# 一、Feign简介

Feign是一个声明式WebService客户端。使用方法是定义一个接口，然后在上面添加注解即可。Feign可以和Eureka和Ribbon组合使用以支持负载均衡。

Feign通过接口的方式调用Rest服务（之前是Ribbon和RestTemplate），通过Feign直接找到服务接口，由于在进行服务调用的时候融合了Ribbon技术，所以也支持负载均衡。

# 二、项目构建

## 1. 修改springcloud-api项目

我们将Feign的接口写在api项目中

### 1. POM添加对Feign的依赖

```
<dependencies><!-- 当前Module需要用到的jar包，按自己需求添加，如果父类已经包含了，可以不用写版本号 -->
        <!-- feign负载均衡相关的 -->
        <dependency>

<groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-feign</artifactId>
        </dependency>
    </dependencies>
```

## 2. 编写DeptClientService接口

```
@FeignClient(value = "springcloud-dept")
public interface DeptClientService {

    @RequestMapping(value = "/dept/get/{id}", method = RequestMethod.GET)
    public Dept get(@PathVariable("id") long id);

    @RequestMapping(value = "/dept/list", method = RequestMethod.GET)
    public List<Dept> list();

    @RequestMapping(value = "/dept/add", method = RequestMethod.POST)
```

```
    public boolean add(Dept dept);

}
```

## 2.新建feign消费端项目

新建maven子模块springcloud-consumer-dept-feign

### 1. POM依赖

```
<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>
  <parent>
    <groupId>com.haoge.cloud</groupId>
    <artifactId>springcloud</artifactId>
    <version>0.0.1-SNAPSHOT</version>
  </parent>
  <artifactId>springcloud-consumer-dept-feign</artifactId>

   <dependencies>
        <dependency>
```

```
            <groupId>com.haoge.cloud</groupId>
            <artifactId>springcloud-api</artifactId>
            <version>${project.version}</version>
        </dependency>
        <!-- Ribbon相关 ，负载均衡-->
        <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.cloud</groupId>
            <artifactId>spring-cloud-starter-config</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
```

```
31              <artifactId>spring-boot-starter-
web</artifactId>
32          </dependency>
33          <!-- 修改后立即生效，热部署 -->
34          <dependency>
35
<groupId>org.springframework</groupId>
36
<artifactId>springloaded</artifactId>
37          </dependency>
38          <dependency>
39
<groupId>org.springframework.boot</groupId>
40              <artifactId>spring-boot-
devtools</artifactId>
41          </dependency>
42          <dependency>
43
<groupId>org.springframework.cloud</groupId>
44              <artifactId>spring-cloud-starter-
feign</artifactId>
45          </dependency>
46      </dependencies>
47
48  </project>
```

## 2. yaml文件

```yaml
server:
  port: 80

eureka:
  client:
    register-with-eureka: false #自己不能注册
    service-url:
      defaultZone:
http://eureka7001.com:7001/eureka/,http://eureka7002.com:7002/eureka/,http://eureka7003.com:7003/eureka/
```

## 3. 主启动类

```
@SpringBootApplication
@EnableEurekaClient
@EnableFeignClients(basePackages=
{"com.haoge.cloud"})
@ComponentScan("com.haoge.cloud")
public class DeptConsumerFeign_App {

    public static void main(String[] args) {

SpringApplication.run(DeptConsumerFeign_App.clas
s, args);
    }

}
```

## 4. Controller

```
@RestController
public class DeptController_Consumer {
    @Autowired
    private DeptClientService service;



@RequestMapping(value="/consumer/dept/get/{id}")
    public Dept get(@PathVariable("id") Long id)
{

```

```
10          return this.service.get(id);
11
12      }
13
14      @RequestMapping(value="/consumer/dept/list")
15      public List<Dept> list() {
16
17          return this.service.list();
18
19      }
20
21      @RequestMapping(value="/consumer/dept/add")
22      public boolean add(Dept dept) {
23
24          return this.service.add(dept);
25
26      }
27  }
```

### 5. ConfigBean配置类

自定义负载均衡算法的

```
1  @Configuration //@Configuration配置   ConfigBean
   = applicationContext.xml
2  public class ConfigBean {
3      @Bean
```

```
4       @LoadBalanced//Spring Cloud Ribbon是基于Netflix Ribbon实现的一套客户端        负载均衡的工具。
5       //@LoadBalanced内置7种不同的负载均衡的算法，如果我们不显示的申明我们想要的算法，就使用默认的轮训算法。
6       //如果我们显示的声明我们需要的算法，则会替代默认的轮训算法
7       public RestTemplate getRestTemplete() {
8           return new RestTemplate();
9       }
10  //如果我们要显式的指定自己想要的算法，则改变返回算法的名字即可。例子如下
11      @Bean
12      public IRule myRule(){
13          //return new  RetryRule();//如果服务提供者全部可用，则和轮训算法一样。当某一个服务不可用的时候
14                                      //查询该服务不可用几次之后，自动的不会再次查找该服务。在剩下的服务中进行轮训
15
16              return new RandomRule();//随机算法。达到的目的，用我们重新选择的随机算法替代默认的轮询。
17      }
18  }
```

## 6. 测试

启动7001,7002,7003三个Eureka Server,然后提供8001,8002,8003三个服务提供者。之后启动springcloud-consumer-dept-feign。

访问：http://localhost/consumer/dept/list