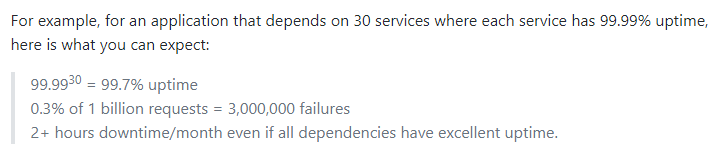
# Spring Cloud断路器Hystrix

上文讲到我们服务间调用使用Feign——声明式Web服务客户端，在分布式系统中，一个服务很可能会调用多个其他微服务，而在调用过程中很容易因为某些原因（网络连接慢，资源紧张，阻塞等）导致某个微服务无法正常提供服务，而作为调用方应该有一种容错机制，来隔离这种异常，从而不让一个微服务的异常影响整体的运行。

Hystrix官网举的例子：



大意是1个系统依赖30个微服务，假设每个微服务可用时间是99.99%，换算成整个系统就是99.99的30次方约99.7%可用时间，0.3%不可用。

1亿的请求有30万失败。30天有2个多小时不能正常提供服务。

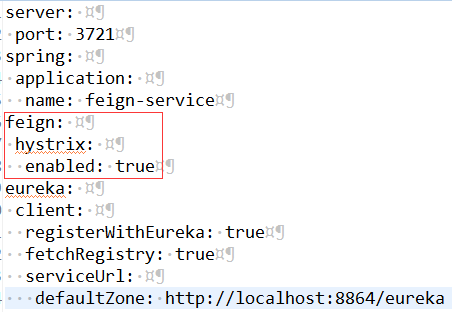
现实通常比这更糟糕。

我举个栗子：假设我们在电商网站买东西，从下单到完成要依赖三个微服务：订单、支付、物流。遇上双11或618，客户的订单蜂拥而至，这时候物流微服务可能因为程序优化不好或者硬件资源不够而不能及时提供服务，如果因为物流的问题而导致客户无法下单，这到手的钱挣不到了就很可惜了。

这时候就可以对物流微服务进行熔断，暂时不调用，全力以赴把订单、支付给完成了。熬过去这段峰值，回过头来再处理物流的问题，反正钱已到手。不会因为一颗老鼠屎坏一锅汤。

Spring Cloud的断路器是Hystrix，两种使用方法，一种是直接使用Hystrix，一种是通过FeignClient。本文介绍第二种。

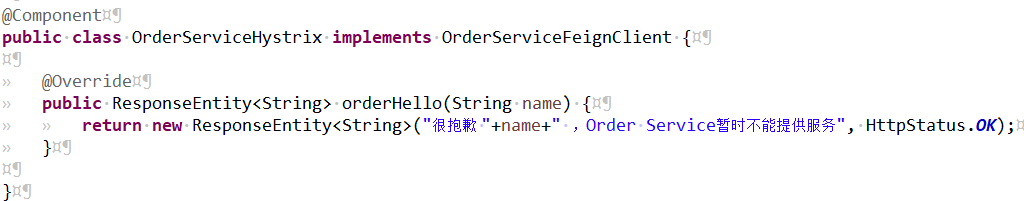
application.yml：



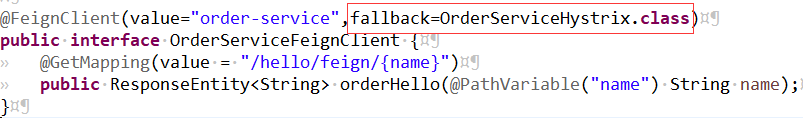
FeignClient自带Hystrix，在配置中启用就好。

代码：

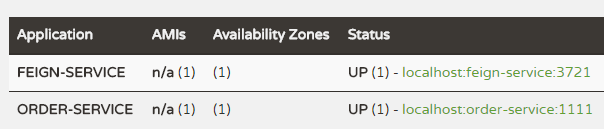
新建一个Fallback处理类，实现FeignClient接口：



修改FeignClient接口，指定Fallback处理类：



依次启动Eureka注册中心、Order Service、Feign Service：



正常访问如下图，调用没问题：



关掉Order Service，熔断机制启动，结果如下图：

