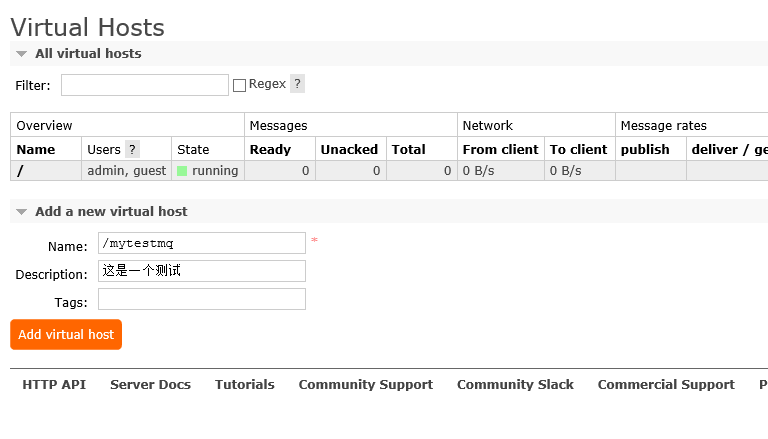
像mysql拥有数据库的概念并且可以指定用户对库和表等操作的权限。那RabbitMQ呢？RabbitMQ也有类似的权限管理。在RabbitMQ中可以虚拟消息服务器VirtualHost，每个VirtualHost相当月一个相对独立的RabbitMQ服务器，每个VirtualHost之间是相互隔离的。exchange、queue、message不能互通。 相当于mysql的db。Virtual Name一般以/开头

进入管理页面

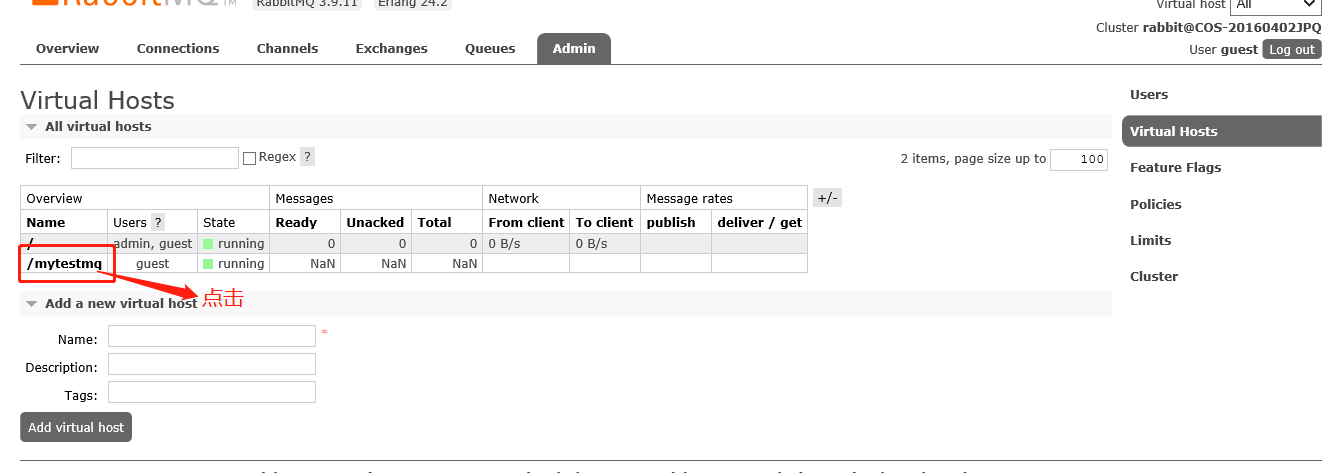


点击右侧的VirtualHost

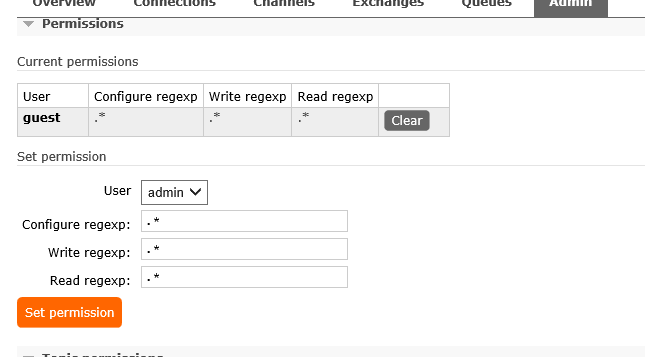
填写Name：/mytestmq（根据自己具体情况定义）



 对用户进行授权，点击需要授权的vhosts



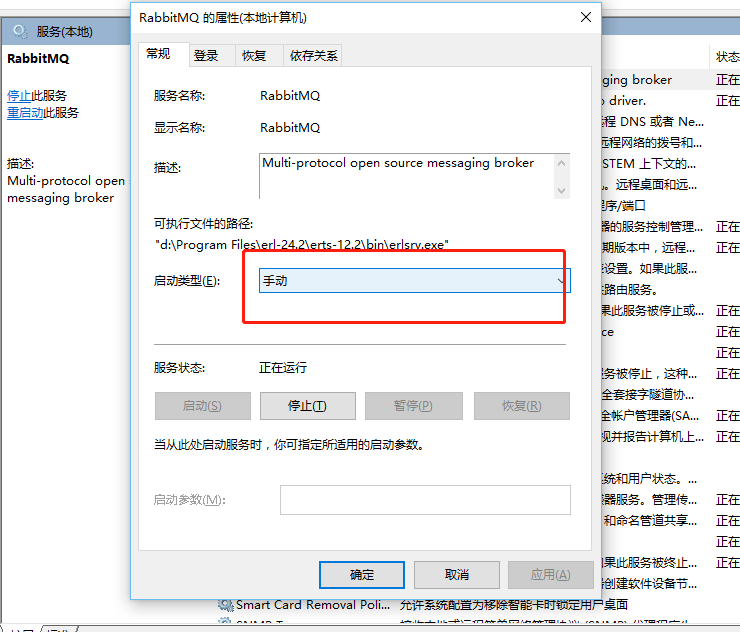
点击“Set permission”按钮，选择用户（自己创建的用户） admin

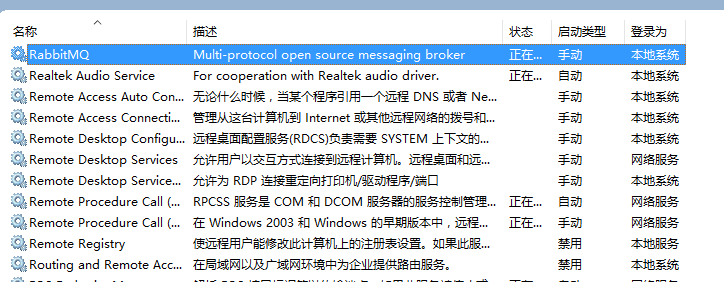


## Windows下禁用服务

在服务中，找到RabbitMQ服务，

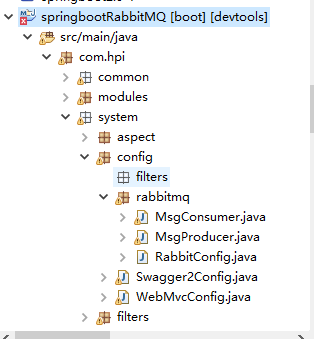






# 项目例子一

**对应工程为：springbootRabbitMQ，把生产者和消费者合并在一起实例**



## pom.xml中增加依赖包

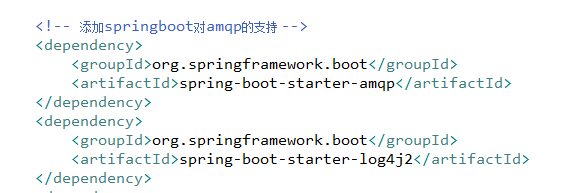
<!-- 添加springboot对amqp的支持 -->

<dependency>

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

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

</dependency>



## 在application.yml中增加rabbit服务端信息：

# rabbitmq 配置

spring.rabbitmq.host=127.0.0.1

spring.rabbitmq.port=5672

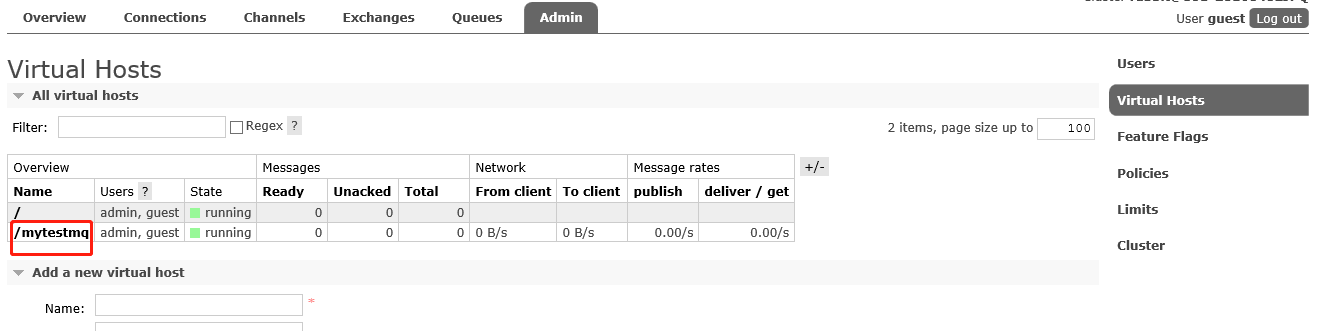
spring.rabbitmq.username=admin

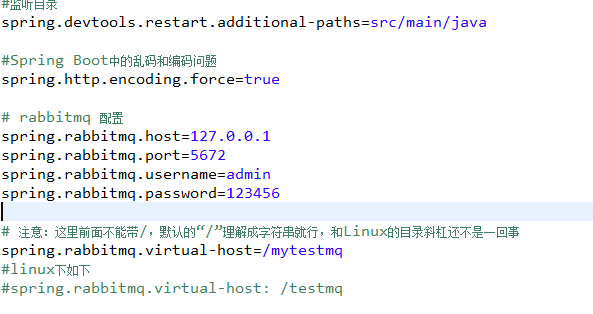
spring.rabbitmq.password=123456

# 注意：这里前面不能带/，默认的“/”理解成字符串就行，和Linux的目录斜杠还不是一回事

spring.rabbitmq.virtual-host=/mytestmq

其中spring.rabbitmq.virtual-host为自己添加的virtual-host





## 增加配置类：

增加rabbit的配置类，定义所需要绑定的queue，exchange，routingkey，binding等，我们

**import** org.springframework.amqp.core.\*;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

@Configuration

**public** **class** RabbitConfig

{

//队列名

**public** **static** **final** String ***FANOUT\_QUEUE\_NAME*** = "test\_fanout\_queue";

**public** **static** **final** String ***FANOUT\_QUEUE\_NAME1*** = "test\_fanout\_queue1";

**public** **static** **final** String ***TEST\_FANOUT\_EXCHANGE*** = "testFanoutExchange";

**public** **static** **final** String ***DIRECT\_QUEUE\_NAME*** = "test\_direct\_queue";

**public** **static** **final** String ***TEST\_DIRECT\_EXCHANGE*** = "testDirectExchange";

**public** **static** **final** String ***DIRECT\_ROUTINGKEY*** = "test";

**public** **static** **final** String ***TOPIC\_QUEUE\_NAME*** = "test\_topic\_queue";

**public** **static** **final** String ***TEST\_TOPIC\_EXCHANGE*** = "testTopicExchange";

**public** **static** **final** String ***TOPIC\_ROUTINGKEY*** = "test.\*";

//队列 起名：TestDirectQueue,我自己创建的

@Bean

**public** Queue TestDirectQueue() {

**return** **new** Queue("TestDirectQueue",**true**);

}

//创建队列

@Bean

**public** Queue createFanoutQueue() {

**return** **new** Queue(***FANOUT\_QUEUE\_NAME***);

}

//创建队列

@Bean

**public** Queue createFanoutQueue1() {

**return** **new** Queue(***FANOUT\_QUEUE\_NAME1***);

}

//创建队列

@Bean

**public** Queue createDirectQueue() {

**return** **new** Queue(***DIRECT\_QUEUE\_NAME***);

}

//创建队列

@Bean

**public** Queue createTopicQueue() {

**return** **new** Queue(***TOPIC\_QUEUE\_NAME***);

}

//创建交换机

@Bean

**public** FanoutExchange defFanoutExchange() {

**return** **new** FanoutExchange(***TEST\_FANOUT\_EXCHANGE***);

}

//队列与交换机进行绑定

@Bean

Binding bindingFanout() {

**return** BindingBuilder.*bind*(createFanoutQueue()).

to(defFanoutExchange());

}

//队列与交换机进行绑定

@Bean

Binding bindingFanout1() {

**return** BindingBuilder.*bind*(createFanoutQueue1()).

to(defFanoutExchange());

}

@Bean

DirectExchange directExchange(){

**return** **new** DirectExchange(***TEST\_DIRECT\_EXCHANGE***);

}

//绑定 将队列和交换机绑定, 并设置用于匹配键：TestDirectRouting 我自己创建的，公用directExchange交换机

@Bean

Binding bindingDirect() {

**return** BindingBuilder.*bind*(TestDirectQueue()).to(directExchange()).with("TestDirectRouting");

}

@Bean

Binding bindingDirect1() {

**return** BindingBuilder.*bind*(createDirectQueue()).

to(directExchange()).

with(***DIRECT\_ROUTINGKEY***);

}

@Bean

TopicExchange defTopicExchange(){

**return** **new** TopicExchange(***TEST\_TOPIC\_EXCHANGE***);

}

@Bean

Binding bindingTopic() {

**return** BindingBuilder.*bind*(createTopicQueue()).

to(defTopicExchange()).

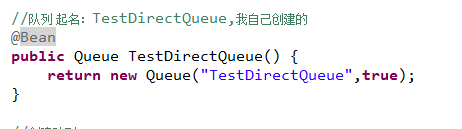
with(***TOPIC\_ROUTINGKEY***);

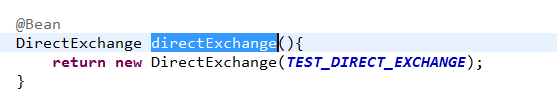
}

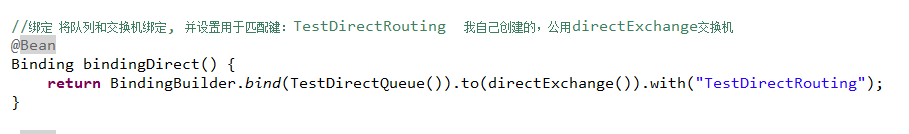
}

此处定义了三种类型的queue，exchange，和routingkey，预先定义，并且将队列，绑定到指定Exchange上，定义其routingKey。并使用@Bean注解定义为实体

可以创建多个绑定，我创建的一个是







## 创建生产者和消费者：

新建MsgProducer和MsgConsumer类，定义消费者和生产者，用于消费队列和发送消息

消费者：

package com.hpi.system.config.rabbitmq;

import lombok.extern.slf4j.Slf4j;

import org.springframework.amqp.core.Message;

import org.springframework.amqp.rabbit.annotation.\*;

import org.springframework.stereotype.Component;

import java.nio.charset.StandardCharsets;

import java.util.Map;

@Component

@Slf4j

public class MsgConsumer

{

@RabbitListener(

bindings =

{

@QueueBinding(value = @Queue(value = RabbitConfig.FANOUT\_QUEUE\_NAME, durable = "true"),

exchange = @Exchange(value = RabbitConfig.TEST\_FANOUT\_EXCHANGE, type = "fanout"))

})

@RabbitHandler

public void processFanoutMsg(Message massage) {

String msg = new String(massage.getBody(), StandardCharsets.UTF\_8);

log.info("received Fanout message : " + msg);

}

@RabbitListener(

bindings =

{

@QueueBinding(value = @Queue(value = RabbitConfig.FANOUT\_QUEUE\_NAME1, durable = "true"),

exchange = @Exchange(value = RabbitConfig.TEST\_FANOUT\_EXCHANGE, type = "fanout"))

})

@RabbitHandler

public void processFanout1Msg(Message massage) {

String msg = new String(massage.getBody(), StandardCharsets.UTF\_8);

log.info("received Fanout1 message : " + msg);

}

@RabbitListener(

bindings =

{

@QueueBinding(value = @Queue(value = RabbitConfig.DIRECT\_QUEUE\_NAME, durable = "true"),

exchange = @Exchange(value = RabbitConfig.TEST\_DIRECT\_EXCHANGE),

key = RabbitConfig.DIRECT\_ROUTINGKEY)

})

@RabbitHandler

public void processDirectMsg(Message massage) {

String msg = new String(massage.getBody(), StandardCharsets.UTF\_8);

log.info("received Direct message : " + msg);

}

@RabbitListener(

bindings =

{

@QueueBinding(value = @Queue(value = RabbitConfig.TOPIC\_QUEUE\_NAME, durable = "true"),

exchange = @Exchange(value = RabbitConfig.TEST\_TOPIC\_EXCHANGE, type = "topic"),

key = RabbitConfig.TOPIC\_ROUTINGKEY)

})

@RabbitHandler

public void processTopicMsg(Message massage) {

String msg = new String(massage.getBody(), StandardCharsets.UTF\_8);

log.info("received Topic message : " + msg);

}

/\*\*

\*

\* @方法说明: 我自己创建的TestDirectQueue监听

\* @参数： @param testMessage

\* @返回值： void

\* @异常：

\* @作者： duhj

\* @创建日期 2021年12月30日

\*

\* 历史记录

\* 1、修改日期：

\* 修改人：

\* 修改内容：

\*/

@RabbitListener(

bindings =

{

@QueueBinding(value = @Queue(value = "TestDirectQueue", durable = "true"),

exchange = @Exchange(value = RabbitConfig.TEST\_DIRECT\_EXCHANGE))

})

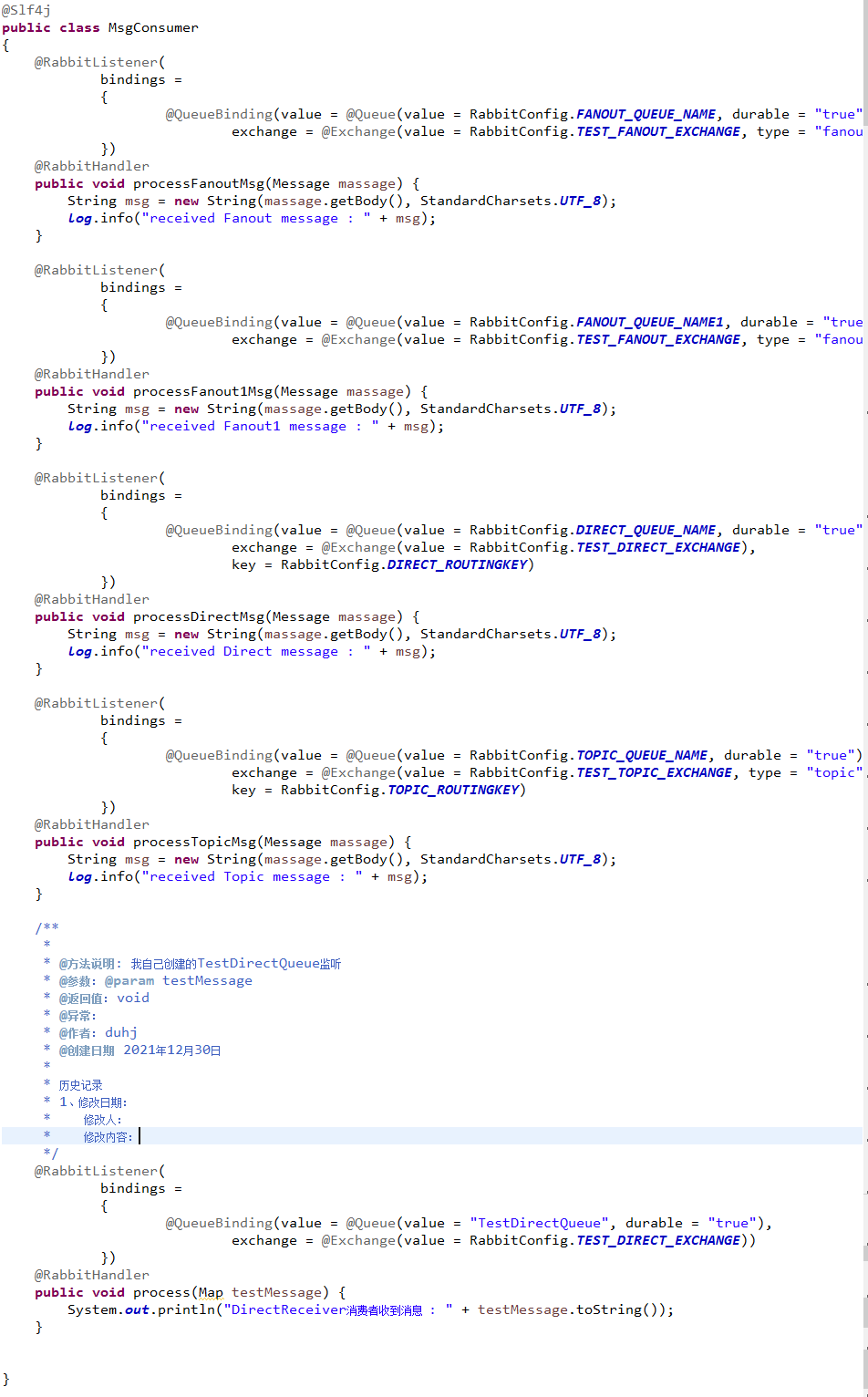
@RabbitHandler

public void process(Map testMessage) {

System.out.println("DirectReceiver消费者收到消息 : " + testMessage.toString());

}

}



创建生产者：

@Component

**public** **class** MsgProducer

{

@Autowired

**private** RabbitTemplate rabbitTemplate;

**public** **void** send2FanoutTestQueue(String massage){

rabbitTemplate.convertAndSend(RabbitConfig.***TEST\_FANOUT\_EXCHANGE***,

"", massage);

}

**public** **void** send2DirectTestQueue(String massage){

rabbitTemplate.convertAndSend(RabbitConfig.***TEST\_DIRECT\_EXCHANGE***,

RabbitConfig.***DIRECT\_ROUTINGKEY***, massage);

}

**public** **void** send2TopicTestAQueue(String massage){

rabbitTemplate.convertAndSend(RabbitConfig.***TEST\_TOPIC\_EXCHANGE***,

"test.aaa", massage);

}

**public** **void** send2TopicTestBQueue(String massage){

rabbitTemplate.convertAndSend(RabbitConfig.***TEST\_TOPIC\_EXCHANGE***,

"test.bbb", massage);

}

/\*\*

\*

\* **@方法说明**: **TODO**

\* **@参数**： **@param** map

\* **@返回值**： void

\* **@异常**：

\* **@作者**： duhj

\* **@创建日期** 2021年12月30日

\*

\* 历史记录

\* 1、修改日期：

\* 修改人：

\* 修改内容：

\*/

**public** **void** send2ExchangeQueue(Map<String,Object> map){

//将消息携带绑定键值：TestDirectRouting 发送到交换机TestDirectExchange

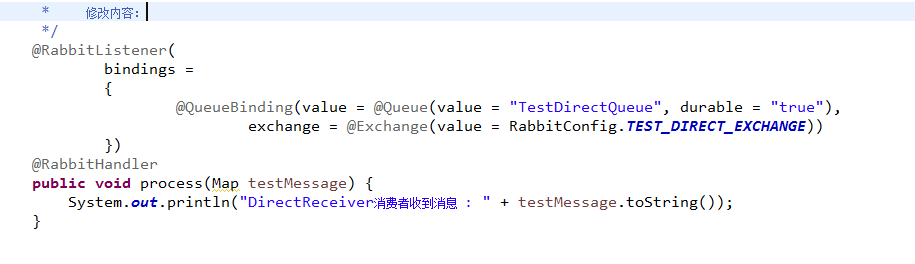
rabbitTemplate.convertAndSend("TestDirectExchange", "TestDirectRouting", map);

}

}



消费者类中通过@RabbitListener和@RabbitHandler注解将一个方法定义为消息监听的方法，使用方法如下



## 发送消息测试：

定义controller，在controller下新建RabbitMQController类，发送消息，定义几个接口

@RequestMapping("/mq/sms")

@Controller

@Slf4j

**public** **class** RabbitMQController

{

@Autowired

**private** MsgProducer msgProducer;

@GetMapping(value = "/sendFanout")

@ResponseBody

@Transactional(rollbackFor = Exception.**class**)

**public** **void** sendMsg(){

msgProducer.send2FanoutTestQueue("this is a test fanout message!");

}

@GetMapping(value = "/sendDirect")

@ResponseBody

@Transactional(rollbackFor = Exception.**class**)

**public** **void** sendDirectMsg(){

msgProducer.send2DirectTestQueue("this is a test direct message!");

}

@GetMapping(value = "/sendDirectA")

@ResponseBody

@Transactional(rollbackFor = Exception.**class**)

**public** **void** sendTopicAMsg(){

msgProducer.send2TopicTestAQueue("this is a test topic aaa message!");

}

@GetMapping(value = "/sendTopicB")

@ResponseBody

@Transactional(rollbackFor = Exception.**class**)

**public** **void** sendTopicBMsg(){

msgProducer.send2TopicTestBQueue("this is a test topic bbb message!");

}

@GetMapping("/sendDirectMessage")

@ResponseBody

@Transactional(rollbackFor = Exception.**class**)

**public** String sendDirectMessage() {

String messageId = String.*valueOf*(UUID.*randomUUID*());

String messageData = "test message, hello!";

String createTime = LocalDateTime.*now*().format(DateTimeFormatter.*ofPattern*("yyyy-MM-dd HH:mm:ss"));

Map<String,Object> map=**new** HashMap<>();

map.put("messageId",messageId);

map.put("messageData",messageData);

map.put("createTime",createTime);

//将消息携带绑定键值：TestDirectRouting 发送到交换机TestDirectExchange

msgProducer.send2ExchangeQueue(map);

**return** "ok";

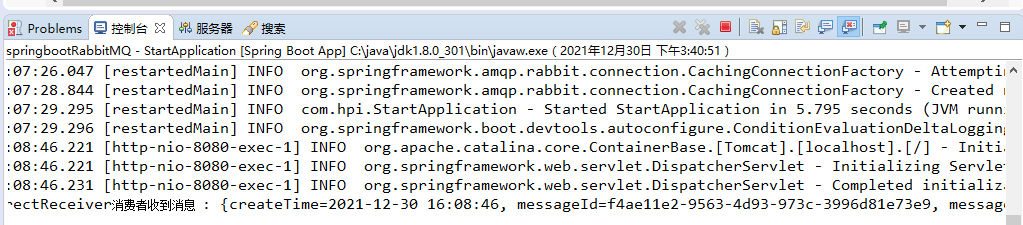
}

}

启动服务，测试

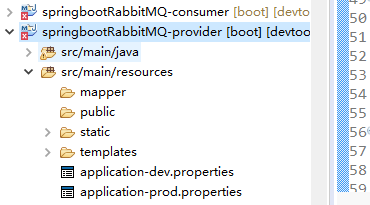
<http://localhost:8080/mq/sms/sendDirectMessage>

监听检测到消息



# 项目例子二

生产者和消费者分开，明了易懂，项目是springbootRabbitMQ-consumer和springbootRabbitMQ-provider，springbootRabbitMQ-consumer消费者，springbootRabbitMQ-provider生产者



## 创建生产者（provider）

### 1.1pom.xml中增加依赖包

<!-- 添加springboot对amqp的支持 -->

<dependency>

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

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

</dependency>

### 1.2配置文件

# rabbitmq 配置

spring.rabbitmq.host=127.0.0.1

spring.rabbitmq.port=5672

spring.rabbitmq.username=admin

spring.rabbitmq.password=123456

# 注意：这里前面不能带/，默认的“/”理解成字符串就行，和Linux的目录斜杠还不是一回事

spring.rabbitmq.virtual-host=/mytestmq

### 1.3创建配置类

@Configuration

**public** **class** DirectRabbitConfig

{

//队列 起名：TestDirectQueue

@Bean

**public** Queue TestDirectQueue() {

// durable:是否持久化,默认是false,持久化队列：会被存储在磁盘上，当消息代理重启时仍然存在，暂存队列：当前连接有效

// exclusive:默认也是false，只能被当前创建的连接使用，而且当连接关闭后队列即被删除。此参考优先级高于durable

// autoDelete:是否自动删除，当没有生产者或者消费者使用此队列，该队列会自动删除。

// return new Queue("TestDirectQueue",true,true,false);

//一般设置一下队列的持久化就好,其余两个就是默认false

**return** **new** Queue("TestDirectQueue",**true**);

}

//Direct交换机 起名：TestDirectExchange

@Bean

DirectExchange TestDirectExchange() {

// return new DirectExchange("TestDirectExchange",true,true);

**return** **new** DirectExchange("TestDirectExchange",**true**,**false**);

}

//绑定 将队列和交换机绑定, 并设置用于匹配键：TestDirectRouting

@Bean

Binding bindingDirect() {

**return** BindingBuilder.*bind*(TestDirectQueue()).to(TestDirectExchange()).with("TestDirectRouting");

}

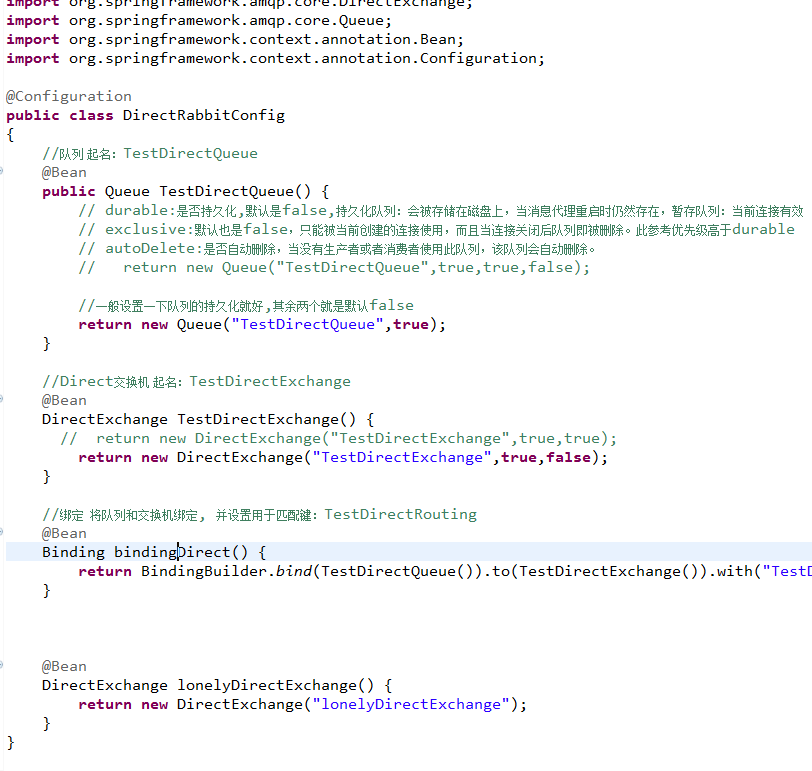
@Bean

DirectExchange lonelyDirectExchange() {

**return** **new** DirectExchange("lonelyDirectExchange");

}

}



### 1.4发送消息

创建一个Controller，用来在前台调用发送消息

@Slf4j

@RestController

@RequestMapping("/mq/sms")

**public** **class** SendMessageController

{

@Autowired

RabbitTemplate rabbitTemplate; //使用RabbitTemplate,这提供了接收/发送等等方法

@GetMapping("/sendDirectMessage")

**public** String sendDirectMessage() {

String messageId = String.*valueOf*(UUID.*randomUUID*());

String messageData = "test message, hello!";

String createTime = LocalDateTime.*now*().format(DateTimeFormatter.*ofPattern*("yyyy-MM-dd HH:mm:ss"));

Map<String,Object> map=**new** HashMap<>();

map.put("messageId",messageId);

map.put("messageData",messageData);

map.put("createTime",createTime);

//将消息携带绑定键值：TestDirectRouting 发送到交换机TestDirectExchange

rabbitTemplate.convertAndSend("TestDirectExchange", "TestDirectRouting", map);

**return** "ok";

}

}

### 1.5启动服务发送消息

启动服务 后，输入如下地址：

<http://localhost:8081/mq/sms/sendDirectMessage>

在管理页面看到有一条等待信息



## 创建消费者（consumer）

### 1.1pom.xml中增加依赖包

和生产者一样

### 1.2配置文件

和生产者一样

### 1.3创建配置类

基本上和生产者一样

@Configuration

**public** **class** DirectRabbitConfig

{

//队列 起名：TestDirectQueue

@Bean

**public** Queue TestDirectQueue() {

**return** **new** Queue("TestDirectQueue",**true**);

}

//Direct交换机 起名：TestDirectExchange

@Bean

DirectExchange TestDirectExchange() {

**return** **new** DirectExchange("TestDirectExchange");

}

//绑定 将队列和交换机绑定, 并设置用于匹配键：TestDirectRouting

@Bean

Binding bindingDirect() {

**return** BindingBuilder.*bind*(TestDirectQueue()).to(TestDirectExchange()).with("TestDirectRouting");

}

}



### 1.3创建消息监听类

创建消息接收监听类，DirectReceiver.java：

@Component

@RabbitListener(queues = "TestDirectQueue")//监听的队列名称 TestDirectQueue

**public** **class** DirectReceiver

{

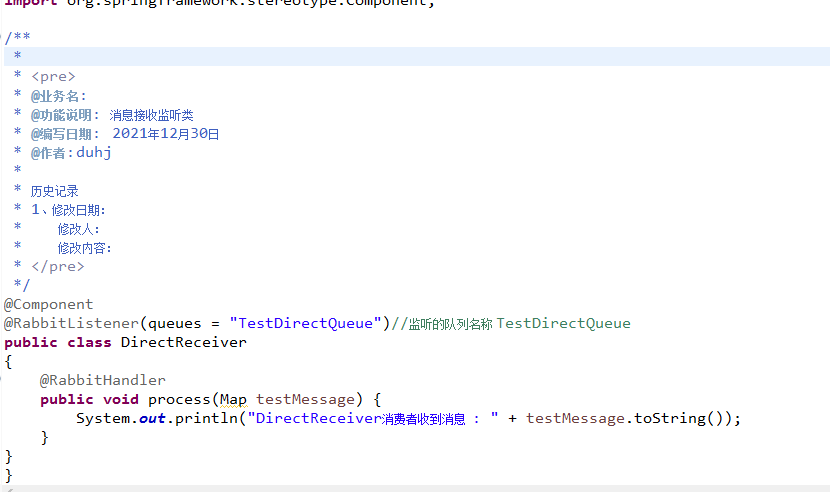
@RabbitHandler

**public** **void** process(Map testMessage) {

System.***out***.println("DirectReceiver消费者收到消息 : " + testMessage.toString());

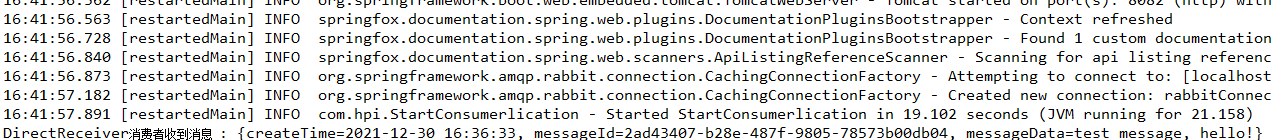
}

}



### 1.3启动消费者

启动后，后台会监听到消息



同时,管理页面等待的已经为0了

