**Spring Cloud and Spring Boot, Part 1: Implementing Eureka Server**

## What is Eureka Server?

Eureka Server is service discovery for your microservices, where all client applications can register by themselves and other microservices look up the Eureka Server to get independent microservices to get the job complete.

Eureka Server is also known as Discovery Server and it contains all the information about client microservices running on which IP address and port.

To achieve this you need to create a Eureka Server application and add the below dependency in POM.xml.

<dependency>

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

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

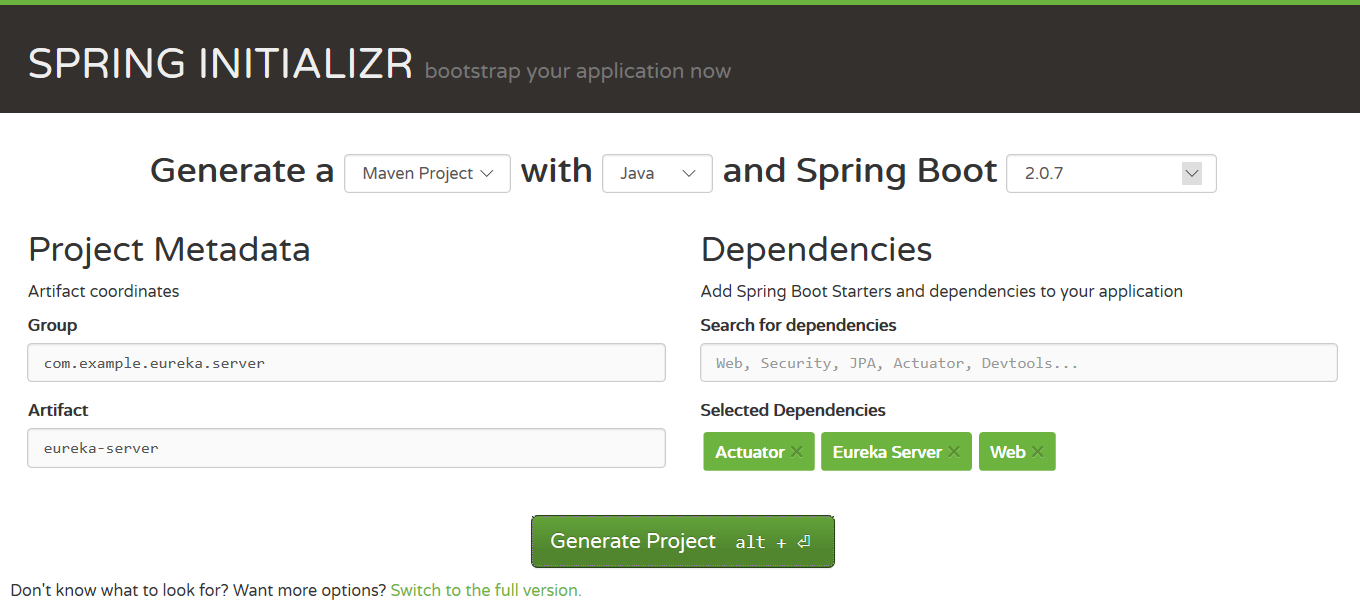
</dependency>

## Implementing Eureka Server

### **Creating Eureka Server Spring Application**

Navigate to<https://start.spring.io/> and create a project template. Provide project metadata like Group, Artifact, and add the below dependencies/modules. Click Generate Project and it will download .zip Spring-based project and you can unzip the downloaded project and import in Eclipse as a Maven project.

* Eureka Server
* Web
* Actuator



### **POM.xml**

Verify the POM.xml and looks like it does below:

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

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

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

<version>2.0.7.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.example.eureka.server</groupId>

<artifactId>eureka-server</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>eureka-server</name>

<description>Demo project for Spring Boot</description>

<properties>

<java.version>1.8</java.version>

<spring-cloud.version>Finchley.SR2</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-server</artifactId>

</dependency>

<dependency>

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

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

</project>

### **Spring Boot Starter Application**

Now you need to open EurekaServerApplication.javaand add the annotation @EnableEurekaServer   on the top of the class as shown below.

package com.example.eureka.server.eurekaserver;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication

@EnableEurekaServer

public class EurekaServerApplication {

public static void main(String[] args) {

SpringApplication.run(EurekaServerApplication.class, args);

}

}

### **Application Properties File**

You need to add the below list of properties in application.properties located at src/main/resources on your application.

spring.application.name=eureka-server

server.port=8761

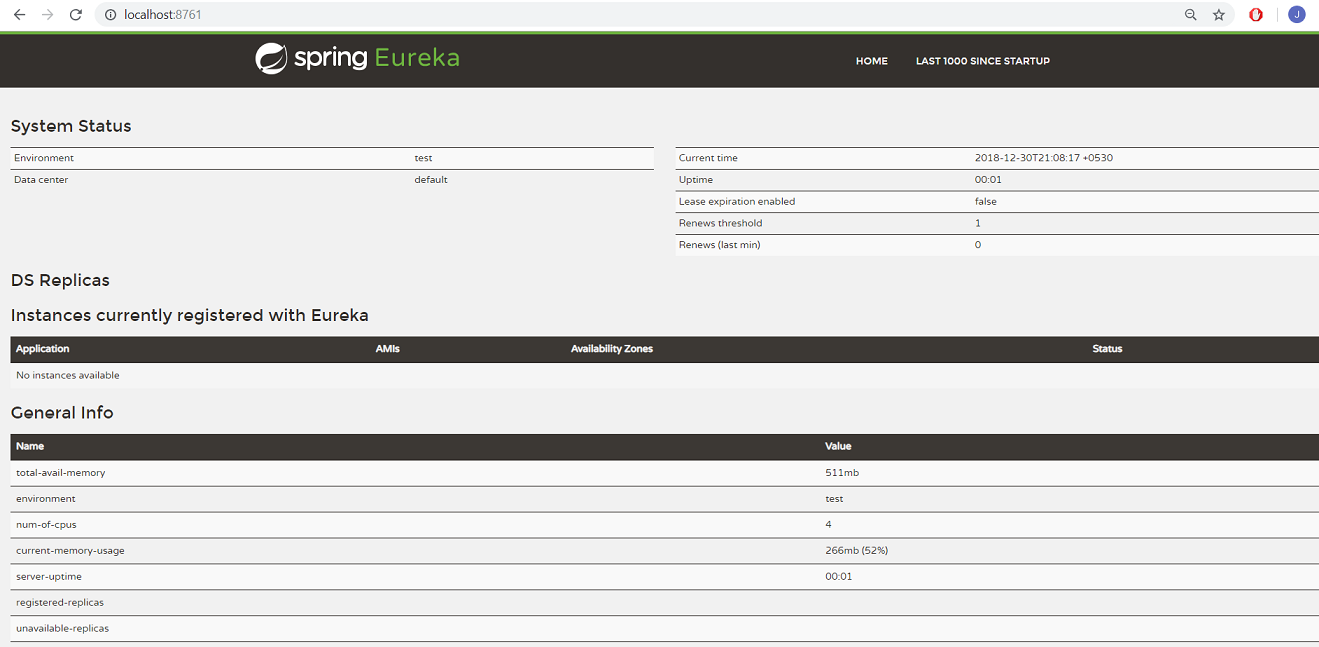
eureka.client.register-with-eureka=false

eureka.client.fetch-registry=false

* **spring.application.name** is a unique name for your application.
* **server.port** in which your application will be bound and wewill use default port 8761 for eureka server.
* **eureka.client.fetch-registry** doesn't register itself in eureka server.
* **eureka.client.register-with-eureka** is determines if service register itself as a client in eureka server.

### **Running the Eureka Server**

Run the Eureka server as Java application and go the URL: http://localhost:8761/



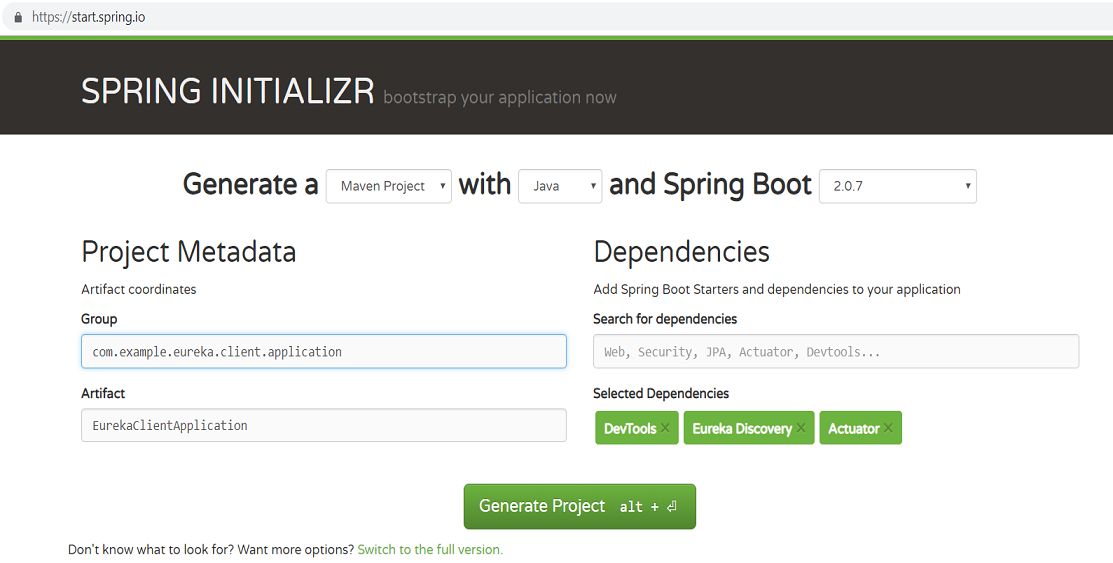
You can see the Eureka server is up and running but no application is registered with it yet.

## Registering Client Application With Eureka Server

### **Creating Spring Boot Client Application**

Navigate to<https://start.spring.io/> and create a project template. Provide project metadata like Group, Artifact and add below dependencies/modules. Click Generate Project and it will download a .zip Spring-based project and you can unzip the downloaded project and import it to Eclipse as Maven project.

* DevTools
* Actuator
* Discovery Client



### **POM.xml**

Verify the POM.xml looks as it does below:

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

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

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

<version>2.0.7.RELEASE</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.example.eureka.client</groupId>

<artifactId>EurekaClientApplication</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>EurekaClientApplication</name>

<description>Demo project for Spring Boot</description>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>

<java.version>1.8</java.version>

<spring-cloud.version>Finchley.SR2</spring-cloud.version>

</properties>

<dependencies>

<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>

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

<dependency>

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

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

</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>

</project>

### **Java Spring Server Application**

Now you need to open the SpringBootApplication EurekaClientApplication.java and add the annotation @EnableDiscoveryClient   on the top of the class as shown below.

package com.example.eureka.client.application;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication

@EnableDiscoveryClient

public class EurekaClientApplication{

public static void main(String[] args) {

SpringApplication.run(EurekaClientApplication.class, args);

}

}

### **Adding REST Controller**

You need to add the class HelloWorldController  under package com.example.eureka.client.application and implement GET method in the class.

package com.example.eureka.client.application;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class HelloWorldController {

@GetMapping("/hello-worlds/{name}")

public String getHelloWorld (@PathVariable String name)

{

return "Hello World "+name;

}

}

### **Application Properties File**

You need to add the below list of properties in application.properties located at src/main/resources.

spring.application.name=eureka-client-service

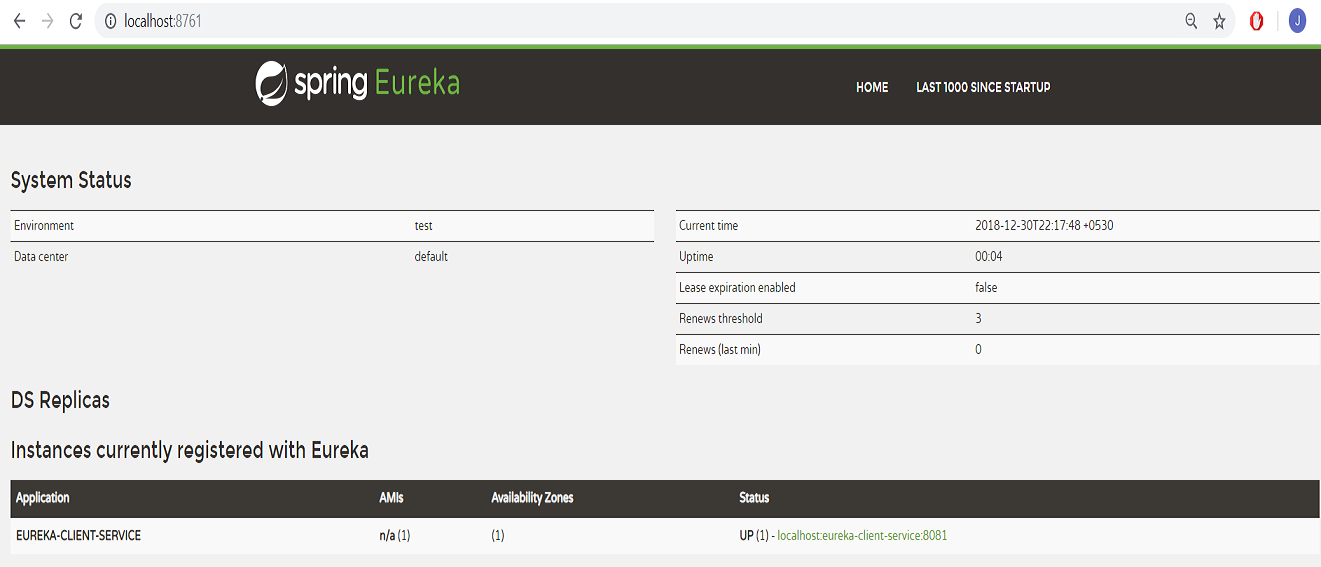
server.port=8081

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

**eureka.client.service-url.defaultZone** determines the address where the Eureka Server is running so the client application can register itself in Eureka Server.

### **Running Client Application**

Before running the application, you need to make sure the Eureka Server is up and running. Run the client application as a Java application and navigate to the Eureka Server at http://localhost:8761/. This time you should see that client application registered in Eureka Server.



This is how you can implement [Eureka Server](https://dzone.com/articles/spring-cloud-amp-spring-bootimplementing-zipkin-se) for your microservices