**Module 5: Assignment -Microservices with Springboot and Cloud**

## ****1. Introduction****

To implement as per task, microservices design patterns using Spring Cloud:

* **Service Registry and Discovery** using Eureka Server
* **Client-Side Load Balancing** using Spring Cloud LoadBalancer
* **API Gateway** using Spring Cloud Gateway

## ****2. Service Registry and Discovery (Eureka Server)****

### ****Step 1: Create a Eureka Server****

#### ****1. Create a Spring Boot project****

* Use Spring Initializr and include Eureka Server dependency.

#### ****2. Configure Eureka Server****

spring:

application:

name: EduEurekaServer

server:

port: 8761

eureka:

client:

registerWithEureka: false

fetchRegistry: false

server:

waitTimeInMsWhenSyncEmpty: 0

enableSelfPreservation: false

#### ****3. Enable Eureka Server****

* In the main class:

package com.eduexcelence.edueurekaserver;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

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

@SpringBootApplication

@EnableEurekaServer

public class EduEurekaServerApplication {

public static void main(String[] args) {

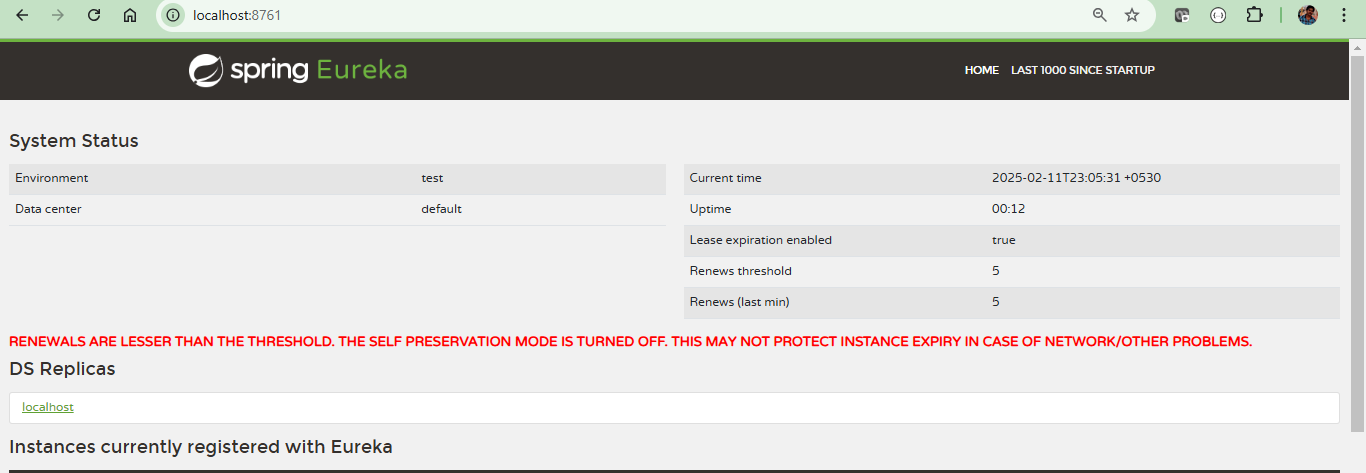
SpringApplication.run(EduEurekaServerApplication.class, args);

}

}

#### ****4. Run Eureka Server****

* Start the application and access http://localhost:8761 to see the Eureka Dashboard.



## ****3. Registering Microservices with Eureka****

### ****Step 1: Create a Microservice (StudentMs service , FeesMS and Auth Server)****

Include dependencies: Spring Boot Web, Eureka Client

<dependency>

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

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

</dependency>

### ****Step 2: Configure Microservices****

* In application.properties of StudentMS:

spring.application.name=studentms

spring.cloud.discovery.enabled=true

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

eureka.client.register-with-eureka=true

eureka.client.fetch-registry=true

* In application.properties of FeesMS:

spring.application.name=feesms

spring.cloud.discovery.enabled=true

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

eureka.client.register-with-eureka=true

eureka.client.fetch-registry=true

In application.properties of AuthService:

spring.application.name= authservice

spring.cloud.discovery.enabled=true

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

eureka.client.register-with-eureka=true

eureka.client.fetch-registry=true

### ****Step 3: Enable Eureka Client****

1. In StudentMsApplication.java

@SpringBootApplication

@EnableDiscoveryClient

**public** **class** StudentMsApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(StudentMsApplication.**class**, args);

}

}

1. In FeesmsApplication

@SpringBootApplication

@EnableDiscoveryClient

**public** **class** FeesmsApplication {

**public** **static** **void** main(String[] args) {

SpringApplication.*run*(FeesmsApplication.**class**, args);

}

}

1. In AuthserviceApplication application,

@SpringBootApplication

@EnableDiscoveryClient

**public** **class** AuthserviceApplication {

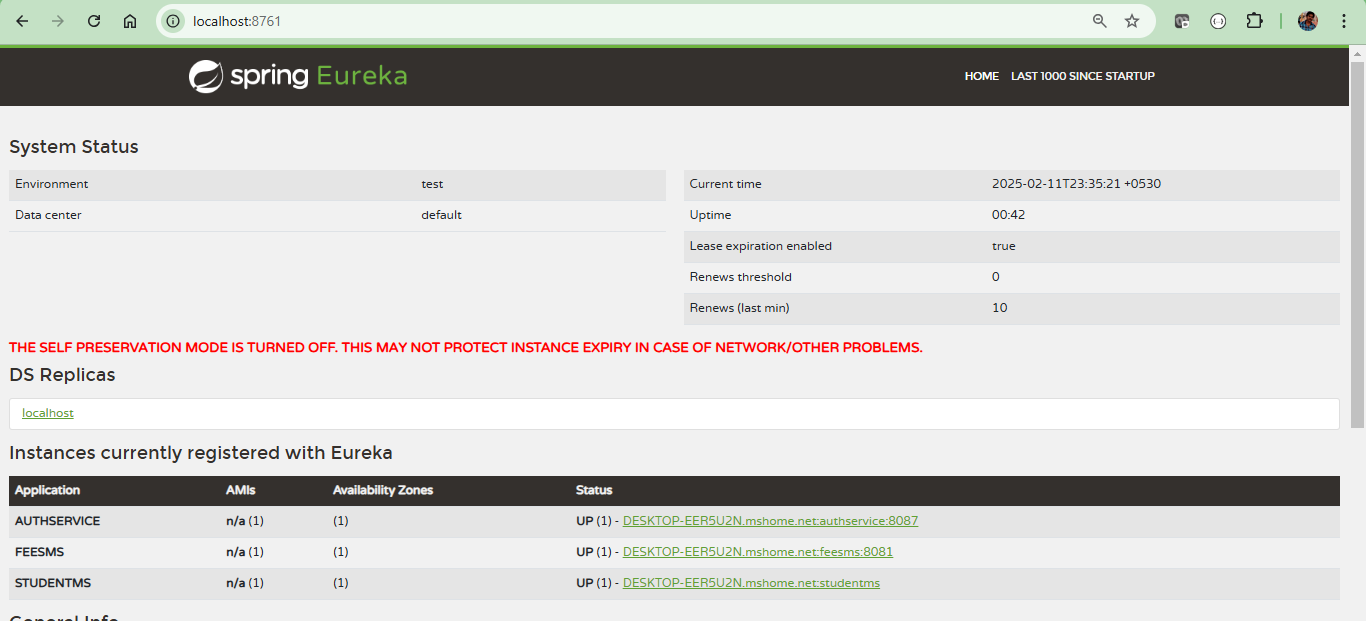
**public** **static** **void** main(String[] args) {

SpringApplication.*run*(AuthserviceApplication.**class**, args);

}

}

Start both services and verify their registration in <http://localhost:8761>.



## ****4. Client-Side Load Balancing with Spring Cloud LoadBalancer****

### ****Step 1: Enable Load Balancing in a Client Service****

In a @RestController class:

@RestController

public class ClientController {

@Autowired

private WebClient.Builder webClientBuilder;

@GetMapping("/fetch-data")

public String fetchData() {

return webClientBuilder.build()

.get()

.uri("http://studentms")

.retrieve()

.bodyToMono(String.class)

.block();

}

}

### ****Step 2: Configure WebClient LoadBalancer****

* In application.yml:

spring:

cloud:

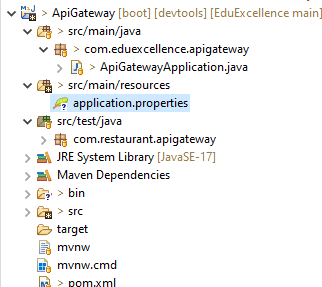
loadbalancer:

enabled: true

## ****API Gateway with Spring Cloud Gateway****

### ****Step 1: Create an API Gateway Service****

* API Gate Way Service : ApiGateWay
* Add dependency: Spring Cloud Gateway, Eureka Client



### ****Step 2: Configure Routes in**** application.properties

spring.application.name=gateway

# API Gateway will run on port 8080

server.port=9000

spring.security.user.name=admin

spring.security.user.password=password

# Eureka Client Configuration

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

eureka.client.register-with-eureka=true

# Spring Cloud Gateway Routes

spring.cloud.gateway.routes[0].id=studentms

spring.cloud.gateway.routes[0].uri=lb://stundentms

spring.cloud.gateway.routes[0].predicates[0]=Path=/api/v1/\*\*

spring.cloud.gateway.routes[1].id=feesms

spring.cloud.gateway.routes[1].uri=lb://feesms

spring.cloud.gateway.routes[1].predicates[0]=Path=/api/v1/\*\*

### ****Step 3: Enable Discovery Client in API Gateway****

In ApiGatewayApplication.java:

@SpringBootApplication

@EnableDiscoveryClient

public class ApiGatewayApplication {

public static void main(String[] args) {

SpringApplication.run(ApiGatewayApplication.class, args);

}

}

### ****Step 4: Run API Gateway and Test Routes****

* Start API Gateway and test routes:
  + http://localhost:9000/studentms/data
  + http://localhost:9000/fessms/info

## ****6. Expected Results****

| **Service** | **URL** | **Expected Response** |
| --- | --- | --- |
| Studentms | http://localhost:8081/api/v1 | "Response from Service Student Ms" |
| feesms | <http://localhost:8082/api/v1> | "Response from Service Fees MS" |
| API Gateway | http://localhost:9000/studentms | "Response from Service " |
| API Gateway | http://localhost:9000/fees | "Response from Service B" |
|  |  |  |

### ****Screenshots****

### 

