# Week 5 Hands-On:

# Microservices with Eureka Server and API Gateway:

**Creating Microservices for account and loan**

Student Name: Jitesh Sen

Email: jiteshsen2002@gmail.com

Superset ID: 6387362

**Objective:** Created two Spring Boot microservices (account-service and loan-service), register them in a Eureka Discovery Server, and expose them through a Spring Cloud API Gateway with global request logging.

## 1. Summary of Solution

This week’s exercise implements a fully functional microservices architecture consisting of:

* **Account Service** (port 8082): Provides a REST endpoint /accounts/{number} returning dummy account details.
* **Loan Service** (port 8081): Provides a REST endpoint /loans/{number} returning dummy loan details.
* **Eureka Discovery Server** (port 8761): Acts as a registry for both microservices.
* **API Gateway** (port 9090): Routes incoming requests to the appropriate service, supports discovery-based and static routing, and logs every request via a global filter.

Key achievements:

* Services registered and visible in Eureka dashboard.
* Direct access to each microservice via its port.
* Routing via API Gateway functioning with both dynamic discovery and explicit routes.
* Global log filter prints each request path in gateway console.

## 2. Steps & Configuration

### 2.1 Eureka Discovery Server

* **application.yml**:

server:  
 port: 8761  
  
eureka:  
 client:  
 register-with-eureka: false  
 fetch-registry: false

* **Main Class** annotated with @EnableEurekaServer.

### 2.2 Account Service

* **application.yml**:

server:  
 port: 8082  
  
spring:  
 application:  
 name: account-service  
  
eureka:  
 client:  
 service-url:  
 defaultZone: http://localhost:8761/eureka  
 instance:  
 homePageUrl: http://localhost:9090/account-service  
 statusPageUrl: http://localhost:9090/account-service/actuator/health  
 healthCheckUrl: http://localhost:9090/account-service/actuator/health

* **Controller**:

@RestController  
@RequestMapping("/accounts")  
public class AccountController {  
 @GetMapping("/{number}")  
 public Map<String,Object> getAccount(@PathVariable String number) { ... }  
}

### 2.3 Loan Service

* **application.yml**:

server:  
 port: 8081  
  
spring:  
 application:  
 name: loan-service  
  
eureka:  
 client:  
 service-url:  
 defaultZone: http://localhost:8761/eureka  
 instance:  
 homePageUrl: http://localhost:9090/loan-service  
 statusPageUrl: http://localhost:9090/loan-service/actuator/health  
 healthCheckUrl: http://localhost:9090/loan-service/actuator/health

* **Controller**:

@RestController  
@RequestMapping("/loans")  
public class LoanController {  
 @GetMapping("/{number}")  
 public Map<String,Object> getLoan(@PathVariable String number) { ... }  
}

### 2.4 API Gateway

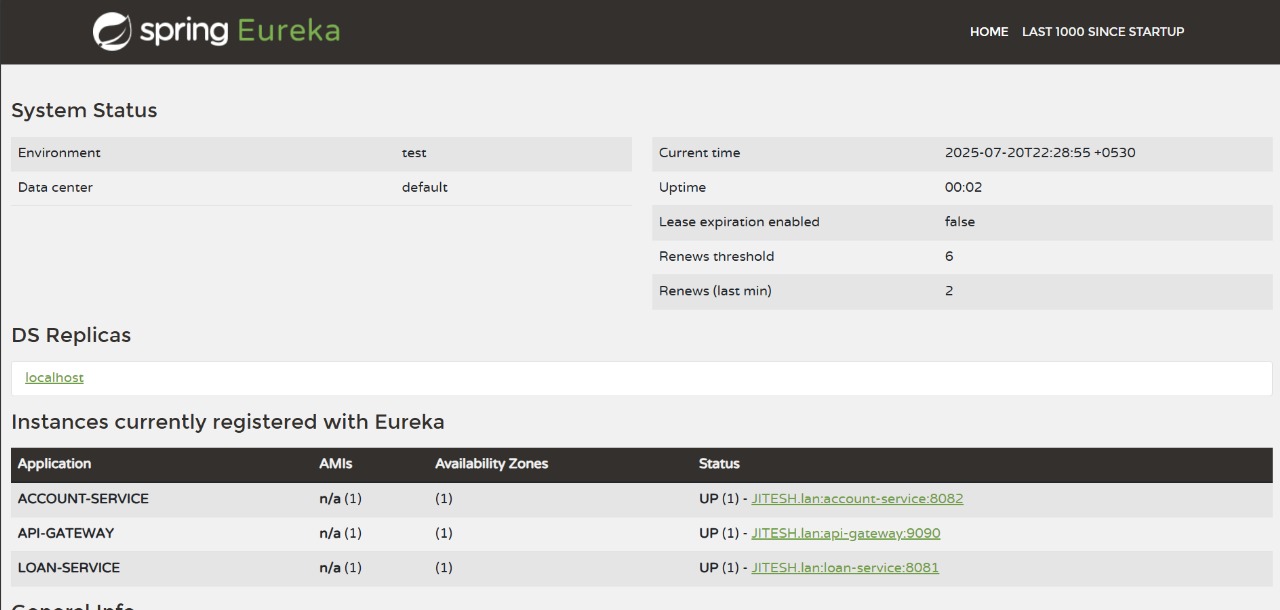
* **application.yml** with explicit routes and discovery:

server:  
 port: 9090  
  
spring:  
 application:  
 name: api-gateway  
  
 cloud:  
 discovery:  
 client:  
 service-url:  
 defaultZone: http://localhost:8761/eureka  
  
 gateway:  
 routes:  
 - id: account-service  
 uri: lb://account-service  
 predicates:  
 - Path=/account-service/\*\*  
 - id: loan-service  
 uri: lb://loan-service  
 predicates:  
 - Path=/loan-service/\*\*  
  
 discovery:  
 locator:  
 enabled: true  
 lower-case-service-id: true  
  
management:  
 endpoints:  
 web:  
 exposure:  
 include: '\*'

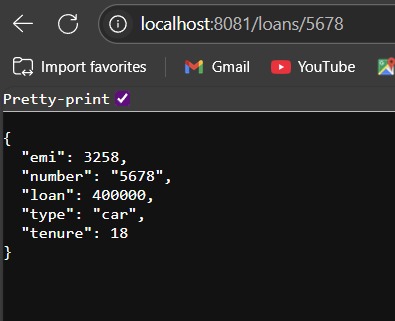
* **LogFilter.java** implements GlobalFilter, Ordered to log each request.

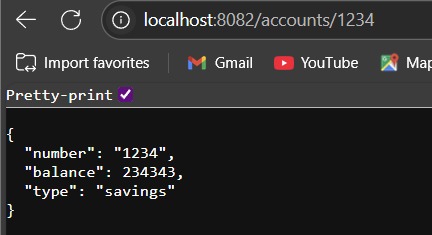
## 3. Screenshots

1. **Eureka Dashboard** – Both services registered

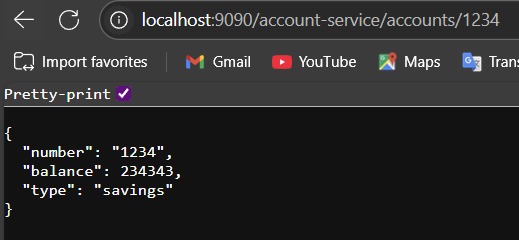
****

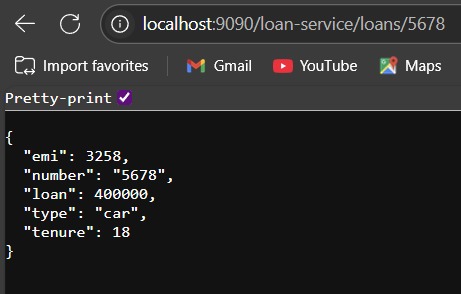
1. **Direct Service Access** – JSON response from ACCOUNT and LOAN endpoints



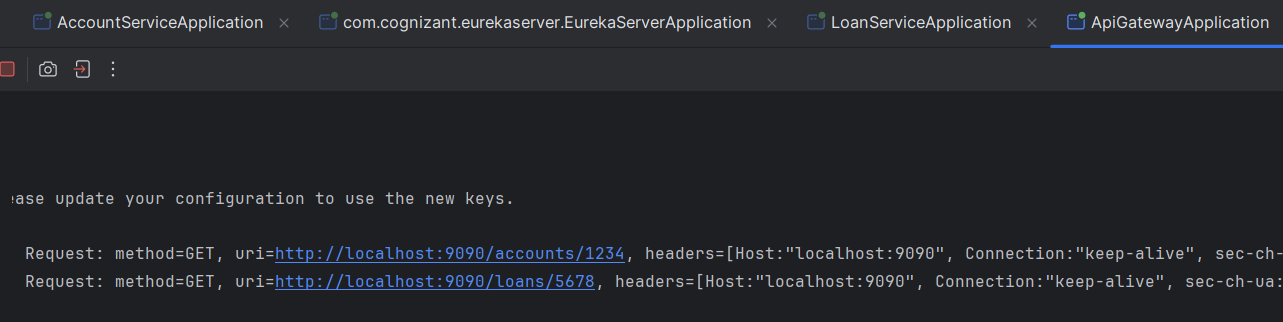


1. **API Gateway Routing** – JSON response via gateway

****



1. **Gateway Console Logs** – LogFilter output

**