* 1. **基于Feign实现服务调用**

# 什么是Feign

Feign是Spring Cloud提供的一个声明式的伪Http客户端， 它使得调用远程服务就像调用本地服务一样简单， 只需要创建一个接口并添加一个注解即可。

Nacos很好的兼容了Feign， Feign默认集成了 Ribbon， 所以在Nacos下使用Fegin默认就实现了负载均衡的效果。

## Feign的使用

1. 加入Fegin的依赖

<!--fegin组件-->

<dependency>

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

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

</dependency>



1. 在主类上添加Fegin的注解

@SpringBootApplication @EnableDiscoveryClient @EnableFeignClients//开启Fegin public class OrderApplication {}

1. 创建一个service， 并使用Fegin实现微服务调用

@FeignClient("service-product")//声明调用的提供者的name public interface ProductService {

//指定调用提供者的哪个方法

//@FeignClient+@GetMapping 就是一个完整的请求路径 [http://service-](http://service-/) product/product/{pid}

@GetMapping(value = "/product/{pid}")

Product findByPid(@PathVariable("pid") Integer pid);

}



1. 修改controller代码，并启动验证

@RestController @Slf4j

public class OrderController { @Autowired

private OrderService orderService;

@Autowired

private ProductService productService;

//准备买1件商品

@GetMapping("/order/prod/{pid}")

public Order order(@PathVariable("pid") Integer pid) { log.info(">>客户下单,这时候要调用商品微服务查询商品信息");

//通过fegin调用商品微服务

Product product = productService.findByPid(pid);

log.info(">>商品信息,查询结果:" + JSON.toJSONString(product));

Order order = new Order(); order.setUid(1);

order.setUsername("测试用户");

order.setPid(product.getPid()); order.setPname(product.getPname()); order.setPprice(product.getPprice()); order.setNumber(1);

orderService.save(order);

return order;

}

}

1. 重启order微服务,查看效果