**Digital Nurture 4.0 Deep Skilling - Java FSE**  
**WEEK –5 Additional Hands-on Exercises  
Module 8 - Microservices with Spring Boot 3 and Spring Cloud**

**Microservices with API gateway**

**B.) Create Eureka Discovery Server and register**

**api\_gateway.filter**

**LogFilter.java**

package com.cognizant.api\_gateway.filter;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.cloud.gateway.filter.GlobalFilter;

import org.springframework.core.Ordered;

import org.springframework.stereotype.Component;

import org.springframework.web.server.ServerWebExchange;

import reactor.core.publisher.Mono;

import org.springframework.cloud.gateway.filter.GatewayFilterChain;

@Component

public class LogFilter implements GlobalFilter, Ordered {

private static final Logger logger = LoggerFactory.getLogger(LogFilter.class);

@Override

public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {

logger.info("Request Path: {}", exchange.getRequest().getPath());

return chain.filter(exchange);

}

@Override

public int getOrder() {

return -1;

}

}

**ApiGatewayApplication.java**

package com.cognizant.api\_gateway;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

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

@SpringBootApplication

@EnableDiscoveryClient

public class ApiGatewayApplication {

public static void main(String[] args) {

SpringApplication.run(ApiGatewayApplication.class, args);

}

}

**application.properties**

server.port=9090

spring.application.name=api-gateway

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

spring.cloud.gateway.server.webflux.discovery.locator.enabled=true

spring.cloud.gateway.server.webflux.discovery.locator.lower-case-service-id=true

**ApiGatewayApplicationTests.java**

package com.cognizant.api\_gateway;

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest

class ApiGatewayApplicationTests {

@Test

void contextLoads() {

}

}

**EurekaServerApplicationFolder**

**EurekaServerApplication.java**

package com.cognizant.eureka\_server;

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**

spring.application.name=eureka-server

server.port=8761

eureka.client.register-with-eureka=false

eureka.client.fetch-registry=false

logging.level.com.netflix.eureka=OFF

logging.level.com.netflix.discovery=OFF

**GreetController.java**

package com.cognizant.greet\_service.controller;

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

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

@RestController

public class GreetController {

@GetMapping("/greet")

public String greet() {

return "Hello World";

}

}

**GreetServiceApplication.java**

package com.cognizant.greet\_service;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

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

@SpringBootApplication

@EnableDiscoveryClient

public class GreetServiceApplication {

public static void main(String[] args) {

SpringApplication.run(GreetServiceApplication.class, args);

}

}

**GreetServiceApplicationTests.java**

package com.cognizant.greet\_service;

import org.junit.jupiter.api.Test;

import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest

class GreetServiceApplicationTests {

@Test

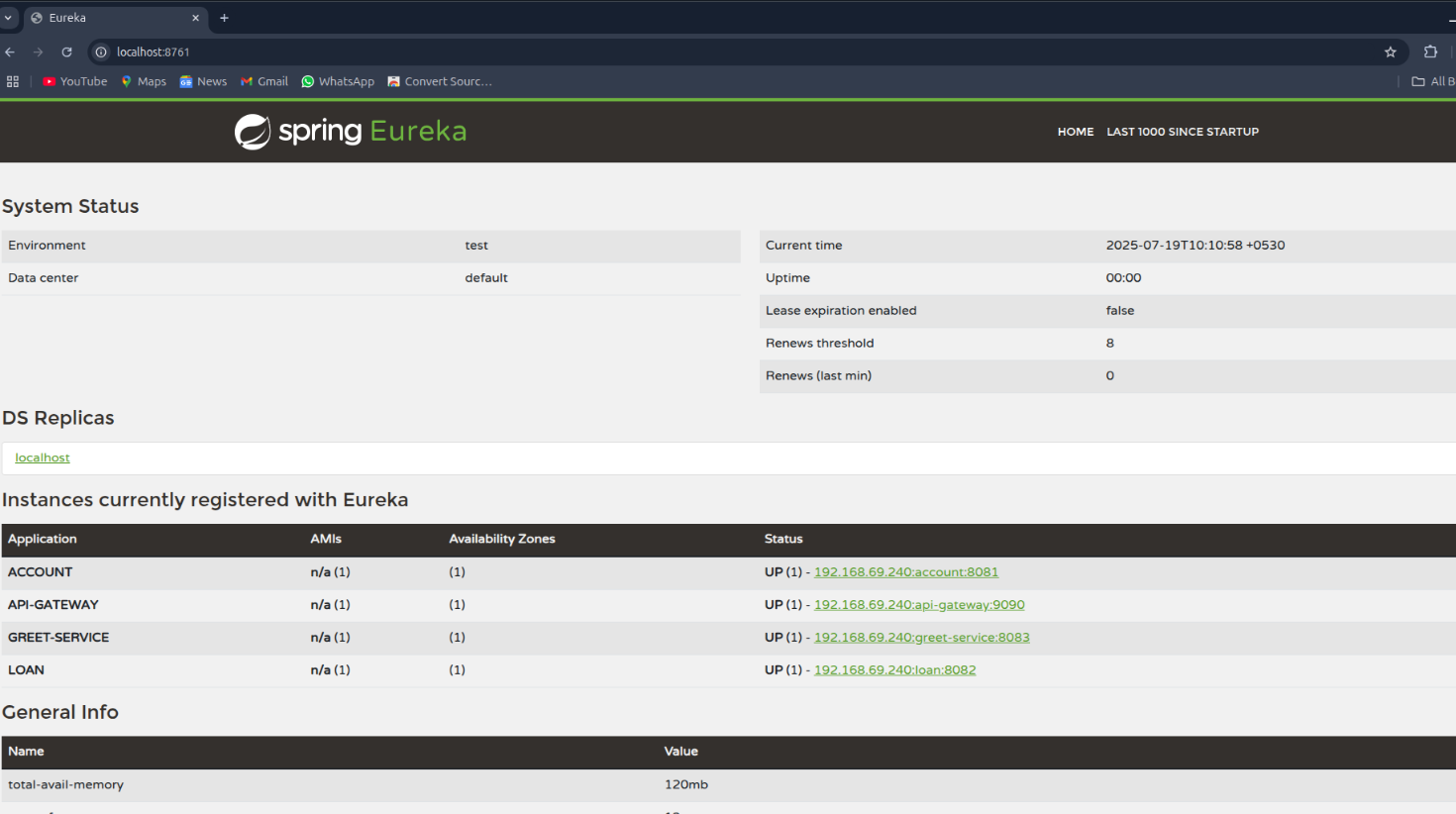
void contextLoads() {

}

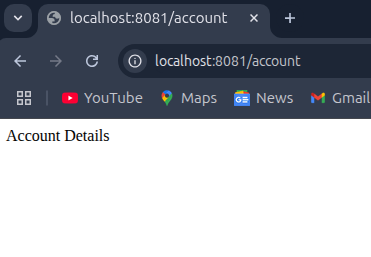
}

**OUTPUT**

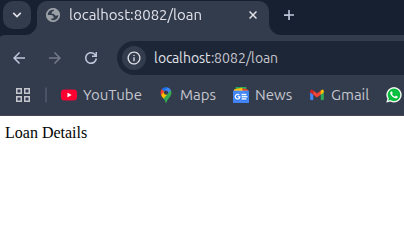
**Eureka.png**

****

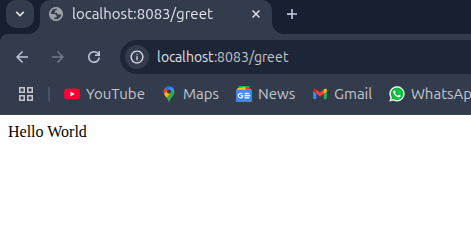
**Account.png**

****

**Loan.png**

****

**greet.png**

****