此文档是建立在[服务的注册于发现(Eureka)](onenote:" \l "服务的注册于发现(Eureka)&section-id={B46804C8-2079-4343-A399-0DFB75448724}&page-id={6088FC12-1292-4553-8CEC-7B6F6D0723A2}&end&base-path=https://d.docs.live.net/6599c42f24631913/文档/总结/SpringCloud.one)之上。

在微服务架构中，业务都会被拆分成一个独立的服务，服务与服务的通讯是基于http restful的。Spring Cloud有两种服务调用方式，一种是ribbon+restTemplate，另一种是fegin。本文先介绍ribbon+rest

Ribbon is a client side load balancer which gives you a lot of control over the behaviour of HTTP and TCP clients. Feign already uses Ribbon, so if you are using @FeignClient then this section also applies.

—–摘自官网

ribbon是一个负载均衡客户端，可以很好的控制htt和tcp的一些行为。Feign默认集成了ribbon。

ribbon 已经默认实现了这些配置bean：

* + IClientConfig ribbonClientConfig: DefaultClientConfigImpl
  + IRule ribbonRule: ZoneAvoidanceRule
  + IPing ribbonPing: NoOpPing
  + ServerList ribbonServerList: ConfigurationBasedServerList
  + ServerListFilter ribbonServerListFilter: ZonePreferenceServerListFilter
  + ILoadBalancer ribbonLoadBalancer: ZoneAwareLoadBalancer

**创建ribbon模块**

New Module 
Dependencies 
Core 
web 
Template Engines 
SQL 
NOSQL 
Integration 
Cloud Core 
Cloud Config 
Cloud Discovery 
loud Routing 
Cloud Circuit Breaker 
Cloud Tracing 
Cloud Messaging 
Cloud AWS 
Cloud Contract 
Pivotal Cloud Foundry 
Azure 
spring Cloud GCP 
ops 
spring Boot 
Selected Dependencies 
Cloud Routing 
zuul 
Gateway 
Feign 
Ribbon 
Client side load balancing with spring-cloud-netflix ane 
Ribbon 
A Client Side Load Balancinq with Ribbon and Sprinq 
Previous 
Next 
Ribbon 
Cancel 
Help 

配置文件

**eureka:**

**client:**

**serviceUrl:**

**defaultZone:**http://localhost:8761/eureka/

**server:**

**port:**8764

**spring:**

**application:**

**name:**service-ribbon

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。

Or ingBDotApp1 i cation 
ßEnab1 eDi s coveryC1i entl 
public class RibbonApp1ication { 
public static void main (StringC] args) 
ßBean 
ßLoadBa1anced 
RestTemp1ate restTemp1ate() { 
return new RestTemp1ate() 
{ SpringApp1 ication. run(RibbonApp1 ication. class, 
args) 

ßService 
public class TestService { 
private RestTemp1ate restTemplate: 
publ ic String testService (String name) { 
. "http://SERVlCElJEMØ/demo'?name="+name, String. class) , 
return restTemp1ate. getFor0bject( url• 

注意：

使用ribbon实现负载均衡的时候，服务名称不能用下划线，换成中划线。

service-demo不能换成service\_demo

在浏览器上多次访问http://localhost:8764/test?name=xxxx，浏览器交替显示

hixxxx, i am from 8762

hixxxx, i am from 8763

这说明当我们通过调用restTemplate.getForObject(“[http://SERVICE-DEMO/test?name=](http://service-hi/hi?name=)“+name,String.class)方法时，已经做了负载均衡，访问了不同的端口的服务实例。

此时架构为



* + 一个服务注册中心，eureka server,端口为8761
  + service-demo工程跑了两个实例，端口分别为8762,8763，分别向服务注册中心注册
  + ribbon端口为8764,向服务注册中心注册
  + 当ribbon通过restTemplate调用service-demo的hi接口时，因为用ribbon进行了负载均衡，会轮流的调用service-demo：8762和8763 两个端口的test接口；