

原

spring cloud：二、服务的消费（ribbon和feign两种方式）

2018年03月07日 14:53:46

0

收藏

评论

微信

微博

QQ

阅读数：208

上一篇文章总结了在分布式服务系统中，服务是如何在服务中心注册与发现的，这篇文章主要总结一下的服务之间是怎么调用的。传统的一个项目当中，通常会有系统管理、用户管理、角色管理、人员管理等等这些模块，在分布式的系统以往的同一个项目里的每一个功能模块可以相应独立拆分成一个项目也就是一个服务，然后这些服务在服务中心注册，服务与服务之直接调用，而通过服务中心来调用。

Spring cloud 服务间调用一般有两种方式，第一种：ribbon+restTemplate第二种：fegin

第一种ribbon+restTemplate:

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

新建一个maven项目：zhangsanService，作为服务消费者，去调用上一篇文章注册到服务中心的helloService。

Pom.xml:

```
<?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/4.0.0http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.gaox.zhangsanService</groupId>
  <artifactId>zhangsanService</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

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

  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.5.10.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>
    <spring-cloud.version>Edgware.SR2</spring-cloud.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
```

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-netflix-ribbon</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>${spring-cloud.version}</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>

Application.properties:

```
server.port=8764
spring.application.name=zhangsanService
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/
```

ZhangsanServiceApplication.java



```
package com.gaox.zhangsanService;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.loadbalancer.LoadBalanced;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import org.springframework.context.annotation.Bean;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;

@SpringBootApplication
@EnableEurekaClient
@RestController
public class ZhangsanServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(ZhangsanServiceApplication.class, args);
    }
    @Bean
    @LoadBalanced
    public RestTemplate restTemplate(){
        return new RestTemplate();
    }

    @Autowired
    private RestTemplate restTemplate;

    @RequestMapping("/hello")
    public String hello(String name){
        String result=restTemplate.getForObject("http://HELLOSERVICE/hello?name="+name,String.class);
        return result;
    }
}
```



依次启动serviceCenter,helloService:8762,helloService:8763,zhangsanService,在浏览器上多次访问<http://localhost:8764/hello?name=zhangsan>，浏览器交替显示：

你好，zhangsan。我是helloService,端口是8762

你好，zhangsan。我是helloService,端口是8763

这说明当我通地调用restTemplate.getForObject(<http://HELLOSERVICE/hello?name=zhangsan>,String.class)方法时，已经作了负载均衡，访问了不同端口的服务实例。

这时项目的架构就是

一个服务注册中心：serviceCenter,端口8761

在服务注册中心注册的两个helloService实例，端口分别是8762和8763

在服务注册中心注册的另一个实例：zhangsanService,端口8764

当zhangsanService通过restTemplate调用helloService的hello接口时，因为用ribbon进行了负载均衡，所以才会出现轮流的调用8762和8763两个端口的helloService实例。

第二种方法feign：

新建一个lisiService，通过feign的方法去调用上一篇文章在服务注册中心注册的helloService。

Feign是一个声明式的伪http客户端，它使得写http客户端变得更简单，使用fegin，只需要创建一个接口，并注解，它具有可插拔的注解特性，可使用Feign 注解和JAX-RS注解。Feign支持可插拔的编码器和解码器。Feign默认集成了Ribbon，并和Eureka结合，默认实现了负载均衡的效果。

Pom.xml

```
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.gaox.lisiService</groupId>
<artifactId>lisiService</artifactId>
<version>0.0.1-SNAPSHOT</version>
<packaging>jar</packaging>

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

<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.5.10.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>
  <spring-cloud.version>Edgware.SR2</spring-cloud.version>
</properties>

<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-eureka</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-feign</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>${spring-cloud.version}</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>
```



```
server.port=8765
spring.application.name=lisiService
eureka.client.service-url.defaultZone=http://localhost:8761/eureka/
```

LisiServiceApplication.java

```
package com.gaox.lisiService;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import org.springframework.cloud.netflix.feign.EnableFeignClients;
import org.springframework.cloud.netflix.feign.FeignClient;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@SpringBootApplication
@EnableEurekaClient
@EnableFeignClients
@RestController
public class LisiServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(LisiServiceApplication.class, args);
    }

    @FeignClient("helloService")
    public interface HelloService {
        @RequestMapping(value = "/hello", method = RequestMethod.GET)
        public String hello(@RequestParam("name") String name);
    }

    @Autowired
    private HelloService helloService;

    @RequestMapping("/hello")
    public String hello(@RequestParam("name")String name){
        return helloService.hello(name);
    }
}
```

0

收藏

评论

微信

微博

QQ

依次启动serviceCenter,helloService:8762,helloService:8763,lisiService , 多次访问<http://localhost:8765/hello?name=lisi> , 浏览器会交替显示

你好 , lisi。我是helloService,端口是8762

你好 , lisi。我是helloService,端口是8763

这说明feign确实默认集成了ribbon , 实现了负载均衡。

版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/fox9916/article/details/79471179>

个人分类：[spring cloud](#)
所属专栏：[spring cloud](#)

[查看更多>>](#)

springCloud学习01之eureka服务发现-提供者-消费者ribbon/feign-负载均衡

微服务的其中一个特点就是有许许多多的粒度小（功能单一，比如用户管理，短信发送管理，邮件发送管理，文件管理等）、能独立部署、扩展、运行的小应用，可以称为api,也就是服务提供者。api之间可以相互调用，但...

dream_broken 2017-07-27 18:25:03 阅读数：3108

springcloud实战之5 服务消费者（ribbon）

前几篇介绍了服务注册中心以及服务提供者，目前需要一个服务消费者。本篇介绍如何用Ribbon消费注册中心上注册的服务。Ribbon介绍Ribbon是一个基于HTTP和TCP客户端的负载均衡器。Feign...

u012806787 2017-12-13 20:47:32 阅读数：316

新出的微赚钱方式被曝光，网友：白玩这么多年！

汇奇林科技·顶新

Feign正确的使用姿势和性能优化注意事项

1. feign自定义Configuration和root 容器有效隔离。用@Configuration注解不能在主@ComponentScan (or @SpringBootApplication)...

belalds 2018-05-28 12:55:42 阅读数：95

springcloud用feign消费服务记录

springcloud提供了两种消费服务的方式，feign和ribbon。实质上feign是集成了ribbon的，最终也是利用ribbon做了负载均衡，发了http请求。ribbon的方式编码的时候更像...

qq_16397481 2018-01-04 11:33:33 阅读数：178

Feign 的简单使用及传参方式

Feign 的简单使用 0. 简介 Feign 是简化Java HTTP客户端开发的工具。它使用注解的方式将HTTP的URL封装成接口，每个URL对应一个接口，大大简化了HTTP客户端的开发。...

u013451048 2018-03-20 16:52:09 阅读数：582

spring-cloud-feign集成feign的几个注意事项

那些用feign踩过的坑

u013257425 2017-01-19 11:13:14 阅读数：4482

“性”生活太快？问题究竟出在哪里？

北京登蓝·顶新

spring cloud feign调用service的两种POST传值方式

1、application/json包引用compile("org.springframework.cloud:spring-cloud-starter-openfeign");fe...

YapingGe2014 2018-05-31 17:31:05 阅读数：22

Feign学习笔记2-源码解读

Feign源码

w8452960 2017-08-13 21:07:26 阅读数：814

2、springcloud微服务：基于Feign的服务调用

摘要：Feign是一个声明式、模板化的HTTP客户端调用组件，它可以像调用本地方法一样调用远程服务。创建一个新的服务：microservice-provider-user，在microservice-...

errorandexception 2018-03-07 14:03:22 阅读数：17

Spring Cloud学习系列第一章·使用Feign调用服务

0

收藏

评论

微信

微博

QQ

一、Feign简介 Feign是一个声明式的Web服务客户端。这使得Web服务客户端的写入更加方便 要使用Feign创建一个界面并对其进行注释。它具有可插入注释支持，包括Feign注释和JAX-R...

 wangjinnan16

2017-08-11 21:31:00

阅读数：923

Spring Cloud Feign 的使用注意事项

1.传递多个参数,API接口中的参数需要使用@RequestParam("参数名") 2.@RequestMapping中的method的值不能是多个,只能是单个 (随着学习feign,继续完善)...

 qq_33547169

2017-10-24 18:39:07

阅读数：106

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

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 在上一篇文章，讲了服

 forezp

2017-04-08 23:25:26

阅读数：314257

微服务之间的调用 (Ribbon与Feign)

概述在前面的文章中，我们讲了使用Eureka作为服务注册中心，在服务启动后，各个微服务会将自己注册到Eureka s 那么服务之间是如何调用？又是如何进行负载均衡的呢？本文讲讲服务之间调用及...

 jrn1012

2017-09-04 15:05:08

阅读数：9378

高效、准确、低成本的文字识别api

证件、文件一键扫描，准确率高达99%。经过海量用户和多种复杂场景考验

 0

 收藏

 评论

 微信

 微博

 QQ



spring cloud-使用feign来消费Restful服务同时加入Ribbon来实现负载均衡

前言 在前面的示例中，我们消费spring boot提供的Restful服务的时候，使用的是RestTemplate来实现的，实现起来还是比较复杂的，尤其是在消费复杂的Restful服务的时候，还需...

 liuchuanhong1

2017-01-25 09:57:28

阅读数：12448

3.springcloud中使用Ribbon和Feign调用服务以及服务的高可用

springcloud中使用Ribbon和Feign调用服务以及服务的高可用

 u014532775

2017-11-10 17:28:00

阅读数：1128

微服务：Eureka+Zuul+Ribbon+Feign+Hystrix构建微服务架构

本案例将打造一个微服务框架，参考来源官方参考文档 微服务：是什么？网上有一堆资料。不做叙述。标题提到的框架是spring-cloud-netflix相关开源框架。d emo源码gi...

 qq_18675693

2016-11-22 13:51:19

阅读数：31058

SpringCloud--服务消费者 (rest+ribbon)

一、ribbon简介 Ribbon is a client side load balancer which gives you a lot of control over the beha...

 u012470138

2017-08-09 10:51:56

阅读数：243

一起来学SpringCloud之 - 服务消费者 (Ribbon)

上一篇文章，简单概述了服务注册与发现，在微服务架构中，业务都会被拆分成一个独立的服务，服务之间的通讯是基于http restful的，Ribbon可以很好地控制HTTP和TCP客户端的行为，Sprin...

 memmsc

2017-09-28 12:57:35

阅读数：395

免费云主机试用一年

云服务器免费试用

百度广告



Spring Cloud架构教程 (十一) 服务消费 (Ribbon)

Spring Cloud Ribbon Spring Cloud Ribbon是基于Netflix Ribbon实现的一套客户端负载均衡的工具。它是一个基于HTTP和TCP的客户端负载均衡器。它...

SpringCloud教程 | 第二篇: 服务消费者 (rest+ribbon)

在上一篇文章，讲了服务的注册和发现。在服务架构中，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有两种调用方式，一种是ribbon+rest...

qq_36330643 2017-08-14 10:50:14 阅读数：345

个人资料



冰与火IAF

关注

原创 21 粉丝 9 喜欢 7 评论 9

等级： 博客 已 访问：2万+
积分：414 排名：12万+
勋章： 恒

云虚拟主机-免费版

每人限购一台，如果连续90天，未生成激活码，账号将会被删除，该虚拟主机将被关闭。

1G 内存空间 10G 存储空间 50M 数据流量

立即购买

虚拟主机 免费

最新文章

- 关于springboot+shiro+thymeleaf页面级元素的权限控制的问题
- Spring +mybatisplus+shiro权限管理集成整合
- spring cloud ：五、分布式配置中心 (spring cloud config)
- spring cloud ：四、路由网关 (zuul)
- spring cloud ：三、断路器 (hystrix)

博主专栏



spring cloud

阅读量：12436 5 篇

个人分类

- spring 3篇
- Java基础 5篇
- database 1篇
- git 2篇
- springboot 4篇

展开

归档

2018年5月 2篇

0

收藏

评论

微信

微博

QQ

2017年10月	5篇
2017年7月	1篇
...	...
展开	

热门文章

- spring cloud ：五、分布式配置中心（spring cloud config）

阅读量：11397
- git本地项目代码上传至码云远程仓库总结

阅读量：3891
- spring webflow 学习小结

阅读量：3630
- 关于spring boot属性文件中中文参数输出到页面乱码

阅读量：573
- spring cloud ：一、服务的注册与发现

阅读量：268

最新评论

- spring cloud ：五、分...

u010897136：....
- spring webflow 学习...

fox9916：[reply]qq_24646129[reply] 能跑通的
- spring cloud ：五、分...

fox9916：[reply]qq_31559209[reply] git的用户名密码是需要配置的
- spring cloud ：五、分...

qq_31559209：问下我在将分布式配置中心加入到eurake中后在客户端就读取不到git上的配置属性了是怎么回事。
- spring webflow 学习...

qq_24646129：感觉zhengzhou.xml，web-application-config.xml这些文件都不全啊...



工业键盘



联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

👤 QQ客服 🗨 客服论坛

关于 招聘 广告服务 百度

©1999-2018 CSDN版权所有

京ICP证09002463号

经营性网站备案信息

网络110报警服务

中国互联网举报中心

北京互联网违法和不良信息举报中心



0



收藏



评论



微信



微博



QQ