**史上最简单的SpringCloud教程 | 第二篇: 服务消费者（rest+ribbon）**

2017年04月08日 23:25:26

阅读数：356749

转载请标明出处：   
<http://blog.csdn.net/forezp/article/details/69788938>   
本文出自[方志朋的博客](http://blog.csdn.net/forezp" \t "_blank)

* 最新版本：
* [史上最简单的SpringCloud教程 | 第二篇: 服务消费者（rest+ribbon）(Finchley版本)](http://blog.csdn.net/forezp/article/details/81040946)

在上一篇文章，讲了服务的注册和发现。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feign。在这一篇文章首先讲解下基于ribbon+rest。

**一、ribbon简介**

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

**二、准备工作**

这一篇文章基于上一篇文章的工程，启动eureka-server 工程；启动service-hi工程，它的端口为8762；将service-hi的配置文件的端口改为8763,并启动，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。访问localhost:8761如图所示：

如何一个工程启动多个实例，请看这篇文章:<https://blog.csdn.net/forezp/article/details/76408139>



**三、建一个服务消费者**

重新新建一个spring-boot工程，取名为：service-ribbon;   
在它的pom.xml文件分别引入起步依赖spring-cloud-starter-eureka、spring-cloud-starter-ribbon、spring-boot-starter-web，代码如下：

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.forezp</groupId>

<artifactId>service-ribbon</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>service-ribbon</name>

<description>Demo project for Spring Boot</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>1.5.2.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-eureka</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-ribbon</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-dependencies</artifactId>

<version>Dalston.RC1</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

<repositories>

<repository>

<id>spring-milestones</id>

<name>Spring Milestones</name>

<url>https://repo.spring.io/milestone</url>

<snapshots>

<enabled>false</enabled>

</snapshots>

</repository>

</repositories>

</project>

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8

在工程的配置文件指定服务的注册中心地址为[http://localhost:8761/eureka/](http://localhost:8761/eureka/" \t "_blank)，程序名称为 service-ribbon，程序端口为8764。配置文件application.yml如下：

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。

@SpringBootApplication

@EnableDiscoveryClient

public class ServiceRibbonApplication {

public static void main(String[] args) {

SpringApplication.run(ServiceRibbonApplication.class, args);

}

@Bean

@LoadBalanced

RestTemplate restTemplate() {

return new RestTemplate();

}

}

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：

@Service

public class HelloService {

@Autowired

RestTemplate restTemplate;

public String hiService(String name) {

return restTemplate.getForObject("http://SERVICE-HI/hi?name="+name,String.class);

}

}

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

/\*\*

\* Created by fangzhipeng on 2017/4/6.

\*/

@RestController

public class HelloControler {

@Autowired

HelloService helloService;

@RequestMapping(value = "/hi")

public String hi(@RequestParam String name){

return helloService.hiService(name);

}

}

在浏览器上多次访问[http://localhost:8764/hi?name=forezp](http://localhost:8764/hi?name=forezp" \t "_blank)，浏览器交替显示：

hi forezp,i am from port:8762

hi forezp,i am from port:8763

这说明当我们通过调用restTemplate.getForObject(“[http://SERVICE-HI/hi?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。

**四、此时的架构**



* 一个服务注册中心，eureka server,端口为8761
* service-hi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
* sercvice-ribbon端口为8764,向服务注册中心注册
* 当sercvice-ribbon通过restTemplate调用service-hi的hi接口时，因为用ribbon进行了负载均衡，会轮流的调用service-hi：8762和8763 两个端口的hi接口；

源码下载：<https://github.com/forezp/SpringCloudLearning/tree/master/chapter2>

**五、参考资料**