Spring Cloud OpenFeign：基于 Ribbon 和 Hystrix 的声明式服务调用

原创 梦想de星空 macrozheng 2019-10-08 08:32

收录于合集
#Spring Cloud学习教程

26个

> Spring Cloud OpenFeign 是声明式的服务调用工具，它整合了 Ribbon 和 Hystrix，拥有负载均衡和服务容错功能，本文将对其用法进行详细介绍。

Feign 简介

Feign 是声明式的服务调用工具，我们只需创建一个接口并用注解的方式来配置它，就可以实现对某个服务接口的调用，简化了直接使用 RestTemplate 来调用服务接口的开发量。Feign 具备可插拔的注解支持，同时支持 Feign 注解、JAX-RS 注解及 SpringMvc 注解。当使用 Feign 时，Spring Cloud 集成了 Ribbon 和 Eureka 以提供负载均衡的服务调用及基于 Hystrix 的服务容错保护功能。

创建一个 feign-service 模块

> 这里我们创建一个 feign-service 模块来演示 feign 的常用功能。

在 pom.xml 中添加相关依赖

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-openfeign</artifactId>
```

```
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

在 **application.yml** 中进行配置

```
server:
  port:8701
spring:
  application:
    name:feign-service
eureka:
  client:
    register-with-eureka:true
    fetch-registry:true
    service-url:
      defaultZone:http://localhost:8001/eureka/
```

在启动类上添加**@EnableFeignClients** 注解来启用 **Feign** 的客户端功能

```
@EnableFeignClients
@EnableDiscoveryClient
@SpringBootApplication
publicclass FeignServiceApplication {

    public static void main(String[] args) {
        SpringApplication.run(FeignServiceApplication.class, args);
    }

}
```

添加 **UserService** 接口完成对 **user-service** 服务的接口绑定

> 我们通过@FeignClient 注解实现了一个 Feign 客户端，其中的 value 为 user-service 表示这是对 user-service 服务的接口调用客户端。我们可以回想下 user-service 中的 UserController，只需将其改为接口，保留原来的 SpringMvc 注释即可。

```
/**
 * Created by macro on 2019/9/5.
 */
@FeignClient(value = "user-service")
publicinterface UserService {
    @PostMapping("/user/create")
    CommonResult create(@RequestBody User user);

    @GetMapping("/user/{id}")
    CommonResult<User> getUser(@PathVariable Long id);

    @GetMapping("/user/getByUsername")
    CommonResult<User> getByUsername(@RequestParam String username);

    @PostMapping("/user/update")
    CommonResult update(@RequestBody User user);

    @PostMapping("/user/delete/{id}")
    CommonResult delete(@PathVariable Long id);
}
```

添加 **UserFeignController** 调用 **UserService** 实现服务调用

```
/**
 * Created by macro on 2019/8/29.
 */
@RestController
@RequestMapping("/user")
publicclass UserFeignController {
    @Autowired
    private UserService userService;

    @GetMapping("/{id}")
    public CommonResult getUser(@PathVariable Long id) {
```

```
        return userService.getUser(id);
    }

    @GetMapping("/getByUsername")
    public CommonResult getByUsername(@RequestParam String username) {
        return userService.getByUsername(username);
    }

    @PostMapping("/create")
    public CommonResult create(@RequestBody User user) {
        return userService.create(user);
    }

    @PostMapping("/update")
    public CommonResult update(@RequestBody User user) {
        return userService.update(user);
    }

    @PostMapping("/delete/{id}")
    public CommonResult delete(@PathVariable Long id) {
        return userService.delete(id);
    }
}
```

负载均衡功能演示

- 启动 eureka-service，两个 user-service，feign-service 服务，启动后注册中心显示如下：

- 多次调用http://localhost:8701/user/1进行测试，可以发现运行在 8201 和 8202 的 user-service 服务交替打印如下信息：

```
2019-10-04 15:15:34.829  INFO 9236 --- [nio-8201-exec-5] c.macro.cloud.controller.UserController
2019-10-04 15:15:35.492  INFO 9236 --- [io-8201-exec-10] c.macro.cloud.controller.UserController
2019-10-04 15:15:35.825  INFO 9236 --- [nio-8201-exec-9] c.macro.cloud.controller.UserController
```

Feign 中的服务降级

> Feign 中的服务降级使用起来非常方便，只需要为 Feign 客户端定义的接口添加一个服务降级处理的实现类即可，下面我们为 UserService 接口添加一个服务降级实现类。

添加服务降级实现类 UserFallbackService

> 需要注意的是它实现了 UserService 接口，并且对接口中的每个实现方法进行了服务降级逻辑的实现。

```
/**
 * Created by macro on 2019/9/5.
 */
@Component
publicclass UserFallbackService implements UserService {
    @Override
    public CommonResult create(User user) {
        User defaultUser = new User(-1L, "defaultUser", "123456");
        returnnew CommonResult<>(defaultUser);
    }

    @Override
    public CommonResult<User> getUser(Long id) {
        User defaultUser = new User(-1L, "defaultUser", "123456");
        returnnew CommonResult<>(defaultUser);
    }

    @Override
    public CommonResult<User> getByUsername(String username) {
        User defaultUser = new User(-1L, "defaultUser", "123456");
        returnnew CommonResult<>(defaultUser);
    }

    @Override
    public CommonResult update(User user) {
        returnnew CommonResult("调用失败，服务被降级",500);
    }
```

```
    @Override
    public CommonResult delete(Long id) {
        returnnew CommonResult("调用失败，服务被降级",500);
    }
}
```

修改 **UserService** 接口，设置服务降级处理类为 **UserFallbackService**

> 修改@FeignClient 注解中的参数，设置 fallback 为 UserFallbackService.class 即可。

```
@FeignClient(value = "user-service",fallback = UserFallbackService.class)
publicinterface UserService {
}
```

修改 **application.yml**，开启 **Hystrix** 功能

```
feign:
  hystrix:
    enabled:true#在Feign中开启Hystrix
```

服务降级功能演示

- 关闭两个 user-service 服务，重新启动 feign-service;
- 调用http://localhost:8701/user/1进行测试，可以发现返回了服务降级信息。

日志打印功能

> Feign 提供了日志打印功能，我们可以通过配置来调整日志级别，从而了解 Feign 中 Http 请求的细节。

日志级别

- NONE：默认的，不显示任何日志；
- BASIC：仅记录请求方法、URL、响应状态码及执行时间；
- HEADERS：除了 BASIC 中定义的信息之外，还有请求和响应的头信息；
- FULL：除了 HEADERS 中定义的信息之外，还有请求和响应的正文及元数据。

通过配置开启更为详细的日志

> 我们通过 java 配置来使 Feign 打印最详细的 Http 请求日志信息。

```
/**
 * Created by macro on 2019/9/5.
 */
@Configuration
publicclass FeignConfig {
    @Bean
    Logger.Level feignLoggerLevel() {
```

```
        return Logger.Level.FULL;
    }
}
```

在 **application.yml** 中配置需要开启日志的 **Feign** 客户端

> 配置 UserService 的日志级别为 debug。

```
logging:
  level:
    com.macro.cloud.service.UserService:debug
```

查看日志

> 调用http://localhost:8701/user/1进行测试，可以看到以下日志。

```
2019-10-04 15:44:03.248 DEBUG 5204 --- [-user-service-2] com.macro.cloud.service.UserService
2019-10-04 15:44:03.248 DEBUG 5204 --- [-user-service-2] com.macro.cloud.service.UserService
2019-10-04 15:44:03.257 DEBUG 5204 --- [-user-service-2] com.macro.cloud.service.UserService
2019-10-04 15:44:03.257 DEBUG 5204 --- [-user-service-2] com.macro.cloud.service.UserService
2019-10-04 15:44:03.258 DEBUG 5204 --- [-user-service-2] com.macro.cloud.service.UserService
2019-10-04 15:44:03.258 DEBUG 5204 --- [-user-service-2] com.macro.cloud.service.UserService
2019-10-04 15:44:03.258 DEBUG 5204 --- [-user-service-2] com.macro.cloud.service.UserService
2019-10-04 15:44:03.258 DEBUG 5204 --- [-user-service-2] com.macro.cloud.service.UserService
2019-10-04 15:44:03.258 DEBUG 5204 --- [-user-service-2] com.macro.cloud.service.UserService
```

Feign 的常用配置

Feign 自己的配置

```
feign:
  hystrix:
    enabled:true#在Feign中开启Hystrix
  compression:
    request:
      enabled:false#是否对请求进行GZIP压缩
      mime-types:text/xml,application/xml,application/json#指定压缩的请求数据类型
      min-request-size:2048#超过该大小的请求会被压缩
    response:
      enabled:false#是否对响应进行GZIP压缩
logging:
  level:#修改日志级别
    com.macro.cloud.service.UserService:debug
```

Feign 中的 Ribbon 配置

在 Feign 中配置 Ribbon 可以直接使用 Ribbon 的配置，具体可以参考**Spring Cloud Ribbon：负载均衡的服务调用**。

Feign 中的 Hystrix 配置

在 Feign 中配置 Hystrix 可以直接使用 Hystrix 的配置，具体可以参考**Spring Cloud Hystrix：服务容错保护**。

使用到的模块

```
springcloud-learning
├── eureka-server -- eureka注册中心
├── user-service -- 提供User对象CRUD接口的服务
└── feign-service -- feign服务调用测试服务
```

项目源码地址

https://github.com/macrozheng/springcloud-learning

推荐阅读

- 那些年，我们见过的 **Java** 服务端“问题”
- **Tomcat** 在 **SpringBoot** 中是如何启动的
- 消灭 **Java** 代码的“坏味道”
- **IDEA** 中创建和启动 **SpringBoot** 应用的正确姿势
- **IDEA** 中的 **Git** 操作，看这一篇就够了！
- 我的 **Github** 开源项目，从 **0** 到 **20000 Star**！
- **Hystrix Dashboard**：断路器执行监控
- **Spring Cloud Hystrix**：服务容错保护
- **Spring Cloud Ribbon**：负载均衡的服务调用
- **Spring Cloud Eureka**：服务注册与发现
- **SpringCloud** 整体架构概览

欢迎关注，点个在看

阅读原文

喜欢此内容的人还喜欢

项目中到底该不该用Lombok?

macrozheng