13 | OpenFeign 实战：如何实现服务间调用功能？

2022-01-10 姚秋辰

《Spring Cloud 微服务项目实战》 课程介绍 >

讲述：姚秋辰
时长 17:09 大小 15.72M

你好，我是姚秋辰。

在上一讲中，我带你了解了 OpenFeign 组件的设计目标和要解决的问题。今天我们来学习如何使用 OpenFeign 实现**跨服务的调用**，通过这节课的学习，你可以对实战项目中的 WebClient 请求做大幅度的简化，让跨服务请求就像调用本地方法一样简单。

今天我要带你改造的项目是 coupon-customer-serv 服务，因为它内部需要调用 template 和 calculation 两个服务完成自己的业务逻辑，非常适合用 Feign 来做跨服务调用的改造。

在集成 OpenFeign 组件之前，我们需要把它的依赖项 spring-cloud-starter-OpenFeign 添加到 coupon-customer-impl 子模块内的 pom.xml 文件中。

复制代码

```
<!-- OpenFeign组件 -->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-openfeign</artifactId>
</dependency>
```

在上面的代码中，你并不需要指定组件的版本号，因为我们在顶层项目中定义的 spring-cloud-dependencies 依赖项中已经定义了各个 Spring Cloud 的版本号，它们会随着 Maven 项目的继承关系传递到子模块中。

添加好依赖项之后，我们就可以进行大刀阔斧的 OpenFeign 改造了。在 coupon-customer-impl 子模块下的 CouponCustomerServiceImpl 类中，我们通过 WebClient 分别调用了 template 和 calculation 的服务。这节课我先来带你对 template 的远程调用过程进行改造，将其替换为 OpenFeign 风格的调用。

改造 Template 远程调用

通过上节课的内容我们了解到，OpenFeign 组件通过接口代理的方式发起远程调用，那么我们改造过程的第一步就是要定义一个 OpenFeign 接口。

我在 coupon-customer-impl 项目下创建了一个 package，它的路径是 com.geekbang.coupon.customer.feign。在这个路径下我定义了一个叫做 TemplateService 的 Interface，用来实现对 coupon-template-serv 的远程调用代理。我们来看一下这个接口的源代码。

复制代码

```
@FeignClient(value = "coupon-template-serv", path = "/template")
public interface TemplateService {
    // 读取优惠券
    @GetMapping("/getTemplate")
    CouponTemplateInfo getTemplate(@RequestParam("id") Long id);

    // 批量获取
    @GetMapping("/getBatch")
    Map<Long, CouponTemplateInfo> getTemplateInBatch(@RequestParam("ids") Coll
}
```

在上面的代码中，我们在接口上声明了一个 FeignClient 注解，它专门用来标记被 OpenFeign 托管的接口。

在 FeignClient 注解中声明的 value 属性是目标服务的名称，在代码中我指定了 coupon-template-serv，你需要确保这里的服务名称和 Nacos 服务器上显示的服务注册名称是一样的。

此外，FeignClient 注解中的 path 属性是一个可选项，如果你要调用的目标服务有一个统一的前置访问路径，比如 coupon-template-serv 所有接口的访问路径都以 /template 开头，那么你可以通过 path 属性来声明这个前置路径，这样一来，你就不用在每一个方法名上的注解中带上前置 Path 了。

在项目的启动阶段，OpenFeign 会查找所有被 FeignClient 注解修饰的接口，并代理该接口的所有方法调用。当我们调用接口方法的时候，OpenFeign 就会根据方法上定义的注解自动拼装 HTTP 请求路径和参数，并向目标服务发起真实调用。

因此，我们还需要在方法上定义 spring-web 注解（如 GetMapping、PostMapping），让 OpenFeign 拼装出正确的 Request URL 和请求参数。这时你要注意，**OpenFeign 接口中定义的路径和参数必须与你要调用的目标服务中的保持一致**。

完成了 Feign 接口的定义，接下来你就可以替换 CouponCustomerServiceImpl 中的业务逻辑调用了。

首先，我们在 CouponCustomerServiceImpl 接口中注入刚才定义的 TemplateService 接口。

复制代码

```
@Autowired
private TemplateService templateService;
```

被 FeignClient 注解修饰的对象，也会被添加到 Spring 上下文中。因此我们可以通过 Autowired 注入的方式来使用这些接口。

然后，我们就可以对具体的业务逻辑进行替换了。以 CouponCustomerServiceImpl 类中的 placeOrder 下单接口为例，其中有一步是调用 coupon-template-serv 获取优惠券模板数据，这个服务请求是使用 WebClient 发起的，我们来看一下改造之前的方法实现。

复制代码

```
webClientBuilder.build().get()
    .uri("http://coupon-template-serv/template/getTemplate?id=" + templateId)
    .retrieve()
    .bodyToMono(CouponTemplateInfo.class)
    .block();
```

从上面的代码中你可以看出，我们写了一大长串的代码，只为了发起一次服务请求。如果使用 **OpenFeign 接口**来替换，那画风就不一样了，我们看一下改造后的服务调用过程。

复制代码

```
templateService.getTemplate(couponInfo.getTemplateId())
```

你可以看到，使用 OpenFeign 接口发起远程调用就像使用本地服务一样简单。和 WebClient 的调用方式相比，**OpenFeign 组件不光可以提高代码可读性和可维护性，还降低了远程调用的 Coding 成本**。

在 CouponCustomerServiceImpl 类中的 findCoupon 方法里，我们调用了 coupon-template-serv 的批量查询接口获取模板信息，这个过程也可以使用 OpenFeign 接口实现，下面是具体的实现代码。

复制代码

```
// 获取这些优惠券的模板ID
List<Long> templateIds = coupons.stream()
        .map(Coupon::getTemplateId)
        .distinct()
        .collect(Collectors.toList());

// 发起请求批量查询券模板
Map<Long, CouponTemplateInfo> templateMap = templateService
        .getTemplateInBatch(templateIds);
```

到这里，我们已经把 template 服务的远程调用改成了 OpenFeign 接口调用的方式，那么接下来让我们趁热打铁，去搞定 calculation 服务的远程调用。

改造 Calculation 远程调用

首先，我们在 TemplateService 同样的目录下创建一个新的接口，名字是 CalculationService，后面你会使用它作为 coupon-calculation-serv 的代理接口。我们来看一下这个接口的源码。

复制代码

```
@FeignClient(value = "coupon-calculation-serv", path = "/calculator")
public interface CalculationService {

    // 订单结算
    @PostMapping("/checkout")
    ShoppingCart checkout(ShoppingCart settlement);

    // 优惠券试算
    @PostMapping("/simulate")
    SimulationResponse simulate(SimulationOrder simulator);
}
```

我在接口类之上声明了一个 FeignClient 注解，指向了 coupon-calculation-serv 服务，并且在 path 属性中注明了服务访问的前置路径是 /calculator。

在接口中我还定义了两个方法，分别指向 checkout 用户下单接口和 simulate 优惠券试算接口，这两个接口的访问路径和 coupon-calculation-serv 中定义的路径是一模一样的。

有了前面 template 服务的改造经验，相信你应该很轻松就能搞定 calculation 服务调用的改造。首先，我们需要把刚才定义的 CalculationService 注入到 CouponCustomerServiceImpl 中。

复制代码

```
@Autowired
private CalculationService calculationService;
```

然后，你只用在调用 coupon-calculation-serv 服务的地方，将 WebClient 调用替换成下面这种 OpenFeign 调用的方式就可以了，是不是很简单呢？

复制代码

```
// order清算
ShoppingCart checkoutInfo = calculationService.checkout(order);

// order试算
calculationService.simulate(order)
```

到这里，我们就完成了 template 和 calculation 服务调用过程的改造。在我们启动项目来验证改造成果之前，还有最为关键的一步需要完成，那就是配置 OpenFeign 的加载路径。

配置 OpenFeign 的加载路径

我们打开 coupon-customer-serv 项目的启动类，你可以通过在类名之上添加一个 EnableFeignClients 注解的方式定义 OpenFeign 接口的加载路径，你可以参考以下代码。

复制代码

```
// 省略其他无关注解
@EnableFeignClients(basePackages = {"com.geekbang"})
public class Application {

}
```

在这段代码中，我们在 EnableFeignClients 注解的 basePackages 属性中定义了一个 com.geekbang 的包名，这个注解就会告诉 OpenFeign 在启动项目的时候做一件事儿：找到所有位于 com.geekbang 包路径（包括子 package）之下使用 FeignClient 修饰的接口，然后生成相关的代理类并添加到 Spring 的上下文中。这样一来，我们才能够在项目中用 Autowired 注解注入 OpenFeign 接口。

如果你忘记声明 EnableFeignClients 注解了呢？那么启动项目的时候，你就会收到一段异常，告诉你目标服务在 Spring 上下文中未找到。我把具体的报错信息贴在了这里，你可以

参考一下。如果碰到这类启动异常，你就可以先去查看启动类上有没有定义 EnableFeignClients 注解。

```
Field templateService in com.geekbang.coupon.customer.service.CouponCustomerSe
required a bean of type 'com.geekbang.coupon.customer.feign.TemplateService' t
```

上面就是使用包路径扫描的方式来加载 FeignClient 接口。除此之外，你还可以通过直接加载指定 FeignClient 接口类的方式，或者从指定类所在的目录进行扫包的方式来加载 FeignClient 接口。我把这两种加载方式的代码写在了下面，你可以参考一下。

```
// 通过指定Client类来加载
@EnableFeignClients(clients = {TemplateService.class, CalculationService.class

// 扫描特定类所在的包路径下的FeignClient
@EnableFeignClients(basePackageClasses = {TemplateService.class})
```

在这三种加载方式中，我比较推荐你在项目中使用一劳永逸的“包路径”加载的方式。因为不管以后你添加了多少新的 FeignClient 接口，只要这些接口位于 com.geekbang 包路径之下，你就不用操心加载路径的配置。

到这里，我们就完成了 OpenFeign 的实战项目改造，你可以在本地启动项目来验证改造后的程序是否可以正常工作。

总结

现在，我们来回顾一下这节课的重点内容。今天我们使用 OpenFeign 替代了项目中的 WebClient 组件，实现了跨服务的远程调用。在这个过程中有两个重要步骤。

FeignClient：使用该注解修饰 OpenFeign 的代理接口，你需要确保接口中每个方法的寻址路径和你要调用的目标服务保持一致。除此之外，FeignClient 中指定的服务名称也要和 Nacos 服务端中的服务注册名称保持一致；

EnableFeignClients：在启动类上声明 EnableFeignClients 注解，使用本课程中学习的三种扫包方式的任意一种加载 FeignClient 接口，这样 OpenFeign 组件才能在你的

程序启动之后对 FeignClient 接口进行初始化和动态代理。

通过这节课的学习，相信你已经能够掌握 Spring Cloud 体系下的微服务远程调用的方法了。在后面的课程中，我将带你进一步了解 OpenFeign 组件的其他高级玩法。

思考题

在这节课中，我把 OpenFeign 接口定义在了调用方这一端。如果你的服务需要暴露给很多业务方使用，每个业务方都要维护一套独立的 OpenFeign 接口似乎也不太方便，你能想到什么更好的接口管理办法吗？欢迎在留言区写下自己的思考，与我一起讨论。

好啦，这节课就结束啦。欢迎你把这节课分享给更多对 Spring Cloud 感兴趣的朋友。我是姚秋辰，我们下节课再见！

上一篇 12 | OpenFeign：服务间调用组件 OpenFeign 是怎么“隔空取物”的？

精选留言 (6)

写留言

金灶沐
2022-01-10

服务提供方提取一层接口出来， 由服务提供方维护请求路径， 服务消费方，直接声明一个接口extends消费方的接口， 加上@FeignClients即可

作者回复: Bingo！同学在extends的时候加上@FeignClients的方式很好，规避了bean override的问题


tornado
2022-01-10

能讲讲feign的负载均衡么？查了一下了解的是feign集成了robbin？那robbin和LoadBalancer之间有什么关系呢？

作者回复: 在早期版本里是使用的ribbon，但由于Ribbon在最新版本里已经被剔除出局了，你会发现依赖项里已经找不到netflix组件的身影了，所以现在大家在需要负载均衡的地方就用官方的load balancer组件就可以了

共 3 条评论 >

会飞的鱼
2022-01-10

老师，这个课程啥时候可以全部更新完咧，有点着急。。。

编辑回复: 预计到3月中旬哦~


so long
2022-01-10

老师，我用spring cloud alibaba搭建了公司的一个项目，服务启动后，接口的首次请求需要2-3秒钟，后续请求都在100ms左右，请问有哪些优化措施可以提高首次接口请求速度？之前使用ribbon可是设置为饿汉式加载，但是spring cloud loadbalancer好像没有饿汉式加载的配置。

展开

作者回复: 讲真loadbalancer的功能相比ribbon是差一截的，奈何spring cloud不愿意带ribbon玩了，也没辙。ribbon规避懒加载的原因是RibbonClient在调用期才进行初始化，不过ribbon在这个过程花费的时间并不多，只会在网络环境不好的情况下超时概率有所增加。对于loadbalancer来说，实际场景下大多数公司的做法是设置connection timeout + retry的方式来解决。如果对于一致性要求高的接口，底层要注意实现幂等性以防多次调用


so long
2022-01-10

每个服务提供方单独添加一个openfeign的模块，服务调用方添加对应的openfeign模块即可

作者回复: bingo，我也推荐这种做法


Geek_e93c48

2022-01-10

关于老师的思考题：
做成将提供方的OpenFeign做成中间件抽离出来。
个人建议：老师是否可以在后边的文章中不仅仅讲技术落地，加入一些使用该技术在生产上的遇到的问题和排查思路，这些才是我们需要的（手动滑稽）

作者回复: 同学这个建议很好，专栏整体偏入门，没有加入太多线上案例分析，后面会分享一些线上的使用场景



1