# SpringCloud框架的学习搭建

本项目使用了当下比较流行的微服务架构进行搭建，在了解微服务架构之前，我们要先了解几个概念。

分布式：把一个业务划分为多个子业务，部署到不同的服务器上

集群：把同一个业务部署到不同的服务器上

负载均衡：当一个业务部署到不同服务器时（实现集群），用户请求一个业务处理该访问哪个服务器呢，这就需要负载均衡器实现负载均衡算法，拦截用户请求，通过负载均衡算法将请求转发到一台具体的服务器，服务器再响应请求返回给负载均衡器，负载均衡器再响应给用户。

服务消费者：使用服务的服务。

服务生产者：被使用的服务

断路器：当一个业务在服务器挂掉时，用户请求它会会执行迅速失败的回调方法，而不是等待响应超时，不会造成请求阻塞。

路由：路由功能是微服务的一部分，比如将／api/user转发到user服务。

接下来就是搭建微服务架构的具体步骤了(项目的搭建基于springboot框架)

## 服务注册和发现

项目采用Eureka作为服务注册与发现的组件

1，首先创建一个主Maven工程，在其pom文件引入依赖，这个pom文件作为父pom文件，起到依赖版本控制的作用，本文其他module工程继承该pom。



2，然后创建1个model作为服务注册中心，即eureka-server

eureka-server的pom文件

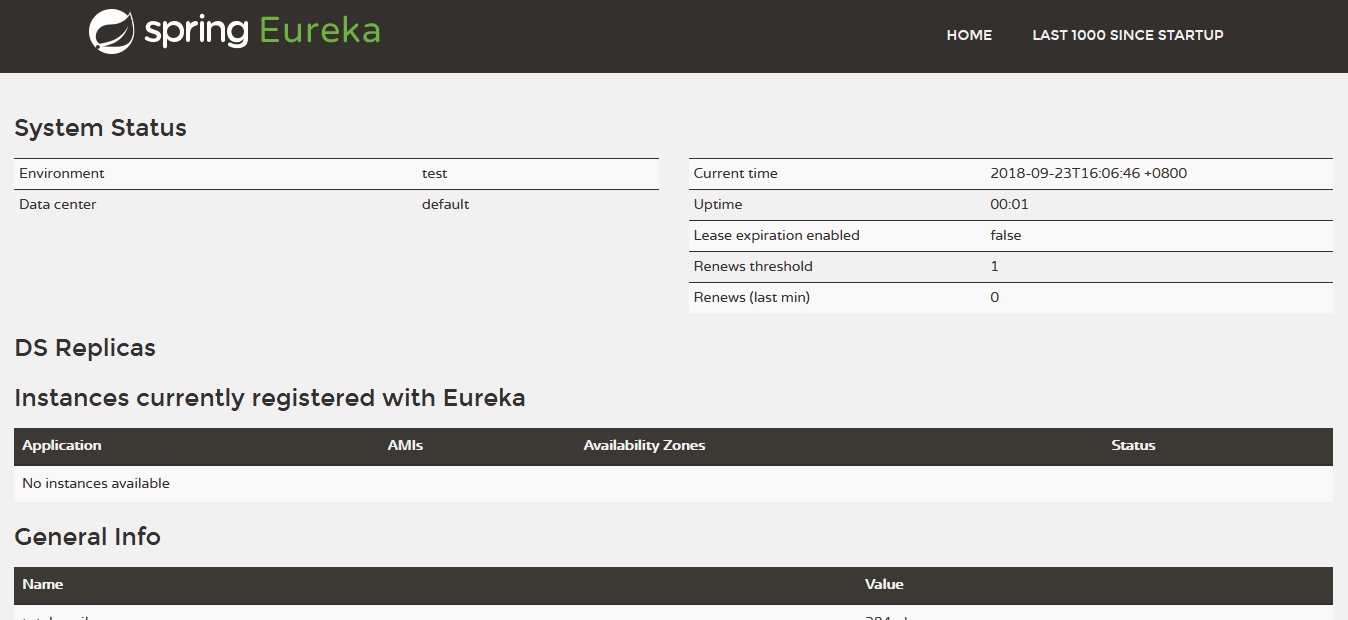


3，在eureka-serve的springboot启动类中添加@EnableEurekaServer注解启动服务注册中心，并增添appication.yml 配置文件



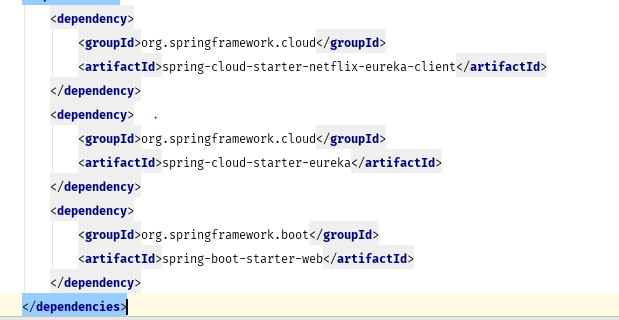
eureka.client.registerWithEureka：false和fetchRegistry：false来表明自己是一个eureka server.

4，启动eureka-server，打开http://localhost:8889，出现如下界面



5，服务注册中心搭建好了之后我们就需要创建一个服务提供者了，然后创建1个model作为服务提供者，即eureka-hi

eureka-hi的pom文件

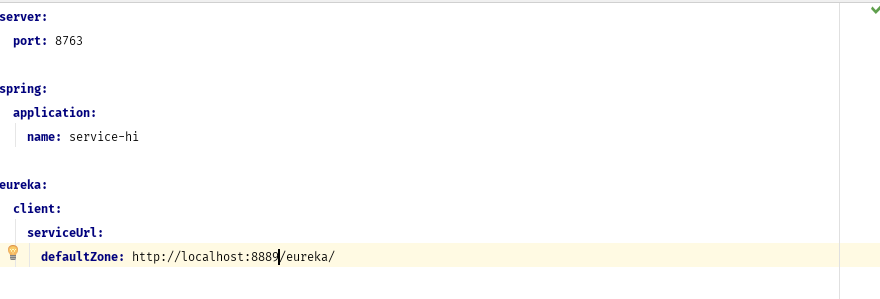


6，在eureka-hi的springboot启动类中添加@EnableEurekaClient 表明自己是一个eurekaclient并增添配置文件

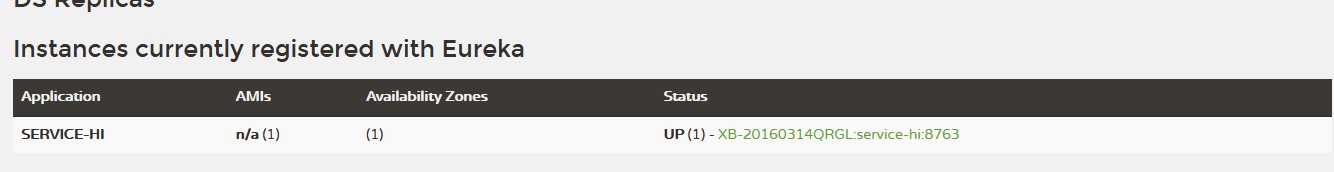
启动类如下



配置文件如下：（需要指明spring.application.name,这个很重要，这在以后的服务与服务之间相互调用一般都是根据这个name。）



启动工程，打开<http://localhost:8889>，即可发现server-hi已经注册在服务注册中心



7，这时打开 http://localhost:8763/hi?name=skycrumbing ，你会在浏览器上看到

hi skycrumbing ,i am from port:8763



## 服务消费者

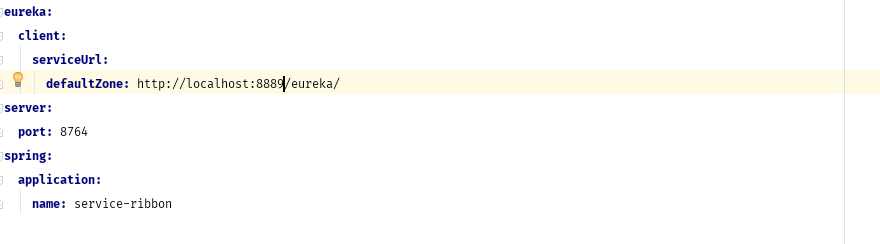
这里要用ribbon组件提供负载均衡的作用，当然也用到了Hystrix组件提供断路器作用。

还可以用 feign实现负载均衡的作用，就不在展示。

1，创建1个model，service-ribbon并引入如下依赖



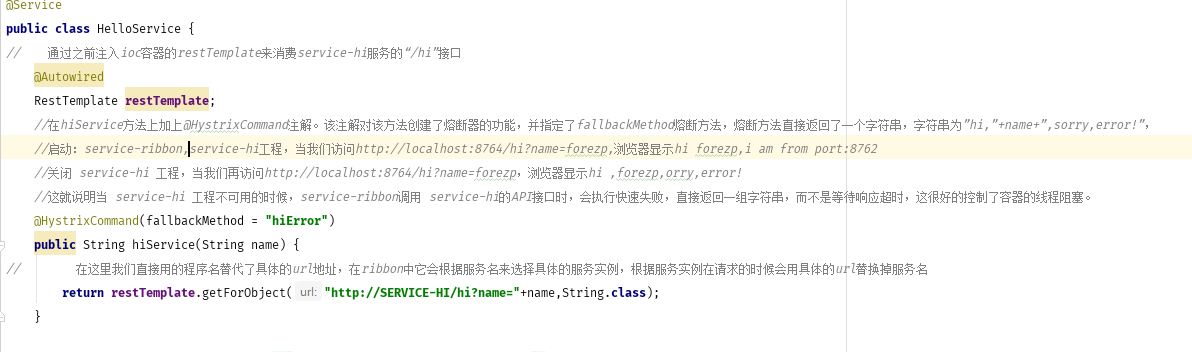
2，配置文件如下：



3，在工程的启动类中,通过@EnableDiscoveryClient向服务中心注册；并且向程序的ioc注入一个bean: restTemplate;并通过@LoadBalanced注解表明这个restRemplate开启负载均衡的功能。



4，写一个测试类HelloService，通过之前注入ioc容器的restTemplate来消费service-hi服务的“/hi”接口，在这里我们直接用的程序名替代了具体的url地址，在ribbon中它会根据服务名来选择具体的服务实例，根据服务实例在请求的时候会用具体的url替换掉服务名，代码如下：



5，写一个controller，在controller中用调用HelloService 的方法，代码如下



6，启动service-hi，eureka-server工程，它的端口为8763；将service-hi的配置文件的端口改为8762,并再次启动，再启动（在启动配置中将single instance only的勾选去掉），这时你会发现：service-hi在eureka-server注册了2个实例，这就相当于一个小的集群。

7，启动server-ribbon工程，访问<http://localhost:8764/hi?name=skycrumbing>

这时会轮流显示

hi skycrumbing ,i am from port:8763

hi skycrumbing ,i am from port:8762

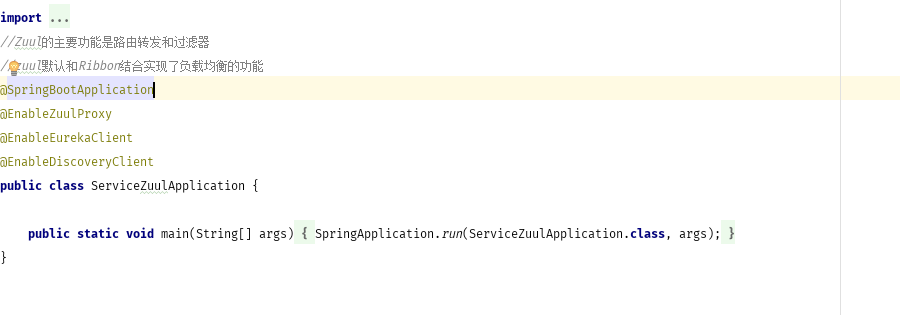
## 路由网关

这里使用了zuul组件完成路由的转发和过滤。

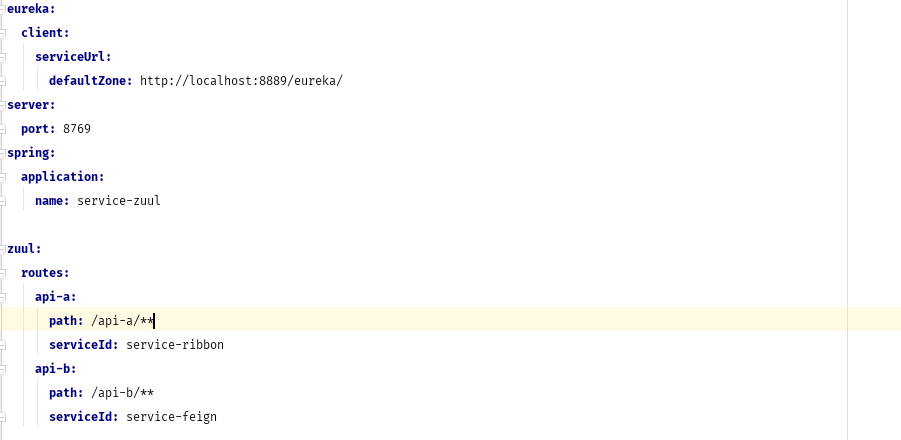
1，创建1个model，service-ribbon并引入如下依赖



2，在其入口applicaton类加上注解@EnableZuulProxy，开启zuul的功能



3，添加配置文件application.yml



4，开启所有相关服务，访问http://localhost:8769/api-a/hi?name=skycrumbing

会轮流显示：

hi skycrumbing ,i am from port:8763

hi skycrumbing ,i am from port:8762

这说明zuul起到了路由的作用。

同时zuul也可以实现过滤的作用，可以做安全验证。

5，增加MyFilter类



6，这时访问：http://localhost:8769/api-a/hi?name=skycrumbing

token is empty

访问：<http://localhost:8769/api-a/hi?name=skycrumbing&&token=22>

会轮流显示：

hi skycrumbing ,i am from port:8763

hi skycrumbing ,i am from port:8762

## 结尾

当然springcloud还涉及到分布式配置，消息总线，服务链追踪，断路器监控等，这些将在后续的学习中继续深入