Descubrimiento de servicios con Spring Boot Eureka

Tecnología de Software para Electrónica

Trabajo de Extra

Autor:
Jordy Bayas Escobar (alumno)

Tutor/es:
Darwin Alulema Flores (tutor1)

12 de Junio de 2019

Descubrimiento de servicios con Spring Boot Eureka

Subtítulo del proyecto

Autor/es
Jordy Bayas Escobar (alumno)

Tutor/es
Darwin Alulema Flores (tutor1)
Departamento de Eléctrica y Electrónica

Tecnología de Software para Electrónica

Quito, 12 de Junio de 2019

Índice general

Índice de figuras

1 Objetivos

1.1 Objetivo General

Desarrollar una aplicación que permita la inicialización del Servidor de Descubrimiento Eureka

1.1.1 Objetivos específicos

- Demostrar los distintos factores que intervienen en el resgitro de descubrimiento.
- Deducir las variables que componen a Eureka, además de la configuración de sus dependencias.
- Diseñar un prototipo basado en Maven para la inicialización y registro de servicios.

2 Marco Teórico

2.1 Spring Cloud Netflix Eureka

Eureka es un servicio REST, utilizándose principalmente en la nube de AWS, a la cual está estrechamente ligado. Eureka se comporta como servidor, cuyo objetivo es registrar y localizar microservicios existentes, informar de su localización, su estado y datos relevantes de cada uno de ellos. Además, nos facilita el balanceo de carga y tolerancia a fallos. Por último, añadir que gracias a Spring Cloud Netflix, nos proporciona una fácil integración con el proyecto Netflix OSS (Open Source Software) para aplicaciones Spring Boot.

2.2 Funcionamiento

Lo primero es entender que un microservicio debe definirse como un cliente de Eureka. Una vez aclarado este detalle, vamos a explicar su funcionamiento. Cuando un microservicio arranca, se comunicará con el servidor Eureka para notificarle que está disponible para ser consumido. El servidor Eureka mantendrá la información de todos los microservicios registrados y su estado. Cada microservicio le notificará su estado mediante heartbeats cada 30 segundos. Si pasados tres periodos heartbeats no recibe ninguna notificación del microservicio, lo eliminará de su registro. Al igual que si después de sacarlo del registro recibe tres notificaciones, entenderá que ese microservicio vuelve a estar disponible. Cada cliente o microservicio puede recuperar el registro de otros microservicios registrados y quedará cacheado en dicho cliente. Además, Eureka se puede configurar para funcionar en modo cluster, para que varias instancias intercambien su información.

2.3 Configuración

Como se ha mencionado anteriormente, existen dos elementos diferenciados: un servidor Eureka y un cliente Eureka. Cabe destacar las anotaciones que proporciona Spring Cloud Netflix para facilitar la configuración de ambos.

- **Servidor Eureka** La configuración que hay que llevar a cabo para crear un servidor Eureka y usarlo como un registro de servicios es sencillo. El primer paso es

agregar la dependencia de maven“spring-cloud-starter-eureka-server” para poder añadir a nuestra clase de configuración de Spring Boot la anotación @EnableEurekaServer.
Una vez hecho esto, se añade una configuración básica en el fichero de propiedades “application.yml”. En este caso, el formato que se va a emplear será YAML.
Como se puede observar, se indicará el puerto donde se quiere que despliegue.

```
server:
  port: 8761
eureka:
  client:
    registerWithEureka: false
    fetchRegistry: false
```

Figura 2.1: COnfiguración del local host

De igual modo, se indicará que la propiedad “registerWithEureka” no se registre a sí misma, porque se quiere que actúe como servidor. Para comprobar que está levantado, el punto de acceso será: http://localhost:8761.

- **Cliente Eureka** Para que una aplicación sea identificada en el registro de Eureka, también se debe añadir la dependencia “spring-cloud-starter-eureka-server”, agregar a la clase de configuración de Spring Boot la anotación @EnableEurekaClient y configurar su fichero de propiedades. Finalmente, se define el nombre con el que se quiere registrar la aplicación, el puerto dónde se va a levantar y el servidor Eureka dónde debe registrarse.

2.4 Servicios y clientes no basados en Java

Si el servicio no está basado en Java, Eureka ofrece la posibilidad de implementar la parte cliente de Eureka en el lenguaje de programación que se desee. También cabe la posibilidad de ejecutar una aplicación Java que contiene un cliente Eureka integrado. Hay que indicar que, además, Eureka expone todas sus operaciones

```
spring:
  application:
    name: my-eureka-client
server:
  port: 8080
eureka:
  client:
    serviceUrl:
      defaultZone: ${EUREKA_URI:http://localhost:8761/eureka}
```

Figura 2.2: Configuración del local host

REST que soporta mediante puntos de entrada, «endpoints», para obtener más información sobre los clientes Eureka.

3 Diagramas

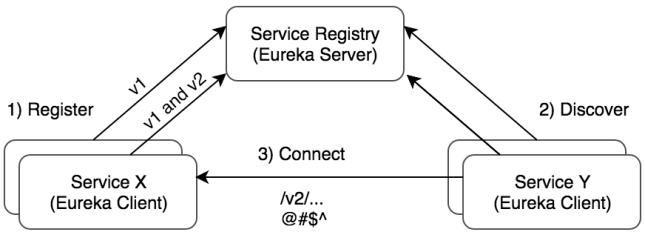

Figura 3.1: Comunicación con Eureka Server

4 Código Fuente

Bueno en base al tema se realizo la correspondiente programación tanto de la dependencias, como el host colocado al eureka, ademas de agregar un archivo de propiedades, y la aplicación en si de ellas.

4.0.1 EurekaServer

AplicaciónEurekaServer

```
    package Tecno.SpringEurekaServer;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@EnableEurekaServer
@SpringBootApplication
public class SpringEurekaServerApplication {

public static void main(String[] args) throws Exception {

SpringApplication.run(SpringEurekaServerApplication.class, args);
}

}
```

Propiedades

```
    #Application Name
spring:
  application:
    name: eureka-server

server:
  port: 8761
```

```
eureka:
  client:
    #telling the server not to register himself in the service registry
    registerWithEureka: false
    fetchRegistry: false
    serviceUrl:
      defaultZone: http://localhost:8761/eureka
  server:
    waitTimeInMsWhenSyncEmpty: 0    #wait time for subsequent sync
```

Dependencias

```
    <?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/mav
<modelVersion>4.0.0</modelVersion>
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
<version>2.2.0.BUILD-SNAPSHOT</version>
<relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>Tecno</groupId>
<artifactId>SpringEurekaServer</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>SpringEurekaServer</name>
<description>Demo project for Spring Boot</description>

<properties>
<java.version>1.8</java.version>
<spring-cloud.version>Hoxton.BUILD-SNAPSHOT</spring-cloud.version>
</properties>

<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
```

```
</dependency>
<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-eureka-server</artifactId>
</dependency>

<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
<exclusions>
<exclusion>
<groupId>org.junit.vintage</groupId>
<artifactId>junit-vintage-engine</artifactId>
</exclusion>
<exclusion>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
</exclusion>
</exclusions>
</dependency>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-contract-verifier</artifactId>
<scope>test</scope>
</dependency>
</dependencies>

<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>

<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
</plugins>
</build>

<repositories>
<repository>
<id>spring-snapshots</id>
<name>Spring Snapshots</name>
<url>https://repo.spring.io/snapshot</url>
<snapshots>
<enabled>true</enabled>
</snapshots>
</repository>
<repository>
<id>spring-milestones</id>
<name>Spring Milestones</name>
<url>https://repo.spring.io/milestone</url>
</repository>
</repositories>
<pluginRepositories>
<pluginRepository>
<id>spring-snapshots</id>
<name>Spring Snapshots</name>
<url>https://repo.spring.io/snapshot</url>
<snapshots>
<enabled>true</enabled>
</snapshots>
</pluginRepository>
<pluginRepository>
<id>spring-milestones</id>
<name>Spring Milestones</name>
<url>https://repo.spring.io/milestone</url>
</pluginRepository>
</pluginRepositories>
```

```
</project>
```

4.0.2 CountriesAplicación

AplicaciónCountries

```
    package tecno.coutriesser;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class CoutriesserApplication {

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

}
```

Propiedades

```
spring.application.name=paises-service
eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka
server.port=8000
#Configuacion JPA
spring.jpa.show-sql=true
spring.h2.console.enabled=true
```

Pom.xml

```
    <?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/200
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/m
<modelVersion>4.0.0</modelVersion>
<parent>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-parent</artifactId>
```

```
<version>2.2.0.BUILD-SNAPSHOT</version>
<relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>tecno</groupId>
<artifactId>coutriesser</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>coutriesser</name>
<description>Demo project for Spring Boot</description>

<properties>
<java.version>1.8</java.version>
<spring-cloud.version>Hoxton.BUILD-SNAPSHOT</spring-cloud.version>
</properties>

<dependencies>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.cloud</groupId>
<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>

<dependency>
<groupId>com.h2database</groupId>
<artifactId>h2</artifactId>
<scope>runtime</scope>
</dependency>
<dependency>
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<optional>true</optional>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-test</artifactId>
<scope>test</scope>
```

```
<exclusions>
<exclusion>
<groupId>org.junit.vintage</groupId>
<artifactId>junit-vintage-engine</artifactId>
</exclusion>
<exclusion>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
</exclusion>
</exclusions>
</dependency>
</dependencies>

<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>

<build>
<plugins>
<plugin>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-maven-plugin</artifactId>
</plugin>
</plugins>
</build>

<repositories>
<repository>
<id>spring-snapshots</id>
<name>Spring Snapshots</name>
<url>https://repo.spring.io/snapshot</url>
<snapshots>
<enabled>true</enabled>
</snapshots>
```

```
</repository>
<repository>
<id>spring-milestones</id>
<name>Spring Milestones</name>
<url>https://repo.spring.io/milestone</url>
</repository>
</repositories>
<pluginRepositories>
<pluginRepository>
<id>spring-snapshots</id>
<name>Spring Snapshots</name>
<url>https://repo.spring.io/snapshot</url>
<snapshots>
<enabled>true</enabled>
</snapshots>
</pluginRepository>
<pluginRepository>
<id>spring-milestones</id>
<name>Spring Milestones</name>
<url>https://repo.spring.io/milestone</url>
</pluginRepository>
</pluginRepositories>

</project>
```

5 Prerequisitos

Se recomienda tener la versión del maven de Spring 2.2.0 Snapshoot

Dependencias:

- Eureka Server
- Spring-boot-starter-web
- Actuator
- Test

6 Conclusiones

- Se puede aseverar Eureka proporciona soporte a multiregión, pudiendo definir diferentes agrupaciones de microservicios.
- Mediante el uso de el resgistro de descubrimiento Eureka se puede conocer el estado de nuestro ecosistema de microservicios: Eureka proporciona un dashboard que permite ver los microservicios existentes actualmente en el registro.
- Cualquier microservicio que sea un cliente Eureka solo necesita conocer el identificador del microservicio al que desea invocar y Eureka resolverá su localizacióna.

7 Recomendaciones

- Se recomienda realizar las dependencias en el Spring Boot, snapshoot: 2.2.0, porque si se trabaj on el modelo de ItellijIdea, este solo trabaja con la versión 2.0 en adelante.
- Comprender la naturaleza que corresponde a un servicio de Descubrimiento
- Se recomienda verificar el funcionamiento, mediante el registro de un servicio.

8 Repositorio

8.1 Git Hub

https://github.com/Jhordyb3/EurekaServer

8.2 Links Overleaf

https://es.overleaf.com/read/ckstvjngmmbm

Bibliografía

[1] Francisco Cilleruelo. Gestor de transacciones distribuidas asíncronas en arquitecturas de microservicios Máster Universitario de Investigación en Ingeniería de Software y Sistemas Informáticos Itinerario de Ingeniería de Software Gestor de transacciones distribuidas asíncronas. 2017.

[2] Silvia Mariana and Mina García. Universidad técnica del norte universidad mariana de pasto instituto de postgrado. 2012.

[3] Matías Gabriel. Desarrollo e implementación de una arquitectura horizontalmente escalable. 2017.

[4] D Barredo Gil. Escuela politécnica de ingeniería de gijón. 2019.