springboot单体项目集成springcloud

eureka

注册中心服务端的组件

spring-cloud-starter-eureka-server

1.4.7.RELEASE

服务端登录验证模块：进入 eureka 的网页时候需要输入登录密码的模块

spring-boot-starter-security

1.4.1.RELEASE

- 配置 eureka 登录密码

```
security:
  basic:
    enabled: true
  user:
    name: admin
    password: admin
```

启动类加上eurekaserver注解

@EnableEurekaServer

配置 eureka 首页的路径

```
  dashboard:
    enabled: true   # 如果这里配置为 false 的话，那么 Eureka 的首页将无法访问
    path: /x        # 默认配置是 / ，但是这里配置成 /x 的话，那么访问的首页路径为: http://localhost:8761/x
```

eureka常用配置

```
eureka:
  server:
```

```
    enable-self-preservation: false #自我保护模式关闭
    eviction-interval-timer-in-ms: 60000  # 清理间隔（单位毫秒，默认是60*1000）
  datacenter: SpringCloud   # 修改 http://localhost:8761 地址 Eureka 首页上面 System Status 的 Data center 显示信息
  environment: Test         # 修改 http://localhost:8761 地址 Eureka 首页上面 System Status 的 Environment 显示信息
  client:
    register-with-eureka: false #因为默认情况下一个eureka-server也是一个eureka-client,所以设置注册eureka为false
    fetch-registry: false #表示表示是否从EurekaServer获取注册信息，默认为true。单节点不需要同步其他的EurekaServer节点的数据
    healthcheck:  # 健康检查（需要spring-boot-starter-actuator依赖）
      enabled: true
  instance:
    hostname: localhost
```

eureka.server.enable-self-preservation

是否开启自我保护模式，默认为true。

默认情况下，如果Eureka Server在一定时间内没有接收到某个微服务实例的心跳，Eureka Server将会注销该实例（默认90秒）。但是当网络分区故障发生时，微服务与Eureka Server之间无法正常通信，以上行为可能变得非常危险了——因为微服务本身其实是健康的，此时本不应该注销这个微服务。
Eureka通过“自我保护模式”来解决这个问题——当Eureka Server节点在短时间内丢失过多客户端时（可能发生了网络分区故障），那么这个节点就会进入自我保护模式。一旦进入该模式，Eureka Server就会保护服务注册表中的信息，不再删除服务注册表中的数据（也就是不会注销任何微服务）。当网络故障恢复后，该Eureka Server节点会自动退出自我保护模式。
综上，自我保护模式是一种应对网络异常的安全保护措施。它的架构哲学是宁可同时保留所有微服务（健康的微服务和不健康的微服务都会保留），也不盲目

注销任何健康的微服务。使用自我保护模式，可以让Eureka集群更加的健壮、稳定。

开发环境或小项目中建议关闭自我保护模式

gateway

网关

spring-cloud-starter-gateway

2.1.2.RELEASE

监控和管理Spring Boot应用

spring-boot-starter-actuator

2.1.6.RELEASE

eureka客户端

spring-cloud-starter-netflix-eureka-client

2.1.1.RELEASE

版本问题可能需要gson

gson

2.8.5

网关跨域配置

```
@Configuration
public class CorsConfig {
   @Bean
   public CorsWebFilter corsFilter() {
      CorsConfiguration config = new CorsConfiguration();
      config.addAllowedMethod("*");
      config.addAllowedOrigin("*");
      config.addAllowedHeader("*");

      UrlBasedCorsConfigurationSource source = new
UrlBasedCorsConfigurationSource(new PathPatternParser());
      source.registerCorsConfiguration("/**", config);
      return new CorsWebFilter(source);
```

```
    }
}
```

此处注意如果之前未接入网关前如果已经有了跨域配置的代码，则单体应用的跨域配置应该干掉

启动类加上eurekaClient注解

```
@EnableEurekaClient
```

常用配置

gateway配置

```
spring:
  application:
    name: gateway
  # 开启 Gateway 服务注册中心服务发现
  cloud:
    gateway:
      discovery:
        locator:
          enabled: false
          lowerCaseServiceId: true
      routes:
        - id: CompositeDiscoveryClient_OMS
          uri: lb://OMS
          order: 0
          predicates:
            - Path=/**
          filters:
            - StripPrefix=0
#           - RequestTime=true
```

子主题 1

eurekaClient配置

```
eureka:
  instance:
    instance-id: ${spring.cloud.client.ip-address}:${spring.application.name}:${server.port}
    prefer-ip-address: true
  #    lease-renewal-interval-in-seconds: 5
  #    lease-expiration-duration-in-seconds: 15
```

registerWithEureka: true
fetchRegistry: true
healthcheck: # 健康检查
enabled: false

如果 cloud:gateway:discovery: locator:enabled: 为true，会添加默认路由，建议关闭

routes.filters- StripPrefix等于几就在往后台服务发送时的路径上去掉几个前缀

lb://OMS

OMS为服务名

order

默认为0，越低优先级越高

远程eurekaServer配置

eureka:
client:
serviceUrl:
defaultZone:
http://admin:admin@localhost:8181/eureka

开启actuator管理api，后面要关闭

management:
endpoints:
web: # http://localhost:8080/actuator/gateway/routes
exposure:
include: "*"