# 基本概念

1、Feign是声明式Web Service客户端，它让微服务之间的调用变得更简单，类似controller调用service。SpringCloud集成了Ribbon和Eureka，可以使用Feigin提供负载均衡的http客户端。

2、只需要创建一个接口，然后添加注解即可使用Feign。

# Feign与Open Feign的区别

## 相同点

1、[Feign](https://so.csdn.net/so/search?q=Feign&spm=1001.2101.3001.7020)和OpenFeign作用一样，都是进行远程调用的组件。

2、里面都内置了Ribbon。

3、都是加在消费端的注解，让消费端可以调用其他生产者的服务。

## 不同点

### 1、核心依赖库

Feign是Netflix开源的一个项目，而OpenFeign是Spring Cloud对Feign进行了二次封装后提供的一个模块。

一个是spring-cloud-starter-feign，一个是spring-cloud-starter-openfeign。

### 2、注解支持

在使用Feign时，我们需要使用Feign自带的注解，如@FeignClient、@RequestMapping等。而在使用OpenFeign时，可以使用Spring Cloud的注解，如@FeignClient、@GetMapping等，这样更符合Spring Cloud的开发习惯。

### 3、编码器和解码器

在Feign中，编码器和解码器是通过配置项进行配置的，而在OpenFeign中，它们是通过使用Spring Cloud的编码器和解码器来处理请求和响应的，默认使用的是Spring MVC的方式。

### 4、Hystrix支持

Hystrix是Netflix开源的一套容错框架，在处理服务调用的过程中，可以提供服务降级、熔断、限流等功能。在Feign中，需要通过@EnableCircuitBreaker注解来开启Hystrix支持，而在OpenFeign中，默认就已经集成了Hystrix，无需额外配置。

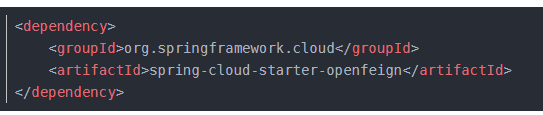
### 5、日志支持

Feign提供了自带的日志输出功能，可以通过配置项进行配置，包括NONE、BASIC、HEADERS、FULL等级别。而OpenFeign在Spring Cloud中提供了更丰富的日志输出功能，可以通过配置项进行配置，如FULL、HEADERS、BODY等级别。

# Spring Cloud OpenFeign的优缺点

## 优点

# maven依赖

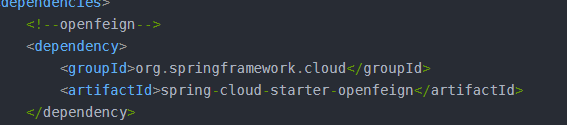


spring-cloud-starter-openfeign

# 使用方法

## 1、新建项目

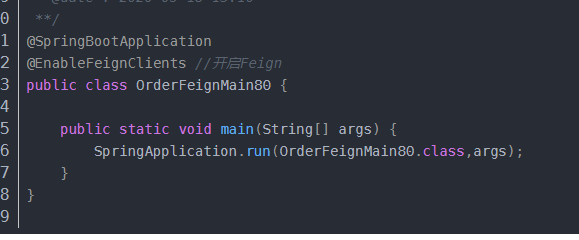
## 2、添加依赖



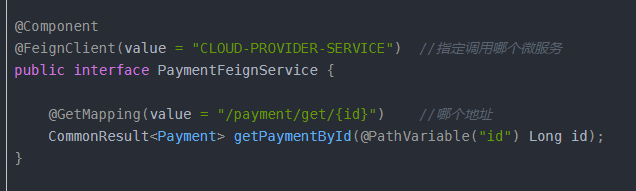
## 3、修改application.yml文件

*# Tomcat*server:  
 port: 9900  
  
*# Spring*spring:  
 application:  
 *# 应用名称* name: platform-gateway  
 profiles:  
 *# 环境配置* active: dev  
 main:  
 allow-bean-definition-overriding: true  
 cloud:  
 nacos:  
 discovery:  
 *# 服务注册地址* server-addr: platform-config-nacos:8848  
 *# 配置命名空间* namespace: dev  
 config:  
 *# 配置中心地址* server-addr: platform-config-nacos:8848  
 *# 配置命名空间* namespace: dev  
 *# 配置文件格式* file-extension: yml  
 *# 共享配置* shared-configs:  
 - application-${spring.profiles.active}.${spring.cloud.nacos.config.file-extension}  
 sentinel:  
 *# 取消控制台懒加载* eager: true  
 sentinel:  
 transport:  
 *# 指定sentinel 控制台的地址* dashboard: platform-config-nacos:8080

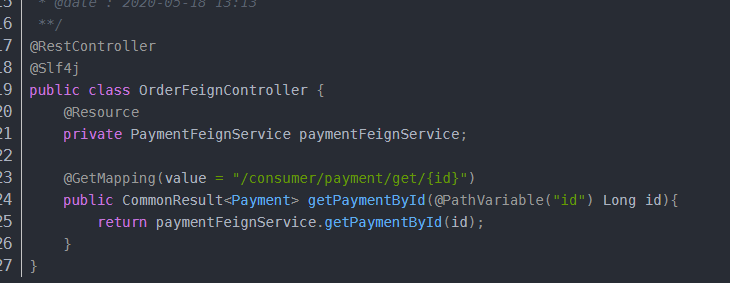
## 4、在主启动类上添加@EnableFeignClients注解



## 5、编写Feign接口



## 6、Controller层调用Feign接口



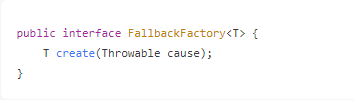
# 接口

## FallbackFactory

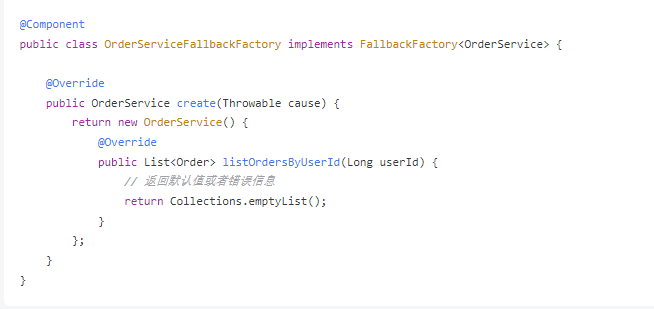
### 基本概念

在使用OpenFeign进行服务调用时，如果被调用的服务出现了异常或者超时等情况，常见的处理方法是通过定义fallback函数来实现服务降级，返回一个默认值或者错误信息。在OpenFeign中，fallback函数可以通过实现FallbackFactory接口来定义。

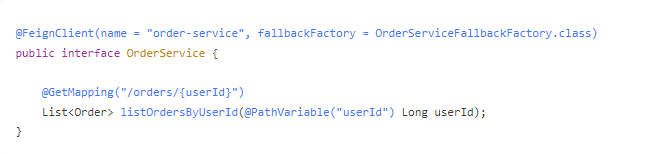
FallbackFactory接口是一个泛型接口，它有一个参数类型，表示被调用的服务接口类型。该接口包含一个create()方法，用于创建fallback实例，具体实现如下：



其中，create()方法接收一个Throwable类型的cause参数，表示异常的原因。当被调用的服务出现异常时，OpenFeign会调用create()方法来创建fallback实例，然后调用fallback实例的对应方法进行服务降级处理。FallbackFactory接口的实现类需要实现create()方法，并返回一个实现了被调用服务接口类型的fallback对象，例如：



在上述代码中，OrderServiceFallbackFactory实现了FallbackFactory<OrderService>接口，并重写了create()方法，返回一个匿名内部类实现了OrderService接口，并覆写了对应的方法，返回一个默认值或者错误信息。最后，在使用OpenFeign的时候，我们需要将FallbackFactory实例注入到对应的调用服务接口中，例如：



在上述代码中，我们通过在@FeignClient注解中指定fallbackFactory属性，将OrderServiceFallbackFactory实例注入到OrderService接口中。当被调用的服务出现异常时，OpenFeign会使用OrderServiceFallbackFactory创建fallback实例，并调用其对应的方法进行服务降级处理。总之，FallbackFactory接口提供了一种更加灵活的方式来定义fallback函数，使得我们可以在fallback函数中进行更加复杂的逻辑处理。同时，FallbackFactory还可以将异常信息传递给fallback实例，从而实现更加精细化的服务降级处理。

## RequestInterceptor

### 基本概念

RequestInterceptor 接口是 Feign 提供的一个拦截器接口，其作用是在发送请求前或响应返回后对请求进行拦截和处理。

## ErrorDecoder

# 注解

## @EnableFeignClients

### 基本概念

主要用于开启Feign客户端的功能，

### 属性

#### defaultConfiguration

## @FeignClient

### 基本概念

### 作用

通过定义接口来声明一个基于接口的远程服务客户端。它简化了与远程服务的交互过程，并且提供了服务发现、负载均衡和服务URL配置等功能。通过使用@FeignClient注解，我们可以在Spring Cloud项目中轻松地实现微服务之间的通信。

### 属性

#### contextId

用于设置该Feign客户端的上下文ID。在微服务架构中，一个应用通常由多个模块或服务组成。每个模块或服务都可能使用Feign来调用其他模块或服务提供的API。在这种情况下，我们需要区分不同的Feign客户端实例，以便能够正确地引用和配置它们。

需要注意的是，contextId属性是可选的。如果不指定该属性，Feign客户端的上下文ID将默认为客户端接口的全限定类名。

##### 作用

1、**区分不同的Feign客户端**：通过为每个Feign客户端指定唯一的contextId，可以在应用程序中区分不同的客户端实例。这样就能够使用不同的配置和属性来独立地管理和控制它们。

2、**创建独立的Spring容器上下文**：当使用contextId属性时，Feign客户端将在Spring容器中创建一个独立的上下文。这意味着Feign客户端的配置、Bean定义等将在该独立的上下文中隔离执行，而不影响其他Feign客户端或应用程序的上下文。

#### name

指定远程服务的名称，通常与服务发现和负载均衡相关。

#### url

指定远程服务的URL地址。与name二选一，用于直接指定服务的URL。

#### value

与name属性作用相同，用于指定远程服务的名称。

#### path

可选属性，用于指定远程服务的前缀路径。

#### configuration

用于指定一个配置类，该配置类可以为Feign客户端提供额外的配置和自定义行为。在开发微服务时，我们可能需要对Feign客户端进行一些自定义的配置，例如设置超时时间、添加拦截器、启用日志等。为了实现这些额外的配置，我们可以创建一个专门的配置类，并将其指定为@FeignClient注解的configuration属性。

需要注意的是，configuration属性可以指定多个配置类，以数组的形式进行设置。

#### fallbackFactory

用于指定服务调用失败时的容错处理。当调用一个服务时，如果该服务不可用或者出现了其他错误，就会触发回退机制。fallbackFactory属性所指定的类就是用来处理这个回退逻辑的，该类需要实现FallbackFactory接口，并且提供create方法，用于创建回退逻辑处理类对象。

#### fallback

#### decode404

指定是否将404状态码视为正常的响应，通常情况下，当使用Feign调用一个服务接口时，如果目标服务返回了404状态码，Feign会将其视为错误，并抛出FeignException异常。但是有些情况下，我们希望将404状态码视为正常的响应，这时就可以使用@FeignClient注解中的decode404属性来控制Feign对404状态码的处理行为。

当将decode404属性设置为false时，Feign会将404状态码视为错误，触发FeignException异常；当设置为true时，Feign会将404状态码视为正常的响应，不会触发异常。这个属性的作用在于控制Feign对于不同状态码的处理方式，从而实现定制化的错误处理逻辑。

# 类

## FeignException

# 问题

## 1、Method has too many Body parameters