Spring Cloud Netflix Eureka

1. Introduction

Client-side service discovery allows services to find and communicate with each other without hard-coding the hostname and port. One drawback is that all clients must implement a certain logic to interact with this fixed point. This assumes an additional network round trip before the actual request.

With Netflix Eureka, each client can simultaneously act as a server to replicate its status to a connected peer. In other words, a client retrieves a list of all connected peers in a service registry, and makes all further requests to other services through a load-balancing algorithm.

For this tutorial, it will be implemented two main types of microservices:

- a service registry (Eureka Server);
- a REST service, which registers itself at the registry (Eureka Client).

2. Eureka Server

Implementing a Eureka Server for service registry is as easy as:

- adding spring-cloud-starter-netflix-eureka-server to the dependencies;
- enabling the Eureka Server in a @SpringBootApplication by annotating it with @EnableEurekaServer;
- configuring some properties.

First of all, you have to create a new Maven project called “discovery-server”. After this step you have to add the dependencies in the pom.xml file:

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

Then you should create the main application class:

```
@SpringBootApplication
@EnableEurekaServer
public class DiscoveryServerApplication {

    no usages   oana_pop
    public static void main(String[] args) { SpringApplication.run(DiscoveryServerApplication.class, args); }

}
```

The last step for the server is to configure the application.properties file in this way:

```
eureka.instance.hostname=localhost
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
server.port=8761
```

3. Eureka Client

Your clients will be the user microservice and the calendar microservice. For both of them you have to add the dependencies in the pom.xml file:

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

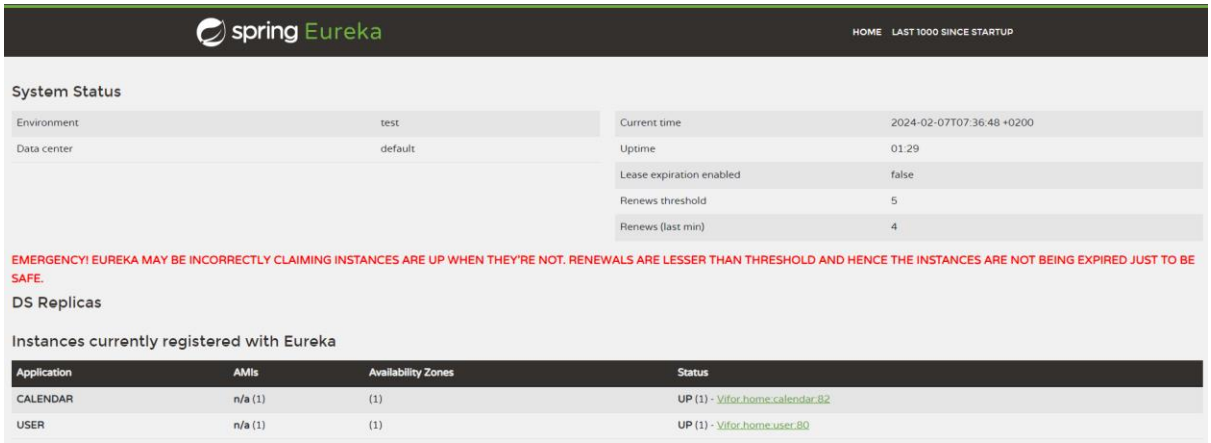
Then you should create the main application class with the specific annotation:

```
1 usage   oana_pop
@SpringBootApplication
@EnableDiscoveryClient
public class CalendarApplication {

    no usages   oana_pop
    public static void main(String[] args) { SpringApplication.run(CalendarApplication.class, args); }

}
```

Finally, you have to set the properties in the application.yml:

```
eureka:
  client:
    serviceUrl:
      defaultZone: http://localhost:8761/eureka
```

After making these changes for user and calendar microservices, you can go to http://localhost:8761/ to check if the clients are registered:



As you can see in the image, both of the microservices are successfully registered!