1. 什么是Feign?

Feign是GitHub上的一个开源项目，目的是简化REST/WebService客户端开发。使用Feign，我们可以通过注解来修饰接口和参数，从而达到简单优雅地访问REST/WebService服务的目的。Feign不光支持自带的注解，还支持使用第三方注解，并且还支持插件式的编码器和解码器，我们可以通过这些特性，自定义编码解码规则。

Spring Cloud将Feign集成到Netflix项目中，当与Eureka、Ribbon一起使用时，Feign就具有了负载均衡的功能，它很方便在Spring Cloud中进行微服务之间的调用，因为操作非常简单，可以大大降低开发工作量。

2. 在Spring Cloud中使用Feign

*（NOTE: 需要源码的盆友可以去这个里面找哦, 找到feign-CSDN目录）*

*https://github.com/aharddreamer/chendong/tree/master/springcloud/*

本次演示的项目结构还是老三样：

eureka-server 集群服务器，供provider和invoker注册和调用

first-service-provider 服务提供端

first-service-invoker 服务调用端

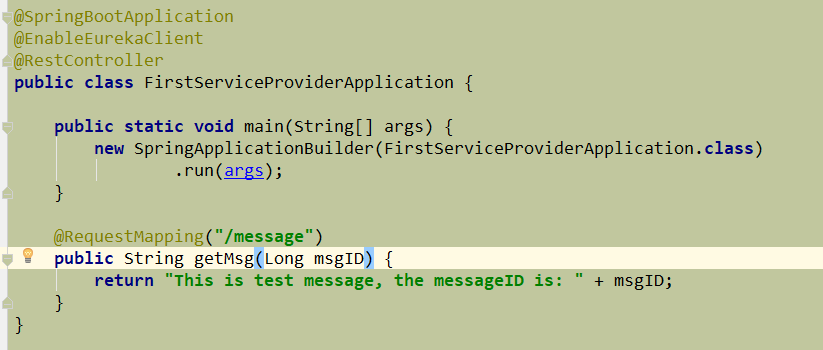
另外使用方式也和其他Spring Cloud项目很类似，添加依赖->添加配置->编写代码->测试。

first-service-provider的改动：

application.properties (注册eureka)

**spring.application.name**=**first-service-provider  
server.port**=**8080  
eureka.instance.hostname**=**localhost  
eureka.client.service-url.default-zone**=**http://localhost:8761/eureka/**

添加message接口，给Invoker模块做测试用：



first-service-invoker的改动：

1）添加依赖：

在POM.xml中加入以下依赖：

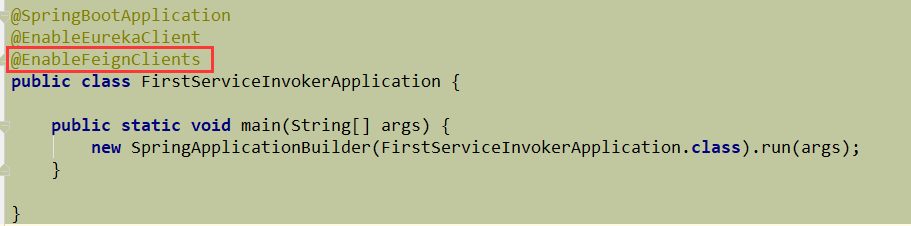
<**dependency**>  
 <**groupId**>org.springframework.cloud</**groupId**>  
 <**artifactId**>spring-cloud-starter-feign</**artifactId**>  
</**dependency**>

2）添加配置：

application.properties (注册eureka):

**server.port**=**9000  
spring.application.name**=**first-service-invoker  
eureka.instance.hostname**=**localhost  
eureka.client.service-url.default-zone**=**http://localhost:8761/eureka/**

在启动类中添加Feign的启用开关

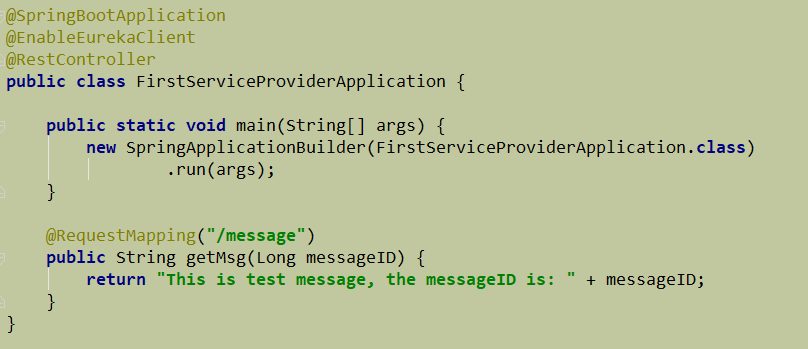


新建一个MyFeignClient, 测试通过Feign来实现微服务之间的调用：



@FeignClient里面的参数表示这个是要调用哪个微服务，下面的message方法的参数表示请求微服务first-service-provider的/message接口的messageID参数，上面的@RequestMapping大家应该非常熟悉了，就是Controller中定义接口的注解，在这里也一样好用。 是不是非常简洁？我们一会还要去provider里面建一个message接口供它调用测试。

新建一个InvokerController，我们在first-service-invoker里面写一个测试接口，一会我们通过浏览器访问这个接口，然后这个接口通过MyFeignClient去调用first-service-provider里面的message接口，从而实现微服务之间的调用。

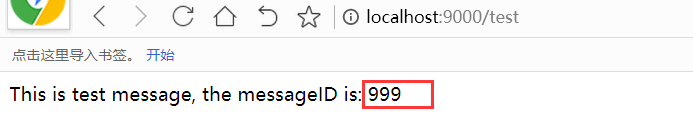


好了，依次启动eureka-server，first-service-provider, first-service-invoker。

如果运行起来有错误，说Jackson的方法或类未找到，就在pom里面加入这个依赖：

<**dependency**>  
 <**groupId**>com.fasterxml.jackson.core</**groupId**>  
 <**artifactId**>jackson-core</**artifactId**>  
 <**version**>2.9.8</**version**>  
</**dependency**>  
<**dependency**>  
 <**groupId**>com.fasterxml.jackson.core</**groupId**>  
 <**artifactId**>jackson-databind</**artifactId**>  
 <**version**>2.9.8</**version**>  
</**dependency**>

启动成功后，在浏览器输入 <http://localhost:9000/test>



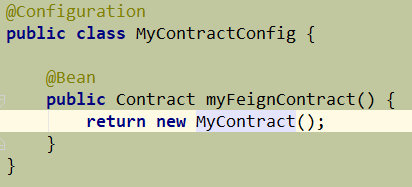
可以看到返回的messageID 999，这是刚才在test接口里面设置的测试参数。微服务之间的调用已经打通了，是不是so easy，你可以根据具体场景，加入更加复杂的请求和响应，比如在请求端使用@PathVariabe, @RequestHeader等，ContentType可以选择application/json等。在响应端可以定义复杂类型作为返回类型，同样的在请求端这里也要用同样的类型来接收，因为Feign会通过json转换工具（比如Jackson）来进行编码和解码，最终会将响应端返回的对象来填充请求端指定的对象。

有一个坑需要非常注意，响应端和请求端的接口名/参数名/返回类型最好完全一致，避免出错。返回类型里面要保证有无参的构造函数，有完整的get set方法，并且最好不要重写get set方法，否则可能会出现序列化错误的问题。

3. 自定义配置

如果需要使用自定义的Feign实现，可以配置一个Bean，这个Bean是注解翻译器，类型是Contract：

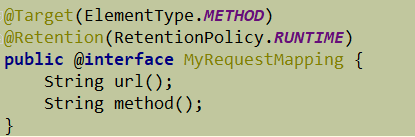
MyContractConfig.java



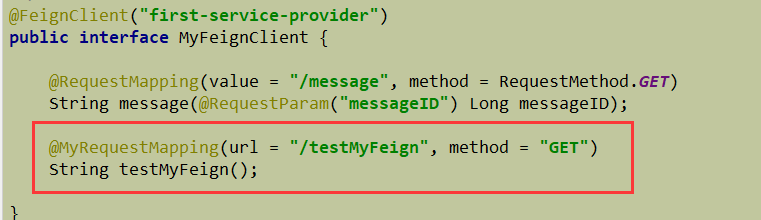
MyContract.java 注意看注释部分，这个是集成了SpringMvc的Contract，所以可以具有Spring的RequestMapping的功能，我们在控制台打印一下我们自定义的注解参数信息。



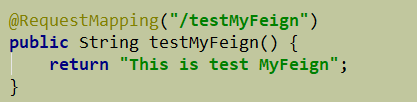
这是自定义的注解类：



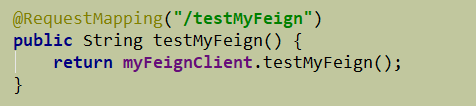
配置好了之后，在MyFeignClient里面去加一个方法，测试一下能不能通过自定义的注解去请求provider微服务里面的接口：



当然，first-service-provider微服务里面肯定要有对应的接口给它调用，所以加一个testMyFeign:

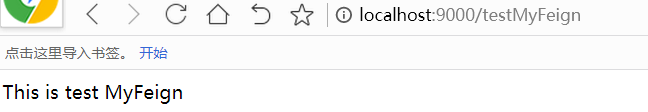


好了，在first-service-invoker的InvokerController里面再加一个测试接口, 这个是使用了我们自定义注解的FeignClient



OK，还是依次启动eureka-server, first-service-provider, first-service-invoker项目，然后在浏览器访问：

<http://localhost:9000/testMyFeign>



可以发现，自定义的注解已经起作用了。