Zoe Jimenez

DH  CTD

BACKEND II

MICROSERVICIOS

**Eureka**

Registrar automáticamente los microservicios y sus instancias, tan pronto como se encuentren ejecutando y eliminarlas del registro en cuanto dejan de ejecutarse o dejan de responder.

El cliente debe ser capaz de enviar una solicitud a los microservicios sin conocer su ubicación.

Las solicitudes a las instancias de un microservicio deben poder ser equilibradas mediante un balanceador de carga.

Se puede dividir en dos componentes.

|  |  |
| --- | --- |
| **Client** | **Server** |
| Encargado de publicar la información del microservicio en el que se encuentra. | Encargado de recopilar la información de todos los clientes. |

**Agregar a un proyecto:**

* Eureka Client:

Agregar dependencia:

* spring-cloud-starter-netflix-eureka-client

En Maven:

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>

</dependency>

Con agregar la dependencia se registra la app en Eureka server, hay que indicar donde se esta ejecutando (default: puerto 8761).

Dentro de application.properties:

eureka.client.service-url.defaultZone= <http://localhost:8761/eureka/>

Mas propiedades a configurar:

Los clientes para comunicarse forman la url con el nombre de la aplicación:

spring.application.name=mi-servicio

Para desactivar Eureka client en nuestro proyecto podemos agregar:

eureka.client.enabled=false

Eureka Server:

Crear un proyecto nuevo con Spring Boot, y agregar las dependencias:

* spring-cloud-starter-netflix-eureka-server
* spring-boot-starter-web
* spring-boot-starter-actuator

Con Maven:

<dependency>

<groupId>org.springframework.cloud</groupId>

<artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

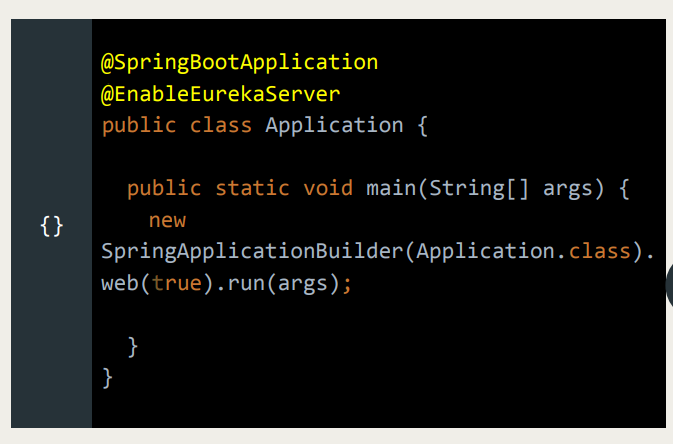
<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

Agregar anotación @EnableEurekaServer en la clase principal del proyecto.

Ej: 

En application.properties, Eureka server puede funcionar como cliente y servidor a la vez. Para evitar que se registre como cliente poner:

eureka.client.register-with-eureka=false eureka.client.fetch-registry=false

Configuramos el puerto:

server.port=8761

**Spring Boot Actuator (heartbeat)**

En un ecosistema de microservicios es necesario llevar un control por errores, para balanceo de carga o para analizar métricas (ver si se necesitan mas instancias de un servidor en particular, etc.).

Los errores mas comunes en las instancias son por excepción de aplicación, timeout (la respuesta llega después de un determinado umbral), y errores de instancia (la instancia se queda sin espacio en disco, sin memoria, etc.)

Spring Actuator genera los endpoints necesarios para el monitoreo, y las respuestas pueden ser sondeadas por balanceadores de carga, Eureka u otras aplicaciones de administración de tráfico.

|  |  |
| --- | --- |
| Endpoints útiles | |
| /health | Información sobre la salud de la aplicación. Un “estado” si estamos sin autenticación, o información más detallada al autenticarse. |
| /info | Información arbitraria. |
| /metrics | Información de métricas. |
| /trace | Información de seguimiento (default: ultimas peticiones HTTP). |
| /serviceregistry | Estado del registro en Eureka Server. |

Por defecto solo esta habilitado /health, para habilitar el resto hay que configurar application.properties:

**management.endpoints.web.exposure.include=serviceregistry,health,info**

Para habilitar todos:

**management.endpoints.web.exposure.include=\***

Eureka Server utiliza /health e /info.

Por defecto, petición GET a /health responde:

{

“status”:”UP”

}

Petición GET a /info debe ser configurada, ejemplo:

info.app.name=mi-servicio  
info.app.description=Servicio probando Eureka  
info.app.version=1.0.0

Respuesta:

{  
"app" : {  
"version" : "1.0.0",  
"description" : "Servicio probando Eureka",  
"name" : "mi-servicio"  
}  
}

Según la versión de Actuator, el endpoint /info no lee la configuración en application.properties. Para habilitarlo:

**management.info.env.enabled = true**

Agregarlo a nuestro proyecto con Maven:

<dependency>

<groupid>org.springframework.boot</groupid>

<artifactid>spring-boot-starter-actuator</artifactid>

</dependency>