# 前言

有了 Eureka，RestTemplate，Ribbon 我们就可以愉快地进行服务间的调用了，但是使用 RestTemplate 还是不方便，我们每次都要进行这样的调用。



**这样每次都调用 RestTemplate 的 API 是否太麻烦，而且每次都要写URL，设置返回值类型，如果是一个需要传大量参数的请求，还要构造一个复杂的请求，接着发送请求过去，最后对返回的响应结果再写一大堆代码来处理，我能不能像调用原来代码一样进行各个服务间的调用呢？答案是确定的有，Netflix已经为我们提供了一个框架Feign(菲恩)。Spring Cloud Feign在Netflix的Feign的基础上进行了封装，简化了咱们的使用。**

# 什么是Feign

**官网解释**：

http://projects.spring.io/spring-cloud/spring-cloud.html#spring-cloud-feign

Feign是一个声明式WebService客户端。**使用Feign能让编写Web Service客户端更加简单**, 它的使用方法是**定义一个接口**，然后**在上面添加注解**，同时也支持JAX-RS标准的注解。Feign也支持可拔插式的编码器和解码器。Spring Cloud对Feign进行了封装，使其支持了Spring MVC标准注解和HttpMessageConverters。**Feign可以与Eureka和Ribbon组合使用以支持负载均衡**。



Feign是一个声明式的Web服务客户端，使得**编写Web服务客户端**变得非常容易，

**只需要创建一个接口，然后在上面添加注解即可**。

参考官网：<https://github.com/OpenFeign/feign>

# Feign能干什么

Feign 是一种声明式、模板化的 HTTP 客户端(仅在 consumer 中使用)，Feign旨在使编写Java Http客户端变得更容易。**声明式调用就像调用本地方法一样调用远程方法**。Spring Cloud 的声明式调用, 可以做到使用 HTTP 请求远程服务时能就像调用本地方法一样的体验，开发者完全感知不到这是远程方法，更感知不到这是个 HTTP 请求。**它解决了让开发者调用远程接口就跟调用本地方法一样**，无需关注与远程的交互细节，更无需关注分布式环境开发。

前面在使用Ribbon+RestTemplate时，利用RestTemplate对http请求的封装处理，形成了一套模版化的调用方法。但是在实际开发中，由于对服务依赖的调用可能不止一处，往往一个接口会被多处调用，所以通常都会针对**每个微服务自行封装一些客户端类来包装这些依赖服务的调用**。所以，Feign在此基础上做了进一步封装，由他来帮助我们定义和实现依赖服务接口的定义。在Feign的实现下，我们**只需创建一个接口并使用注解的方式**来配置它(以前是Dao接口上面标注Mapper注解,现在是一个微服务接口上面标注一个Feign注解即可)，即可完成对服务提供方的接口绑定，简化了使用Spring cloud Ribbon时，自动封装服务调用客户端的开发量。

# Feign集成了Ribbon

Feign也是运行在消费者端的，使用 Ribbon 进行负载均衡，所以 Feign 直接内置了 Ribbon。而与Ribbon不同的是，通过feign只需要定义服务绑定接口且以声明式的方法，优雅而简单的实现了服务调用。

# Feign使用步骤

1. **在服务提供者接口模块添加Feign的依赖**

|  |
| --- |
| <!--引入Feign的依赖--> <!-- https://mvnrepository.com/artifact/org.springframework.cloud/spring-cloud-starter-feign --> <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-feign</artifactId>  <version>1.4.7.RELEASE</version> </dependency> |

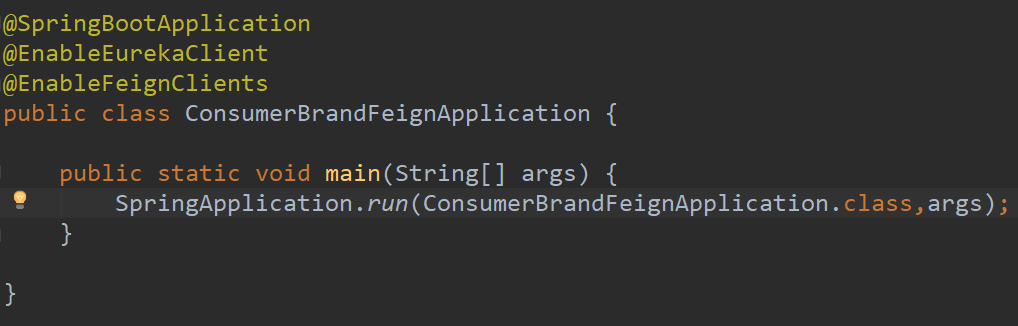
1. **在具体的服务提供者接口模块中创建接口，并在上面加上相应的注解**



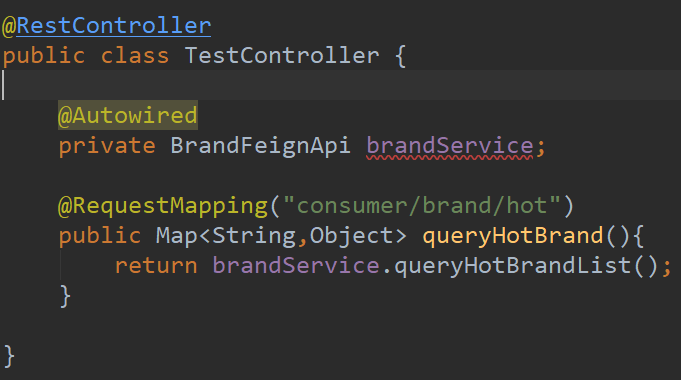
1. **在服务消费者中引入具体的服务提供者接口模块的依赖**

|  |
| --- |
| <!--引入Eureka客户端的依赖--> <dependency>  <groupId>org.springframework.cloud</groupId>  <artifactId>spring-cloud-starter-eureka</artifactId>  <version>1.4.7.RELEASE</version> </dependency> <!--引入品牌微服务提供者接口模块的依赖--> <dependency>  <groupId>com.fh</groupId>  <artifactId>shopcloud-provider-brand-api</artifactId>  <version>1.0-SNAPSHOT</version> </dependency>  <!--引入SpringBoot的Web模块的依赖--> <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-web</artifactId> </dependency> |

1. **在服务消费者的启动类上加上Feign的注解**



1. **在服务消费者项目中新建一个Controller进行测试**



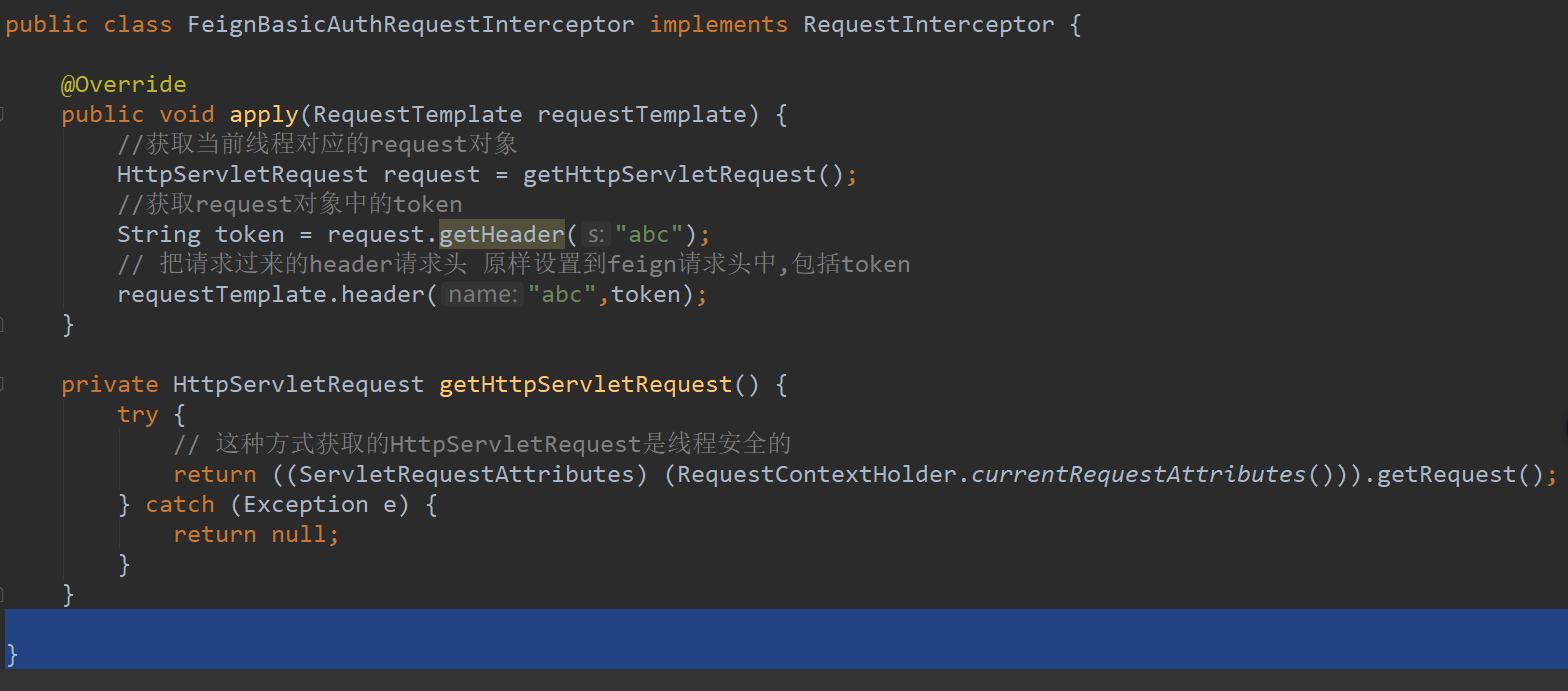
# Feign的工作原理

* 主程序入口添加了@EnableFeignClients注解开启对FeignClient扫描加载处理。根据Feign Client的开发规范，定义接口并加@FeignClient注解。
* 当程序启动时，会进行包扫描，扫描所有@FeignClients的注解的类，并且将这些信息注入Spring IOC容器中，当定义的的Feign接口中的方法被调用时，通过JDK的代理方式，来生成具体的RequestTemplate.当生成代理时，Feign会为每个接口方法创建一个RequestTemplate。当生成代理时，Feign会为每个接口方法创建一个RequestTemplate对象，该对象封装HTTP请求需要的全部信息，如请求参数名，请求方法等信息都是在这个过程中确定的。
* 然后RequestTemplate生成Request,然后把Request交给Client去处理，这里指的是Client可以是JDK原生的URLConnection,Apache的HttpClient,也可以是OKhttp，最后Client被封装到LoadBalanceClient类，这个类结合Ribbon负载均衡发送服务器之间的调用。

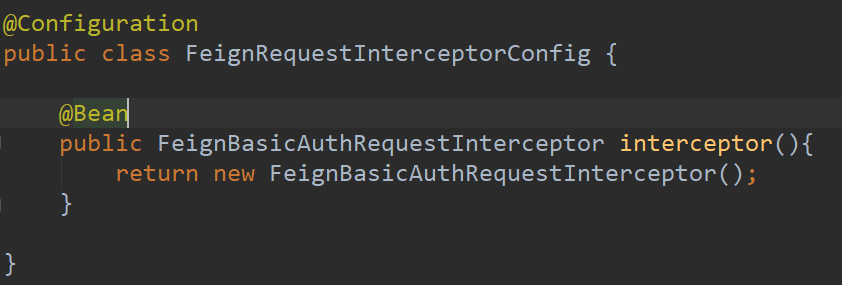
# 服务之间调用如何传递token

1. **在公共模块下新建一个Feign的拦截器类并实现RequestInterceptor的接口**

**前提条件：别忘了在公共模块下pom.xml中引入feign的配置**



1. **在服务消费者调用服务提供者需要传递token时，服务消费者项目下新建一个配置类， 让咱们在公共模块下编写的feign拦截器生效**



注意：不要在服务提供者为服务中轻易打断点，有可能会触发某种重试机制，这个重试机制暂时还不确定是由Feign导致的还是由Ribbon导致的。