SpringCLoud复习总结

复习线路：

Eureka 注册中心

|

|（切记应用名中间不能用下划线）

EurekaClient 提供者(构建)

|

EurekaClient 消费者（构建）

|

|服务的调用：Ribbon /Feign

1 RestTemplate getForEntity(url,resultype，values)

传递参数：/USER-PROVIDER/{name}/{id} values则是一个map,有key为name 和id

（LoadBalancerClient 是RestTemplage的底层）

2 FeignClient

（有点类似有Mybatis的接口编程）

GetMapping PostMapping

|

|（服务调用时难免出现异常：系统负载过高 流量或网络异常）

Hystrix 熔断 降级 隔离 fallback cache 监控（容错机制）

| 1 熔断

2 降级策略：响应超时设置

3 隔离策略：线程隔离 信号量隔离

4 fallback 为熔断和降级后的补偿操作：提示用户 或 异常报警（可以结合redis）

回调方式一：直接在controller中写回调方法

@RequestMapping(“/fallBack”)

public String fallback(参数和对应方法一致){}

回调方式二：hystrix+feignClient

5 可视化界面：Dashboard仪表盘

启动类注解（必须）@EnableHystrixDashboard

配置配置（必须）

**management:  
 endpoints:  
 web:  
 exposure:  
 include:** "\*"

Hystrix Stream一致loading，是因为没有请求

|

|之后会对服务的调用链路进行监控

链路追踪 Sleuth/ Zipkin

Sleuth：链路抓取（跟踪的项目都要添加）

Zipkin: 数据展示（一个需要部署服务）

用处（好处）：

1 从整体到局部，可视化的集中展示跨应用的链路调用。

2 通过集中展示可以快速排查故障位置 ，性能监控

3 对链路进行分析，明确依赖关系以及优化

4 对链路分析获取用户行为习惯数据，汇总分析应用在很多业务场景。

|

|

配置中心 ConfigServer（独立项目）

Git服务器结合Config搭建分布式配置中心

1 修改配置文件bootstrap.yml 服务名+git地址

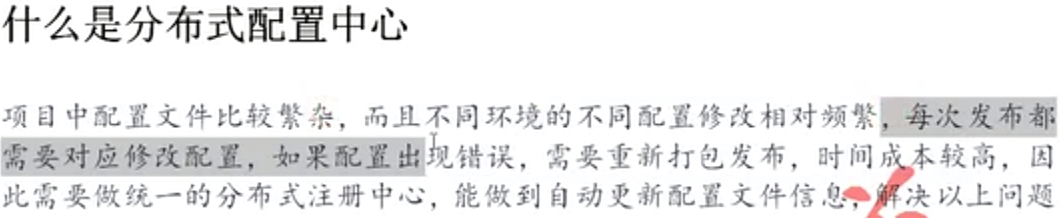
2 客户端添加配置中心地址

传统的配置文件有什么缺点：

如果修改了配置文件，就需要重新打包发布，并重启项目；另外，每个环境变更配置文件比较复杂。

分布式配置中心优点：

将配置文件信息注册到配置中心平台上，可以直接使用分布式配置中心实时修改更新配置文件，既能统一管理，又不需要重新打包发布或重启项目。



|

|

|之后是网关

Zuul

基础：请求智能路由 流量监控 权限校验

高级：熔断降级（ZuulFallbackProvider FallbackProvider）

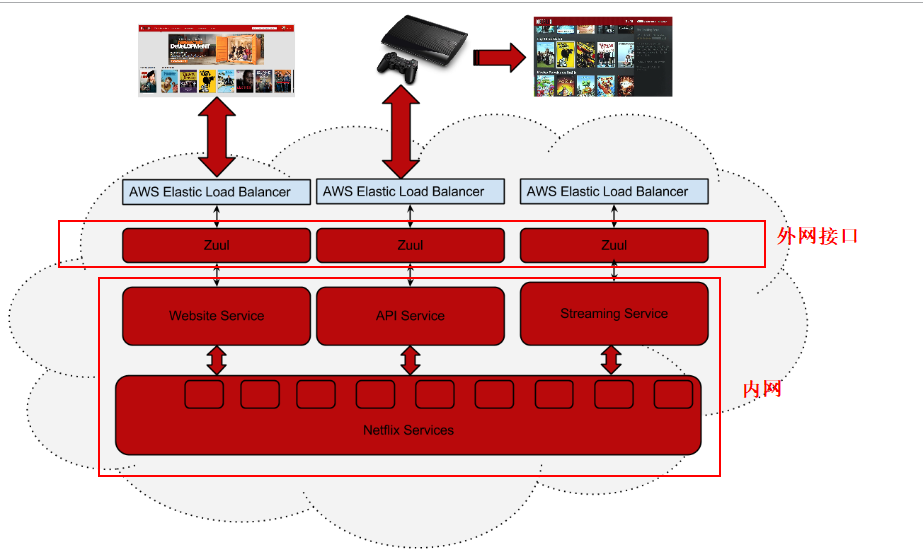
请求重试 (ReTry)依赖

<dependency>

<groupId>org.springframework.retry</groupId>

<artifactId>spring-retry</artifactId>

</dependency>



按照我们的理解，配置有以下几个属性：

* **配置是独立于程序的只读变量**
  + 配置首先是独立于程序的，同一份程序在不同的配置下会有不同的行为。
  + 其次，配置对于程序是只读的，程序通过读取配置来改变自己的行为，但是程序不应该去改变配置。
  + 常见的配置有：DB Connection Str、Thread Pool Size、Buffer Size、Request Timeout、Feature Switch、Server Urls等。
* **配置伴随应用的整个生命周期**
  + 配置贯穿于应用的整个生命周期，应用在启动时通过读取配置来初始化，在运行时根据配置调整行为。
* **配置可以有多种加载方式**
  + 配置也有很多种加载方式，常见的有程序内部hard code，配置文件，环境变量，启动参数，基于数据库等
* **配置需要治理**
  + 权限控制
    - 由于配置能改变程序的行为，不正确的配置甚至能引起灾难，所以对配置的修改必须有比较完善的权限控制
  + 不同环境、集群配置管理
    - 同一份程序在不同的环境（开发，测试，生产）、不同的集群（如不同的数据中心）经常需要有不同的配置，所以需要有完善的环境、集群配置管理
  + 框架类组件配置管理
    - 还有一类比较特殊的配置 - 框架类组件配置，比如CAT客户端的配置。
    - 虽然这类框架类组件是由其他团队开发、维护，但是运行时是在业务实际应用内的，所以本质上可以认为框架类组件也是应用的一部分。
    - 这类组件对应的配置也需要有比较完善的管理方式。