## Spring Cloud Feign

Spring Cloud Feign是一套基于Netflix Feign实现的声明式服务调用客户端。它使得编写Web服务客户端变得更加简单。我们只需要通过创建接口并用注解来配置它既可完成对Web服务接口的绑定。它具备可插拔的注解支持，包括Feign注解、JAX-RS注解。它也支持可插拔的编码器和解码器。Spring Cloud Feign还扩展了对Spring MVC注解的支持，同时还整合了Ribbon和Eureka来提供均衡负载的HTTP客户端实现。

接下来的示例将分为三个模块：

* 1.服务接口定义模块：通过Spring MVC注解定义抽象的interface服务接口
* 服务接口实现模块：实现服务接口定义模块的interface，该模块作为服务提供者注册到eureka
* 服务接口消费模块：服务接口定义模块的客户端实现，该模块通过注册到eureka来消费服务接口

## 使用示例

1.服务接口的定义

* 创建一个Spring Boot项目：eureka-feign-api，pom.xml的主要内容如下：

<parent>

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

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

<version>1.5.6.RELEASE</version>

<relativePath/>

</parent>

<dependencies>

<dependency>

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

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

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

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

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

<version>Dalston.SR2</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

* 使用Spring MVC注解来定义服务接口：

public interface HelloService {

@GetMapping("/hello")

String hello(@RequestParam(value = "name") String name);

}

* 完成了上述构建之后，我们使用mvn install将该模块构建到本地的Maven仓库中。

2.服务接口的实现

* 创建一个Spring Boot项目：eureka-feign-client，pom.xml的主要内容如下：

<parent>

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

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

<version>1.5.6.RELEASE</version>

<relativePath/>

</parent>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>com.didispace</groupId>

<artifactId>eureka-feign-api</artifactId>

<version>1.0.0</version>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

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

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

<version>Dalston.SR2</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

该模块需要依赖上面定义的eureka-feign-api，将使用上述定义的HelloService接口来实现对应的REST服务。同时依赖Eureka是为了将该服务注册到Eureka上供服务消费者发现。

* 创建应用主类。使用[@EnableDiscoveryClient](https://github.com/EnableDiscoveryClient)注解开启服务注册与发现，并实现HelloService接口的REST服务：

@EnableDiscoveryClient

@SpringBootApplication

public class Application {

@RestController

class HelloController implements HelloService {

@Override

public String hello(String name) {

return "hello " + name;

}

}

public static void main(String[] args) {

new SpringApplicationBuilder(Application.class).web(true).run(args);

}

}

* 编辑application.properties配置内容：

spring.application.name=eureka-feign-client

server.port=2101

eureka.client.serviceUrl.defaultZone=http://eureka.didispace.com/eureka/

配置了服务提供者的名称eureka-feign-client，服务提供者的端口号2101，并将该服务注册到我的公益Eureka注册中心上。启动该项目，我们可以通过访问：<http://eureka.didispace.com/> ，在该页面中找到它。

3.服务接口的消费

* 创建一个Spring Boot项目：eureka-feign-consumer，pom.xml的主要内容如下：

<parent>

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

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

<version>1.5.6.RELEASE</version>

<relativePath/>

</parent>

<dependencies>

<dependency>

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

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

</dependency>

<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>

<dependency>

<groupId>com.didispace</groupId>

<artifactId>eureka-feign-api</artifactId>

<version>1.0.0</version>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

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

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

<version>Dalston.SR2</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

该模块较服务提供者的依赖增加了Feign的依赖，因为这里将使用Feign来绑定服务接口的客户端。下面我们将使用Feign的继承特性来轻松的构建Feign客户端。

* 创建应用主类。使用[@EnableDiscoveryClient](https://github.com/EnableDiscoveryClient)注解开启服务注册与发现，并通过[@FeignClient](https://github.com/FeignClient" \o "@FeignClient)注解来声明服务绑定客户端：

@EnableFeignClients

@EnableDiscoveryClient

@SpringBootApplication

public class Application {

@FeignClient("eureka-feign-client")

interface HelloServiceClient extends HelloService {

}

@RestController

class TestController {

@Autowired

private HelloServiceClient helloServiceClient;

@GetMapping("/test")

public String test(String name) {

return helloServiceClient.hello(name);

}

}

public static void main(String[] args) {

new SpringApplicationBuilder(Application.class).web(true).run(args);

}

}

从上述代码中我们可以看到，利用Feign的继承特性，[@FeignClient](https://github.com/FeignClient" \o "@FeignClient)注解只需要通过声明一个接口来继承在API模块中定义的公共interface就能产生服务接口的Feign客户端了。而[@FeignClient](https://github.com/FeignClient" \o "@FeignClient)中的值需要填写该服务的具体服务名（服务提供者的spring.application.name配置值）。

* 编辑服务消费者的application.properties配置内容，将服务消费者注册到eureka上来消费服务：

spring.application.name=eureka-feign-consumer

server.port=2102

eureka.client.serviceUrl.defaultZone=http://eureka.didispace.com/eureka/

* 启动eureka-feign-consumer之后，我们可以通过访问：<http://localhost:2102/test> ，来实验eureka-feign-consumer对eureka-feign-client接口的调用。