springCloud笔记

1、天气api

<http://wthrcdn.etouch.cn/weather_mini?citykey=101280601>

<http://wthrcdn.etouch.cn/weather_mini?city>=深圳

http://mobile.weather.com.cn/js/citylist.xml

2、springboot redis配置

springboot使用spring-boot-starter-data-redis来自动配置lettuce和jedis这两种客户端.默认选用的是lettuce.

由于springboot版本不一样,低版本和高版本的配置会有所不同：

低版本的配置：

1. # REDIS (RedisProperties)
2. # Redis数据库索引（默认为0）
3. spring.redis.database=0
4. # Redis服务器地址
5. spring.redis.host=localhost
6. # Redis服务器连接端口
7. spring.redis.port=6379
8. # Redis服务器连接密码（默认为空）
9. spring.redis.password=
10. # 连接池最大连接数（使用负值表示没有限制）
11. spring.redis.pool.max-active=8
12. # 连接池最大阻塞等待时间（使用负值表示没有限制）
13. spring.redis.pool.max-wait=-1
14. # 连接池中的最大空闲连接
15. spring.redis.pool.max-idle=8
16. # 连接池中的最小空闲连接
17. spring.redis.pool.min-idle=0
18. # 连接超时时间（毫秒）
19. spring.redis.timeout=0

高版本的配置：

1. # REDIS (RedisProperties)
2. # Redis数据库索引（默认为0）
3. spring.redis.database=0
4. # Redis服务器地址
5. spring.redis.host=localhost
6. # Redis服务器连接端口
7. spring.redis.port=6379
8. # Redis服务器连接密码（默认为空）
9. spring.redis.password=
10. # 连接池最大连接数（使用负值表示没有限制）
11. spring.redis.jedis.pool.max-active=8
12. # 连接池最大阻塞等待时间（使用负值表示没有限制）
13. spring.redis.jedis.pool.max-wait=-1ms
14. # 连接池中的最大空闲连接
15. spring.redis.jedis.pool.max-idle=8
16. # 连接池中的最小空闲连接
17. spring.redis.jedis.pool.min-idle=0
18. # 连接超时时间（毫秒）
19. spring.redis.timeout=0ms
20. #############redis哨兵设置##############
21. #redis服务器master的名字
22. spring.redis.sentinel.master=master1
23. #redis-sentinel的配置地址和端口
24. spring.redis.sentinel.nodes=127.0.0.1:6381,127.0.0.1:6382

注意的是,timeout和max-wait如果不在后面加上ms就会报错

lettuce专有的配置

1. #连接池最大连接数,负数表示不限制
2. spring.redis.lettuce.pool.max-active=8
3. #连接池最大空闲数,负数表示不限制
4. spring.redis.lettuce.pool.max-idle=8
5. #等待可用连接的最大时间,负数表示不限制
6. spring.redis.lettuce.pool.max-wait=-1ms
7. #连接池最小空闲数
8. spring.redis.lettuce.pool.min-idle=0
9. #连接超时时间
10. spring.redis.lettuce.shutdown-timeout=1000ms

jedis专有配置(和lettuce类似)：

1. spring.redis.jedis.pool.max-active=8
2. spring.redis.jedis.pool.max-idle=8
3. spring.redis.jedis.pool.max-wait=-1ms
4. spring.redis.jedis.pool.min-idle=0

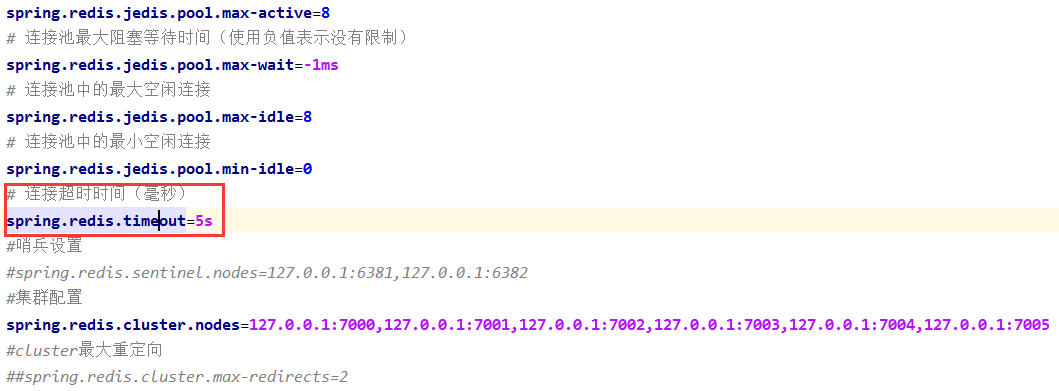
<https://majing.io/posts/10000020931206>

弄了一天的redis，百度springboot redisTemplate都是重写configuration.但是自己重写有错.

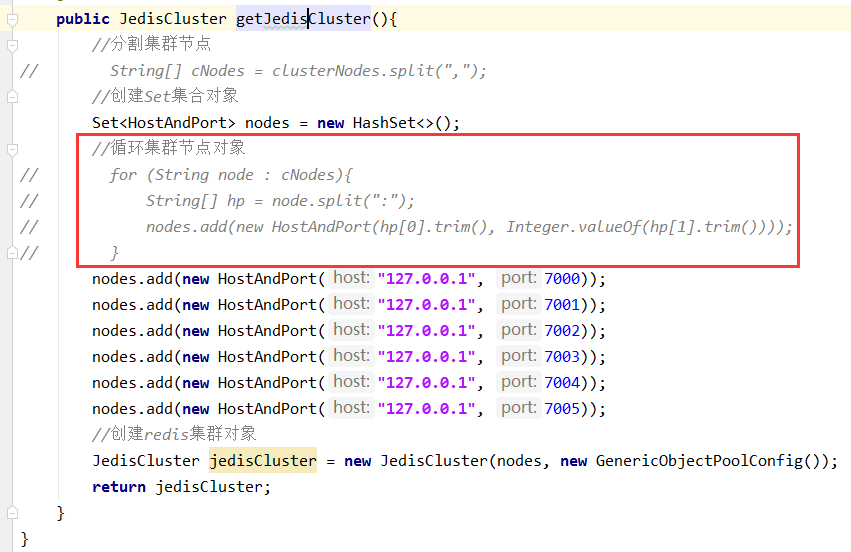
spring-data-redis自2.0后自带lettuce和jedis，默认的是lettuce，像上面那样配置分布式后会报错，但是自己就像用lettuce,因为有优点,但是没成功。

承上：

之前找资料都是使用redisTemplate或者jedis的,由于之前用过了jedis,所以觉得可以搞定.由于redisTemplate配置集群总是不成功,就先放一边了.然后开始用jedis测试,但是测试的时候出问题,在application.properties配置时,会报错java.lang.NumberFormatException: For input string: "5s",因为部分配置如下：



不知道为什么后面加个”s”就报错,不加又不正确(报红).另外,新建springJedisConfig时,如下所示:

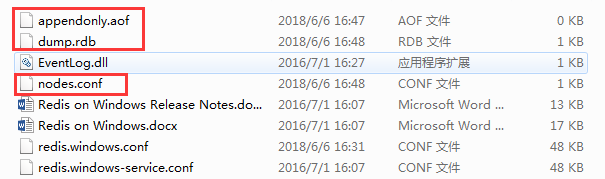


使用红色框里的方法时会报错: java.lang.NumberFormatException: For input string: 具体涉及到add方法里的Integer.parseInt或者Integer.valueof.这里没解决,都说是版本问题,把jedis()于是自己直接配置6个集群,这样虽然在启动时不会报错,但是在访问接口时(<http://localhost:8080/weather/cityName/>武汉),会报错: JedisDataException: ERR This instance has cluster support disabled,百度了下,原来是redis.conf里的配置cluster-enable yes没有打开.然后才明白为什么要jedis.cluster.原来在redis3.0以后配置集群采用的jedis.cluster.之前用的2.x的redis,没有用到cluster,而是采用的shardedredis.正是因为用的3.x的缘故,即使把上一个问题解决了,还是会报错: No reachable node in cluster. 突然想到之前看windows下redis配置时,有个用到了ruby,于是想着是不是3.x的redis安装配置集群和2.x的不一样.然后就百度了windows下怎么配置redis集群：

<https://blog.csdn.net/zsg88/article/details/73715947>

<https://blog.csdn.net/MrXiaGC/article/details/52799081>

第一次搭建集群的时候,不知道为什么第一个和第二个redis总是在写什么东西,强行结束后,再次配置时出现错误: ERR slot 10923 is already busy.百度后才知道是上次配置的时候的配置还保存着,需要删掉redis文件夹下的三个文件



然后重新配置就好了.至此终于把这个redis集群使用jedis的方式弄好了.但是接下来还想用redisTemplate的方式搭建集群.怎么办呢,lets go!

紧接上文:

上文说到使用红色框里的方法时会报一个错误: java.lang.NumberFormatException: For input string: "7005}".当我把每一个端口号打印出来的时候,发现前面的都正常,最后一个7005端口号后面多了一个大括号”}”,百思不得其解,然后换了一种方法:采用三元表达式,若是后面有”}”则截取前面的字符,若没有,保留原来字符:

nodes.add(new HostAndPort(hp[0].trim(), Integer.parseInt(hp[1].endsWith("}")?hp[1].substring(0,hp[1].length()-1):hp[1])));

这样就解决了这个问题.还有上面一个问题就是application.properties里配置的spring.redis.timeout=5s,不加单位就报红,加了单位启动又报错: java.lang.NumberFormatException: For input string 5s,结合刚刚的方法,把后面的单位截取掉后就解决了问题,有毒！

3、quartz scheduler定时

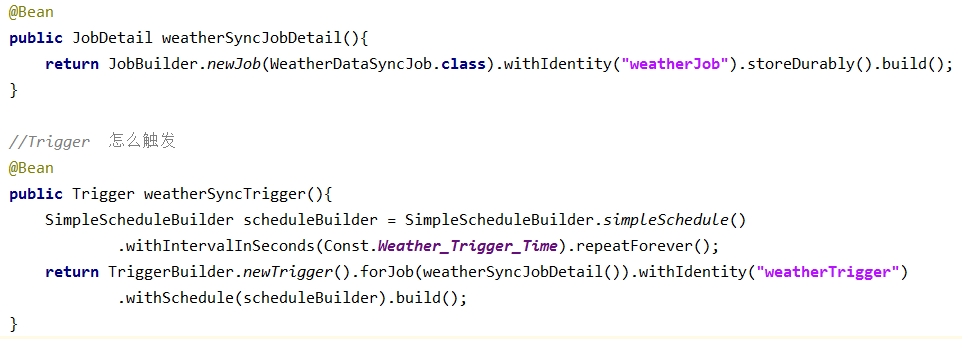
首先引入quartz的依赖：compile(**'org.springframework.boot:spring-boot-starter-quartz'**

quartz中的三个核心概念:

job(任务) scheduler(任务调度器) trigger(触发器)

创建一个任务job,配置相应的job和trigger,通过scheduler来启动任务.

简单示例：



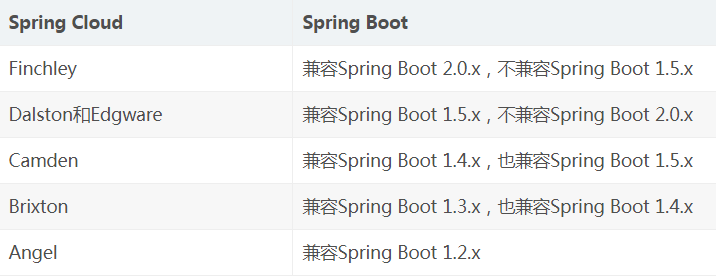
<https://blog.csdn.net/guolong1983811/article/details/51501346>

<https://blog.csdn.net/lkl_csdn/article/details/73613033>

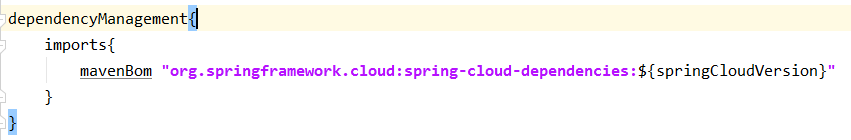
4、eureka 服务注册与发现

springcloud的版本和springboot的版本对应

springcloud版本命名规则是根据伦敦地铁站来命名的



首先添加一个springcloudMangement,用来管理springcloud各个jar之间的依赖

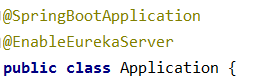


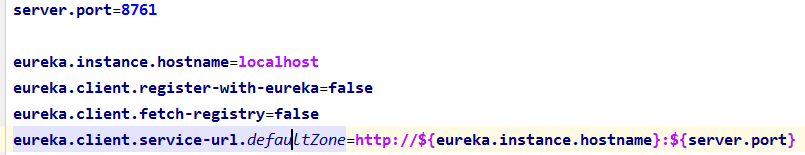
springboot的版本要对应,目前不知道具体的对应情况,这里是springboot:2.0.0M3和Finchley.M2。

添加eureka的依赖,包括服务端和客户端：

服务端的依赖及配置:

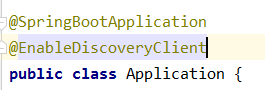
compile('org.springframework.cloud:spring-cloud-starter-netflix-eureka-server')

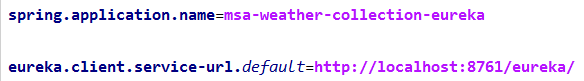




客户端依赖及配置:

compile(**'org.springframework.cloud:spring-cloud-starter-netflix-eureka-client'**)

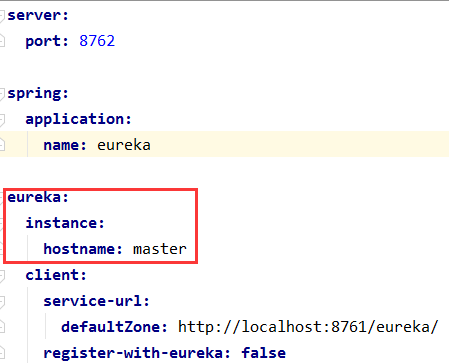
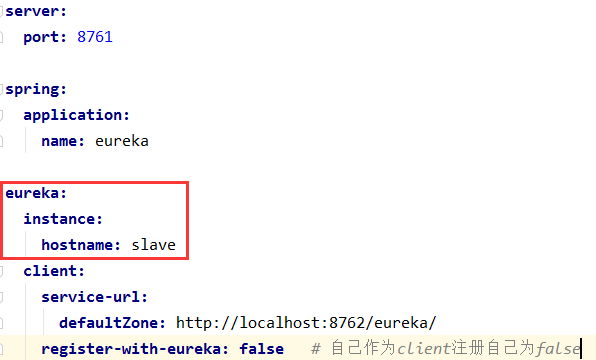


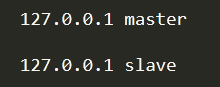


小插曲:使用mvn本地打包时,碰到一个错误：Failure to find io.projectreactor:reactor-bom:pom:Bismuth-M3,解决办法：<https://stackoverflow.com/questions/43783601/failure-to-find-io-projectreactorreactor-bompombismuth-m1>

在配置eureka集群时,碰到问题：eureka-server间总是不能相互注册！

配置时差别就在于application.yml里.以两个eureka-server为例



经过多次试验,spring.application.name可以相同,也可以不同,但是eureka.instance.hostname必须不同,否则无法进行相互注册(windows上如此,linux下还没试),master和slave为hosts文件里的映射,和之前nginx一样,只是一个代理域名. ,defaultZone这里可以随意,其实master和slave都等同于localhost,但是它们的hostname必须区分开.关于application.name最好一样,这样在一个请求到来时,别人不知道访问的是哪个eureka-server(个人觉得…)

必须记:

在看到eureka这里时，用restTemplate实现服务的访问没有问题,但是当用到声明式的客户端feign的时候,一直报错：classNOTfoundException:org/springframework/cloud/client/loadbalancer/LoadBalancedRetryFactory,找了好久也没找到答案,然后看另一个视频看到feign的时候也报了这个错误.哎呀，急死了...,又找了两天,终于看到了一个说版本不对的.我也知道版本是强依赖关系,但是我不知道具体的对应关系,官方文档上只有最新的对应关系...,最开始用的是spring boot 2.0.0.M3和finchely M2,改到版本spring boot 2.0.1.release和finchely RC2,这个错误不报了,其实之前也改到最新的版本了的,报错: Error creating bean with name 'eurekaAutoServiceRegistration': Singleton bean creation not allowed while singletons of this factory are in destruction (Do not request a bean from a BeanFactory in a destroy method implementation!),然后又改回来了,报上面的那个错,就又改回新版本,于是照这个错误的答案:终于找到了一个,说加一个spring-boot-starter-web的依赖就好了,试了下,还真的好了.真的是蛋都碎了,弄了好久.但是还是有一个问题,看之后能不能解决，不知道为什么不能自动注入。但是可以用，很奇怪。<https://github.com/spring-cloud/spring-cloud-netflix/issues/1952>

又报了一个错误: Load balancer does not have available server for client: product,当feign的客户端配置增加了eureka.client.fetch-registry=false时,会出错。去掉就好了

承上：

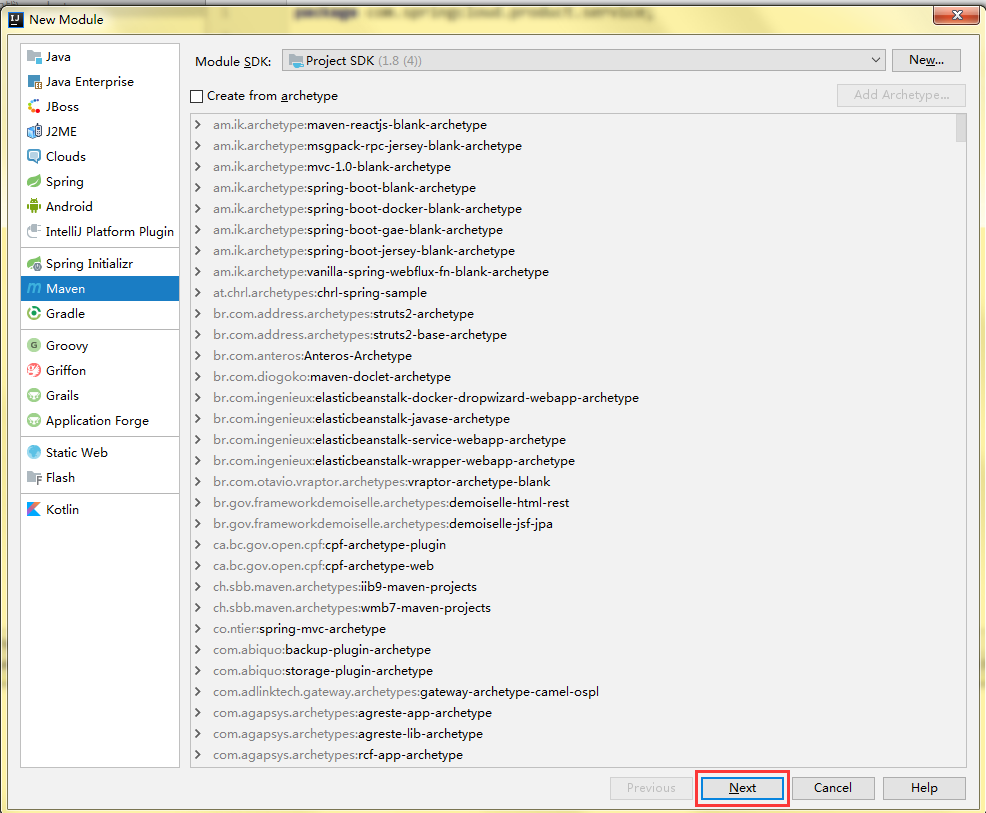
上面说到了不能自动注入的错误,但是不影响使用.百度发现很多说是因为idea的问题,idea比eclipse更严格,所以报错了.ALT+ENTER后,取消相应的检测就行了.目前是这样做的,不知道真正原因是不是这个.

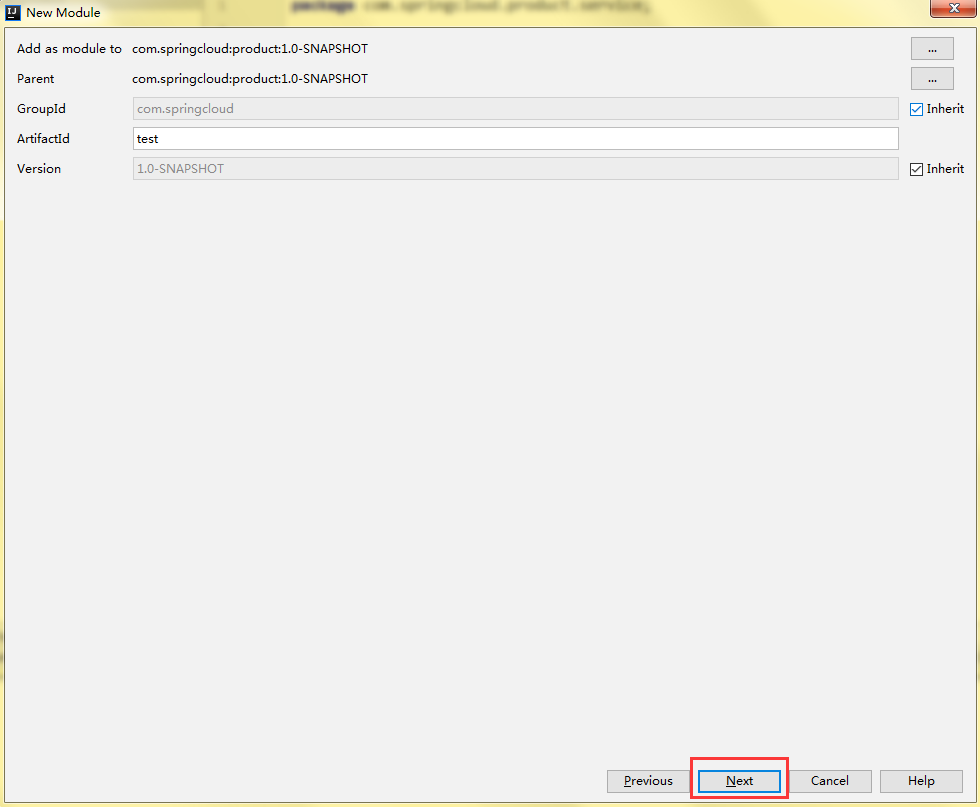
单个项目模块化：

1、idea模块化：

(1) 创建maven项目,不具体截图了,比较简单

(2) 右击新建的项目,new module,选择maven



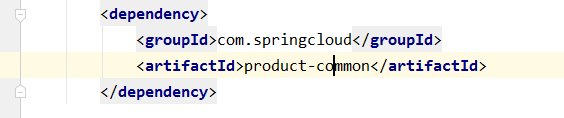


一直next就行了,新建多个module也是如此.当然,在新建module的时候也可以选择spring boot项目,选择maven也可以改造成spring boot项目.module建立完成后,主要工作是修改pom.xml文件。

在主pom.xml中需要有其他module的引入：



在每个单独module中,如果有引用到其他module里的内容,就需要添加相应的依赖：



需要注意的是每个module都需要打包到本地maven,如果是在服务器上,就需要在服务器上打包到服务器上的maven.

在打包的时候如果报错: Failed to execute goal on project product-client: Could not resolve dependencies for project com.springcloud:product-client:jar:1.0-SNAPSHOT: Could not find artifact com.springcloud:common:jar:1.0-SNAPSHOT -> [Help 1],看下是不是pom.xml里引入错了或者可以到每个文件夹下单独打包。打包命令:mvn -Dmaven.test.skip -U clean install.

5、统一配置中心config

1、config-server(服务端)

添加spring.cloud.config.server的依赖

在启动类上加上@EnableConfigServer注解

配置application.yml：



这里的search-paths为github上配置文件所在的目录,如果在根目录,则可以省略。

basedir为本地的git仓库

2、config-client

添加spring.cloud.config.client的客户端

application启动类的注解和其他eureka.client的一样

application.yml的配置要修改:

首先将application.yml改为bootstrap.yml。让程序启动时先加载bootstrap.yml里的配置,然后再去加载其他配置,否则会在程序启动时,同时找配置,当没找到mysql配置时会报错.当使用默认端口(8761)时,不用将eureka的注册配置放在bootstrap.yml中,但是如果改了端口(如8762),那么要将eureka配置放在bootstrap.yml里。因为要先知道在哪里找config服务,否则回报错,找不到一个服务实例。具体配置如下:



当github上有多个配置时,运行时,会将多个配置合并,最终的配置是并集.所以如果有公共配置,可以把公共配置单独拿出来.当配置文件需要减少配置时,需要全部修改。

6、RabbitMQ的基本使用:

首先需要了解rabbitMq的基本概念：

<https://www.jianshu.com/p/79ca08116d57>

<https://blog.csdn.net/whoamiyang/article/details/54954780>

<https://zhuanlan.zhihu.com/p/25069044>

我只说说我在跟着视频做时遇到的问题和解决办法：

引入依赖: spring-cloud-starter-bus-amqp.

对于rabbitmq队列需要有发布者和消费者,发送者只需要调用AqmpTempalte的convertAndSend()方法即可把消息发送到rabbitmq中间件,消费者通过@rabbitlistener监听是否有消息发送过来：



这种方式是rabbitmq的原生方式.spring cloud对rabbitmq进行了封装.

使用spring.cloud.starter.stream.rabbit:

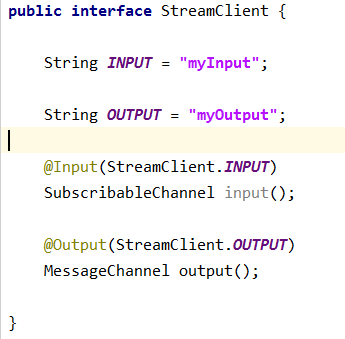
<https://my.oschina.net/u/2342449/blog/1589297>

<https://www.jianshu.com/p/fb7d11c7f798>

<https://blog.csdn.net/jack281706/article/details/73743148>

引入依赖：spring-cloud-starter-stream-rabbit

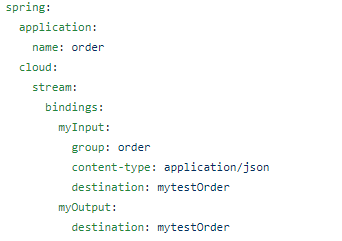
定义接口：



碰到的问题,这里的消息订阅和发布通道的名字不能相同,相同就会报错.开始跟着视频里写的一样,但是报错.

后来看了看知乎上的spring cloud stream.通道连接着binder,binder可以看作是中间件的类的抽象.

只需要我的订阅者和发布者都连接到一个exchange(交换机)或queue(队列),那么一旦有消息发布到中间件上,消费者就可以通过路由键找到消息,并消费掉。而spring cloud stream的exchange就是配置里的destination,所以还需要配置rabbitmq.



这里的group是分组,当有多个相同的实例时,当有消息发布时,我只希望其中一个接收即可,那么只需要把这些的实例的分组设置相同,那么rabbitmq将会把消息随机分发到其中一个实例.