注:

JDK:1.8

IDE:IDEA

参考书籍:Spring+Cloud微服务实战

# 一.Spring Cloud 整体简介

Spring Cloud 是基于Spring Boot 实现的微服务架构开发工具,它为微服务架构中涉及的配置管理、服务治理、断路器、智能路由、微代理、控制总线、全局锁、决策精选、分布式会话和集群状态管理等操作提供了一种简单的开发方式。

Spring Cloud包含了多个子项目：

1. Spring Cloud Config：配置管理工具，支持使用Git存储配置内容，可以使用它实现应用配置的外部化存储，并支持客户端信息刷新、加密/解密配置内容.
2. Spring Cloud Netflix: 核心组件,对多个Netflix OSS开源套件进行整合.
   1. Eureka:服务治理组件,包含服务注册中心,服务注册与发现机制的实现.
   2. Hystrix:容错管理组件,实现断路器模式,帮助服务依赖中出现的延迟,为故障提供强大的容错能力.
   3. Ribbon:客户端负载均衡的服务调用组件.
   4. Feign:基于Ribbon和Hystrix的声明式服务调用组件.
   5. Zuul:网关组件,提供智能路由、访问过滤等功能。
   6. Archaius:外部化配置组件
3. Spring Cloud Bus : 事件,消息总线,用于传播集群中的状态变化或事件,以触发后续的处理,比如用来动态刷新配置等.
4. Spring Cloud Cluster:针对Zookeeper、Redis、Hazelcast、Consul的选举算法和通用状态模式的实现。
5. Spirng Cloud Cloudfoundry：与Pivotal Cloudfoundry的整合支持
6. Spring Cloud Consul：服务发现与配置管理工具。
7. Spring Cloud Stream：通过Redis、Rabbit或者Kafka实现的消费微服务。可以通过简单的声明式模型来发送和接收消息.
8. Spring Cloud AWS:用于简化整合Amazon Web Service的组件.
9. Spring Cloud Security:安全工具包,提供Zuul代理中对OAuth2客户端请求的中继器.
10. Spring Cloud Sleuth:应用的分布式跟踪实现,可以完美整合Zipkin.
11. Spring Cloud Cli:用于在Groovy中快速创建Spring Cloud 应用的Cli插件.

# 二.Eureka

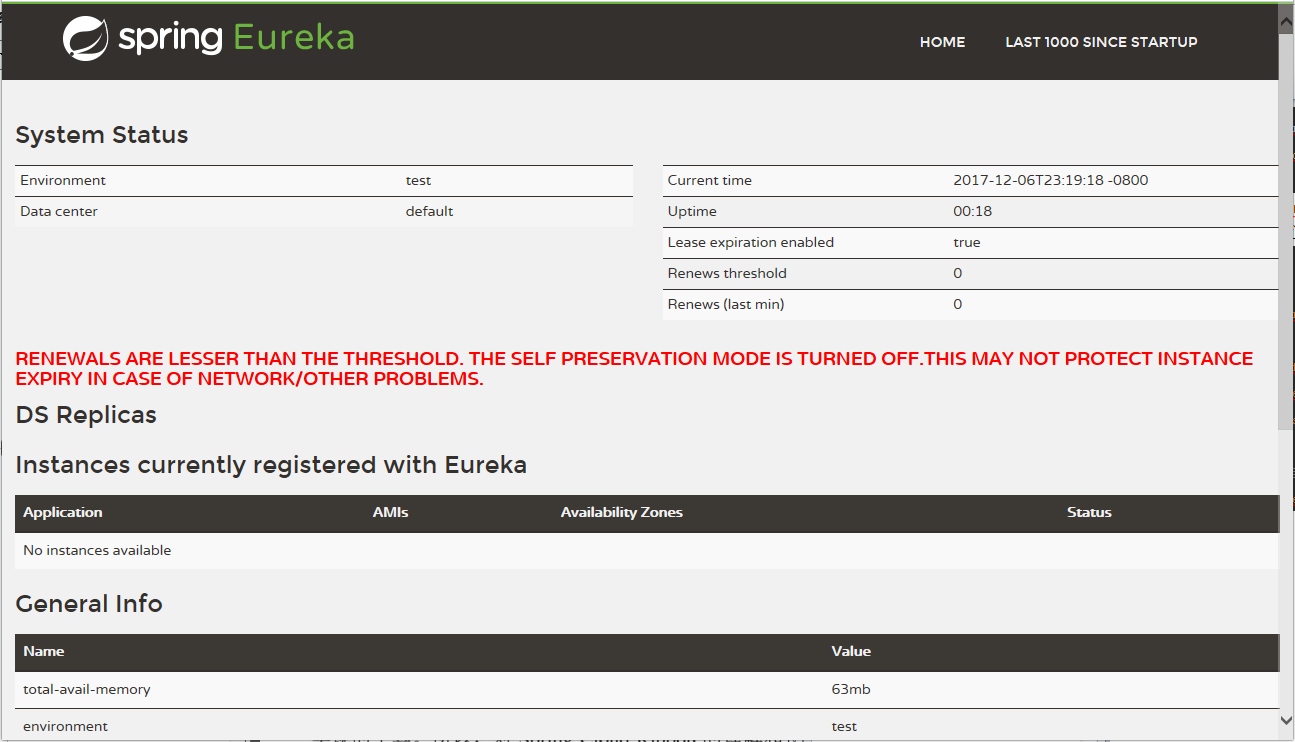
Eureka是Spring Cloud Netflix微服务套件中的一部分,它基于Netflix Eureka做了二次封装,主要负责完成微服务架构中的服务治理功能.

1. 引入依赖配置

<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-eureka-server</artifactId>  
</dependency>

1. 启动注解@EnableEurekaServer
2. 修改注册中心的配置

#配置eureka  
#当前应用为注册中心,不用向注册中心注册自己  
eureka.client.register-with-eureka=false  
#当前应用是注册中心,所以它不需要去检索服务  
eureka.client.fetch-registry=false  
eureka.instance.hostname=localhost  
eureka.client.serviceUrl.defaultZone=http://${eureka.instance.hostname}:${server.port}/eureka/  
#关闭eureka的自我保护  
eureka.server.enable-self-preservation=false

1. 启动注册中心

# 三.Ribbon

Ribbon是基于HTTP和TCP的客户端负载均衡工具，它基于Netflit Ribbon实现，通过Spring Cloud 的封装，可以让我们轻松地讲面向服务的REST模板请求自动转换成客户端负载均衡的服务调用。

Ribbon实现负载均衡的原理,它通过LoadBalancerInterceptor拦截器对RestTemplate的请求进行拦截,并利用Spring Cloud的负载均衡器LoadBalancerClient将以逻辑服务名为host的URI转换成具体的服务实例地址的过程.同时通过分析LoadBalancerClient的Ribbon实现RibbonLoadBalancerClient,可以知道在使用Ribbon实现负载均衡器的时候,实际使用的还是Ribbon中定义的ILoadBalancer接口的实现,自动化配置会采用ZoneAwareLoadBalancer的实例来实现客户端负载均衡.

在Spring Cloud的微服务架构中使用客户端负载均衡调用需要两步:

1. 服务提供者只需要启动多个服务事例并且注册到一个注册中心或者多个相关联的服务注册中心.
2. 服务消费者直接通过调用被@LoadBalanced注解修饰过的RestTemplate来实现面向服务的接口调用.