Eureka服务注册中心module

1. 假如需要引入cloud的一个新技术组件

基本上两步，

1.1新增一个相关mvn坐标

1.2在主启动类标注的启动该新技术的相关注解标签

1.3java业务逻辑编码

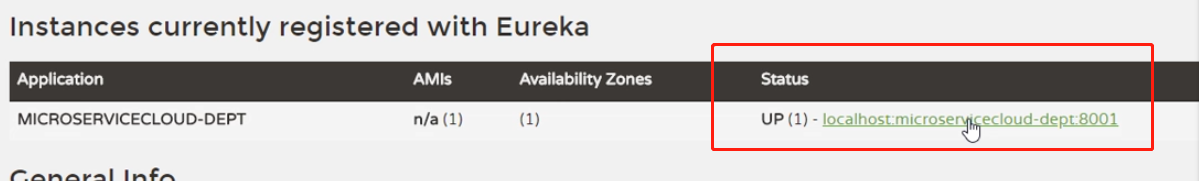
**EMERGENCY! EUREKA MAY BE INCORRECTLY CLAIMING INSTANCES ARE UP WHEN THEY'RE NOT. RENEWALS ARE LESSER THAN THRESHOLD AND HENCE THE INSTANCES ARE NOT BEING EXPIRED JUST TO BE SAFE.**

服务注册后会出现红字，在这里解释：eureka自我保护机制（常考很常见，一段时间没有访问微服务没有心跳就会出现这个），修改配置后也会出现这个故障

什么是自我保护机制：一种应对网络异常的安全保护措施。

某时刻某个微服务不能用了，eureka不会立刻清理，依旧会对该服务的信息进行保存。设计哲学是：宁可保留错误的服务注册信息，也不盲目注销任何可能健康的服务实例

可以关闭自我保护机制，也可以改动等待时间，但不推荐这么做



为服务的id，可以改成别名，方便调用，这个链接可以配置相应的说明页面，对此服务进行说明

Feign：负载均衡，实现了web service客户端负载均衡，使用feign能让编写webservice客户端更加简单，它的使用方法是定义一个借口，在上面添加注解，同时支持jax-rs标准注解，feign可以与eureka和ribbon组合使用以支持负载均衡

Ribbon实现负载均衡，功能很轻大，甚至可以自己定义算法

Feign：直接调用微服务进行访问，面向借口变成，webservice接口，dao接口

通过微服务名称获得调用地址

通过接口+注解，获得调用服务

适应社区其他程序员提出的，面向接口编程的套路，服务接口上打注解

Feign集成了ribbon

利用ribbon维护了microservicecloud-dept的服务列表信息，并且通过轮询实现了客户端的负载均衡，而与ribbon不同的是，通过feign只需要定义服务绑定接口且以声明式方法，优雅而简单实现了服务调用

Hystrix断路器

程序出异常，长时间没有回应，要避免全局系统瘫痪挂起死机。复杂的分布式系统有十来个依赖关系，每一个依赖关系在某些时候会不可避免的失败。

多个微服务之间调用的时候，假设微服务A调用微服务B和微服务C，微服务B和微服务C又调用其他的微服务资源，所谓“扇出”，如果扇出的链路上某个微服务的调用相应时间过长或者不可用，对微服务A的调用就会占用越来越多的资源导致系统崩溃，所谓“雪崩效应”

Hystrix专门处理分布式系统的延迟和容错的开源库。保证在一个依赖出问题情况下，不会导致整体服务失败，避免级联故障

服务降级：

在客户端实现完成，与服务端没有关系；当服务端provider已经down了，但是我们做了服务降级处理，让客户端在服务端不可用时也能得到返回结果，此时可释放资源，避免累计占用过多系统资源