自我保护机制

心跳检测机制有一定的不确定性，比如服务提供者可能是正常的，但是由于网络通信的问题，导致在90s内没有收到心跳请求，那将会导致健康的服务被误杀。

为了避免这种问题，Eureka提供了一种叫自我保护机制的东西。简单来说，就是开启自我保护机制后，Eureka Server会包这些服务实例保护起来，避免过期导致实例被剔除的问题，从而保证Eurreka集群更加健壮和稳定。

进入自我保护状态后，会出现以下几种情况

- Eureka Server不再从注册列表中移除因为长时间没有收到心跳而应该剔除的过期服务，如果在保护期内如果服务刚好这个服务提供者非正常下线了，此时服务消费者就会拿到一个无效的服务实例，此时会调用失败，对于这个问题需要服务消费者端要有一些容错机制，如重试，断路器等！
- Eureka Server仍然能够接受新服务的注册和查询请求，但是不会被同步到其他节点上，保证当前节点依然可用。

Eureka自我保护机制，通过配置 eureka.server.enable-self-preservation 来【true】打开/【false禁用】自我保护机制，默认打开状态，建议生产环境打开此配置。

什么情况下会触发自我保护(打开自我保护的开关)

如果， 超过**85%**的客户端节点都没有发起正常的心跳（renew），那么Eureka Server会认为客户端与服务端出现了网络故障，是的Eureka进入到自我保护状态！

eureka.server.renewal-percent-threshold=0.85

[预期renew值]*85%

renews threshold =服务总数 * 每分钟的续约数量([60s/客户端续约(心跳)间隔时间]) * renewal-percent-threshold

- 如果上一分钟的心跳续约：10<11 ，说明有大量服务不可用，开启自我保护！
- 否则， 正常！心跳超时检测正常（如果存在大于90s没有发起心跳续约的客户端，会被Eureka Server 剔除）

Eureka的实例信息如何存储

> 详见PPT

Eureka多级缓存设计

> 详见PPT

OpenFeign（http通信的客户端）

屏蔽了网络通信的细节，直接面向接口的方式开发，让开发者感知不到网络通信细节。

所有远程调用，都像调用本地方法一样完成！

Feign -> Netflix

OpenFeign(Spring MVC的注解的解析)

OpenFeign集成gpmall

> 详见代码

OpenFeign的特性

- Gzip压缩
- Logger
- 底层通信框架（sun.net.www.protocol.http.HttpURLConnection）
- OKHTTP

OpenFeign原理

> 详见ppt