狗肉是只猫 Lv1
2021年04月07日 阅读 237

关注

feign最佳实践

1. maven坐标

Feign是Netflix 公司开源产品，2018年宣布停止维护。 改由开源社区进行维护，修改了组织归属为OpenFeign，并调整代码仓库到OpenFeign/feign（可以简单理解为换了个仓库地址，换了一批人继续更新维护）。

Netflix停更一些分布式组件后，SpringCloud发布后续版本时，对依赖、maven坐标等也进行了相关调整（由spring-cloud-openfeign项目整合OpenFeign/feign到SpringCloud生态中）。

SpringCloud Finchley及之后的版本，使用spring-cloud-openfeign对应的新maven坐标，旧maven坐标废弃。

xml 复制代码

```xml
<dependency>
  <!--新坐标，SpringCloud Finchley及之后的版本必须使用该坐标-->
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-openfeign</artifactId>
</dependency>
```

SpringCloud Finchley之前的版本，即可使用新maven坐标，也可继续使用旧maven坐标

xml 复制代码

```xml
<!--新坐标，SpringCloud Finchley及之后的版本的版本必须使用该坐标-->
<dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-openfeign</artifactId>
</dependency>

<!--旧坐标，SpringCloud Finchley之前的版本，可使用新坐标，也可继续使用老坐标-->
<dependency>
```

首页 探索掘金 登录

SpringCloud Finchley之前的版本, 不管使用新坐标还是老坐标，包名都是**org.springframework.cloud.netflix.feign**，SpringCloud Finchley及之后的版本包名变更为**org.springframework.cloud.openfeign.FeignClient**，升级版本后，需要修改import路径。

java 复制代码

```java
//示例 SpringCloud Finchley 版本以前
import org.springframework.cloud.netflix.feign.EnableFeignClients;
import org.springframework.cloud.netflix.feign.FeignClient;

//示例 SpringCloud Finchley版本以后
import org.springframework.cloud.openfeign.EnableFeignClients;
import org.springframework.cloud.openfeign.FeignClient;
```

1.1 最佳实践

不管新老版本，统一使用新坐标，官方pom文件里已经声明老坐标废弃。

xml 复制代码

```xml
<artifactId>spring-cloud-starter-feign</artifactId>
<name>spring-cloud-starter-feign</name>
<!--已声明废弃-->
<description>Spring Cloud Starter Feign (deprecated, please use spring-cloud-starter-openfeign)</des
```

2. Feign启用ApacheHttpClient

Feign默认使用JDK原生的HttpURLConnection作为http客户端，支持替换为ApacheHttpClient、OkHttp。SpringCloud 2020.0.2及以后的版本，新增了对ApacheHttpClient5的支持。

建议替换为ApacheHttpClient，原因如下：

- 相比默认的HttpURLConnection，具有连接池的功能
- 可以复用http连接，较少连接创建和销毁损耗
- 相比OkHttp，ApacheHttpClient使用更加广泛，维护学习成本较低
- 具有针对单个HOST/ROUTER配置连接上线的特性，可以做到资源简单隔离的用途

spring-cloud-feign注入ApacheHttpClient实例源码：

java 复制代码

```java
@Configuration
@ConditionalOnClass(ApacheHttpClient.class)
@ConditionalOnProperty(value = "feign.httpclient.enabled", matchIfMissing = true)
class HttpClientFeignLoadBalancedConfiguration {
}
```

加载条件：

1. 能够找到ApacheHttpClient.class，
2. feign.httpclient.enabled=true，或者未配置feign.httpclient.enabled。

ApacheHttpClient是feign.Client接口的实现类，需要引入依赖：

xml 复制代码

```xml
<dependency>
        <groupId>io.github.openfeign</groupId>
        <artifactId>feign-httpclient</artifactId>
</dependency>
```

引入依赖后，Feign默认会使用HttpClient，想恢复使用HttpURLConnection或者使用其他的Http客户端，需要配置：

properties 复制代码

```properties
feign.httpclient.enabled=false
```

3. 配置ApacheHttpClient

3.1 通过参数配置

- 常用参数

配置项	含义	默认值
feign.httpclient.disableSslValidation	是否关闭https证书校验	false
feign.httpclient.maxConnections	全局最大连接数	200
feign.httpclient.maxConnectionsPerRoute	单个HOST最大连接数	50
feign.httpclient.timeToLive	连接存活时间	900
feign.httpclient.timeToLiveUnit	连接存活时间单位	TimeUnit.SECONDS
feign.httpclient.followRedirects	是否支持重定向	true
feign.httpclient.connectionTimerRepeat	定时检测连接存活情况任的调度频率	3000
feign.client.config.default.connectTimeout	连接超时时间	10000
feign.client.config.default.readTimeout	响应超时时间	60000

- 个性化调整超时时间

java 复制代码

```java
// 声明feign，指定name
@FeignClient(name = "serviceName", url = "http://127.0.0.1:7111")
```

修改对应的springCloud配置项：

properties 复制代码

```properties
feign.client.config.serviceName.connectTimeout = 300
feign.client.config.serviceName.readTimeout = 3000
```

未自定义超时时间的情况下，默认采用default对应的超时配置

properties 复制代码

```properties
feign.client.config.default.connectTimeout = 10000
feign.client.config.default.readTimeout = 60000
```

3.2 通过注入bean配置

http客户端。

源码：

```java
@Configuration
@ConditionalOnMissingBean(CloseableHttpClient.class)
protected static class HttpClientFeignConfiguration {
    @Bean
    @ConditionalOnMissingBean(HttpClientConnectionManager.class)
    public HttpClientConnectionManager connectionManager(
            ApacheHttpClientConnectionManagerFactory connectionManagerFactory,
            FeignHttpClientProperties httpClientProperties) {

    }

    @Bean
    @ConditionalOnProperty(value = "feign.compression.response.enabled", havingValue = "false", matc
    public CloseableHttpClient httpClient(
            ApacheHttpClientFactory httpClientFactory,
            HttpClientConnectionManager httpClientConnectionManager,
            FeignHttpClientProperties httpClientProperties) {
        this.httpClient = createClient(httpClientFactory.createBuilder(), httpClientConnectionManage
        return this.httpClient;
    }
}
```

目前Feign提供的配置，已经能满足大部分场景的需要，一般不需要通过注入对象的方式来修改默认配置，**如果你通过Java Config覆盖默认ApacheHttpClient，一定要创建定时器来检测无用连接**

3.3 最佳实践

```properties
#设置通用client连接超时为100毫秒，连接时间不宜过长，防止依赖服务负载过高情况下活跃连接都在长时间尝试建立连接，
feign.client.config.default.connectTimeout = 100

#设置通用client响应超时为1秒，单个接口响应时间不宜过长，建议为1秒，超过1秒的一般都需要优化接口，如果无法优化
feign.client.config.default.readTimeout = 1000

#设置连接存活时间为900秒，超过该时间后空闲连接会被回收，注意的是如果你通过Java Config覆盖默认ApacheHttpClie
feign.httpclient.timeToLive = 900

#设置全局最大连接数为300个连接，可根据具体有多少FeignClient来决定，比如一个HOST最多50个连接，一个有8个HOST，
```

首页 探索掘金 登录

```
#设置单个HOST最大连接数为50个，可根据高峰期调用频率来调整
feign.httpclient.maxConnectionsPerRoute = 50
```

单节点下n秒钟可发起的最大【依赖接口调用次数】计算方式如下：

> 最大连接数 * （n * 1000 / 接口平均响应时间（ms）） = n秒钟内最大接口调用能力

以上述配置为例，最差情况下接口平均响应时间为（readTimeout + connectTimeout）ms，1s内接口最大调用次数为：

> 300 * (1 * 1000 / 1100) = 272 (次)

针对单HOST最大接口调用次数为：

> 50 * (1 * 1000 / 1100) = 45 (次)

全局最大连接数要根据HOST总数、单HOST最大连接数、调用频率来调整，一般取折中值即可。

> 全局最大连接数 <= 单个HOST最大连接数 * HOST总数 <= tomcat最大连接数

4.服务保护

spring-cloud-feign中集成了hystrix，hystrix虽然已不再维护，但应用非常广泛，相比sentinel，sentinel是一个分布式的容错限流框架，上手容易但深入比较难，还需要部署控制台来支持，而resilience4j还不是很完善，应用不广泛，社区并不太活跃，维护成本相对来说比较高，再未充分使用过sentinel和resilience4j的情况下我们还是继续使用hystrix

开启方式：

properties 复制代码

```
feign.hystrix.enabled=true
```

4.1 hystrix的资源隔离

线程池隔离：

- 维护一个线程池用于对一组接口的调用，以feignName为隔离粒度
- 按照feignName进行分组，一个feignName对应一个线程池
- 超时或者流量较大时，会触发熔断，进行降级处理。

1. 线程池隔离策略配置

properties 复制代码

```properties
#设置资源隔离策略为线程池隔离，默认策略
hystrix.command.default.execution.isolation.strategy = THREAD
#设置开启线程执行命令超时限制
hystrix.command.default.execution.timeout.enabled = true
#配置隔离线程执行命令超时时间为10秒
hystrix.command.default.execution.isolation.thread.timeoutInMilliseconds = 10000
#设置当线程执行命令超时时是否中断命令线程
hystrix.command.default.execution.isolation.thread.interruptOnTimeout = true
#降级最多调用次数，超过后，不会调用降级，会抛出异常，对THREAD和SEMAPHORE都生效
hystrix.command.default.fallback.isolation.semaphore.maxConcurrentRequests = 100
```

2. 线程池配置

properties 复制代码

```properties
#设置线程池核心线程数大小为10个
hystrix.threadpool.default.coreSize = 10
#设置线程池最大线程数为50个
hystrix.threadpool.default.maximumSize = 50
#设置是否允许动态调整线程数
hystrix.threadpool.default.allowMaximumSizeToDivergeFromCoreSize = true
#启用等待队列，应对突发流量，上限为50，修改该属性需要重启
hystrix.threadpool.default.maxQueueSize = 50
#动态队列阀值，变相实现实时调整队列大小
hystrix.threadpool.default.queueSizeRejectionThreshold = 10
#设置线程空闲时间为1分钟
hystrix.threadpool.default.keepAliveTimeMinutes = 1
```

3. 个性化配置

线程池隔离方式支持以feignName进行隔离，示例：

java 复制代码

```java
// 声明feign，指定name
@FeignClient(name = "serviceName", url = "http://127.0.0.1:7111")
```

修改对应的springCloud配置项：

首页 探索掘金 登录

properties 复制代码

```properties
#设置线程池核心线程数大小为10个
hystrix.threadpool.serviceName.coreSize = 10
#设置线程池最大线程数为50个
hystrix.threadpool.serviceName.maximumSize = 50
```

> 如何计算线程池数量？ 公式：高峰期每秒的请求数量 / 1000毫秒 / TP99请求延时 + buffer空间

比如说处理一个请求，要50ms，那么TP99，也就是99%的请求里处理一个请求耗时最长是50ms。我们给一点缓冲空间10ms，那就是处理请求接口耗时60ms。 所以一秒钟一个线程可以处理：1000 / 60 = 16，一个线程一秒钟可以处理16个请求。 假设高峰期，每秒最多1200个请求，一个线程每秒可以处理16个请求，需要多少个线程才能处理每秒1200个请求呢？1200 / 16 = 75，最多需要75个线程，每个线程每秒处理16个请求，75个线程每秒才可以处理1200个请求。 线程数不建议超过tomcat线程数，超过的话配置无实际意义，最大并发数受tomcat处理能力限制。

4.1.2 信号量隔离

信号量隔离：

- 一个接口对应一个信号量，信号量控制接口并发数量，以接口为隔离粒度
- 超时或者流量较大时，会触发熔断，进行降级处理

1. 通用配置

properties 复制代码

```properties
#配置资源隔离策略为信号量隔离
hystrix.command.default.execution.isolation.strategy = SEMAPHORE
#设置单个命令最大并发数为50
hystrix.command.default.execution.isolation.semaphore.maxConcurrentRequests = 50
#降级最多调用次数，超过后，不会调用降级，会抛出异常，对THREAD和SEMAPHORE都生效，默认为10
hystrix.command.default.fallback.isolation.semaphore.maxConcurrentRequests = 100
```

2. 个性化配置

properties 复制代码

```properties
#配置资源隔离策略为信号量隔离
hystrix.command.<HystrixCommandKey>.execution.isolation.strategy = SEMAPHORE
#设置单个命令最大并发数为1000
hystrix.command.<HystrixCommandKey>.execution.isolation.semaphore.maxConcurrentRequests = 1000
```

首页 探索掘金 登录

类名#方法名(参数类型) 示例：RemoteProductService#getProduct(int)， 定义API时尽量不要命名相同短名的API，比如package不一样但类名一样，避免影响隔离粒度。

4.1.3 两种隔离方式对比

	是否有线程切换	是否支持异步	是否支持超时	是否支持熔断
信号量	否	否	否	是
线程池	是	是	是	是

信号量的优点：

- 信号量使用的仍是请求线程自身，同步调用

线程池的优点：

- 异步调用，线程池接管了真实的请求，完全隔离了第三方代码
- 支持超时策略，中断请求。

线程池的缺点：

- 线程的调度与上下文切换耗费一定的资源
- 对于依赖线程状态的代码增加了复杂性。

4.2 开启断路器

断路器：熔断条件达到一定比例，触发断路，服务短暂不可调用（CLOSED），一段时间后（默认5s），放入部分流量（HALF-OPEN），如果服务恢复正常，则重新恢复服务（OPEN）。

通用配置：

properties 复制代码

```
#开启断路器
hystrix.command.default.circuitBreaker.enabled = true
#设置达到熔断条件的滑动窗口内最小请求量为20个
hystrix.command.default.circuitBreaker.requestVolumeThreshold = 20
#设置熔断后休眠窗口时间为5秒
hystrix.command.default.circuitBreaker.sleepWindowInMilliseconds = 5
```

首页 探索掘金 登录

```
#关闭缓存
hystrix.command.default.requestCache.enabled = false
```

个性化配置：

properties 复制代码

```
hystrix.command.<HystrixCommandKey>.circuitBreaker.enabled = true
```

4.3 服务降级

feign支持当熔断器被触发或者调用异常时，进行降级处理。

4.3.1 fallback

example：

java 复制代码

```
@FeignClient(name = "serviceName", url = "http://127.0.0.1:7111",
        // 配置降级回调处理类
        fallback = HelloClientApplication.HelloClientFallBack.class
)
interface HelloClient {
    @RequestMapping("/{name}")
    String hello(@PathVariable(value = "name") String name);
}

// 降级处理类实现
@Component
class HelloClientFallBack implements HelloClient {
    @Override
    public String hello(String name) {
        return "you get an error callback";
    }
}
```

4.3.2 fallbackFactory

如果想获取到更详细的异常信息，可以使用fallbackFactory的方式进行降级处理

example：

iava 复制代码

首页 探索掘金 登录

```
        fallbackFactory = HelloClientApplication.HelloClientFallBackFactory.class
)
interface HelloClient {
    @RequestMapping("/{name}")
    String hello(@PathVariable(value = "name") String name);
}

@Component
static class HelloClientFallBackFactory implements FallbackFactory<HelloClientApplication.HelloClier

    @Override
    public HelloClientApplication.HelloClient create(Throwable throwable) {
        return new HelloClientApplication.HelloClient() {
            @Override
            public String hello(String name) {
                return "you get an error, is: " + throwable.getMessage();
            }
        };
    }
}
```

4.4 最佳实践

4.4.1 通用配置

properties 复制代码

```
#开启断路器
feign.hystrix.enabled=true

#配置资源隔离策略为信号量隔离，使用信号量隔离，可达到快速失败的作用，使用线程池隔离主要是为了借用线程池的队列
hystrix.command.default.execution.isolation.strategy = SEMAPHORE

#设置单个命令最大并发数为50,设置并发数的时候需要考虑到tomcat的连接池大小，一般总数不应超过tomcat的连接池最大
hystrix.command.default.execution.isolation.semaphore.maxConcurrentRequests = 50

#开启断路器
hystrix.command.default.circuitBreaker.enabled = true

#设置达到熔断条件的滑动窗口内最小请求量为20个
hystrix.command.default.circuitBreaker.requestVolumeThreshold = 20

#设置熔断后休眠窗口时间为5秒
hystrix.command.default.circuitBreaker.sleepWindowInMilliseconds = 5
```

首页 探索掘金 登录

4.4.2 强制打开/关闭中断

当某个依赖方服务异常时，熔断器未正常开启，可强制打开熔断，阻塞掉所有的请求

```properties
hystrix.command.<HystrixCommandKey>.circuitBreaker.forceOpen = true
```

properties 复制代码

当熔断器被触发，但确认依赖方服务正常时，可强制关闭熔断

```properties
hystrix.command.<HystrixCommandKey>.circuitBreaker.forceClosed = false
```

properties 复制代码

附1：相关配置项的默认值

配置项	含义	默认值
feign.client.config.default.connectTimeout	连接超时时间	10000
feign.client.config.default.readTimeout	响应超时时间	60000
feign.httpclient.timeToLive	连接存活时间	900
feign.httpclient.maxConnetions	全局最大连接数	200
feign.httpclient.maxConnectionsPerRoute	单Host最大连接数	50
feign.httpclient.enabled	是否启用httpclient	true
feign.hystrix.enabled	是否启用hystrix	False
hystrix.command.default.execution.isolation.strategy	隔离策略	THREAD
hystrix.command.default.execution.isolation.semaphore.maxConcurrentRequests	信号量隔离策略，最大并发量	10
hystrix.command.default.circuitBreaker.enabled	是否开启断路器功能	true

首页 探索掘金 登录

hystrix.command.default.metrics.rollingStats.numBuckets	设置统计滚动窗口的桶数量	10
hystrix.command.default.circuitBreaker.requestVolumeThreshold	设置滚动窗口内将使断路器跳闸的最小请求数量	20
hystrix.command.default.circuitBreaker.errorThresholdPercentage	设置失败百分比的阈值	50
hystrix.command.default.circuitBreaker.sleepWindowInMilliseconds	断路器打开后，拒绝访问时间	5000
hystrix.command.default.circuitBreaker.forceOpen	强制打开断路器	false
hystrix.command.default.circuitBreaker.forceClosed	强制关闭断路器	false
hystrix.command.default.fallback.enabled	是否打开降级	true
hystrix.command.default.requestCache.enabled	是否开启请求缓存	true

附2：最佳实践配置汇总

properties 复制代码

```properties
feign.client.config.default.connectTimeout = 100
feign.client.config.default.readTimeout = 1000
feign.httpclient.timeToLive = 900
feign.httpclient.maxConnetions = 300
feign.httpclient.maxConnectionsPerRoute = 50

feign.hystrix.enabled = true
hystrix.command.default.execution.isolation.strategy = SEMAPHORE
hystrix.command.default.execution.isolation.semaphore.maxConcurrentRequests = 50

hystrix.command.default.circuitBreaker.enabled = true
hystrix.command.default.metrics.rollingStats.timeInMilliseconds = 10000
hystrix.command.default.metrics.rollingStats.numBuckets = 10
hystrix.command.default.circuitBreaker.requestVolumeThreshold = 20
hystrix.command.default.circuitBreaker.errorThresholdPercentage = 80
hystrix.command.default.circuitBreaker.sleepWindowInMilliseconds = 5
```

首页 探索掘金 登录

文章分类 后端 文章标签 Spring Cloud

狗肉是只猫 Lv1
java 开发工程师 @ xdja
获得点赞 13 · 获得阅读 4,434

关注

安装掘金浏览器插件
多内容聚合浏览、多引擎快捷搜索、多工具便捷提效、多模式随心畅享，你想要的，这里都有！
前往安装

输入评论（Enter换行，Ctrl + Enter发送）

发表评论

全部评论（1） 最新 最热

稀土君 Lv6 最酷的 @ 稀土 4月前
你的作品数据较昨天有所变化，赶紧来创作者中心看一下；现在体验专栏新功能还有大额京东卡拿，点击查看：sourl.co ； app用户请升级最新版本应用后在我的页面查看哟！
点赞 回复

相关推荐

MacroZheng 20天前 Java Spring Cloud

开箱即用！看看人家的微服务权限解决方案，那叫一个优雅！

记得之前写过一篇文章《微服务权限终极解决方案，Spring Cloud Gateway + Oauth2 实...

4127 74 7

牵牛 20天前 Spring Cloud

Spring Cloud OpenFeign

简介 一开始叫 Feign ， 随着Netflix Feign停止开源，Spring Cloud团队在其基础上开发出来的声明式服务调用组件...

干货满满张哈希 1年前 Spring Cloud

Spring Cloud升级之路 - Hoxton - 4. 使用Resilience4j

由于我们的入口注解类从@SpringCloudApplication替换成了SpringBootApplication，这样不会启用Spring-...

1928 5 1

XiaoLin_Java 29天前 分布式 Spring Cloud 后端

SpringCloudAlibaba全网最全讲解[1]（建议收藏）

简而言之，微服务架构风格是一种将单个应用程序开发为"一套小型服务"的方法，每个服务...

1875 56 1

六脉神剑 1月前 后端 Spring Cloud

SpringCloud原理之feign

Feign是一种声明式、模板化的HTTP客户端(仅在Application Client中使用)。声明式调用是指，就像调用本地方法...

393 5 评论

yanghx 1月前 后端 微信

使用feign对接微信支付v3

使用feign对接微信支付v3 先来个示例 就像写controller一样向微信发送请求。帮你把签名之类的工作全给做了。 用...

179 2 1

zust_Isabella 1月前 Spring Cloud 后端

服务调用Ribbon、OpenFeign

Ribbon 一、概述 Spring Cloud Ribbon是基于Netflix Ribbon实现的一套客户端负载均衡的工具。 简单的说，...

365 24 2

llsydn 1月前 后端 Spring Cloud

spring cloud使用zookeeper作为服务注册中心和配置中心

现在用spring cloud框架搭建微服务已经是比较流行了，毕竟这个是spring提供给我们的一个框架，可以很好配合...

8164 7 评论

西红柿蛋炒饭 2月前 Java Spring Cloud

记一次 Feign 的坑

事情是这样的，最近在代码中需要使用 Feign 调用第三方服务。但是有个 Jar 包下的服务无法注入容器.....

1849 33 7

首页 探索掘金 登录

Feign使用分析

feign使用 在实现的效果上来说Feign = RestTemplate+Ribbon+Hystrix ### Feign实现RestTemplate+Ribbo...

129 点赞 评论

阿鉴 2月前 Spring Cloud 后端

Feign实战技巧篇

Feign在项目中的正确打开方式，90%人不知道的高阶用法，自定义解码器， 如何打印curl...

76 点赞 评论

Rubble 2月前 后端

纳尼？feign报400?

本文已参与好文召集令活动，点击查看：后端、大前端双赛道投稿，2万元奖池等你挑战！ ...

247 11 评论

阿鉴 2月前 Java 后端

Feign远程调用

有关微服务中，服务与服务如何通信，我已经给大家介绍了Ribbon远程调用的相关知识，不知道大家有没有发现...

131 点赞 评论

mghio 2月前 后端

Spring Cloud 整合 Feign 的原理

前言 在 上篇 介绍了 Feign 的核心实现原理，在文末也提到了会再介绍其和 Spring Cloud ...

150 2 评论

mghio 2月前 后端

聊聊 Feign 的实现原理

Feign 是一个 HTTP 请求的轻量级客户端框架。通过 接口 + 注解的方式发起 HTTP 请求调...

818 4 评论

gentten 3月前 源码

openfeign源码阅读

Spring cloud openfeign它是一个基于feign的Http请求框架，它可以集成Sentinel或者...

286 2 评论

卷几你哇 3月前 后端

472 2 评论

LoveLifeSuper 3月前 Spring Cloud 后端

Spring Cloud OpenFeign进阶实战

上一篇：Spring Cloud OpenFeign基础入门 OpenFeign实战 替换默认的Client Feign默...

357 3 评论

LoveLifeSuper 3月前 Spring Cloud 后端

Spring Cloud OpenFeign基础入门

OpenFeign是什么 Feign是一个声明式的Web Service客户端，是一种声明式、模板化的...

339 2 评论

maojunhat 3月前 Spring Boot 后端

使用Eureka + Feign搭建分布式微服务

使用Spring Boot + Eureka + Feign搭建分布式微服务，分离服务提供者、消费者和注册中心。

413 5 评论