Spring Cloud 组件

Eureka

Eureka是SpringCloud中的核心组件，其扮演的角色与作用就是**服务的注册与发现。**

运行原理：

各个服务启动时，Eureka Client都会将服务注册到Eureka Server的一个注册表中，该表会记录所有注册服务的所在的服务器的IP端口，并且Eureka Client还可以反过来从Eureka Server拉取注册表，并缓存在自己服务器，从而知道其他服务在哪里。

同为注册中心，Eureka和Zookeeper的区别：

1. ZooKeeper保证的是CP,Eureka保证的是AP

   ZooKeeper在选举期间注册服务瘫痪,虽然服务最终会恢复,但是选举期间不可用的。Eureka各个节点是平等关系,只要有一台Eureka就可以保证服务可用,而查询到的数据并不是最新的。Eureka可以很好的应对因网络故障导致部分节点失去联系的情况,而不会像ZooKeeper一样使得整个注册系统瘫痪。

   > 自我保护机制会导致：
   > - Eureka不再从注册列表移除因长时间没收到心跳而应该过期的服务。
   > - Eureka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(高可用)。
   > - 当网络稳定时,当前实例新的注册信息会被同步到其他节点中(最终一致性)。

2. ZooKeeper有Leader和Follower角色,Eureka各个节点平等
3. ZooKeeper采用过半数存活原则,Eureka采用自我保护机制解决分区问题
4. Eureka本质上是一个工程,而ZooKeeper只是一个进程。

Feign

feign是声明式的web service客户端，它让微服务之间的调用变得更简单了，类似controller调用service。

Feign的一个关键机制就是使用了动态代理：

- 首先，如果你对某个接口定义了@FeignClient注解，Feign就会针对这个接口创建一个动态代理。
- 接着你要是调用那个接口，本质就是会调用 Feign创建的动态代理，这是核心中的核心。
- Feign的动态代理会根据你在接口上的@RequestMapping等注解，来动态构造出你要请求的服务的地址。
- 最后针对这个地址，发起请求、解析响应。

底层原理：

- 启动时，程序会进行包扫描，扫描所有包下所有@FeignClient注解的类，并将这些类注入到spring的IOC容器中。当定义的Feign中的接口被调用时，通过JDK的动态代理来生成RequestTemplate。
- RequestTemplate中包含请求的所有信息，如请求参数，请求URL等。
- RequestTemplate生产Request，然后将Request交给client处理，这个client默认是JDK的HTTPUrlConnection，也可以是OKhttp、Apache的HTTPClient等。
- 最后client封装成LoadBaLanceClient，结合ribbon负载均衡地发起调用。

Ribbon

Ribbon是一个基于 HTTP 和 TCP 客户端的负载均衡器。Ribbon其实主要是在Spring原有的restTemplate类上做了**负载均衡**处理

运行原理：

- Ribbon是结合Eureka，Feign，先会从 Eureka Client里获取到对应的服务注册表，也就知道了所有的服务都部署在了哪些机器上，在监听哪些端口号。
- 然后Ribbon就可以使用默认的Round Robin算法，从中选择一台机器
- Feign就会针对这台机器，构造并发起请求。

Hystrix

Hystrix是一个熔断器组件。当我们的服务请求链路中，如果某一个环节点出了问题，那么这个时候Hystrix就可以将其熔断，做降级处理等后续操作，保护整个链路的正常进行。不会因为某一个点出现了问题，而导致整条链路失败或是在某个环节卡死。

运行原理：

- Hystrix在服务运行请求时，会为每个服务创建一个线程池。
- 请求进入后，会由线程池去管理。
- 当线程池满了时，会对后来的请求会立即拒绝请求，而不是排队。
- 当请求失败、被拒绝、超时或短路时，执行回退逻辑。
- 并且，如果这种非正常返回的请求量，服务的错误百分比超过了一个阈值，就会触发一个断路器来停止对特定服务的所有请求，无论是手动的还是自动的。这些处理情况，Hystrix会实时监控指标和配置变化，并可通过HystrixDashboard可视化界面查看具体情况。

Zuul

Zuul，也就是微服务网关。这个组件是负责网络路由的。

运行原理：

Zuul会生成很多种Filter，在外部请求过来时，为我们做过滤处理。而在过滤过程中，网关有如下作用：

- 认证和安全：识别每个需要认证的资源，拒绝不符合要求的请求。
- 性能监测：在服务边界追踪并统计数据，提供精确的生产视图。
- 动态路由：根据需要将请求动态路由到后端集群。
- 压力测试：逐渐增加对集群的流量以了解其性能。
- 负载卸载：预先为每种类型的请求分配容量，当请求超过容量时自动丢弃。
- 静态资源处理：直接在边界返回某些响应。