Feign

学习目标

1. Feign是什么
2. 使用Feign有什么优点
3. 使用Feign的注意事项
4. 如何使用Feign

1. 什么是Feing

feign是声明式的web service客户端，它让微服务之间的调用变得更简单了，类似controller调用service。Spring Cloud集成了Ribbon和Eureka，可在使用Feign时提供负载均衡的http客户端。

feing是一个http请求的轻量级框架，可以用java接口注解的方式调用http请求，而不是像java中通过http请求报文的方式直接调用，Feign通过处理注解，将请求模板化，当实际调用的时候，传入参数，根据参数再应用到请求上，进而转化成真正的请求，这种请求相对而言比较直观。

Feign被广泛应用在Spring Cloud 的解决方案中，是学习基于Spring Cloud 微服务架构不可或缺的重要组件。

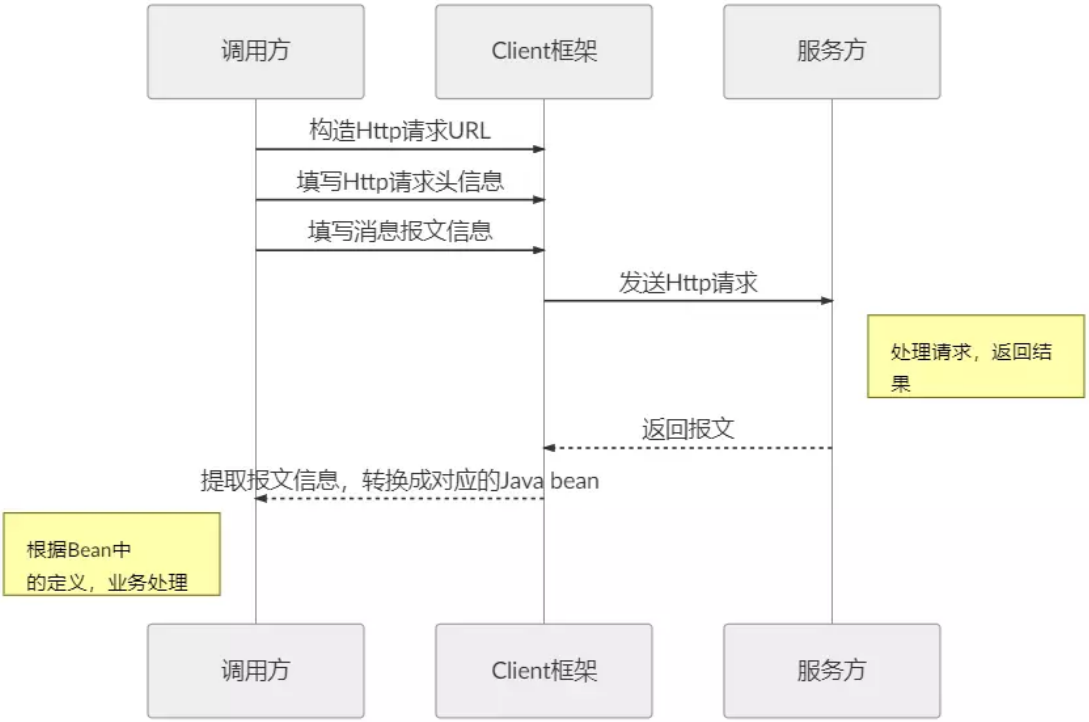
2. 使用Feign有什么优点

2.1.1

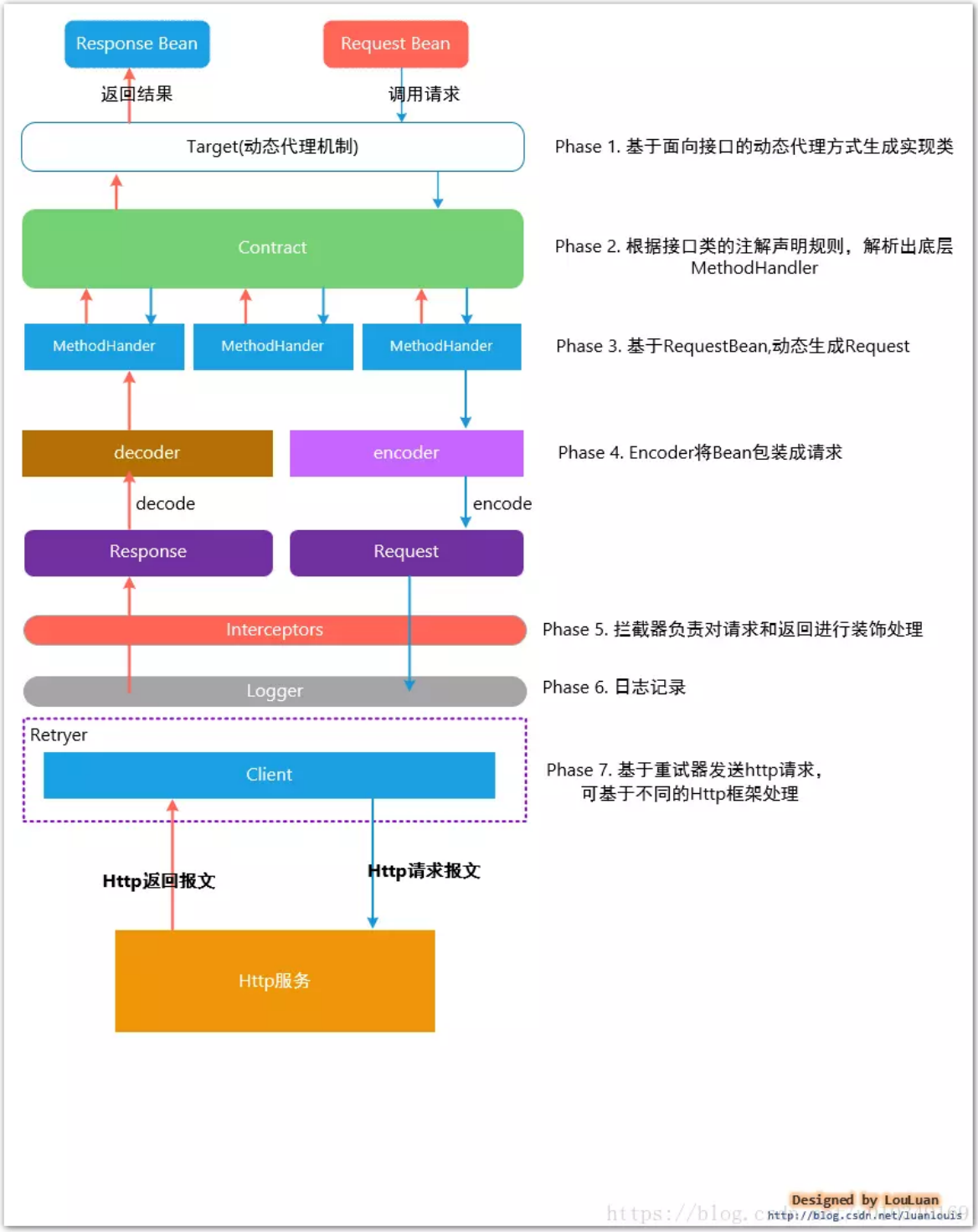
封装了http的调用流程，更适合面向接口化的编程习惯

2.1.2

在服务调用的场景过程中，我们经常需要调用基于http协议的服务，而我们经常用的框架有，HttpURLConnection、Apache HttpComponnets、OkHttp3 、Netty等等，这些框架在基于自身的专注点提供了自身特性。而从角色划分上来看，他们的职能是一致的提供Http调用服务。具体流程如下：



3. Feign是如何设计的?



Feign的性能

由于默认情况下，Feign采用的是JDK的`HttpURLConnection`,所以整体性能并不高

4. 如何使用Feign

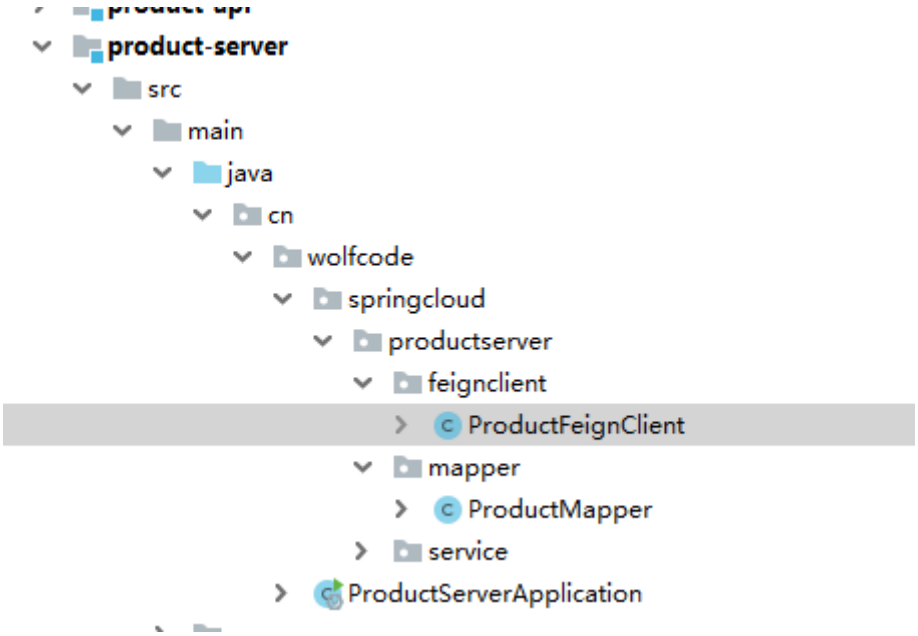
01.在product-api项目中添加openfeign依赖

```xml
<!--开启feign-->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-openfeign</artifactId>
</dependency>
```

02.在product-api项目中添加ProductFeignApi接口

```java
//配置接口  可以把这个接口看做是一个controller
            //服务的名称
@FeignClient(name = "product-server")
public interface ProductFeignApi {
                    //访问的地址
    @RequestMapping("/api/v1/product/find")
    Product find(@RequestParam("id") Long id);
}
```

03.在product-server项目中添加ProductFeignApi的实现类(本质上就是个Controller),注意要把之前的controller删除掉.




```java
@RestController
//编写一个实现类来继承api接口实现内部的方法
public class ProductFeignClient implements ProductFeignApi {
    @Autowired
    private ProductService productService;

    @Value("${server.port}")
    private String port;
    @Override
    public Product find(Long id) {
        Product product = productService.findById(id);
        Product result = new Product();
        BeanUtils.copyProperties(product, result);
```

```
        result.setName(result.getName() + port);
        return result;
    }
}
```

04.在order-server项目中的启动类上贴上@EnableFeignClients注解

```
//用于开启Feing的监听
@EnableFeignClients
```

05.把之前RestTemplate的远程调用替换成Feign方式调用即可 （注意product-api和order-server中包名的问题.）

```
@Service
@Slf4j
public class OrderServiceImpl implements OrderService {
    //    @Autowired
//    private RestTemplate restTemplate;
    @Autowired
    private ProductFeignApi productFeignApi;
    @Override
    public Order findById(Long id) {
        Product product = null;
        Order order = null;
        try {
            product = productFeignApi.find(id);
            log.info("OrderServiceImpl :" + product.toString());
```

以通过java包注入API接口的方式获取这个api的动态代理类 ， 将获取到的动态代理类通过 HttpUrlConnection 的方式进行接口访问