How to use SpringCloud



Introduction

spring cloud 为开发人员提供了快速构建分布式系统的一些工具,包括配置管理、服务发现、断路器、路由、微代理、事件总线、全局锁、决策竞选、分布式会话等等。它运行环境简单,可以在开发人员的电脑上跑。spring cloud是基于springboot的,所以需要开发中对springboot有一定的了解。

Eureka Server

Eureka是一个组件Spring Cloud Netflix上的服务注册和发现模块。

STEP1

创建一个maven主工程。

STEP2

创建两个model工程: 一个model工程作为服务注册中心,即Eureka Server,另一个作为Eureka Client。

STEP3

启动一个服务注册中心,只需要一个注解@EnableEurekaServer。

这个注解需要在springboot工程的启动application类上加:

```
@EnableEurekaServer
@SpringBootApplication
public class EurekaserverApplication {
   public static void main(String[] args) {
        SpringApplication.run(EurekaserverApplication.class, args);
   }
}
```

STEP4

修改eureka server的配置文件appication.yml

```
server:
port: 8761

eureka:
  instance:
  hostname: localhost
  client:
    registerWithEureka: false
    fetchRegistry: false
    serviceUrl:
      defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka/
```

STEP5

打开浏览器访问: http://localhost:8761

Eureka Client

当client向server注册时,它会提供一些元数据,例如主机和端口,URL,主页等。Eureka server从每个client实例接收心跳消息。如果心跳超时,则通常将该实例从注册server中删除。Eureka client创建过程与Eureka server类似。

STEP1

通过注解@EnableEurekaClient 表明自己是一个eurekaclient.

```
@SpringBootApplication
@EnableEurekaClient
@RestController
public class ServiceHiApplication {

   public static void main(String[] args) {
        SpringApplication.run(ServiceHiApplication.class, args);
   }

   @Value("${server.port}")
   String port;
   @RequestMapping("/hi")
   public String home(@RequestParam String name) {
        return "hi "+name+",i am from port:" +port;
   }
}
```

STEP2

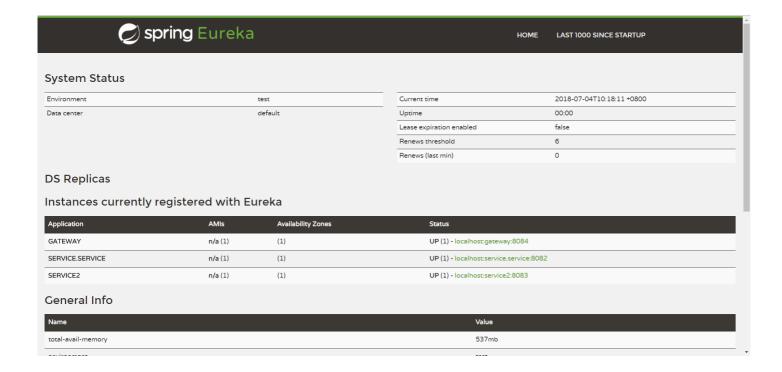
修改eureka client的配置文件appication.yml

```
eureka:
   client:
      serviceUrl:
      defaultZone: http://localhost:8761/eureka/
server:
   port: 8762
spring:
   application:
      name: service-hi
```

STEP3

```
打开浏览器访问: http://localhost:8761
```

运行结果如下图所示。



Zuul

Zuul的主要功能是路由转发和过滤器。路由功能是微服务的一部分,比如 / api/user转发到到user服务,/api/shop转发到到shop服务。

STEP1

在上一节创建的原有工程上,新建一个工程。

STEP2

在其入口applicaton类加上注解@EnableZuulProxy, 开启zuul的功能

```
@EnableZuulProxy
@EnableEurekaClient
@SpringBootApplication
public class ServiceZuulApplication {
   public static void main(String[] args) {
        SpringApplication.run(ServiceZuulApplication.class, args);
   }
}
```

STEP3

修改eureka server的配置文件appication.yml

```
eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:8761/eureka/
server:
  port: 8769
spring:
  application:
    name: service-zuul
zuul:
  routes:
    api-a:
      path: /api-a/**
      serviceId: service-ribbon
    api-b:
      path: /api-b/**
      serviceId: service-feign
```

STEP4

首先指定服务注册中心的地址为http://localhost:8761/eureka/服务的端口为8769,服务名为service-zuul以/api-a/开头的请求都转发给service-ribbon服务以/api-b/开头的请求都转发给service-feign服务

效果如下



Actuator

步骤与上面类似,相关配置如下。

项目依赖

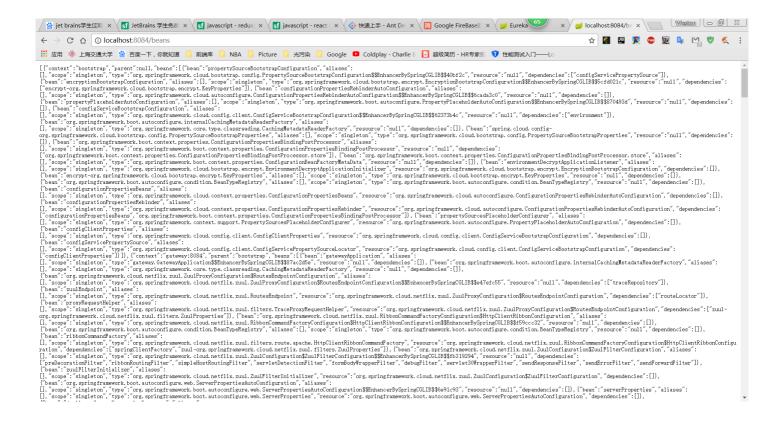
配置文件

```
server:
    port: 8080
management:
    security:
        enabled: false #关掉安全认证
    port: 8088 #管理端口调整成8088
    context-path: /monitor #actuator的访问路径
endpoints:
    shutdown:
        enabled: true

info:
    app:
        name: spring-boot-actuator
        version: 1.0.0
```

- management.security.enabled=false 默认有一部分信息需要安全验证之后才可以查看,如果去掉这些安全认证,直接设置management.security.enabled=false
- management.context-path=/monitor 代表启用单独的url地址来监控Spring Boot应用,为了安全一般都启用独立的端口来访问后端的监控信息
- endpoints.shutdown.enabled=true 启用接口关闭Spring Boot 配置完成之后,启动项目就可以继续验证 各个监控功能了。

我们的监控情况如下。



Reference

• https://spring.io