所有项目引入SpringCloud依赖

<!-- SpringCloud必须的依赖管理-->  
 <dependencyManagement>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-dependencies</artifactId>  
 <version>${spring.cloud-version}</version>  
 <type>pom</type>  
 <scope>import</scope>  
 </dependency>  
 </dependencies>  
 </dependencyManagement>

服务注册中心：

<!-- 服务治理框架服务器-->  
 <dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>  
 </dependency>

配置文件：

server.port

eureka.instance.hostname

spring.application.name

#禁止向注册中心注册自己因为自己就是注册中心  
eureka.client.register-with-eureka=false

#禁止去检索服务，那是客户端干的事  
eureka.client.fetch-registry=false

eureka.client.service-url.defaultZone = http://${eureka.instance.hostname}:${server.port}/eureka/

主类上加

@EnableEurekaServer

服务注册：

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

配置文件：

spring.application.name=eureka-provider

eureka.client.service-url.defaultZone = http://localhost:1111/eureka/

主类上加

@EnableEurekaClient

高可用注册中心

两个服务器

配置1

server.port=1111  
  
eureka.instance.hostname=peer1  
  
spring.application.name=eureka-server

eureka.client.service-url.defaultZone = http://peer2:1112/eureka/

配置2

server.port=1112  
  
eureka.instance.hostname=peer2  
  
spring.application.name=eureka-server

eureka.client.service-url.defaultZone = http://peer1:1111/eureka/

互相注册

测试时：

在hosts文件中修改127.0.0.1 peer1, 127.0.0.1 peer1

如果不想使用主机名，可以使用ip

eureka.instance.prefer-ip-address=true

服务注册：

spring.application.name=eureka-client

eureka.client.service-url.defaultZone = http://peer1:1111/eureka/, http://peer2:1112/eureka/

这样一个客户端同时在两个注册中心注册，其中一个坏了，还是可以用第二个

服务消费（注：测试负载均衡时启动两个提供者）：

引入依赖：

<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  
</dependency>  
  
<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-ribbon</artifactId>  
</dependency>

配置和注册服务一样

开启负载均衡：

在主类里加上

@Bean  
@LoadBalanced  
RestTemplate restTemplate(){  
 return new RestTemplate();  
}

轮询调用服务：

restTemplate.getForEntity("http://服务名/接口名", String.class).getBody();

服务续约配置属性

eureka.instance.lease-renewal-interval-in-seconds=

发送心跳间隔时间  
eureka.instance.lease-expiration-duration-in-seconds=

服务失效时间

服务消费配置属性  
eureka.client.registry-fetch-interval-seconds=

缓存服务清单的更新时间

服务注册中心本地测试关闭保护机制

eureka.server.enable-self-preservation=false

配置（客户端）：

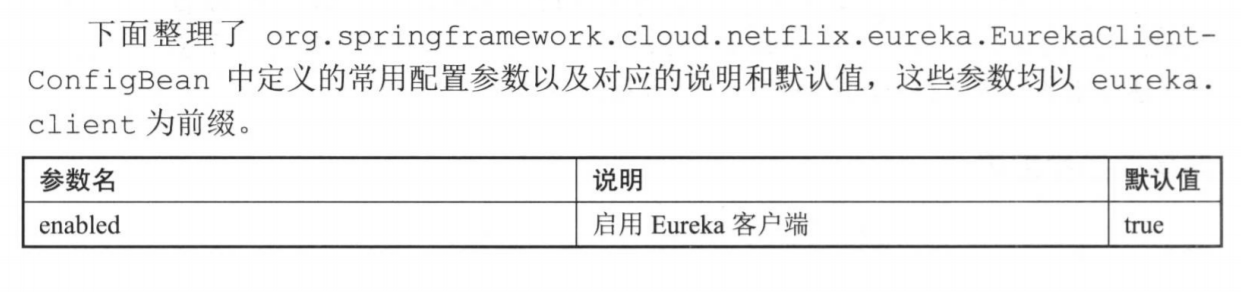
服务注册配置

eureka.client.service-url.defaultZone = 多个注册中心地址用逗号分割

若注册中心加了安全校验

[http://<username>:<password>@localhost:1111/eureka](http://%3cusername%3e:%3cpassword%3e@localhost:1111/eureka)

其他配置：







服务实例配置：

标准化元数据配置：

eureka.instance.属性名=

自定义元数据配置：

eureka.instance.metadata-map.属性名=

实例名配置：

eureka.instance.instance-id=${spring.application.name}:${random.int}

server.port=0

这样实例就根据名字和端口号来确定唯一id

断点配置

若设置了management.server.servlet.context-path=

则

eureka.instance.status-page-url-path=${management.server.servlet.context-path}/info  
eureka.instance.health-check-url-path=${management.server.servlet.context-path}/health

否则不需要配置

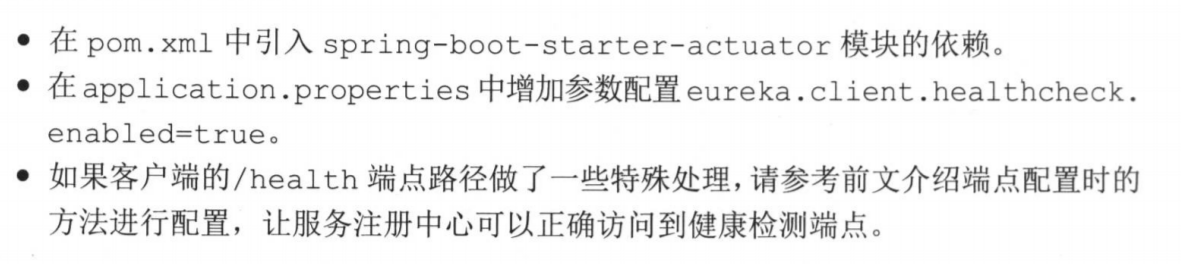
若修改原始路径

endpoints.info.path=/appInfo  
endpoints.health.path=/checkHealth  
eureka.instance.status-page-url-path=/${endpoints.info.path}  
eureka.instance.health-check-url-path=/${endpoints.health.path}

若以https暴露服务，则用绝对路径配置

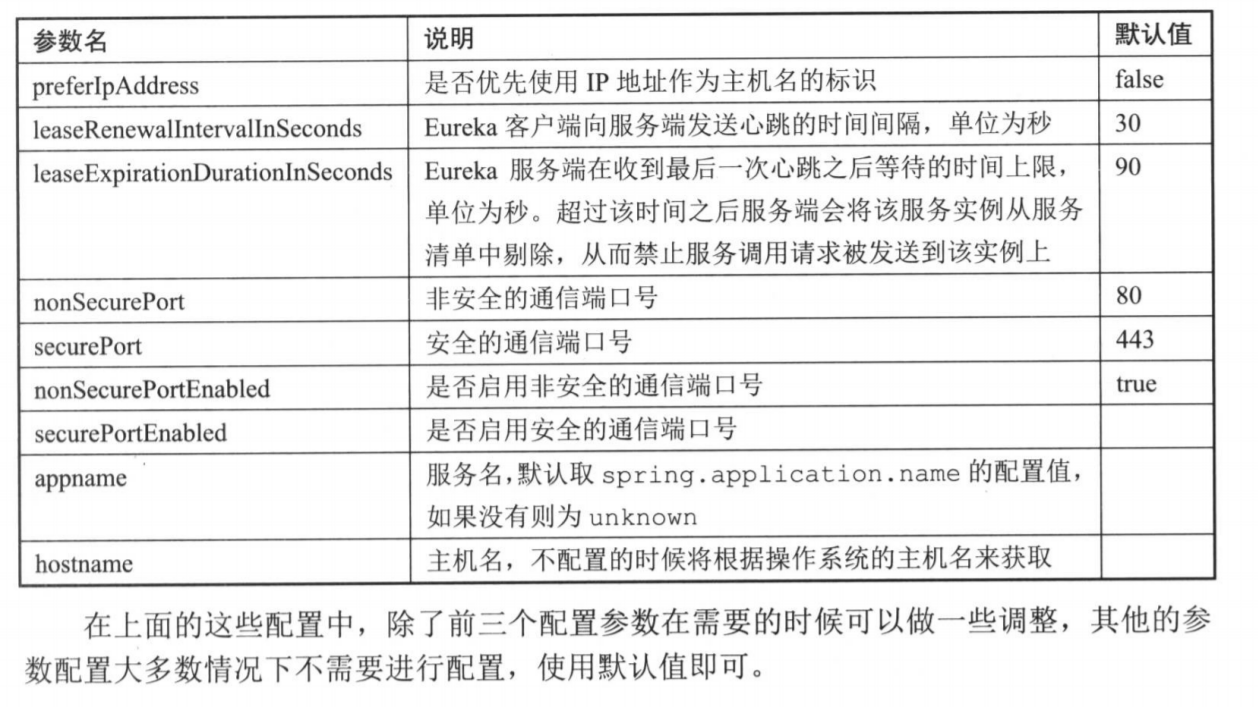
eureka.instance.status-page-url=https://${eureka.instance.hostname}/info  
eureka.instance.health-check-url=https://${eureka.instance.hostname}/health  
eureka.instance.home-page-url=https://${eureka.instance.hostname}/

健康检测：



其他配置：





Eureka的客户端不止用Java语言来实现，也有例如python-eureka的实现，或者eureka-js-client

所以微服务可以跨语言进行合作