# Ribbon引入

使用eureka治理组件，仅仅是将一个集群中管理的信息从之前nginx静态配置upstream变成了进程中内存的动态数据双层map。并不能解决负载均衡调用的问题（服务被调用，服务的相互调用）

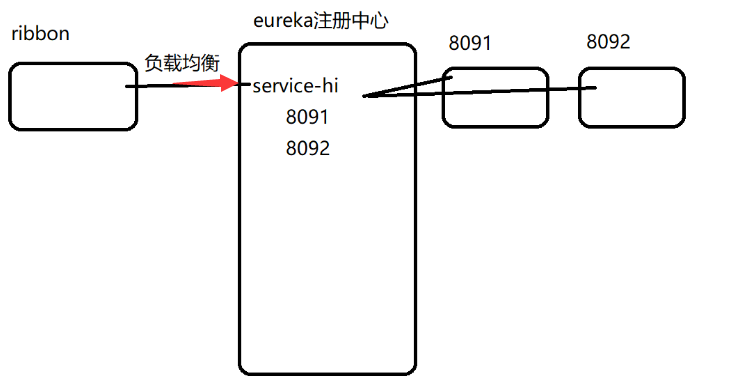
eureka拆分后的项目（微服务）如何相互调用？

Ribbon

# Ribbon简介

负载均衡调用微服务的客户端组件。有了ribbon，就可以通过服务名称负载均衡的访问注册到eureka注册中心的微服务。可以是轮询，也可以是权重，也可以是随机（轮询）

# Ribbon访问结构



Ribbon需要通过“服务名称”调用对应的微服务，微服务信息只能从eureka注册中心中抓取，而只有eureka客户端具有抓取功能，所以ribbon需要配合eureka客户端实现负载均衡调用。

导入依赖时需要导入ribbon依赖，也需要导入eureka客户端依赖。

# 创建Ribbon工程

### pom.xml配置

#### 继承spring-boot

|  |
| --- |
| <!--继承springboot--> <parent>  <artifactId>spring-boot-starter-parent</artifactId>  <groupId>org.springframework.boot</groupId>  <version>1.5.9.RELEASE</version> </parent> |

#### 导入依赖

|  |
| --- |
| <!--eureka-client-->  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-eureka</artifactId>  </dependency>  <!--ribbon-->  <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-ribbon</artifactId>  </dependency> |

### application.properties

|  |
| --- |
| #端口号  server.port=8094  #服务名称  spring.application.name=service-ribbon  #eureka –client  #ip优先  eureka.instance.preferIpAddress=true  #注册中心地址  eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka |

### 启动类

|  |
| --- |
| @SpringBootApplication @EnableEurekaClient public class StarterRibbon {  public static void main(String[] args) {  SpringApplication.*run*(StarterRibbon.class,args);  }  @Bean//注册为bean  @LoadBalanced//ribbon组件会在这个对象上做一个记号，一旦这个对象发起http请求  //经过ribbon负载均衡计算的拦截  ////添加ribbon的拦截逻辑，否则无法实现服务调用  public RestTemplate init(){  return new RestTemplate();  } } |

创建RestTemplate对象添加了一个注解@LoadBalanced,扫描注解之后,所有RestTemplate对象的请求都会经过LoadBalancerIntercptor实现拦截 拦截方法intercept

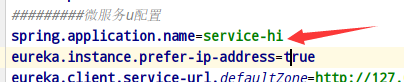
### 创建接口(controller)

|  |
| --- |
| @RestController  public class HelloController {  @Autowired  private HelloService helloService;  //ribbon工程可以直接访问的接口 ribbon/hello  @RequestMapping("ribbon/hello")  public String sayHi(String name){  return "RIBBON:"+helloService.sayHi(name);  }  } |

### 创建service

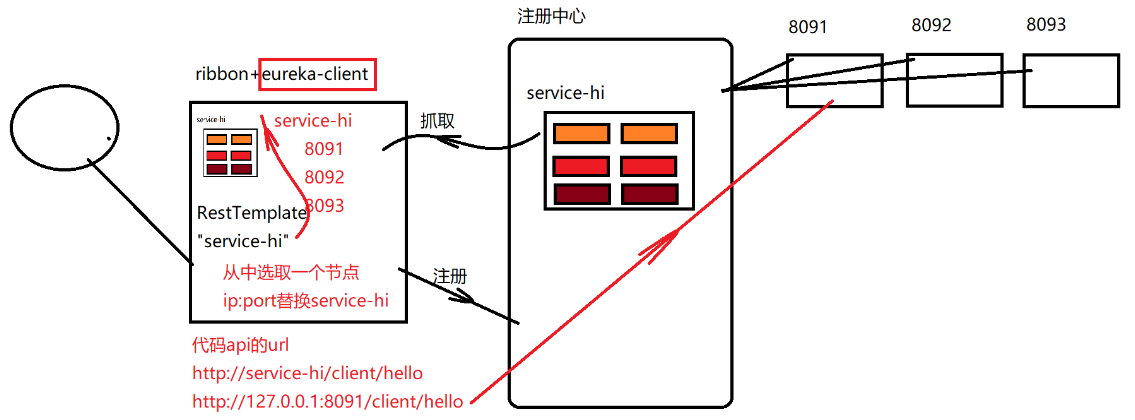
|  |
| --- |
| @Service  public class HelloService {  @Autowired  private RestTemplate template;  public String sayHi(String name) {  //想办法调用8091 8092 8093负载均衡  //通过template对象,发起请求调用service-hi  String url="http://service-hi/client/hello?name="+name;  String responseBody = template.getForObject(url, String.class);  //hello name ,I am from 8091/8092/8093  return responseBody;  }  } |

getForObject的参数解释

* url: 想要用restTemplate的访问的资源url地址
  + service-hi为服务名称
  + 
* responseType: 响应体内容数据类型的反射对象，String.class，会将相应体的内容输出成字符串数据。如果服务器返回的是json，可以直接使用json对应的类型做参数，比如：访问的服务器{"id":1,"name":"刘首付"}，String.class解析相应体类型，就是json字符串本身。自定义一个对应的domain类型

User {int id,String name} User.class

# Ribbon拦截逻辑



ribbon工程启动时，eureka客户端启动时抓取了注册中心的双层map数据

当restTemplate发送请求api调用时,进入到ribbon的拦截逻辑

将url地址中的service-hi解析出来得到服务名称从双层map(第一层 map的key值不就是服务名称),拿到这个服务所有实例提供者

拿到一个list 8091 8092经过负载均衡计算Rule规则得到轮询结果 拿到具体访问节点。

然后将服务名称替换为访问节点的域名、端口拼接url地址,请求发送出去

项目开发中，可以将ribbon组件添加到微服务项目中，实现负载均衡调用其它组件。

虽然微服务之间的调用实现了负载均衡，那么前端调用时如何实现？

引入zuul ( 网关 )组件