**MicroServices with Spring boot**

A **microservice** architecture – a variant of the [SOA](https://en.wikipedia.org/wiki/Service-oriented_architecture) structural style – is an [architectural pattern](https://en.wikipedia.org/wiki/Architectural_pattern) that arranges an application as a collection of [loosely-coupled](https://en.wikipedia.org/wiki/Loose_coupling), [fine-grained](https://en.wikipedia.org/wiki/Granularity) services, communicating through [lightweight protocols](https://en.wikipedia.org/wiki/Lightweight). One of its goals is that teams can develop and [deploy](https://en.wikipedia.org/wiki/Software_deployment) their services independently of others. This is achieved by the reduction of several [dependencies](https://en.wikipedia.org/wiki/Coupling_(computer_programming)) in the code base, allowing for developers to evolve their services with limited restrictions from users, and for users to be hidden from additional complexity. As a consequence, organizations are able to develop software with fast growth and size, as well as use off-the-shelf services more easily. Communication requirements are reduced. These benefits come at a cost to maintaining the decoupling. Interfaces need to be designed carefully and treated as a public API.

In this article we will create two small microservices and discover them using eureka server.

Step 1-Creating microservices

Create at least two spring boot application. These applications will be turned into microservices.For this tutorial I have used maven based spring boot application using java 11.

Step 2-Microservice communiction

Microservices are distributed in nature and communicate with each other in various way.In this step we well see two different ways by which microservices can consume external API-

Using Rest template-

i)include following dependency in pom.xml-

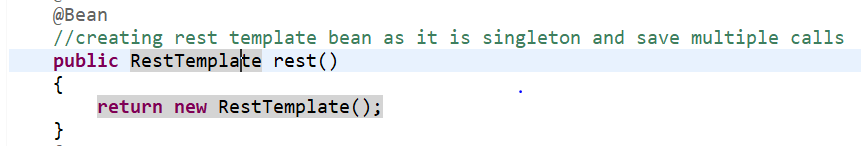
<dependency>

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

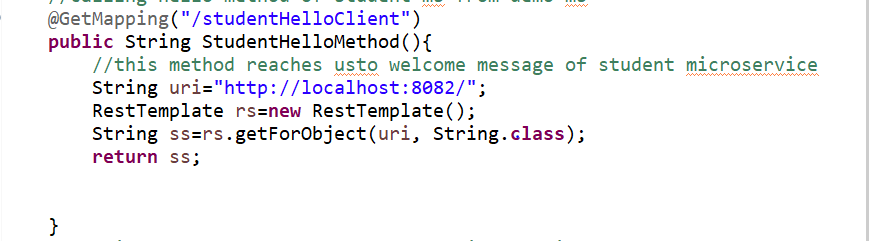
<artifactId>spring-boot-starter-web</artifactId>

</dependency>

ii)Create RestTemplate bean in main class and autowire it.



iii)using RestTemplate to call API from another microservice-



iv)Now you can run the application and it API /studentHelloClient you will be able to hit other microservice running in port 8082

Using Web Client

i)Include dependency in pom.xml-

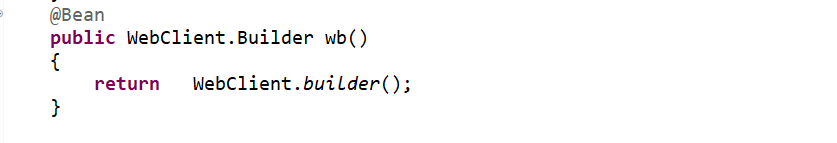
<dependency>

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

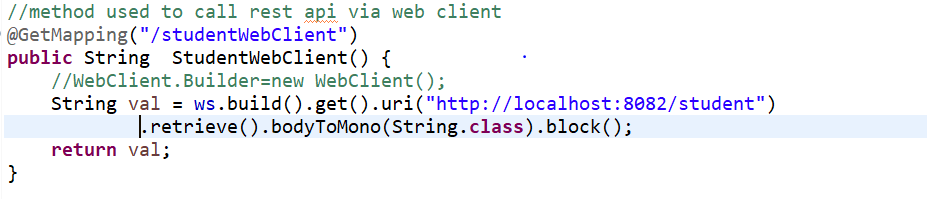
<artifactId>spring-boot-starter-webflux</artifactId>

</dependency>

ii)create web client bean and autowire it



iii)use web client to call API from another microservice-



iv) Now build the application and hit /studentWebClient you will be able to access application running in port 8082

Note-Individual microservices should be running on different ports and application name should be added in application.properties-



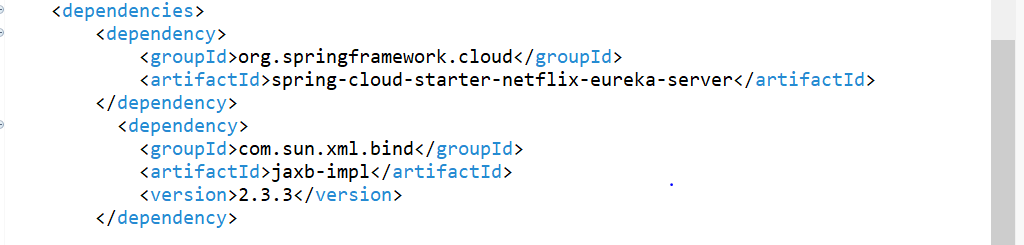
Step 3-Creating Eureka server

Eureka server is basically a spring boot application with specific dependencies and annotation which makes it eureka server.

To create eureka server-

i)setup a simple springboot project

ii)Add below dependencies in pom.xml-

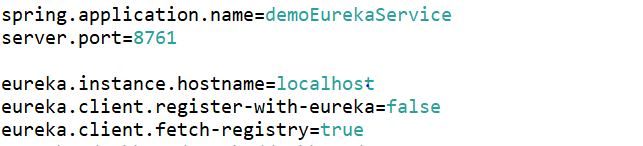


iii)Add following in dependency management tab to import spring cloud dependencies-

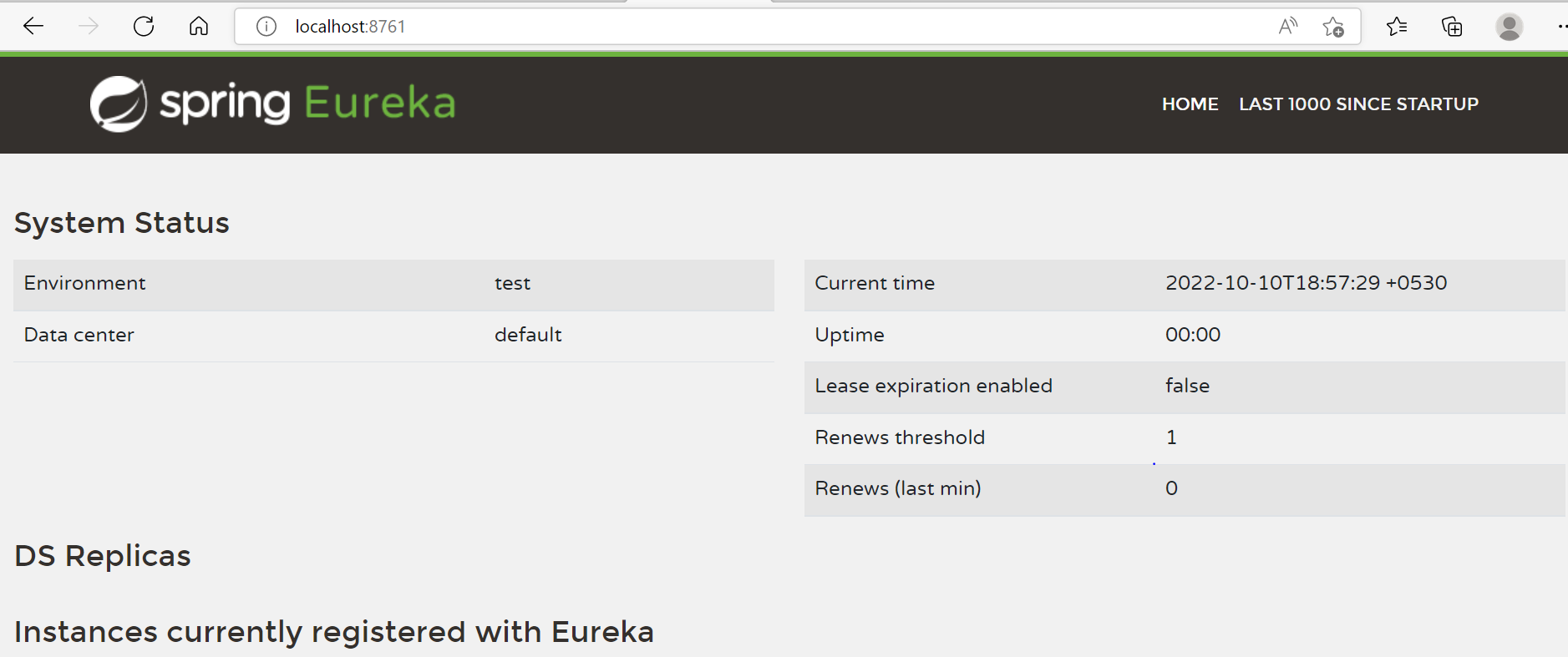


iv)Add annotation @EnableEurekaServer in main class.

v)Add following properties in application.properties-



vi) Now run application and hit url-[localhost](http://localhost:8761/) you will be able to access default page of eureka server-



Step 4-Service Discovery-

Service Discovery is the process of how microservices discover each other over a network. There are two main components of it in terms of Eureka service:

* **Eureka server (service registry):**It is a server that stores the addresses (host and ports) of all the registered microservices.
* **Eureka Client:** Its a microservice registered on the central server and it updates and retrieves addresses to/from the central Eureka server.To create eureka client –

i)Add below dependency in pom.xml your microservices-

<dependency>

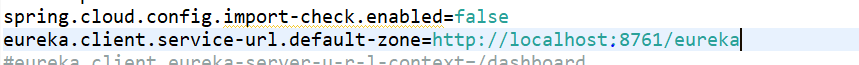
<groupId>org.springframework.cloud</groupId>

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

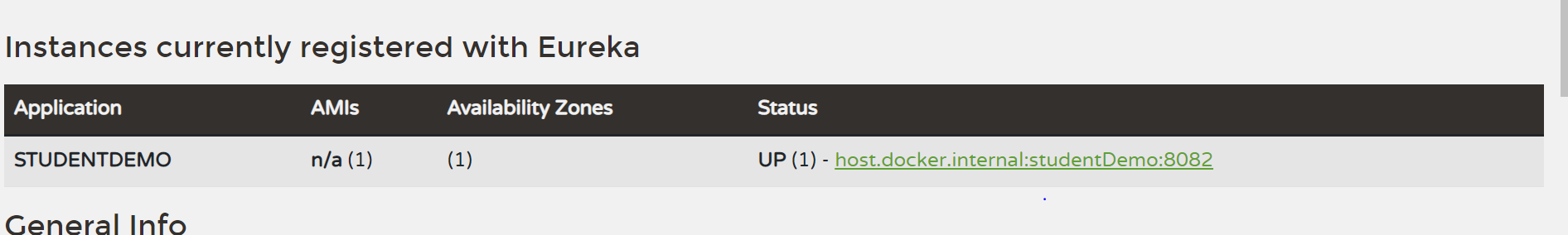
</dependency>

ii)Add the annotation in main class @EnableEurekaClient.In java 11 this is a optional step.

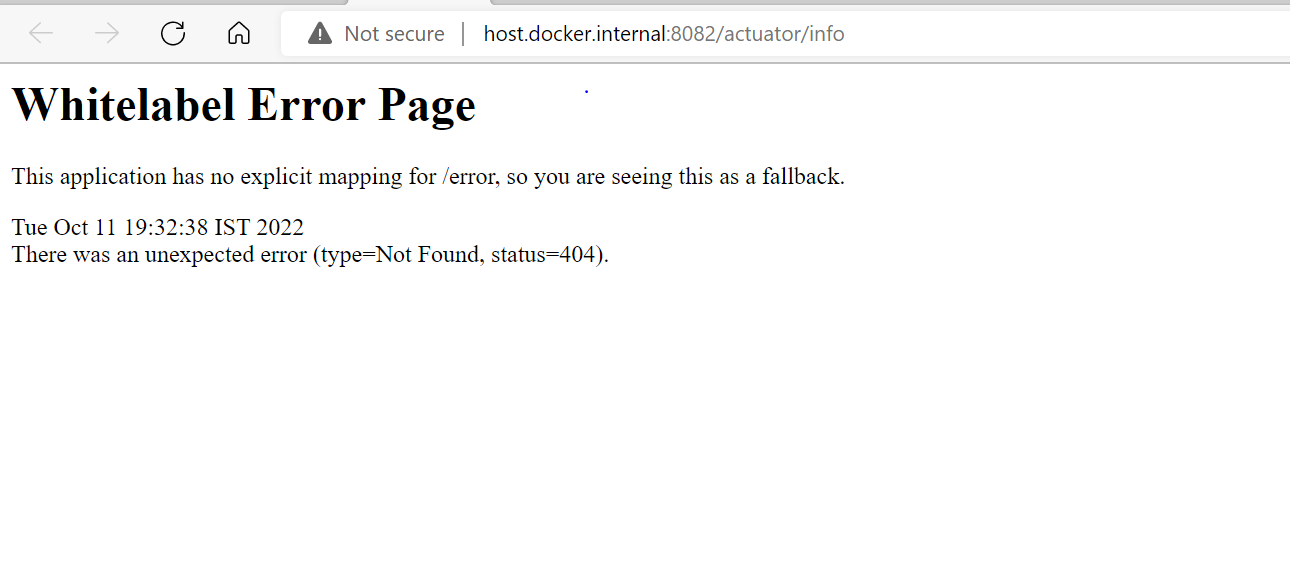
iii)Add following in application.properties-



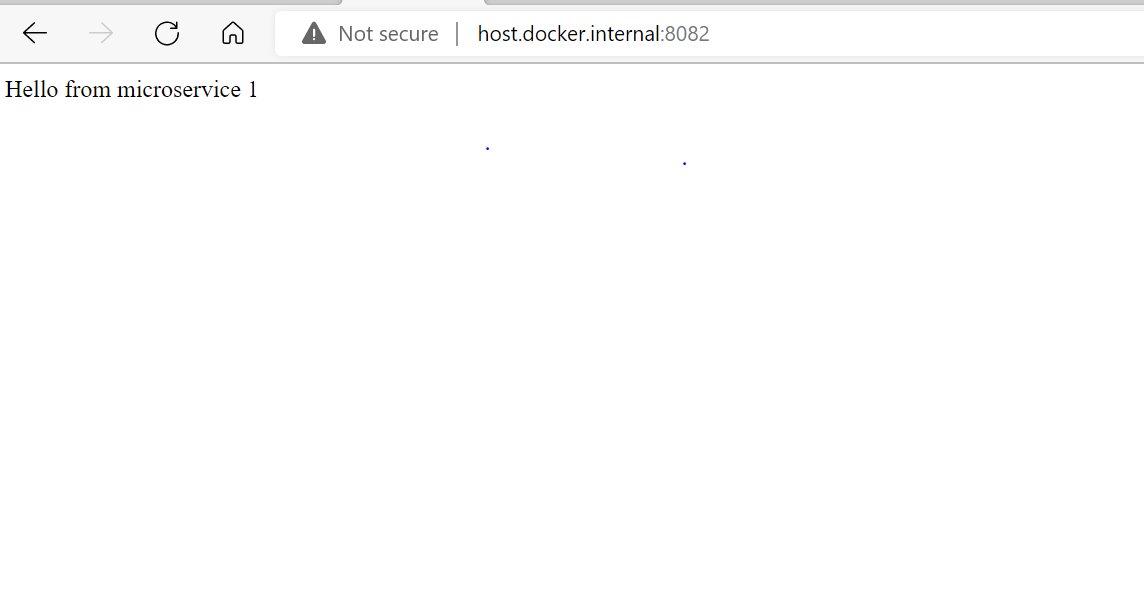
Run the application along with eureka server,you will be able to see your client instance name in eureka’s page-



Now you can use URL details from instance to access microservice-



Modify above url to hit application-



These steps will help you create simple microservices and enable communication among them as well as Eureka server.

You can further use below links for understanding microservices-

* <https://www.youtube.com/watch?v=y8IQb4ofjDo&list=PLqq-6Pq4lTTZSKAFG6aCDVDP86Qx4lNas&index=1>
* [Service Discovery using Eureka in Spring Microservices - Studytonight](https://www.studytonight.com/post/service-discovery-using-eureka-in-spring-microservices#:~:text=Eureka%20provides%20service%20discovery%20in%20a%20microservices%20architecture.,registered%20microservices%20become%20available%20for%20the%20registered%20service.)