**Microservices with Spring Boot 3 and Spring Cloud**

**Create Eureka Discovery Server and register microservices**

1. **Create and Launch Eureka Discovery Server**

**Pom.xml**

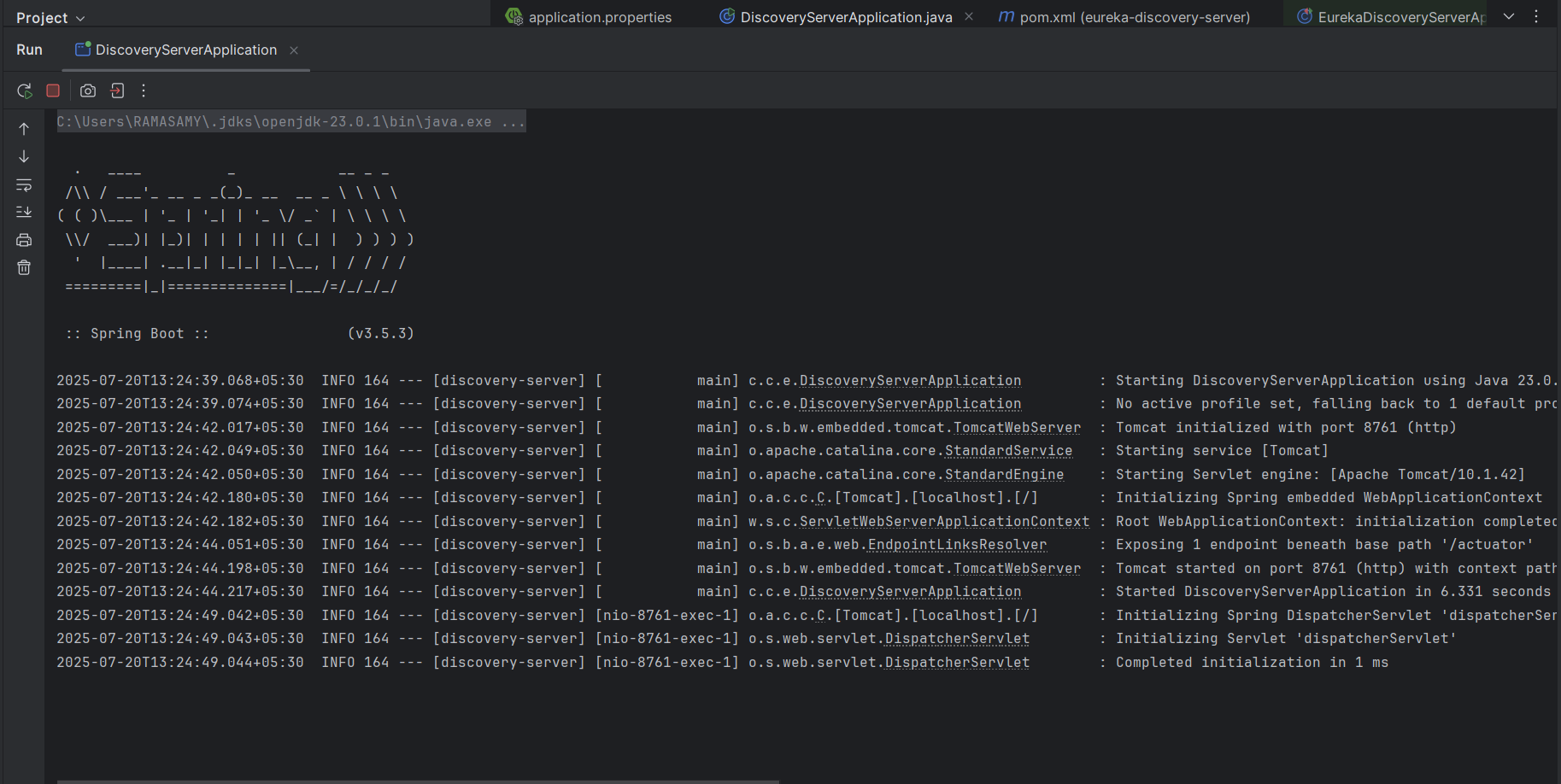
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <groupId>com.example</groupId>  
 <artifactId>eureka-server</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>Eureka Server</name>  
 <description>Eureka Discovery Server</description>  
 <properties>  
 <java.version>17</java.version>  
 <spring-cloud.version>2021.0.8</spring-cloud.version>  
 </properties>  
  
 <dependencies>  
  
 <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-actuator</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>

**Application.properties**

server.port=8761  
spring.application.name=discovery-server  
eureka.client.register-with-eureka=false  
eureka.client.fetch-registry=false

**DiscoveryServerApplication.java**

package com.cognizant.eureka\_discovery\_server;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;  
  
@SpringBootApplication  
@EnableEurekaServer  
public class DiscoveryServerApplication {  
 public static void main(String[] args) {  
 SpringApplication.*run*(DiscoveryServerApplication.class, args);  
 }  
}

**OUTPUT** ****

1. **Register Account REST API to eureka discovery**

**Account**

**Controller** 🡪 **AccountController.java**

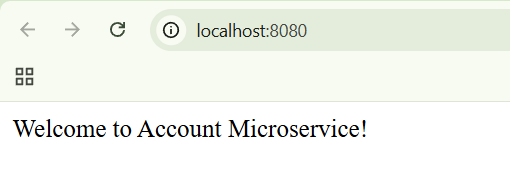
package com.cognizant.account;  
  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
@RestController  
public class AccountController {  
  
 @GetMapping("/")  
 public String home() {  
 return "Welcome to Account Microservice!";  
 }  
}

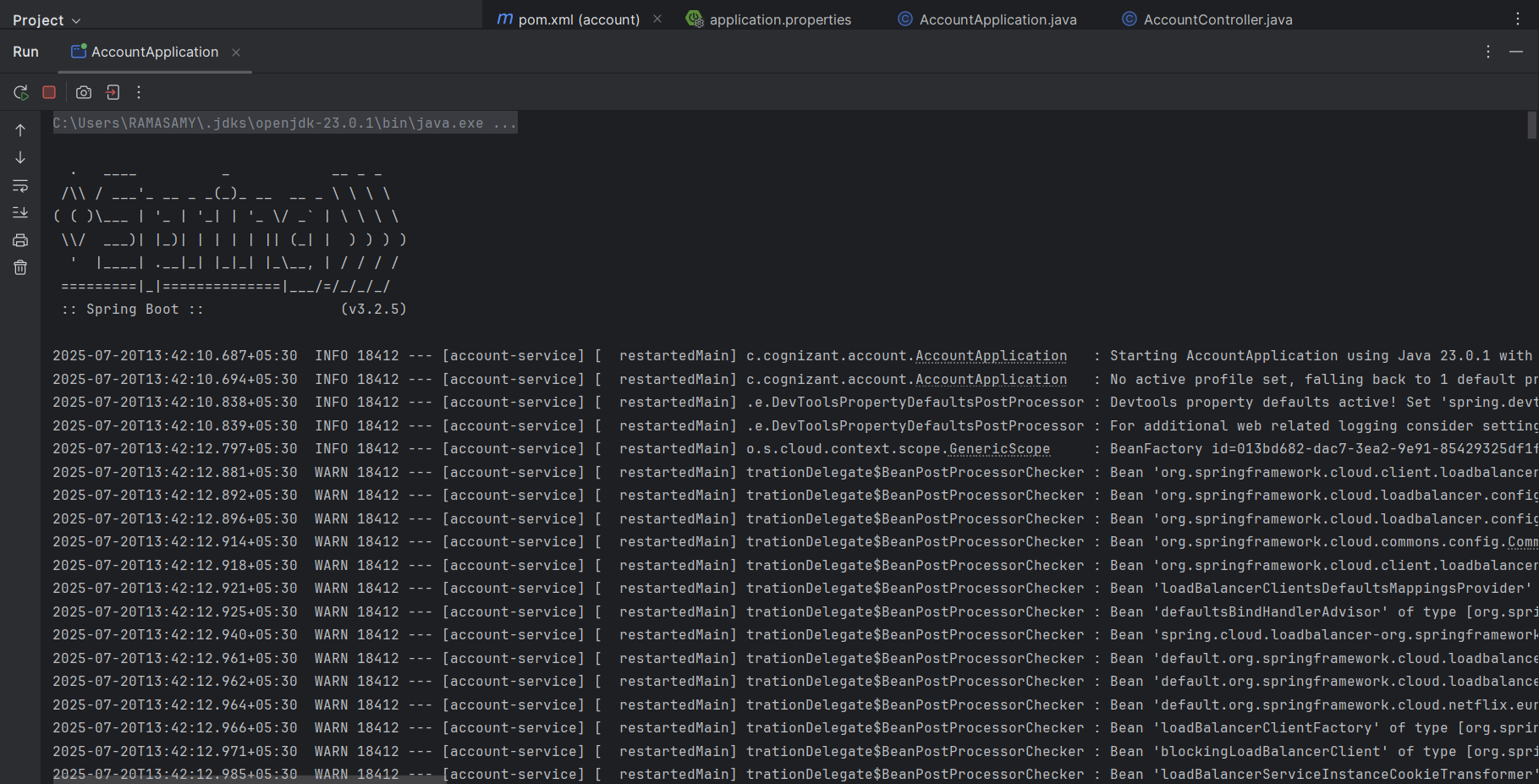
}

**AccountApplication.java**

package com.cognizant.account;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;  
  
@SpringBootApplication  
@EnableDiscoveryClient  
public class AccountApplication {  
 public static void main(String[] args) {  
 SpringApplication.*run*(AccountApplication.class, args);  
 }  
}

**OUTPUT:**

****

****

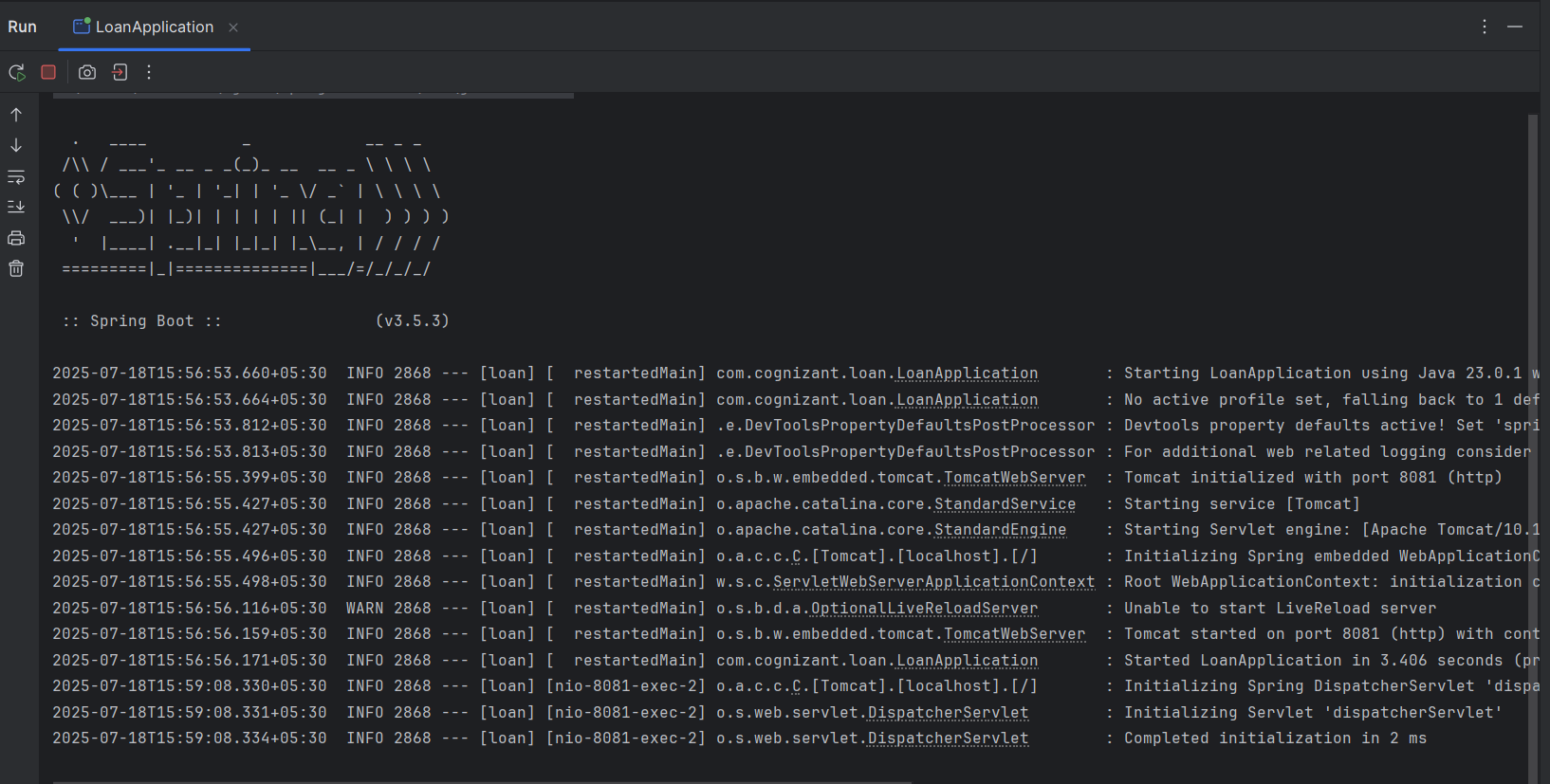
**Loan**

**Controller 🡪 LoanController.java**

package com.cognizant.loan.controller;  
  
import org.springframework.web.bind.annotation.\*;  
  
@RestController  
@RequestMapping("/loans")  
public class LoanController {  
  
 @GetMapping("/{number}")  
 public Loan getLoan(@PathVariable String number) {  
 return new Loan(number, "car", 400000, 3258, 18);  
 }  
}  
  
class Loan {  
 private String number;  
 private String type;  
 private int loan;  
 private int emi;  
 private int tenure;  
  
 public Loan(String number, String type, int loan, int emi, int tenure) {  
 this.number = number;  
 this.type = type;  
 this.loan = loan;  
 this.emi = emi;  
 this.tenure = tenure;  
 }  
  
 public String getNumber() { return number; }  
 public String getType() { return type; }  
 public int getLoan() { return loan; }  
 public int getEmi() { return emi; }  
 public int getTenure() { return tenure; }  
}

**LoanApplication.java**

package com.cognizant.loan;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class LoanApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(LoanApplication.class, args);  
 }  
  
}

**OUTPUT:**

**3. Register Microservices to Eureka & Configure API Gateway**

# 1. Greet-Service Microservice

**application.properties:**

spring.application.name=greet-service  
server.port=8081  
eureka.client.service-url.defaultZone=http://localhost:8761/eureka

Controller class:  
@RestController  
public class GreetController {  
 @GetMapping("/greet")  
 public String greet() {  
 return "Hello World";  
 }  
}

Main class

@SpringBootApplication  
@EnableDiscoveryClient  
public class GreetServiceApplication {  
 public static void main(String[] args) {  
 SpringApplication.run(GreetServiceApplication.class, args);  
 }  
}

# 2. Eureka Discovery Server

**application.properties:**

spring.application.name=eureka-server  
server.port=8761  
eureka.client.register-with-eureka=false  
eureka.client.fetch-registry=false

Main class:

@SpringBootApplication  
@EnableEurekaServer  
public class EurekaServerApplication {  
 public static void main(String[] args) {  
 SpringApplication.run(EurekaServerApplication.class, args);  
 }  
}

# 3. API Gateway Microservice

**application.properties:**

server.port=9090  
spring.application.name=api-gateway  
eureka.client.service-url.defaultZone=http://localhost:8761/eureka  
spring.cloud.gateway.discovery.locator.enabled=true  
spring.cloud.gateway.discovery.locator.lower-case-service-id=true

Main class:

@SpringBootApplication  
@EnableDiscoveryClient  
public class ApiGatewayApplication {  
 public static void main(String[] args) {  
 SpringApplication.run(ApiGatewayApplication.class, args);  
 }  
}

# 4. Global Request Logging Filter

LogFilter.java

@Component  
public class LogFilter implements GlobalFilter, Ordered {  
  
 @Override  
 public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {  
 System.out.println("Request Path: " + exchange.getRequest().getPath());  
 return chain.filter(exchange);  
 }  
  
 @Override  
 public int getOrder() {  
 return -1;  
 }  
}

**OUTPUT:**

