

SpringCloud之四

负载均衡Feign

一、Feign简介

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Feign，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。

简而言之：

- Feign 采用的是基于接口的注解
- Feign 整合了ribbon，具有负载均衡的能力
- 整合了Hystrix，具有熔断的能力

官网描述如图：



官网描述是：

```
1 Feign is a Java to HTTP client binder inspired by Retrofit, JAXRS-2.0,  
2 and WebSocket. Feign's first goal was reducing the complexity of binding  
3 Denominator uniformly to HTTP APIs regardless of ReSTfulness.
```

即：Feign是受到[Retrofit](#)，[JAXRS-2.0](#)和[WebSocket](#)启发的Java到HTTP客户端绑定器。Feign的第一个目标是降低将[Denominator](#)统一绑定到HTTP API 的复杂性，无论[ReSTfulness](#)如何。

Feign官网描述运作说明如下：

```
1 Feign works by processing annotations into a templated request.
2 Arguments are applied to these templates in a straightforward fashion
3 before output. Although Feign is limited to supporting text-based APIs,
4 it dramatically simplifies system aspects such as replaying requests.
5 Furthermore, Feign makes it easy to unit test your conversions knowing this.
```

即：Feign通过将注释处理为模板化请求来工作。在输出之前，参数以直接的方式应用于这些模板。虽然Feign仅限于支持基于文本的API，但它极大地简化了系统方面，例如重放请求。此外，Feign可以轻松地对您的转换进行单元测试。
当然有需要的，还可以查看官网<https://github.com/OpenFeign/feign>的描述、

官网描述的Feign的注解如下：

接口注释		
Feign注释定义了Contract接口与底层客户端应如何工作之间的关系。 Feign的默认合约定义了以下注释：		
注解	接口目标	用法
@RequestMapping	方法	定义HttpMethod和UriTemplatefor请求。 Expressions , {expression}使用相应的带@param注解的参数来解析用大括号包装的值。
@Param	参数	定义模板变量，其值将用于Expression按名称解析相应的模板。
@Headers	方法，类型	定义HeaderTemplate; 一个变种UriTemplate。使用带@param注解的值来解析相应的Expressions。在a上使用时Type，模板将应用于每个请求。在a上使用时Method，模板仅适用于带注解的方法。
@QueryMap	参数	定义一个Map名称 - 值对或POJO，以扩展为查询字符串。
@HeaderMap	参数	定义一个Map名称 - 值对，以扩展为Http Headers
@Body	方法	定义a Template，类似于a UriTemplate和HeaderTemplate，使用带@param注解的值来解析相应的Expressions。

总的来说就是：

- Feign是一个**声明式的REST客户端**，它的目的就是让REST调用更加简单。
- Feign提供了**HTTP请求的模板**，通过编写简单的接口和插入注解，就可以定义好HTTP请求的参数、格式、地址等信息。

- 而Feign则会完全代理HTTP请求，我们只需要像调用方法一样调用它就可以完成服务请求及相关处理。
- SpringCloud对Feign进行了封装，使其支持SpringMVC标准注解和HttpMessageConverters。
- Feign可以与Eureka和Ribbon组合使用以支持负载均衡。

废话够多了，相信很多人会觉得文章马上要和裹脚布有的一拼了，所以开始干货发送！

二、Feign创建

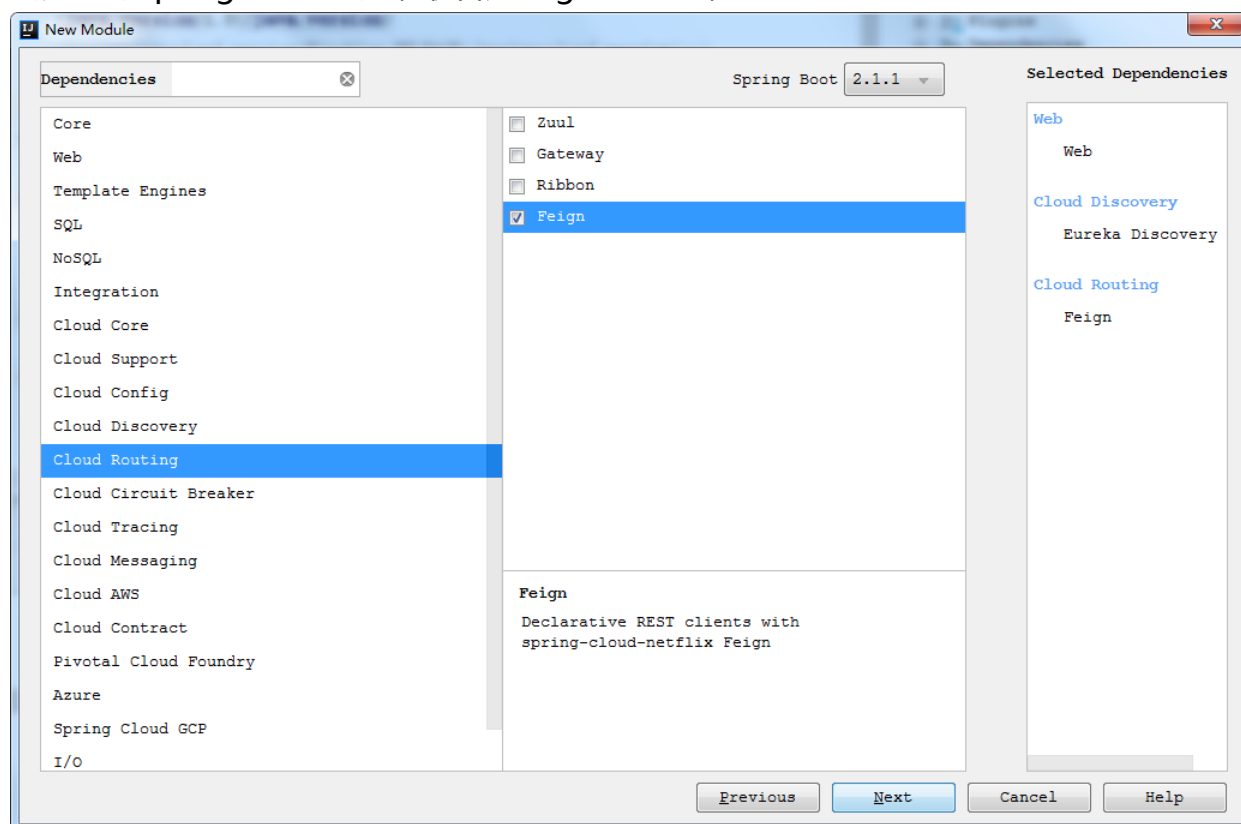
2.1、准备

启动eureka-server，端口为8761；启动login-service两次，端口分别为8762、8773。

具体的可以参考之前的文章。

2.2、创建Feign服务

新建一个spring-boot工程，取名为feign-serice，



在它的pom文件引入Feign的起步依赖spring-cloud-starter-feign、Eureka的起步依赖spring-cloud-starter-eureka、Web的起步依赖spring-boot-starter-web，代码如下：

pom.xml

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.
w3.org/2001/XMLSchema-instance"
3    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apach
e.org/xsd/maven-4.0.0.xsd">
4    <modelVersion>4.0.0</modelVersion>
5    <groupId>xyz.jiangnanke</groupId>
6    <artifactId>feign-service</artifactId>
7    <version>0.0.1-SNAPSHOT</version>
8    <name>feign-service</name>
9    <description>Demo project for Spring Boot</description>
10
11    <parent>
12      <groupId>xyz.jiangnanke</groupId>
13      <artifactId>main</artifactId>
14      <version>0.0.1-SNAPSHOT</version>
15    </parent>
16
17    <dependencies>
18      <dependency>
19        <groupId>org.springframework.boot</groupId>
20        <artifactId>spring-boot-starter-web</artifactId>
21      </dependency>
22      <dependency>
23        <groupId>org.springframework.cloud</groupId>
24        <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
25      </dependency>
26      <dependency>
27        <groupId>org.springframework.cloud</groupId>
28        <artifactId>spring-cloud-starter-openfeign</artifactId>
29      </dependency>
30
31    </dependencies>
32
33  </project>

```

同样在main模块的pom.xml里面加入模块关联

2.3、配置项目

首先是application.yml文件，指定程序名为service-feign，端口号为8765，服务注册地址为http://localhost:8761/eureka/，代码如下：

```

1 server:
2   port: 8765
3
4 spring:
5   application:
6     name: feign-service
7
8 eureka:
9   client:
10    serviceUrl:
11    defaultZone: http://localhost:8761/eureka/

```

然后启动类FeignServiceApplication.java的配置，加上@EnableFeignClients注解开启Feign的功能：

```

1 package xyz.jiangnanke.feignservice;
2
3 import org.springframework.boot.SpringApplication;
4 import org.springframework.boot.autoconfigure.SpringBootApplication;
5 import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
6 import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
7 import org.springframework.cloud.openfeign.EnableFeignClients;
8
9 @SpringBootApplication
10 @EnableEurekaClient
11 @EnableDiscoveryClient
12 @EnableFeignClients
13 public class FeignServiceApplication {
14
15     public static void main(String[] args) {
16         SpringApplication.run(FeignServiceApplication.class, args);
17     }
18
19 }
20

```

2.4、编写测试

首先，定义一个feign接口，通过@ FeignClient（“服务名”），来指定调用哪个服务。比如在代码中调用了login-service服务的“/login”接口，代码如下：
SchedualLoginService.java

```

1 package xyz.jiangnanke.feignservice.service;
2
3 import org.springframework.cloud.openfeign.FeignClient;
4 import org.springframework.web.bind.annotation.RequestMapping;
5 import org.springframework.web.bind.annotation.RequestMethod;
6 import org.springframework.web.bind.annotation.RequestParam;
7
8 /**
9  * @Auther: zhengfeng
10  * @Date: 2018\12\20 0020 15:49
11  * @Description: 指定调用login-service服务
12  */
13 @FeignClient(value = "login-service")
14 public interface SchedualLoginService {
15     /**
16      * 调用login-service服务的login接口请求
17      * @param name
18      * @return
19      */
20     @RequestMapping(value = "/login", method = RequestMethod.GET)
21     String loginOne(@RequestParam(value = "name") String name);
22 }

```

然后，在Web层的controller层，对外暴露一个"/login"的API接口，通过上面定义的Feign客户端SchedualLoginService 来消费服务。代码如下：

LoginController.java

```

1 package xyz.jiangnanke.feignservice.controller;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.web.bind.annotation.GetMapping;
5 import org.springframework.web.bind.annotation.RequestParam;
6 import org.springframework.web.bind.annotation.RestController;
7 import xyz.jiangnanke.feignservice.service.SchedualLoginService;
8
9 /**
10  * @Auther: zhengfeng
11  * @Date: 2018\12\20 0020 15:52
12  * @Description: 对外暴露一个"/login"的API接口，通过上面定义的Feign客户端SchedualLoginService 来消费服务
13  */
14 @RestController

```

```

15 public class LoginController {
16     //编译器报错，无视。 因为这个Bean是在程序启动的时候注入的，编译器感知不到，所以报错。
17     @Autowired
18     SchedualLoginService schedualLoginService;
19
20     @GetMapping(value = "/login")
21     public String login(@RequestParam String name) {
22         return schedualLoginService.loginOne(name);
23     }
24 }

```

2.5、运行访问

可以看到在eureka里面注册了

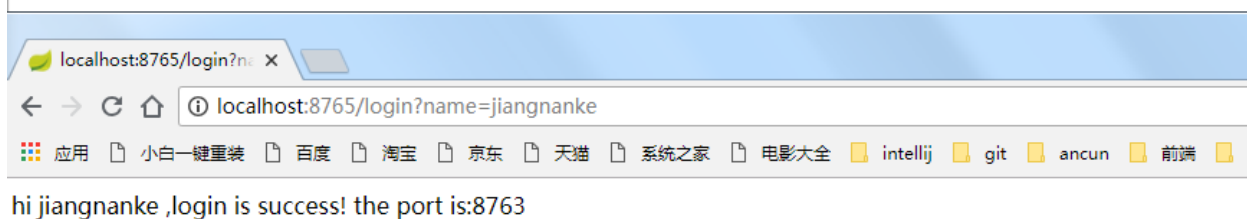
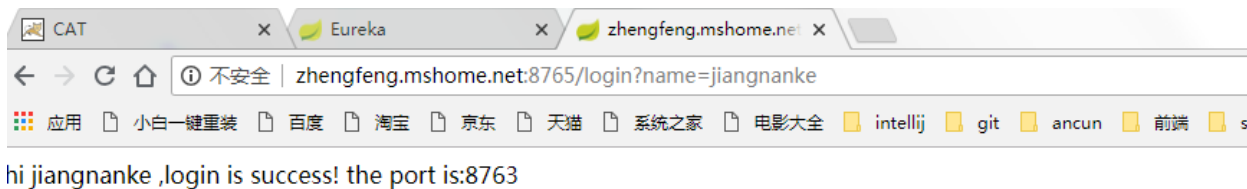
The screenshot shows the Spring Eureka dashboard at localhost:8761. The top navigation bar includes 'HOME' and 'LAST 1000 SINCE STARTUP'. The 'System Status' section displays environment details (test, default) and system metrics (Current time: 2018-12-20T16:00:23+0800, Uptime: 00:03, Lease expiration enabled: false, Renewal threshold: 6, Renewal last min: 3). The 'DS Replicas' section shows 'Instances currently registered with Eureka' in a table:

Application	AMIs	Availability Zones	Status
FEIGN-SERVICE	n/a (1)	(1)	UP (1) - zhengfeng.mshome.net.feign-service:8765
LOGIN-SERVICE	n/a (2)	(2)	UP (2) - zhengfeng.mshome.net.login-service:8762, zhengfeng.mshome.net.login-service:8763

The 'General Info' section provides a summary of system metrics:

Name	Value
total-avail-memory	263mb
environment	test
num-of-cpus	4
current-memory-usage	172mb (65%)
server-uptime	00:03
registered-replicas	

然后多次访问<http://localhost:8765/login?name=jiangnanke> ,结果如下：



3、其他

3.1、JAVA 项目中接口调用怎么做 ？

1. HttpClient
2. Okhttp
3. Httpurlconnection
4. RestTemplate

参考资料

<https://springcloud.cc/>

<https://github.com/OpenFeign/feign>

<https://blog.csdn.net/forezp/article/details/81040965>

用Spring中的RestTemplate来调用rest接口 <http://cxytiandi.com/blog/detail/6157>

<http://cxytiandi.com/blog/detail/12189>

<https://github.com/yinjihuan/spring-cloud>