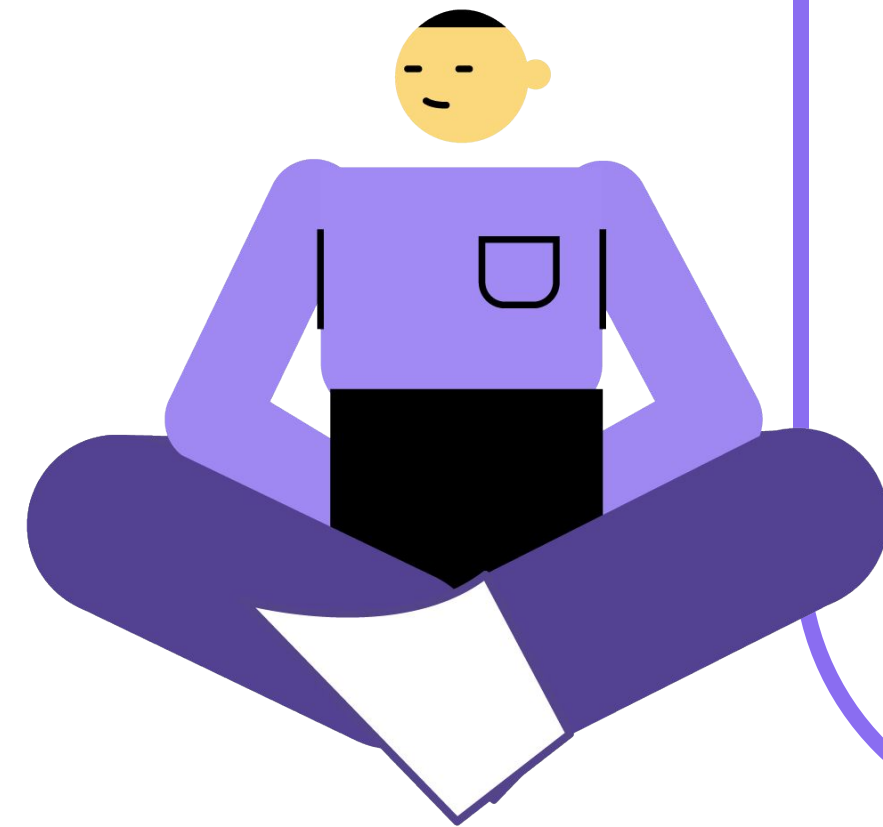


Configuración de Eureka

Configuración de Eureka

Eureka implementa el descubrimiento de servicios del lado del cliente, lo que significa que los clientes son los que se comunican con el servidor de descubrimiento (**Eureka server**) para obtener información sobre las instancias de los microservicios disponibles.

A continuación veremos cómo configurar Eureka client y Eureka server.



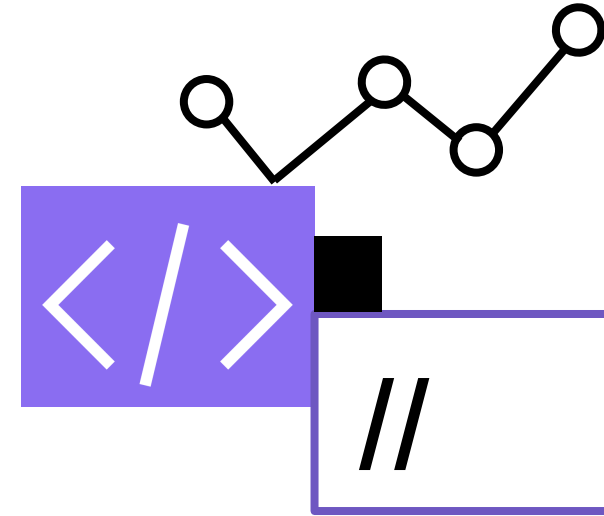
Índice

- 01 [Eureka client](#)
- 02 [Eureka server](#)



01

Eureka client

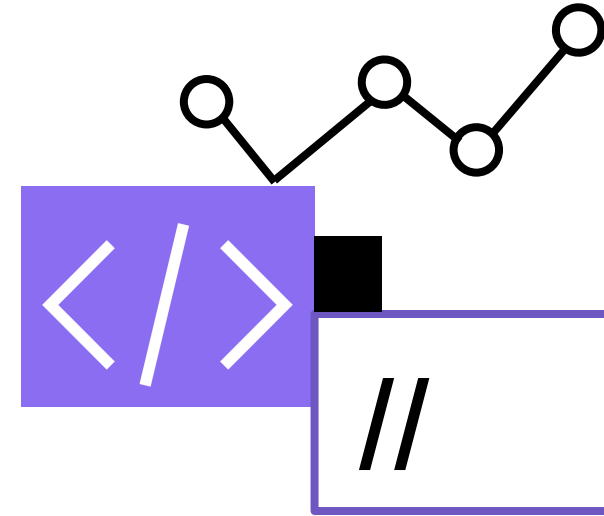


Eureka client

Para configurar **Eureka client** en nuestro proyecto solo debemos agregar la dependencia `spring-cloud-starter-netflix-eureka-client`.

Por ejemplo, utilizando **Maven** como gestor de dependencias:

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



Con agregar la dependencia, nuestra aplicación se registra automáticamente con Eureka server, solo debemos indicarle en dónde se está ejecutando el servidor (por defecto el puerto será 8761).

Dentro del archivo **application.properties**:

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

Una propiedad muy importante a configurar es **spring.application.name**. Esta propiedad se utiliza para darle a cada microservicio el hostname. Luego, los clientes —para comunicarse— utilizarán este nombre para formar la URL:

```
spring.application.name=mi-servicio
```

Para desactivar Eureka client en nuestro proyecto podemos agregar:

```
eureka.client.enabled=false
```

02

Eureka server

Eureka server

Para configurar **Eureka server** debemos seguir los siguientes pasos:

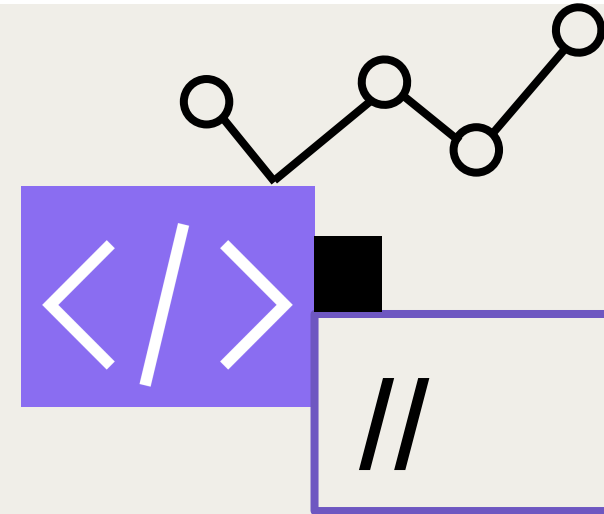
01

Crear un nuevo proyecto con Spring Boot.

02

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>
```

03

Agregar la anotación **@EnableEurekaServer** en la clase principal de nuestro proyecto. Por ejemplo:

```
@SpringBootApplication
@EnableEurekaServer
public class Application {

    public static void main(String[] args) {
        new
        SpringApplicationBuilder(Application.class).
        web(true).run(args);

    }
}
```

04

En **application.properties**, Eureka server es capaz de funcionar como cliente y servidor al mismo tiempo. Para que no se registre como cliente debemos poner las siguientes propiedades en *false*:

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

Configuramos el puerto:


```
server.port=8761
```

05

Ahora sí podemos ejecutar nuestro proyecto y ver la página principal del dashboard de Eureka como se muestran todos los endpoint en:

<http://localhost:8761/eureka/>

Dashboard Eureka Server



[HOME](#) [LAST 1000 SINCE STARTUP](#)

System Status

Environment	N/A	Current time	2022-02-01T22:08:31 -0300
Data center	N/A	Uptime	00:00
		Lease expiration enabled	false
		Renews threshold	1
		Renews (last min)	0

DS Replicas

localhost

Instances currently registered with Eureka

Application	AMIs	Availability Zones	Status
No instances available			

General Info

Name	Value
total-avail-memory	594mb
num-of-cpus	16
current-memory-usage	166mb (27%)

¡Muchas gracias!