2018-03-01T09:27 将服务修改一下端口，服务就多了一个，变成两个。

2018-03-01T10:17 同一个服务运行两个。成功运行Ribbon。【在之前的maven工程中创建一个Ribbon的muldel或者重新建一个Ribbon工程（Spring boot工程）都可】

2018-03-01T10:38 RequestParam表示方法参数应该绑定到Web请求参数的注释。



2018-03-01T10:57 开始实验Feign

使用Feign，只需要创建一个接口并注解。

2018-03-01T11:41 成功在Ribbon中运行Hystrix。Hystrix会执行快速失败，直接返回一组字符串，而不是等待响应超时，这控制了容器的线程阻塞。

2018-03-01T11:43 Feign自带断路器

2018-03-01T13:23 在Spring Cloud微服务系统中，一种常见的负载均衡方式是，客户端的请求首先经过负载均衡（Zuul，Ngnix）再到达服务网关（Zuul集群），然后再到具体的服务。服务统一注册到高可用的服务注册中心集群，服务的所有配置文件由配置服务管理，配置服务的配置文件放在git仓库，方便随时更改配置。

Zuul功能：

1. Authentication
2. Insights
3. Stress Testing
4. Canary Testing
5. Service Migration
6. Load Shedding
7. Security
8. Static Response handling
9. Active/Active traffic management

2018-03-01T14:34 filterType：返回一个字符串代表过滤器的类型，在zuul中定义了四种不同生命周期的过滤器类型，具体：

1. pre：路由之前
2. routing：路由之时
3. post：路由之后
4. error：发送错误调用
5. filterOrder：过滤的顺序
6. shouldFilter：这里可以写逻辑判断，是否要过滤，本文为true，永远过滤
7. run：过滤器的具体逻辑。可用很复杂，包括查sql，nosql去判断该请求到底有没有权限访问。

2018-03-01T16:13 http请求地址和资源文件映射如下：

1、/{application}/{profile}[/{label}]

2、/{application}-{profile}.yml

3、/{label}/{application}-{profile}.yml

4、/{application}-{profile}.properties

5、/{label}/{application}-{profile}.properties

2018-03-01T16:21 史上最简单Spring Cloud教程

Spring.cloud.config.label 指明远程仓库的分支

Spring.cloud.config.profile

1. dev开发环境配置文件
2. test测试环境
3. pro正式环境

spring.cloud.config.uri=http://localhost:8888/指明配置服务中心的网址。

2018-03-01T17:09 configClient配置“bootstrap.properties”

1. spring.cloud.config.discovery.enabled 从配置中心读取文件
2. spring.cloud.config.discovery.serviceId 配置中心的serviceId，即服务名

这时发现，在读取配置文件不再写ip地址，而是服务名，这时如果配置服务部署多份，通过负载均衡，从而高可用。