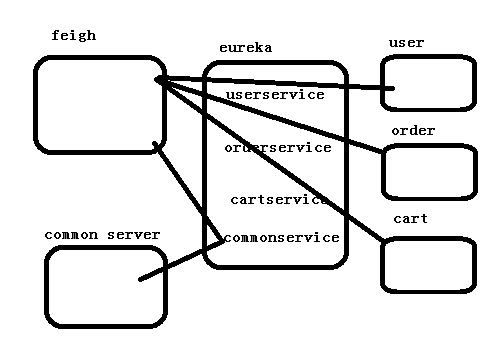
# 14.springcloud的其他组件

## 14.1.feign



feign也是服务调用服务的客户端，可以实现一个代码工程，被需要调用公共服务的服务依赖

搭建feign

1.quickstart（feign-client）

2.pom继承springboot、springcloud、eureka、feign

<parent>

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

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

<version>1.5.9.RELEASE</version>

</parent>

<dependencyManagement>

<dependencies>

<dependency>

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

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

<version>Edgware.RELEASE</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<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>

3.配置文件eureka

server.port=9009

spring.application.name=feignservice

eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka

4.启动类

|  |
| --- |
| **package** cn.tedu;  **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;  @SpringBootApplication  @EnableEurekaClient  @EnableFeignClients  **public** **class** StarterFeign {  **public** **static** **void** main(String[] args) {  SpringApplication.*run*(StarterFeign.**class**, args);  }  } |

5.feign代理Service01（有三个节点：client01，client02，client03）微服务

HelloService

|  |
| --- |
| **package** cn.tedu.feign.service;  **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;  /\*\*  \* 此service可以直接访问公用工程  \*/  //注解是指定需要连接的微服务名称  @FeignClient("service01")  **public** **interface** HelloService {  //相当于调用了service01服务的url为hi的请求，请求参数为name，请求方式为get  @RequestMapping(value="hi",method=RequestMethod.***GET***)  **public** String sayHello(@RequestParam(value="name")String name);  } |

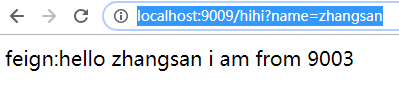
6.可以在任意工程依赖feign，然后在controller中注入HelloService即可使用

|  |
| --- |
| **package** cn.tedu.feign.controller;  **import** org.springframework.beans.factory.annotation.Autowired;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RestController;  **import** cn.tedu.feign.service.HelloService;  @RestController  **public** **class** HelloController {  @Autowired  **private** HelloService helloService;  @RequestMapping("hihi")  **public** String sayhihi(String name){  **return** "feign:"+helloService.sayHello(name);  }  } |

测试：启动feign、client01、client02、client03

访问http://localhost:9009/hihi?name=zhangsan

请求结果发现是三个client轮询



7.实验：

让client01宕机

如果是ribbon，会一直轮询访问1、2、3，直到eureka将宕机节点踢出去

如果是feign，在报错一两次后，直接跳过1，只访问2、3

8.feign的优势：

1.只要依赖feign，调用别的工程时，就像调用自己的工程一样，注入service即可使用

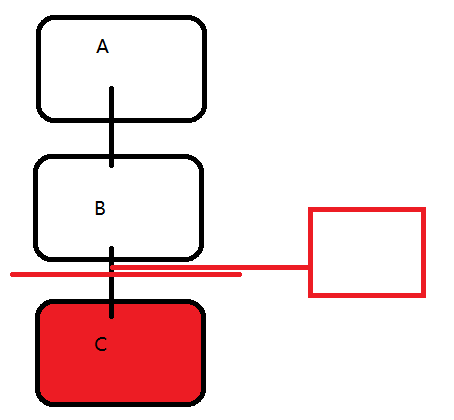
2.feign的客户端支持hystrix访问降级，ribbon不能

访问降级：当访问到的节点出现错误后，不再访问此节点

## 14.2.hystrix

熔断器,对系统中服务间的关系可以起到容错的作用;

在微服务集群中,服务调服务的情况非常常见,一旦出现复杂的调用关系中,由于底层的服务调用失败,引起的压力积累,最终导致多级服务的调用整体瘫痪,所以springcloud引入容错组件熔断器;利用牺牲局部的功能,保全整体的效果,在某个局部中的访问虽然不能获取完美目的数据,可以退而求其次的获取可用数据;



**实现error信息的服务降级**

搭建一个hystrix的ribbon测试工程,实现功能服务降级

service01 9001 9002 9003当其中有一个宕机时,不报异常,直接返回一个降级的需要打印字符串;

在ribbon工程的基础上实现hystrix的添加

pom文件中添加hystrix依赖

<dependency>

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

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

</dependency>

HelloController

@RestController

public class HelloController {

@Autowired

private HelloService helloService;

@RequestMapping("hello")

public String sayHello(String name){

String result=        helloService.sayHello(name);

return "RIBBON:"+result;}

}

HelloService

@Autowired

private RestTemplate client;

//添加一个hystrix熔断器的毁掉命令

@HystrixCommand(fallbackMethod="error")

//这个注解使用的结果,一旦当前方法发生访问服务相关的错误

//例如连接失败,会根据command指定的method名称访问

//error的方法,这个方法就是降级使用的方法

public String sayHello(String name) {

String result = client.getForObject("http://service01/hi?name="+name, String.class);

return result;

}

//自定义一个error降级方法

//方法必须结构和sayHello完全一致,参数,和返回数据类型

public String error(String name){

return "sorry "+name+",error happened";

}

//利用这种代码实现一个服务的调用service01只要还存在一个节点,就能给我正确响应

可以通过hystrix接收的错误,实现日志的输出,或者实现压力测试时对某个服务的节点压力展示(通过日志);

服务提供者到低要多少个节点的集群才是合理的

**互联网架构总结**

