6 - OpenFeign with RestTemplate

Feign VS OpenFeign

Let's compare the similarities and differences between Feign and OpenFeign.

Same point

Feign and OpenFegin have the following similarities:

- Feign and OpenFeign are both remote calls and load balancing components under Spring Cloud.
- Feign and OpenFeign can do the same role to realize remote calls and load balancing of services.
- Both Feign and OpenFeign integrate Ribbon, use Ribbon to maintain the list of available services, and realize the load balancing of the client through Ribbon.
- Feign and OpenFeign both define the service binding interface in the service consumer (client) and configure it through annotation to realize the call of remote services.

Different points

Feign and OpenFeign have the following differences:

- Unlike the dependencies of Feign and OpenFeign, Feign's dependency is spring-cloud-starter-feign, while OpenFeign's dependency is spring-cloud-starter. -openfeign.
- Unlike annotations supported by Feign and OpenFeign, Feign supports Feign annotation and JAX-RS annotation, but not Spring MVC annotations; OpenFeign supports Feign annotations and J In addition to AX-RS annotation, Spring MVC annotation is also supported.

OpenFeign implements remote service calls

Next, let's demonstrate how to implement remote service calls through OpenFeign through an example.

1. Create a Spring Boot module called micro-service-cloud-consumer-dept-feign under DataEngineSwarm and add it to pom.xml The following dependencies.

```xml
<? xml version="1.0" encoding="UTF-8"? >
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-insta nce"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>com.luxbp</groupId>
        <artifactId>DataEngineSwarm</artifactId>
        <version>0.0.1-SNAPSHOT</version>
    </parent>
    <groupId>com.luxbp</groupId>
    <artifactId>micro-service-cloud-consumer-dept-feign</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>micro-service-cloud-consumer-dept-feign</name>
    <description>Demo project for Spring Boot</description>
    <properties>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>com.luxbp</groupId>
            <artifactId>micro-service-cloud-api</artifactId>
            <version>${project.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <!--Eureka Client Dependency-->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
        </dependency>
        <!-- Ribbon Dependency-->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-ribbon</artifactId>
        </dependency>
        <!--Add OpenFeign dependency-->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-openfeign</artifactId>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <excludes>
                        <exclude>
                            <groupId>org.projectlombok</groupId>
                            <artifactId>lombok</artifactId>
                        </exclude>
                    </excludes>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

2. Under the class path under micro-service-cloud-consumer-dept-feign (i.e. /resources directory), add an application.yml, which is configured below.

```
server:
  port: 80
eureka:
  client:
    register-with-eureka: false
    service-url:
      defaultZone: 
http://eureka7001.com:7001/eureka/,http://eureka7002.com:7002/eureka/,http://eureka7003.com:
7003/eureka/
    fetch-registry: true
```

3. Create an interface named DeptFeignService under the net.biancheng.c.service package, and bind the service interface with @FeignClient annotation on this interface, as follows.

```
package net.biancheng.c.service;
Import net.biancheng.c.entity.Dept;
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.stereotype.Component;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
Import java.util.List;
// Add as a component in the container
@Component
// The name of the service provided by the service provider, that is, application.name
@FeignClient(value = "MICROSERVICECLOUDPROVIDERDEPT")
public interface DeptFeignService {
    // Method defined in the corresponding service provider (8001, 8002, 8003) Controller
@RequestMapping(value = "/dept/get/{id}", method = RequestMethod.GET)
    public Dept get(@PathVariable("id") int id);
@RequestMapping(value = "/dept/list", method = RequestMethod.GET)
    public List<Dept> list();
}
```

When writing the service binding interface, you need to pay attention to the following 2 points:

- In the @FeignClient annotation, the value attribute is: the service name of the service provider, that is, the value of spring.application.name in the service provider profile (application.yml).

- Each method defined in the interface corresponds to the service method defined by Controller in the service provider (i.e. micro-service-cloud-provider-dept-8001, etc.).

4. Under the net.biancheng.c.controller package, create a Controller class named DeptController_Consumer, as follows.

```java
package net.biancheng.c.controller;
Import net.biancheng.c.entity.Dept;
import net.biancheng.c.service.DeptFeignService;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import javax.annotation.Resource;
Import java.util.List;
@RestController
public class DeptController_Consumer {

@Resource
    private DeptFeignService deptFeignService;
@RequestMapping(value = "/consumer/dept/get/{id}")
    public Dept get(@PathVariable("id") Integer id) {
        return deptFeignService.get(id);
    }
@RequestMapping(value = "/consumer/dept/list")
    public List<Dept> list() {
        return deptFeignService.list();
    }
}
```

5. Add @EnableFeignClients annotation to the main startup class to turn on the OpenFeign function, the code is as follows.

```java
package net.biancheng.c;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.openfeign.EnableFeignClients;
@SpringBootApplication
@EnableFeignClients // Turn on the OpenFeign function
public class MicroServiceCloudConsumerDeptFeignApplication {
    public static void main(String[] args) {
SpringApplication.run (MicroServiceCloudConsumerDeptFeignApplication.class, args);
    }
}
```

When the Spring Cloud application starts up, OpenFeign scans the interface generation agent marked with @FeignClient annotation and annotates the person into the Spring container.

6. Start the service registry cluster, service provider and micro-service-cloud-consumer-dept-feign in turn. After startup, use the browser to visit http://eureka7001.com/con sumer/dept/list",

7. Visits to http://eureka7001.com/consumer/dept/list several times in a row

Time out Control

wait to be update