**Sample exercises on Centralized Authentication and SSO with Spring Boot 3 and Spring Cloud**

**Exercise 1: Implementing Centralized Authentication with OAuth 2.1/OIDC**

**Solution code:**

**pom.xml**

<dependency>

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

<artifactId>spring-boot-starter-oauth2-client</artifactId>

</dependency>

<dependency>

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

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

</dependency>

**application.yml**

server:

port: 8082

spring:

security:

oauth2:

client:

registration:

google:

client-id: 13682523629-n448786a3ffc88n7n9rtrrdrs17saqg7.apps.googleusercontent.com

client-secret: GOCSPX-TX2U\_EGHuiwcJf3yiAZHFe1OjVuF

scope: openid, profile, email

provider:

google:

authorization-uri: https://accounts.google.com/o/oauth2/v2/auth

token-uri: https://oauth2.googleapis.com/token

user-info-uri: https://openidconnect.googleapis.com/v1/userinfo

user-name-attribute: sub

**SecurityConfig.java**

package com.cognizant.OauthClient.config;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.Customizer;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.web.SecurityFilterChain;

*@Configuration*

public class SecurityConfig {

*@Bean*

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.authorizeHttpRequests(auth -> auth

.anyRequest().authenticated()

)

.oauth2Login(Customizer.*withDefaults*());

return http.build();

}

}

**HomeController.java**

package com.cognizant.OauthClient.controller;

import org.springframework.security.oauth2.core.oidc.user.OidcUser;

import org.springframework.security.core.annotation.AuthenticationPrincipal;

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

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

*@RestController*

public class HomeController {

*@GetMapping*("/")

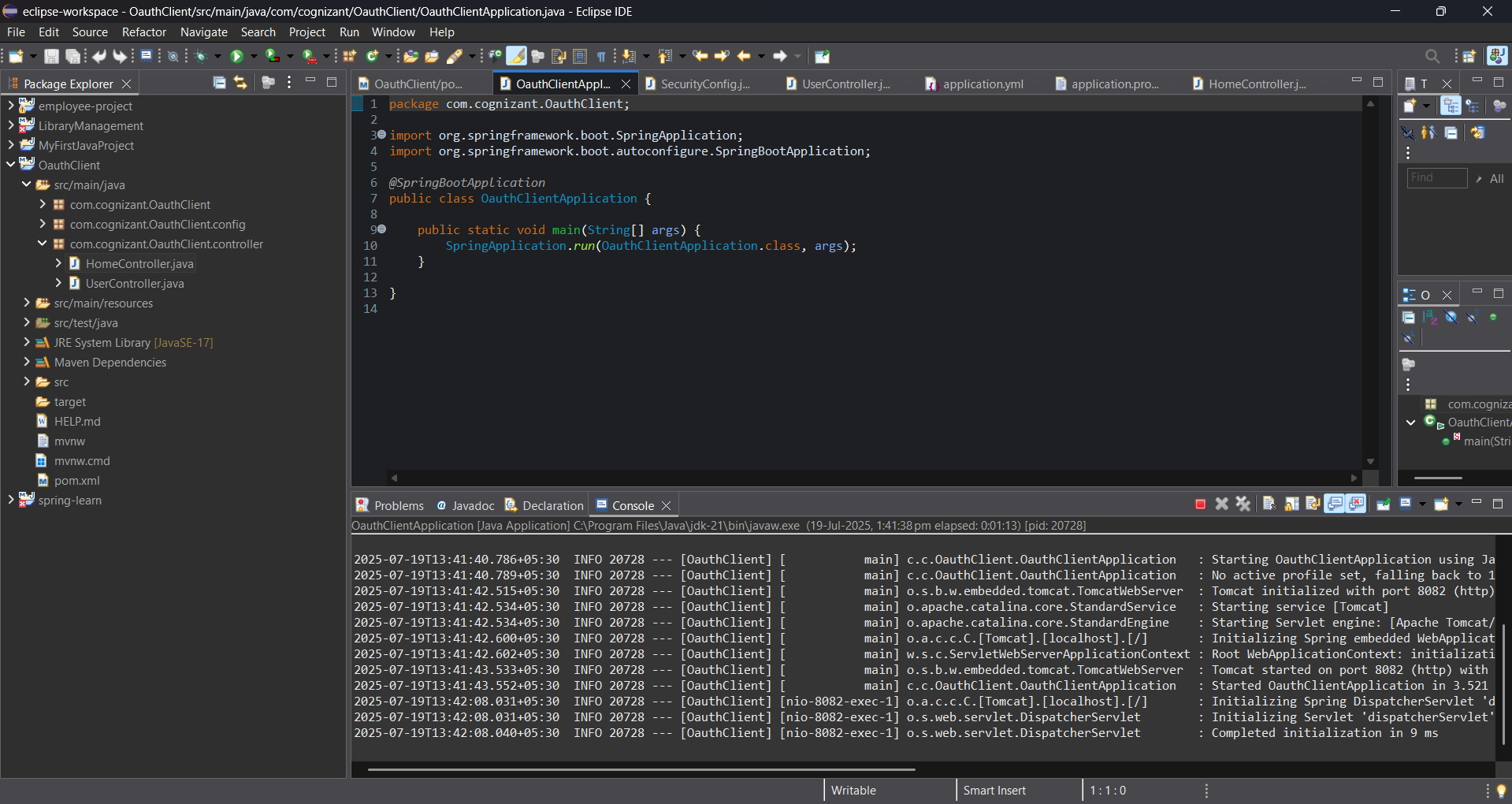
public String home(*@AuthenticationPrincipal* OidcUser user) {

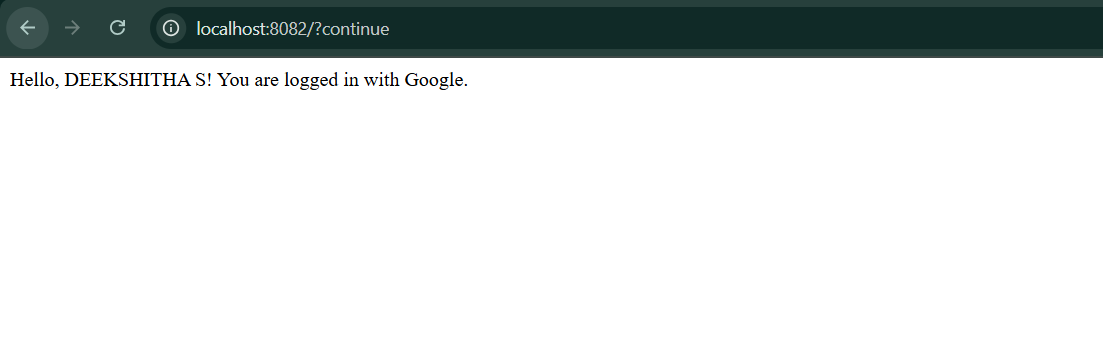
return "Hello, " + user.getFullName() + "! You are logged in with Google.";

}

}

**Output:**



****

**Exercise 2: Configuring Authorization Servers and Resource Servers**

**Pom.xml**

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>spring-boot-starter-oauth2-resource-server</artifactId>

</dependency>

**application.yml**

spring:

security:

oauth2:

resourceserver:

jwt:

issuer-uri: https://dev-5f8cuzt3m8260zii.us.auth0.com/

**ResourceServerConfig.java**

package com.cognizant.exercise2.security;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.config.Customizer;

*@Configuration*

*@EnableWebSecurity*

public class ResourceServerConfig {

*@Bean*

public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {

http

.authorizeHttpRequests(auth -> auth

.anyRequest().authenticated()

)

.oauth2ResourceServer(oauth2 -> oauth2

.jwt(Customizer.*withDefaults*())

);

return http.build();

}

}

**SecureController.java**

package com.cognizant.exercise2.controller;

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

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

*@RestController*

public class SecureController {

*@GetMapping*("/secure")

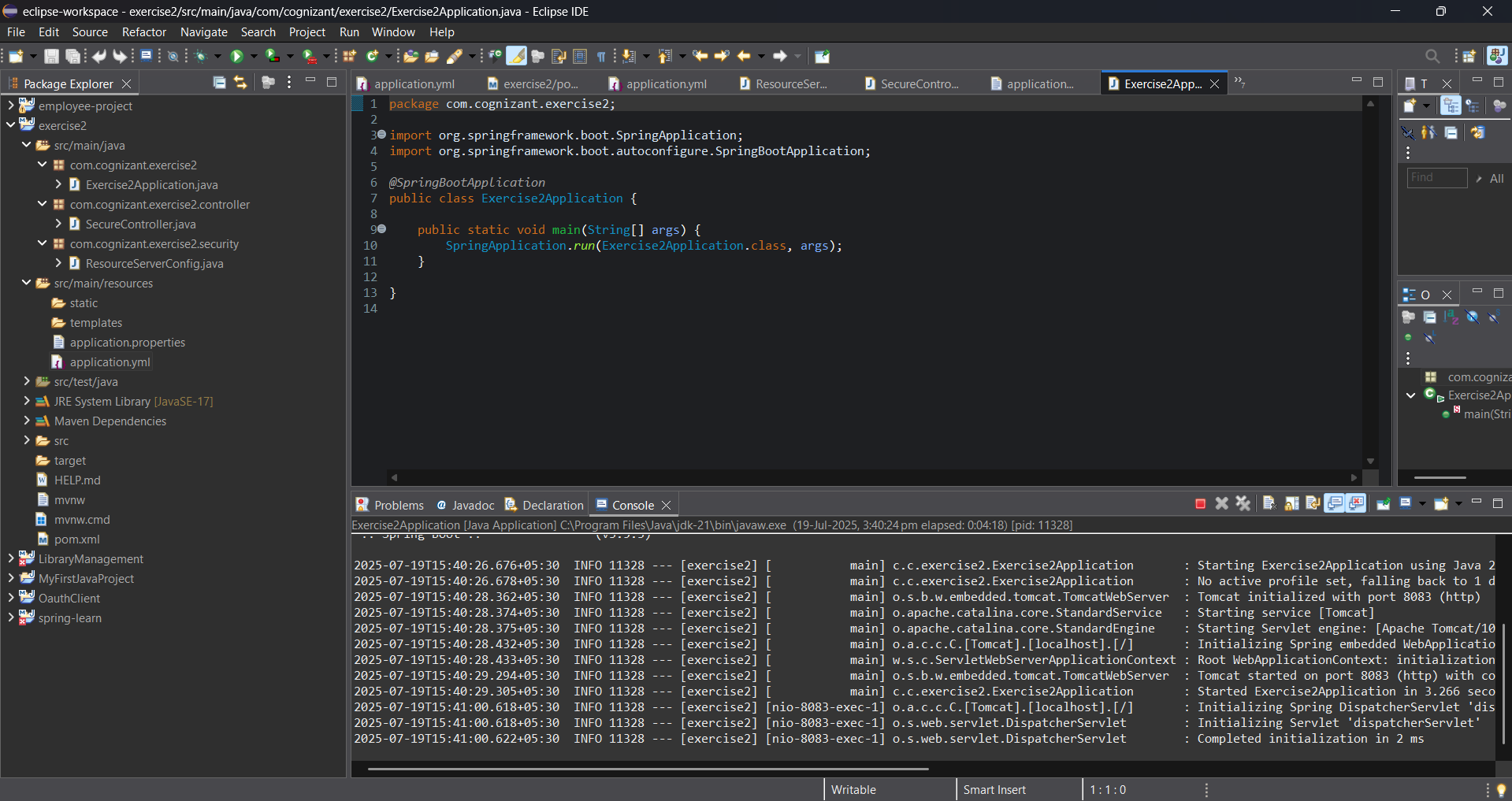
public String secure() {

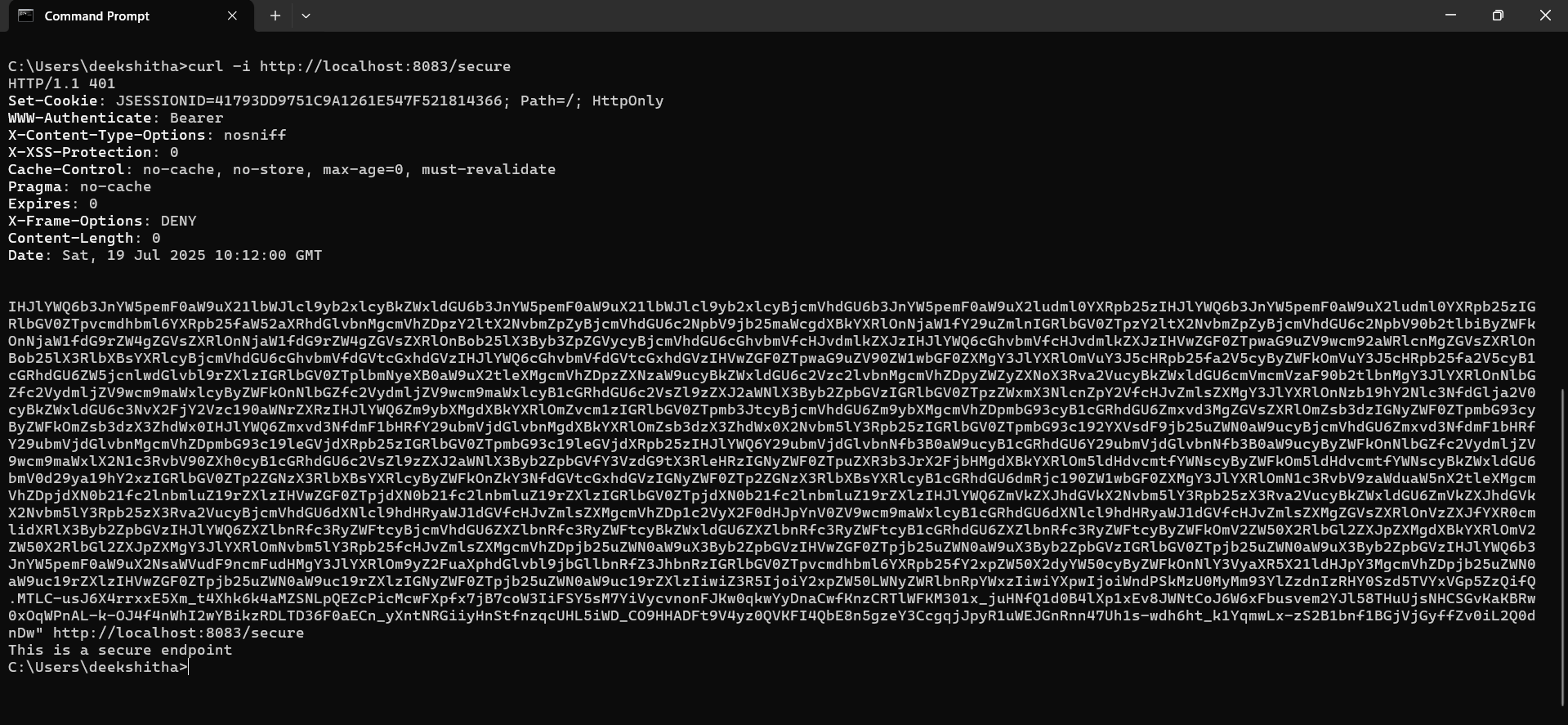
return "This is a secure endpoint";

}

}

**Output**

****



**Exercise 3: Using JSON Web Tokens (JWT) for Secure Communication**

pom.xml

<dependencies>

<!-- Spring Boot Security -->

<dependency>

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

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

</dependency>

<!-- Spring Boot Web -->

<dependency>

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

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

</dependency>

<!-- JWT -->

<dependency>

<groupId>io.jsonwebtoken</groupId>

<artifactId>jjwt</artifactId>

<version>0.9.1</version>

</dependency>

<!-- Lombok -->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<optional>true</optional>

</dependency>

</dependencies>

**application.yml**

server:

port: 8080

spring:

security:

jwt:

secret: MyJwtSecretKey123

**JwtConfig.java**

package com.example.jwtsecurity.config;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.context.annotation.Configuration;

@Configuration

public class JwtConfig {

@Value("${spring.security.jwt.secret}")

private String secret;

public String getSecret() {

return secret;

}

}

**JwtTokenProvider.java**

package com.example.jwtsecurity.security;

import com.example.jwtsecurity.config.JwtConfig;

import io.jsonwebtoken.\*;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

import org.springframework.stereotype.Component;

import java.util.\*;

@Component

public class JwtTokenProvider {

@Autowired

private JwtConfig jwtConfig;

public String createToken(String username) {

Claims claims = Jwts.claims().setSubject(username);

Date now = new Date();

Date expiry = new Date(now.getTime() + 3600000); // 1 hour

return Jwts.builder()

.setClaims(claims)

.setIssuedAt(now)

.setExpiration(expiry)

.signWith(SignatureAlgorithm.HS256, jwtConfig.getSecret())

.compact();

}

public boolean validateToken(String token) {

try {

Jwts.parser().setSigningKey(jwtConfig.getSecret()).parseClaimsJws(token);

return true;

} catch (JwtException | IllegalArgumentException e) {

return false;

}

}

public Authentication getAuthentication(String token) {

String username = Jwts.parser()

.setSigningKey(jwtConfig.getSecret())

.parseClaimsJws(token)

.getBody()

.getSubject();

return new UsernamePasswordAuthenticationToken(username, null, new ArrayList<>());

}

}

**JwtTokenFilter.java**

package com.example.jwtsecurity.security;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.\*;

import javax.servlet.http.\*;

import java.io.IOException;

public class JwtTokenFilter extends OncePerRequestFilter {

@Autowired

private JwtTokenProvider jwtTokenProvider;

@Override

protected void doFilterInternal(HttpServletRequest request,

HttpServletResponse response,

FilterChain filterChain)

throws ServletException, IOException {

String token = resolveToken(request);

if (token != null && jwtTokenProvider.validateToken(token)) {

SecurityContextHolder.getContext()

.setAuthentication(jwtTokenProvider.getAuthentication(token));

}

filterChain.doFilter(request, response);

}

private String resolveToken(HttpServletRequest request) {

String bearer = request.getHeader("Authorization");

return (bearer != null && bearer.startsWith("Bearer ")) ? bearer.substring(7) : null;

}

}

**SecurityConfig.java**

package com.example.jwtsecurity.config;

import com.example.jwtsecurity.security.JwtTokenFilter;

import org.springframework.context.annotation.\*;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.\*;

import org.springframework.security.config.http.SessionCreationPolicy;

import org.springframework.security.web.SecurityFilterChain;

import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

@Configuration

@EnableWebSecurity

public class SecurityConfig {

@Bean

public JwtTokenFilter jwtTokenFilter() {

return new JwtTokenFilter();

}

@Bean

public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {

http.csrf().disable()

.sessionManagement()

.sessionCreationPolicy(SessionCreationPolicy.STATELESS)

.and()

.authorizeHttpRequests()

.antMatchers("/auth/\*\*").permitAll()

.anyRequest().authenticated()

.and()

.addFilterBefore(jwtTokenFilter(), UsernamePasswordAuthenticationFilter.class);

return http.build();

}

}

**AuthController.java**

package com.example.jwtsecurity.controller;

import com.example.jwtsecurity.security.JwtTokenProvider;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import java.util.\*;

@RestController

@RequestMapping("/auth")

public class AuthController {

@Autowired

private JwtTokenProvider jwtTokenProvider;

@PostMapping("/login")

public Map<String, String> login(@RequestParam String username, @RequestParam String password) {

if ("admin".equals(username) && "admin".equals(password)) {

String token = jwtTokenProvider.createToken(username);

return Map.of("token", token);

}

throw new RuntimeException("Invalid credentials");

}

}

**SecureController.java**

package com.example.jwtsecurity.controller;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/secure")

public class SecureController {

@GetMapping

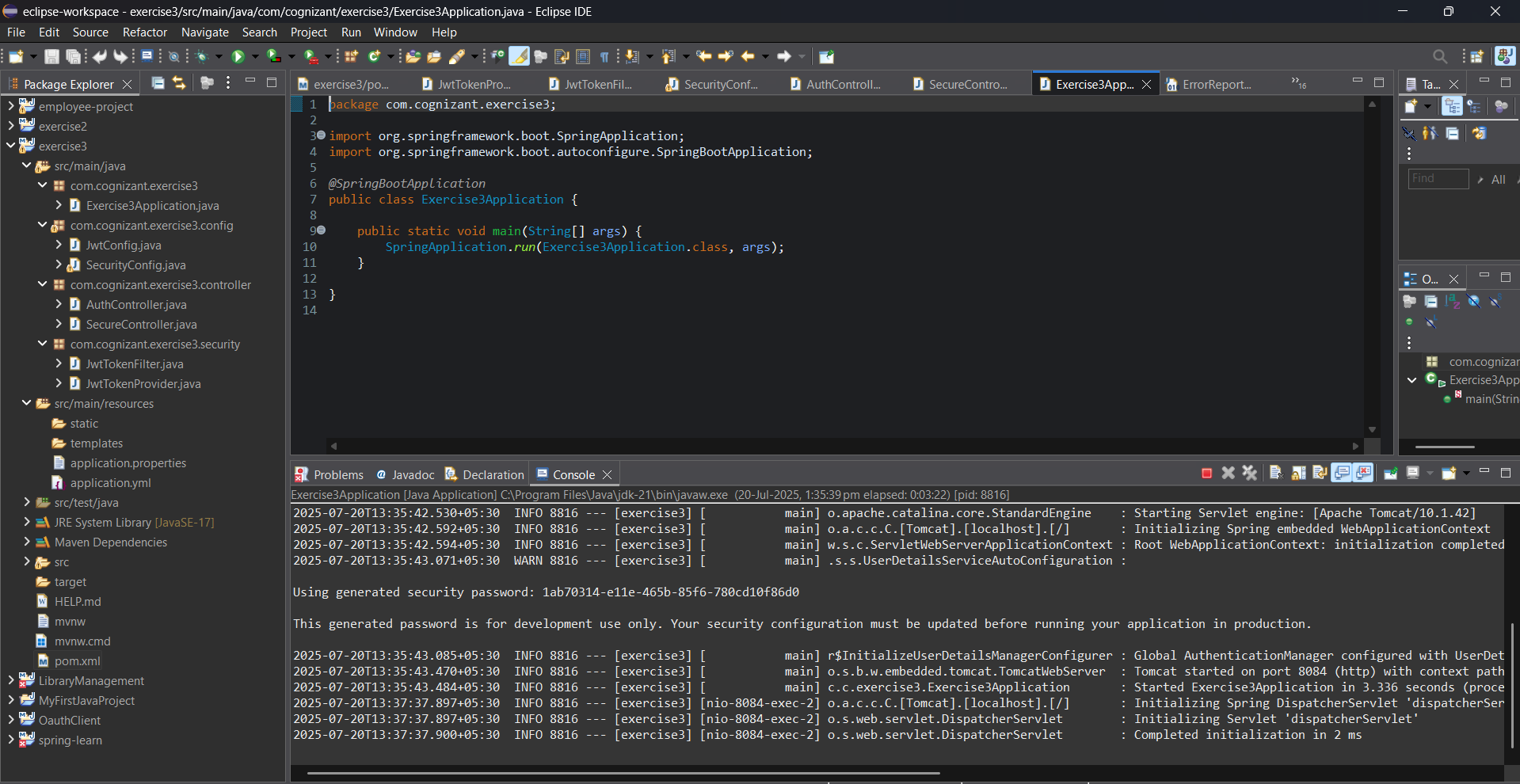
public String secureData() {

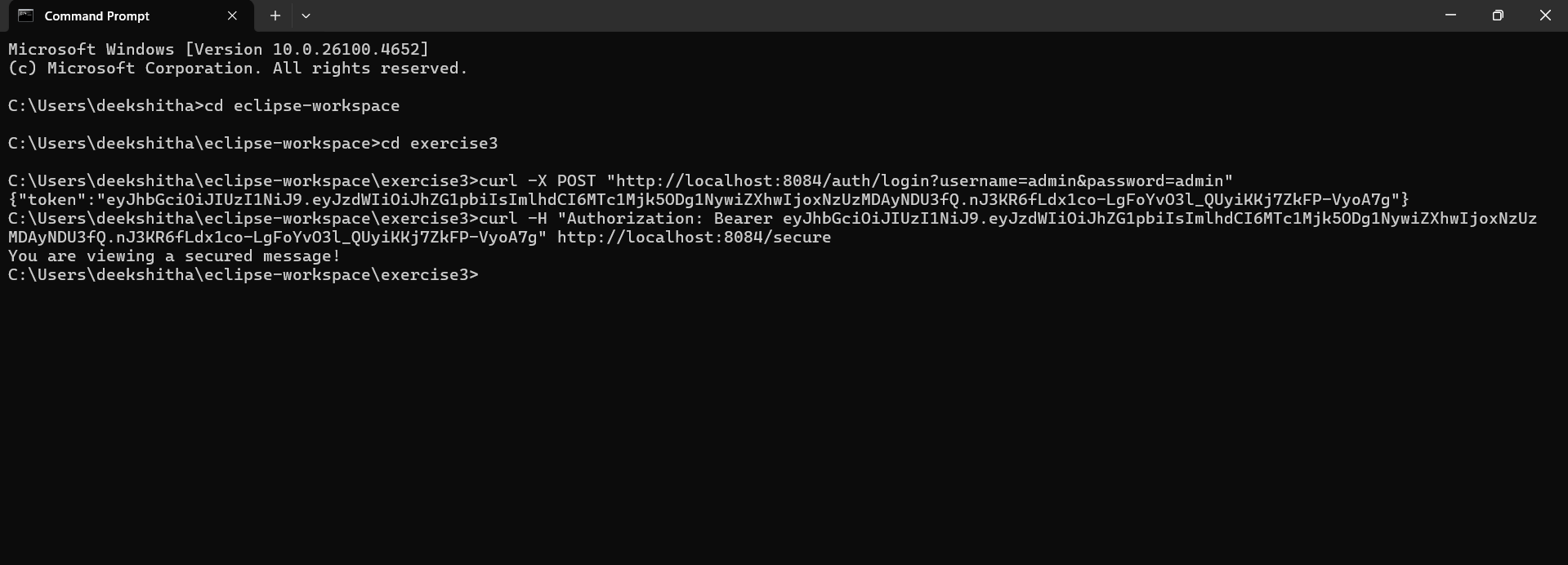
return "You are viewing a secured message!";

}

}

**Output**

****



**Microservices with API gateway**

**Creating Microservices for account and loan**

**AccountController.java**

package com.cognizant.account.controller;

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

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

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

import java.util.Map;

@RestController

public class AccountController {

@GetMapping("/accounts/{number}")

public Map<String, Object> getAccountDetails(@PathVariable String number) {

return Map.of(

"number", number,

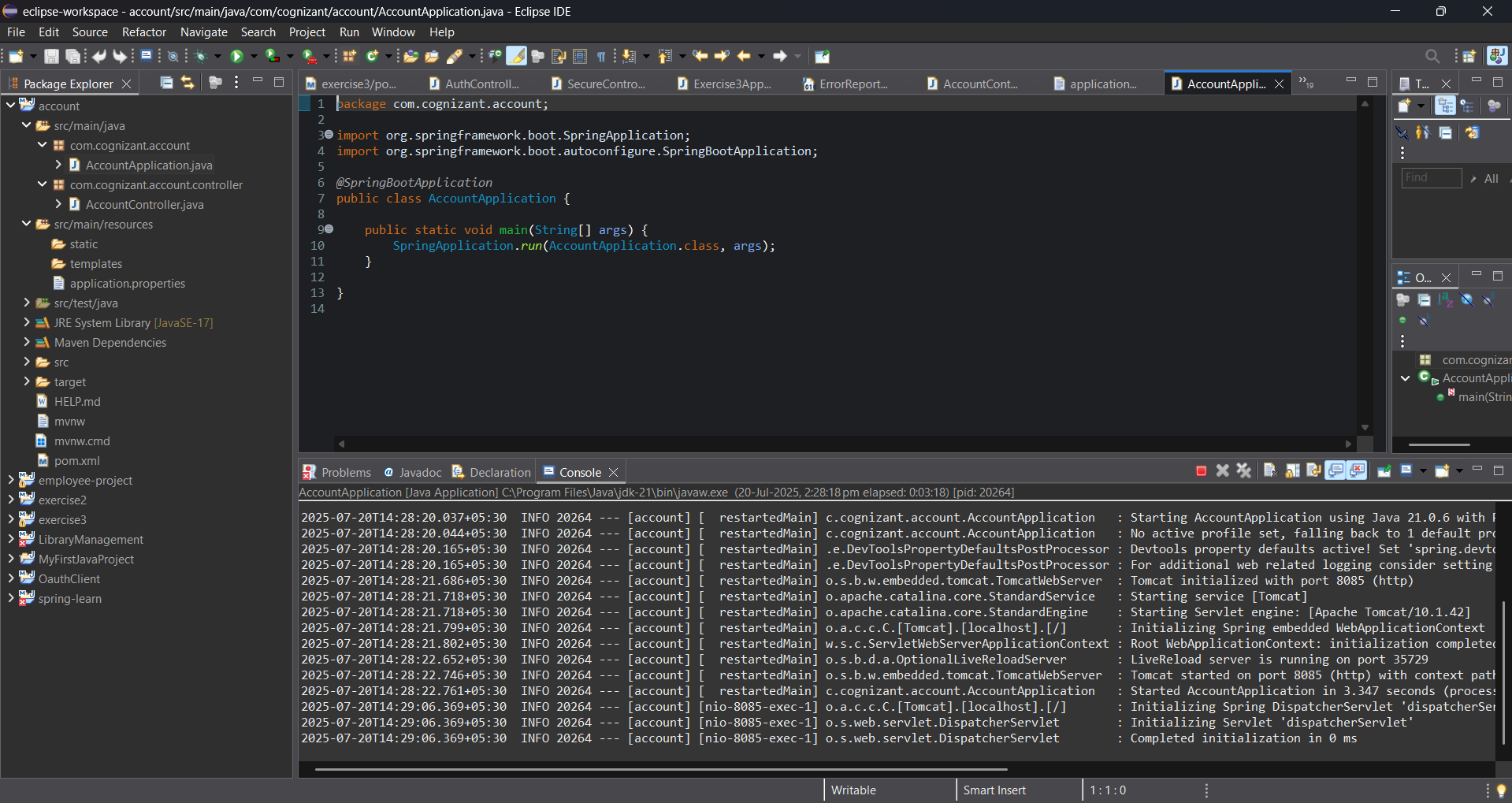
"type", "savings",

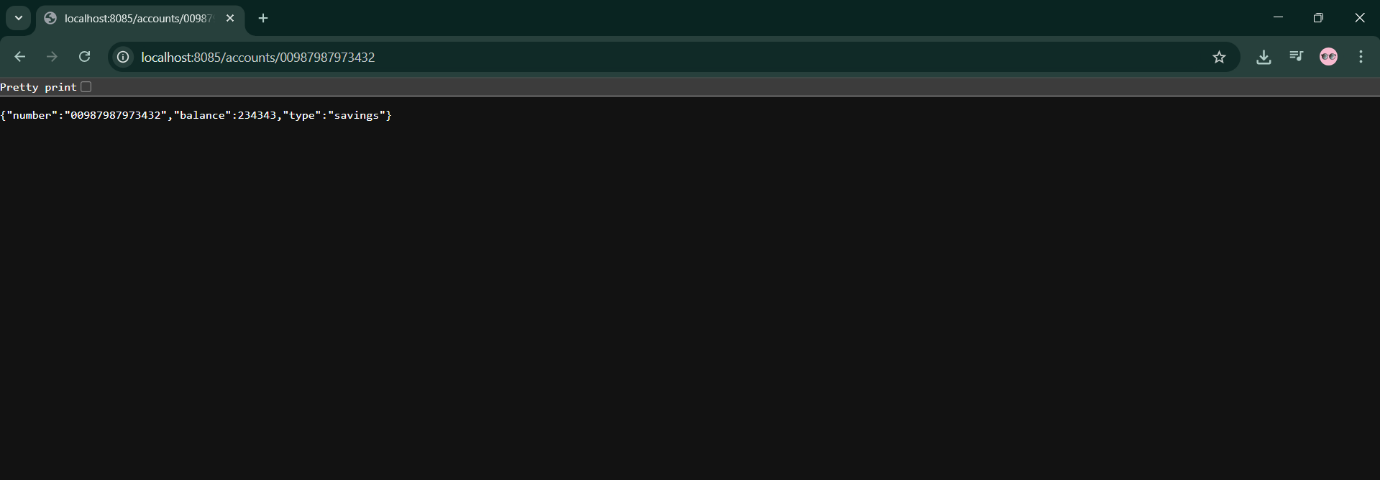
"balance", 234343

);

}

}





**Loan service**

**LoanController.java**

package com.cognizant.loan.controller;

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

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

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

import java.util.Map;

@RestController

public class LoanController {

@GetMapping("/loans/{number}")

public Map<String, Object> getLoanDetails(@PathVariable String number) {

return Map.of(

"number", number,

"type", "car",

"loan", 400000,

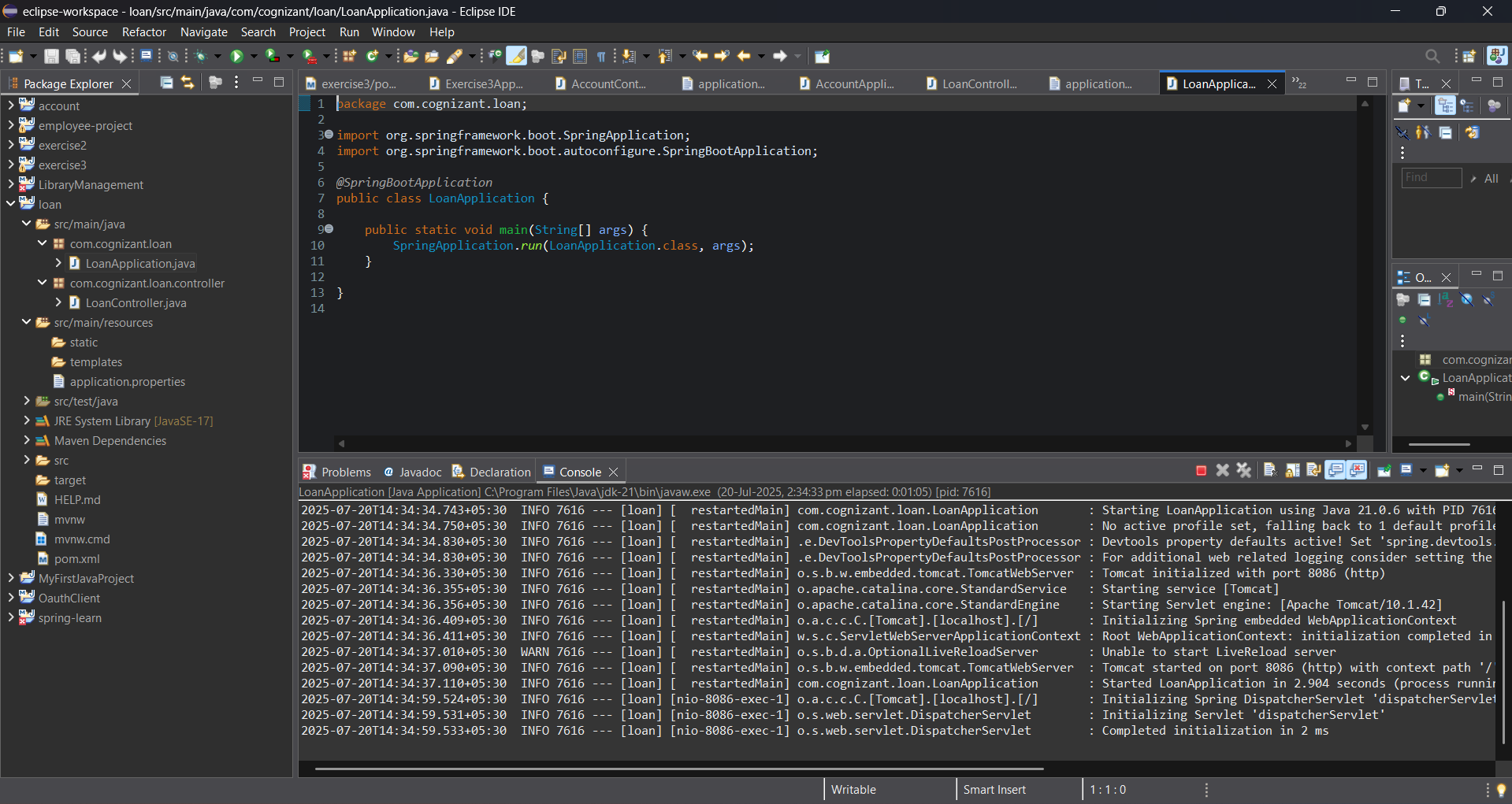
"emi", 3258,

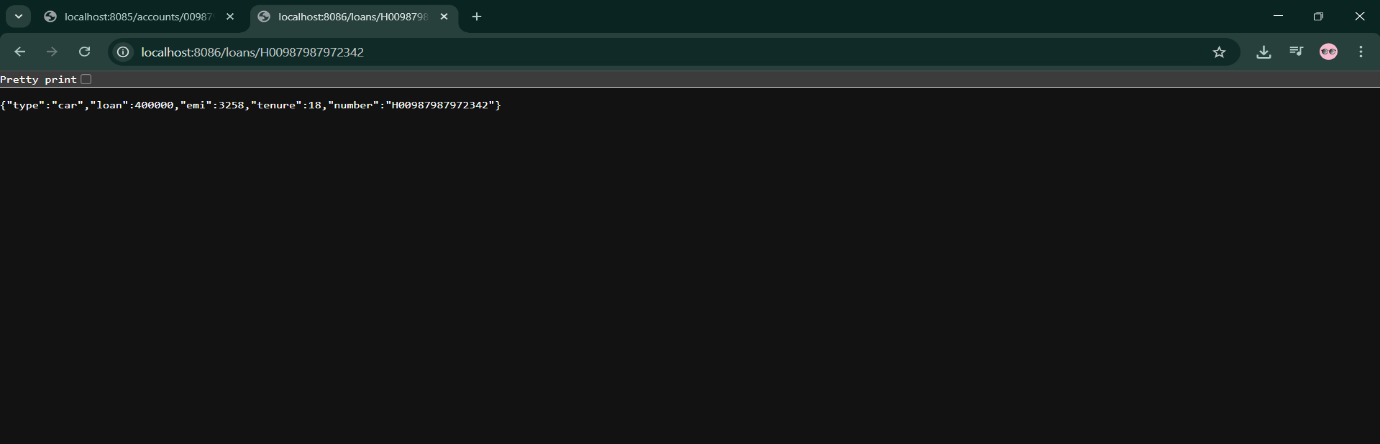
"tenure", 18

);

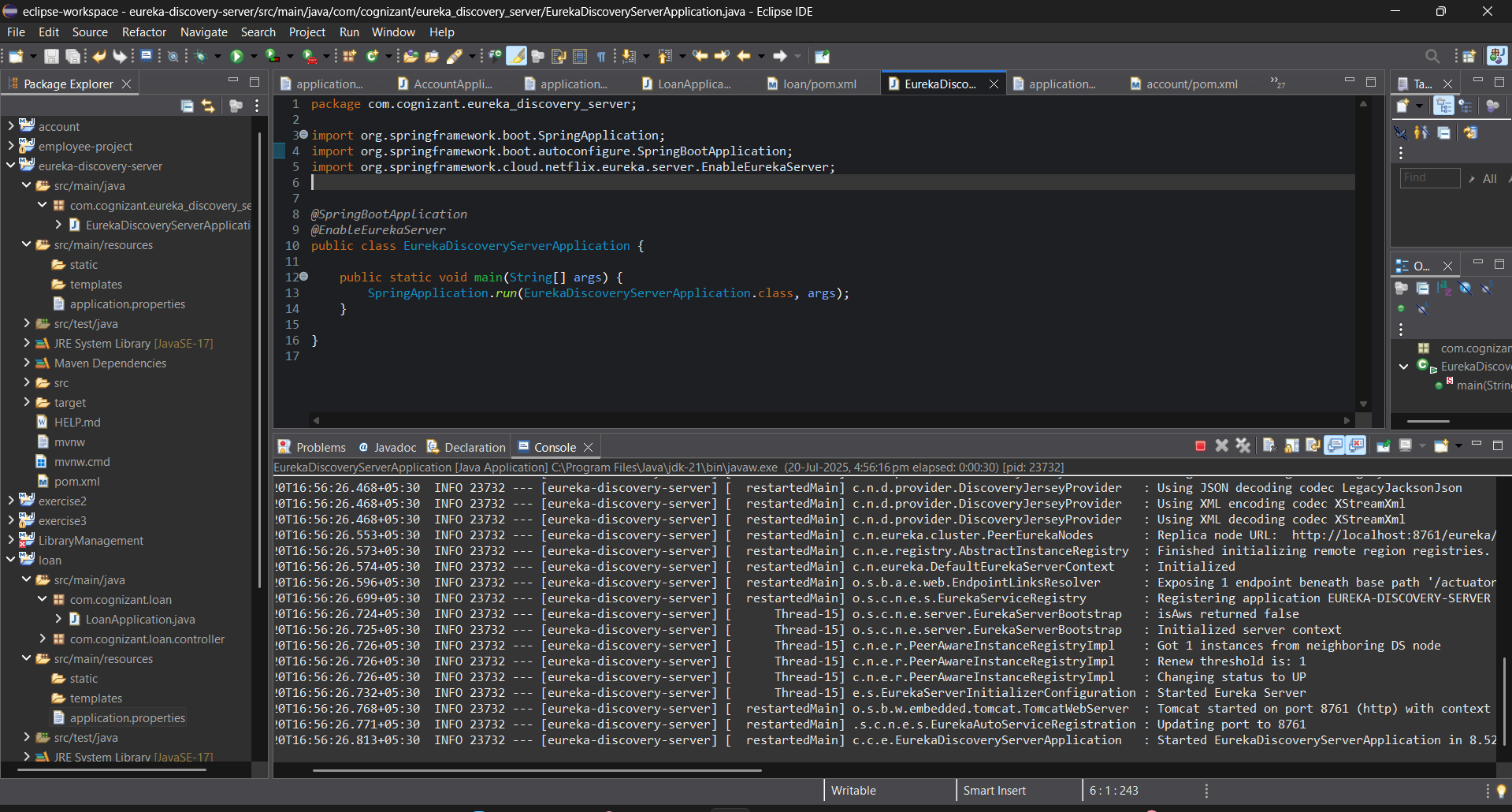
}

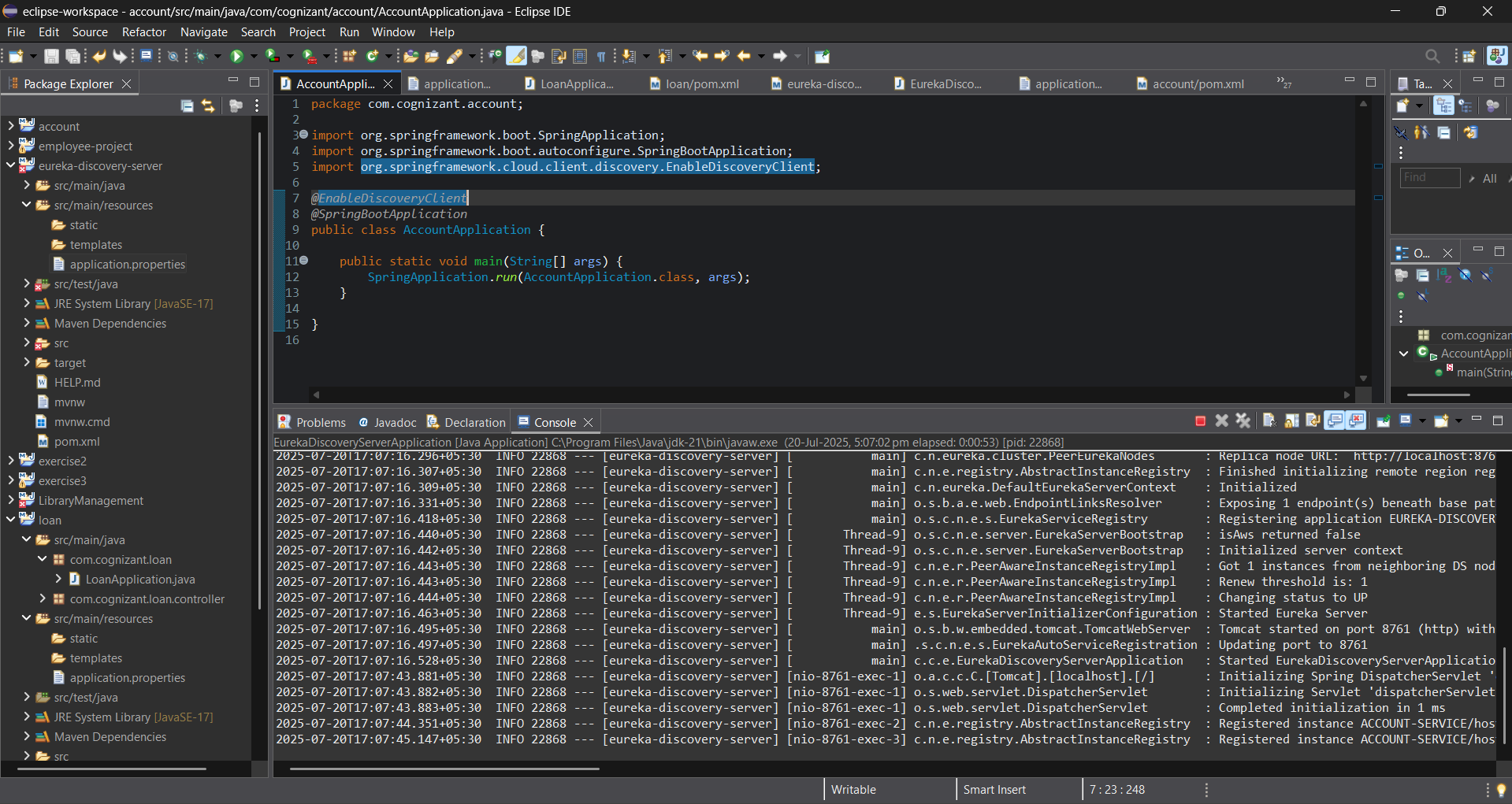
}

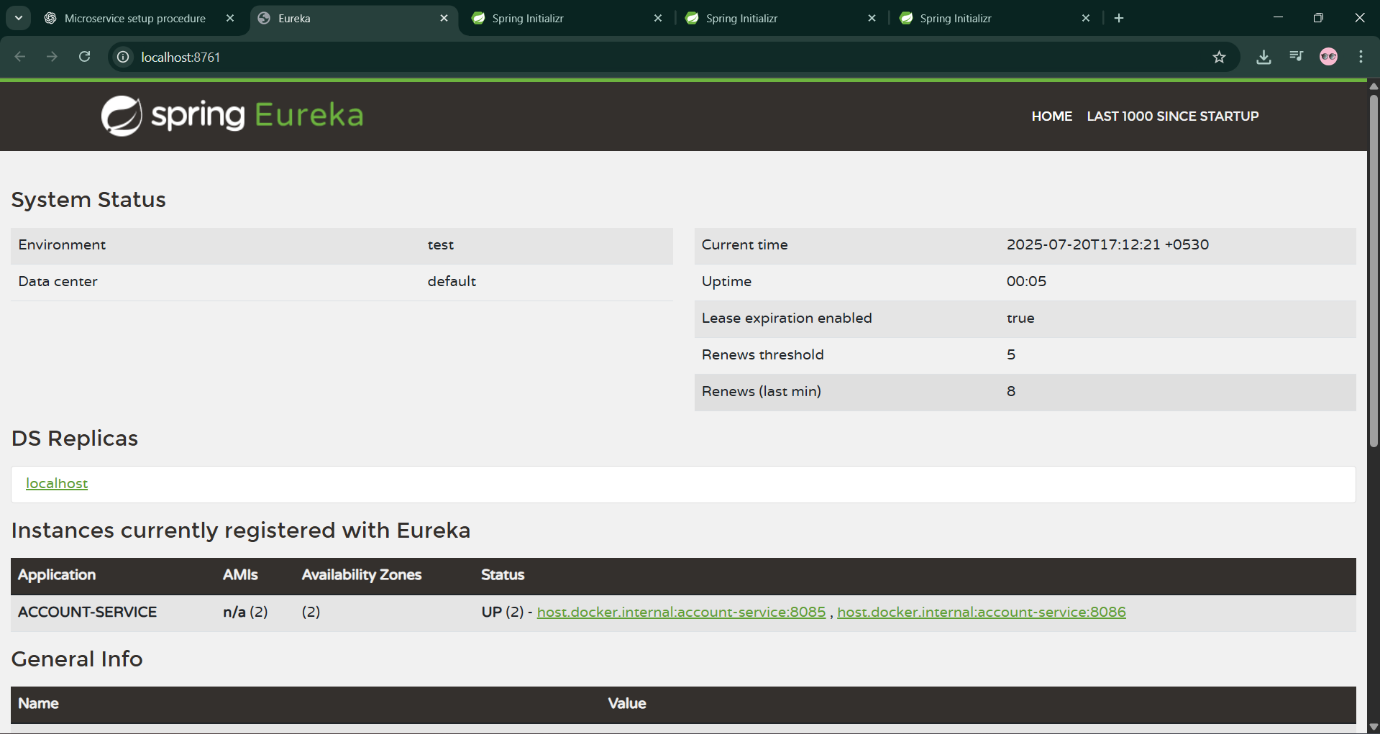




**Create Eureka Discovery Server and register microservices**

****

****



**Sample Hands-on Exercises on Edge Services and API Gateway with Spring Boot 3 and Spring Cloud**

**Exercise 1: Implementing Edge Services for Routing and Filtering**

**pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<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

https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

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

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

<version>3.1.3</version>

<relativePath/>

</parent>

<groupId>com.example</groupId>

<artifactId>edge-routing-filter-service</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>edge-routing-filter-service</name>

<description>Edge Routing and Filtering with Spring Cloud Gateway</description>

<properties>

<java.version>17</java.version>

<spring-cloud.version>2022.0.4</spring-cloud.version>

</properties>

<dependencies>

<dependency>

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

<artifactId>spring-cloud-starter-gateway</artifactId>

</dependency>

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</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=8087

spring.cloud.gateway.routes[0].id=example\_route

spring.cloud.gateway.routes[0].uri=http://example.org

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

**LoggingFilter.java**

package com.example.gateway;

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

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

import org.springframework.stereotype.Component;

import org.springframework.web.server.ServerWebExchange;

import reactor.core.publisher.Mono;

@Component

public class LoggingFilter implements GlobalFilter {

@Override

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

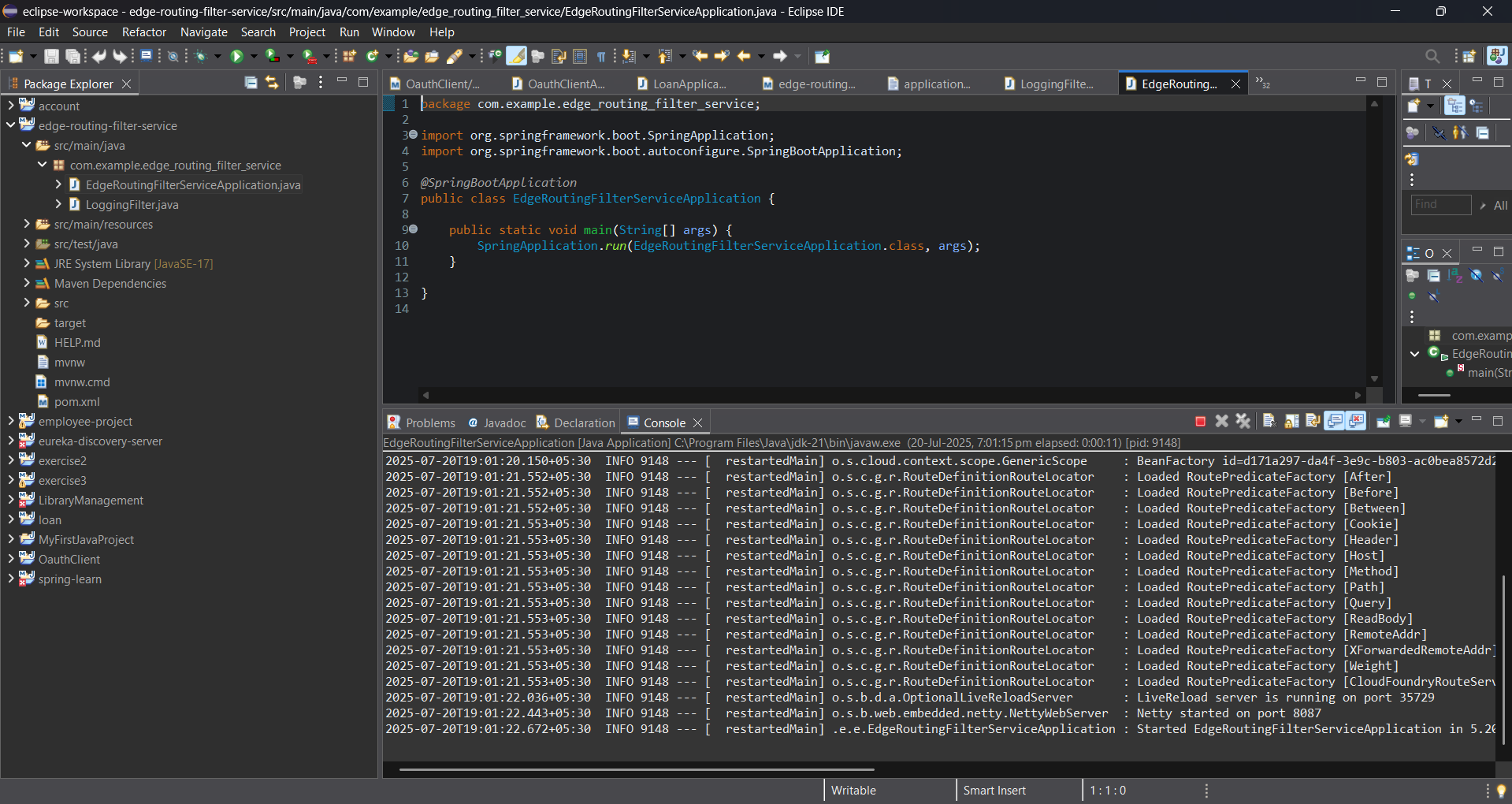
System.out.println("Request URI: " + exchange.getRequest().getURI());

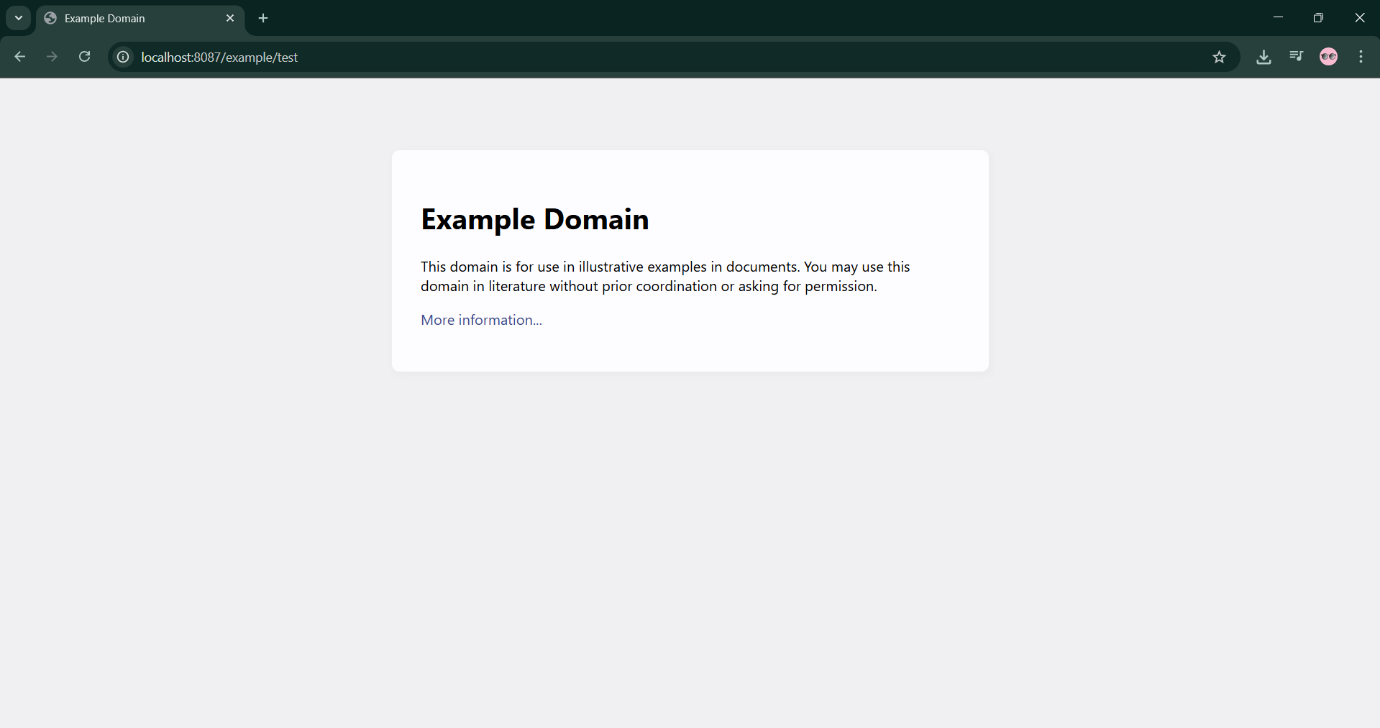
return chain.filter(exchange);

}

}

**Output**

****

****

**Exercise 2: Load Balancing in an API Gateway**

**GatewayLoadbalancerApplication.java**

package com.example.gateway;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class GatewayLoadbalancerApplication {

public static void main(String[] args) {

SpringApplication.run(GatewayLoadbalancerApplication.class, args);

}

}

**application.properties**

server.port=8080

spring.cloud.gateway.routes[0].id=load\_balanced\_route

spring.cloud.gateway.routes[0].uri=lb://example-service

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

spring.cloud.discovery.client.simple.instances.example-service[0].uri=http://localhost:8081

spring.cloud.discovery.client.simple.instances.example-service[1].uri=http://localhost:8082

**Example Service 1**

**ExampleService1Application.java**

package com.example.exampleservice1;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class ExampleService1Application {

public static void main(String[] args) {

SpringApplication.run(ExampleService1Application.class, args);

}

}

**HelloController.java**

package com.example.exampleservice1;

import org.springframework.beans.factory.annotation.Value;

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

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

@RestController

public class HelloController {

@Value("${server.port}")

private String port;

@GetMapping("/hello")

public String hello() {

return "Hello from port: " + port;

}

}

**application.properties**

server.port=8081

spring.application.name=example-service

**Example Service 2**

**ExampleService2Application.java**

package com.example.exampleservice2;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class ExampleService2Application {

public static void main(String[] args) {

SpringApplication.run(ExampleService2Application.class, args);

}

}

**HelloController.java**

package com.example.exampleservice2;

import org.springframework.beans.factory.annotation.Value;

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

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

@RestController

public class HelloController {

@Value("${server.port}")

private String port;

@GetMapping("/hello")

public String hello() {

return "Hello from port: " + port;

}

}

**application.properties**

server.port=8082

spring.application.name=example-service

**Output**

curl http://localhost:8080/loadbalanced/hello

Hello from port: 8081

Hello from port: 8082

Hello from port: 8081

**Exercise 3: Resilience Patterns in an API Gateway**

**ResilienceGatewayApplication.java**

package com.example.gateway;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class ResilienceGatewayApplication {

public static void main(String[] args) {

SpringApplication.run(ResilienceGatewayApplication.class, args);

}

}

**ResilienceConfiguration.java**

package com.example.gateway;

import io.github.resilience4j.circuitbreaker.CircuitBreakerConfig;

import io.github.resilience4j.timelimiter.TimeLimiterConfig;

import org.springframework.cloud.client.circuitbreaker.Customizer;

import org.springframework.cloud.circuitbreaker.resilience4j.ReactiveResilience4JCircuitBreakerFactory;

import org.springframework.cloud.circuitbreaker.resilience4j.Resilience4JConfigBuilder;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class ResilienceConfiguration {

@Bean

public Customizer<ReactiveResilience4JCircuitBreakerFactory> defaultCustomizer() {

return factory -> factory.configureDefault(id -> new Resilience4JConfigBuilder(id)

.circuitBreakerConfig(CircuitBreakerConfig.ofDefaults())

.timeLimiterConfig(TimeLimiterConfig.ofDefaults())

.build());

}

}

**application.properties**

server.port=8080

spring.cloud.gateway.routes[0].id=resilient\_route

spring.cloud.gateway.routes[0].uri=http://localhost:8085

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

spring.cloud.gateway.routes[0].filters[0]=CircuitBreaker=name=exampleCircuitBreaker,fallbackUri=forward:/fallback

resilience4j.circuitbreaker.instances.exampleCircuitBreaker.registerHealthIndicator=true

resilience4j.circuitbreaker.instances.exampleCircuitBreaker.slidingWindowSize=10

resilience4j.circuitbreaker.instances.exampleCircuitBreaker.failureRateThreshold=50

**FallbackController.java**

package com.example.gateway;

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

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

@RestController

public class FallbackController {

@GetMapping("/fallback")

public String fallback() {

return "Fallback response: The service is temporarily unavailable.";

}

}

**Example Backend Service (localhost:8085)**

**ResilientServiceApplication.java**

package com.example.service;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class ResilientServiceApplication {

public static void main(String[] args) {

SpringApplication.run(ResilientServiceApplication.class, args);

}

}

**HelloController.java**

package com.example.service;

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

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

@RestController

public class HelloController {

@GetMapping("/resilient/hello")

public String hello() {

if (Math.random() < 0.5) {

throw new RuntimeException("Simulated failure");

}

return "Hello from Resilient Service!";

}

}

**application.properties**

server.port=8085

spring.application.name=resilient-service

**output:**

C:\Users\deekshitha> curl http://localhost:8080/resilient/hello

Hello from Resilient Service!

C:\Users\deekshitha> curl http://localhost:8080/resilient/hello

Fallback response: The service is temporarily unavailable.

**Exercises on Microservices with Spring Boot 3.0**

**Build a User and Order Management System**

**USER SERVICE**

**application.properties**

server.port=8081

spring.datasource.url=jdbc:mysql://localhost:3306/userdb

spring.datasource.username=root

spring.datasource.password=your\_password

spring.jpa.hibernate.ddl-auto=update

**User.java**

@Entity

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

}

**UserRepository.java**

public interface UserRepository extends JpaRepository<User, Long> {}

**UserController.java**

@RestController

@RequestMapping("/users")

public class UserController {

@Autowired

private UserRepository userRepository;

@GetMapping

public List<User> getAll() {

return userRepository.findAll();

}

@GetMapping("/{id}")

public ResponseEntity<User> getById(@PathVariable Long id) {

return userRepository.findById(id)

.map(ResponseEntity::ok)

.orElse(ResponseEntity.notFound().build());

}

@PostMapping

public User save(@RequestBody User user) {

return userRepository.save(user);

}

}

**UserServiceApplication.java**

@SpringBootApplication

public class UserServiceApplication {

public static void main(String[] args) {

SpringApplication.run(UserServiceApplication.class, args);

}

}

**ORDER SERVICE (Port 8082)**

**application.properties**

server.port=8082

spring.datasource.url=jdbc:mysql://localhost:3306/orderdb

spring.datasource.username=root

spring.datasource.password=your\_password

spring.jpa.hibernate.ddl-auto=update

spring.application.name=order-service

**Order.java**

@Entity

public class Order {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String item;

private Double price;

private Long userId;

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getItem() {

return item;

}

public void setItem(String item) {

this.item = item;

}

public Double getPrice() {

return price;

}

public void setPrice(Double price) {

this.price = price;

}

public Long getUserId() {

return userId;

}

public void setUserId(Long userId) {

this.userId = userId;

}

}

**OrderRepository.java**

public interface OrderRepository extends JpaRepository<Order, Long> {}

**UserClient.java**

@FeignClient(name = "user-service", url = "http://localhost:8081")

public interface UserClient {

@GetMapping("/users/{id}")

User getUserById(@PathVariable("id") Long id);

}

**OrderResponse.java**

public class OrderResponse {

private Order order;

private User user;

public Order getOrder() {

return order;

}

public void setOrder(Order order) {

this.order = order;

}

public User getUser() {

return user;

}

public void setUser(User user) {

this.user = user;

}

}

**OrderController.java**

@RestController

@RequestMapping("/orders")

public class OrderController {

@Autowired

private OrderRepository orderRepo;

@Autowired

private UserClient userClient;

@PostMapping

public Order saveOrder(@RequestBody Order order) {

return orderRepo.save(order);

}

@GetMapping("/user/{userId}")

public ResponseEntity<OrderResponse> getOrderByUserId(@PathVariable Long userId) {

List<Order> orders = orderRepo.findAll().stream()

.filter(o -> o.getUserId().equals(userId)).toList();

User user = userClient.getUserById(userId);

OrderResponse response = new OrderResponse();

response.setUser(user);

if (!orders.isEmpty()) {

response.setOrder(orders.get(0));

}

return ResponseEntity.ok(response);

}

}

**OrderServiceApplication.java**

@SpringBootApplication

@EnableFeignClients

public class OrderServiceApplication {

public static void main(String[] args) {

SpringApplication.run(OrderServiceApplication.class, args);

}

}

**CURL Output:**

$ curl -X POST http://localhost:8081/users -H "Content-Type: application/json" -d '{"name":"Deekshitha","email":"deek@abc.com"}'

{"id":1,"name":"Deekshitha","email":"deek@abc.com"}

$ curl -X POST http://localhost:8082/orders -H "Content-Type: application/json" -d '{"item":"Laptop","price":80000,"userId":1}'

{"id":1,"item":"Laptop","price":80000.0,"userId":1}

$ curl http://localhost:8082/orders/user/1

{"order":{"id":1,"item":"Laptop","price":80000.0,"userId":1},"user":{"id":1,"name":"Deekshitha","email":"deek@abc.com"}}

**Inventory Management System**

**application.properties**

server.port=8888

spring.cloud.config.server.git.uri=https://github.com/your-username/your-config-repo

**ConfigServerApplication.java**

@SpringBootApplication

@EnableConfigServer

public class ConfigServerApplication {

public static void main(String[] args) {

SpringApplication.run(ConfigServerApplication.class, args);

}

}

**DISCOVERY SERVER**

**application.properties**

server.port=8761

spring.application.name=discovery-server

**DiscoveryServerApplication.java**

@SpringBootApplication

@EnableEurekaServer

public class DiscoveryServerApplication {

public static void main(String[] args) {

SpringApplication.run(DiscoveryServerApplication.class, args);

}

}

PRODUCT SERVICE

**bootstrap.properties**

spring.application.name=product-service

spring.config.import=optional:configserver:http://localhost:8888

**application.properties**

server.port=8083

spring.datasource.url=jdbc:mysql://localhost:3306/productdb

spring.datasource.username=root

spring.datasource.password=your\_password

spring.jpa.hibernate.ddl-auto=update

spring.cloud.discovery.enabled=true

**Product.java**

@Entity

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private Double price;

public Product() {}

public Product(Long id, String name, Double price) {

this.id = id;

this.name = name;

this.price = price;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public Double getPrice() {

return price;

}

public void setPrice(Double price) {

this.price = price;

}

}

**ProductRepository.java**

public interface ProductRepository extends JpaRepository<Product, Long> {}

**ProductController.java**

@RestController

@RequestMapping("/products")

public class ProductController {

@Autowired

private ProductRepository repo;

@GetMapping

public List<Product> all() {

return repo.findAll();

}

@PostMapping

public Product save(@RequestBody Product p) {

return repo.save(p);

}

@GetMapping("/{id}")

public ResponseEntity<Product> get(@PathVariable Long id) {

return repo.findById(id)

.map(ResponseEntity::ok)

.orElse(ResponseEntity.notFound().build());

}

}

**ProductServiceApplication.java**

@SpringBootApplication

@EnableDiscoveryClient

public class ProductServiceApplication {

public static void main(String[] args) {

SpringApplication.run(ProductServiceApplication.class, args);

}

}

**INVENTORY SERVICE**

**bootstrap.properties**

spring.application.name=inventory-service

spring.config.import=optional:configserver:http://localhost:8888

**application.properties**

server.port=8084

spring.datasource.url=jdbc:mysql://localhost:3306/inventorydb

spring.datasource.username=root

spring.datasource.password=your\_password

spring.jpa.hibernate.ddl-auto=update

spring.cloud.discovery.enabled=true

**Inventory.java**

@Entity

public class Inventory {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private Long productId;

private int quantity;

public Inventory() {}

public Inventory(Long id, Long productId, int quantity) {

this.id = id;

this.productId = productId;

this.quantity = quantity;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public Long getProductId() {

return productId;

}

public void setProductId(Long productId) {

this.productId = productId;

}

public int getQuantity() {

return quantity;

}

public void setQuantity(int quantity) {

this.quantity = quantity;

}

}

**InventoryRepository.java**

public interface InventoryRepository extends JpaRepository<Inventory, Long> {}

**ProductClient.java**

@FeignClient(name = "product-service")

public interface ProductClient {

@GetMapping("/products/{id}")

Product getProduct(@PathVariable("id") Long id);

}

**InventoryResponse.java**

public class InventoryResponse {

private Inventory inventory;

private Product product;

public InventoryResponse() {}

public InventoryResponse(Inventory inventory, Product product) {

this.inventory = inventory;

this.product = product;

}

public Inventory getInventory() {

return inventory;

}

public void setInventory(Inventory inventory) {

this.inventory = inventory;

}

public Product getProduct() {

return product;

}

public void setProduct(Product product) {

this.product = product;

}

}

**InventoryController.java**

@RestController

@RequestMapping("/inventory")

public class InventoryController {

@Autowired

private InventoryRepository repo;

@Autowired

private ProductClient client;

@PostMapping

public Inventory add(@RequestBody Inventory i) {

return repo.save(i);

}

@GetMapping("/product/{id}")

public ResponseEntity<InventoryResponse> get(@PathVariable Long id) {

Inventory inv = repo.findAll().stream()

.filter(i -> i.getProductId().equals(id)).findFirst().orElse(null);

if (inv == null) return ResponseEntity.notFound().build();

Product prod = client.getProduct(id);

InventoryResponse resp = new InventoryResponse(inv, prod);

return ResponseEntity.ok(resp);

}

}

**InventoryServiceApplication.java**

@SpringBootApplication

@EnableDiscoveryClient

@EnableFeignClients

public class InventoryServiceApplication {

public static void main(String[] args) {

SpringApplication.run(InventoryServiceApplication.class, args);

}

}

**CURL Output:**

$ curl -X POST http://localhost:8083/products -H "Content-Type: application/json" -d '{"name":"Monitor","price":12000}'

{"id":1,"name":"Monitor","price":12000.0}

$ curl -X POST http://localhost:8084/inventory -H "Content-Type: application/json" -d '{"productId":1,"quantity":25}'

{"id":1,"productId":1,"quantity":25}

$ curl http://localhost:8084/inventory/product/1

{"inventory":{"id":1,"productId":1,"quantity":25},"product":{"id":1,"name":"Monitor","price":12000.0}}

**Implement an API Gateway**

**application.yml**

server:

port: 8080

spring:

application:

name: api-gateway

cloud:

gateway:

routes:

- id: customer-service

uri: http://localhost:8081

predicates:

- Path=/customers/\*\*

filters:

- RewritePath=/customers/(?<segment>.\*), /${segment}

- name: RequestRateLimiter

args:

redis-rate-limiter.replenishRate: 5

redis-rate-limiter.burstCapacity: 10

- id: billing-service

uri: http://localhost:8082

predicates:

- Path=/billing/\*\*

filters:

- RewritePath=/billing/(?<segment>.\*), /${segment}

- AddResponseHeader=X-Cache-Status, CACHED

**ApiGatewayApplication.java**

@SpringBootApplication

public class ApiGatewayApplication {

public static void main(String[] args) {

SpringApplication.run(ApiGatewayApplication.class, args);

}

}

**CUSTOMER SERVICE**

**application.properties**

server.port=8081

spring.application.name=customer-service

**Customer.java**

public class Customer {

private Long id;

private String name;

public Customer() {}

public Customer(Long id, String name) {

this.id = id;

this.name = name;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**CustomerController.java**

@RestController

public class CustomerController {

@GetMapping("/")

public String home() {

return "Customer Service Home";

}

@GetMapping("/{id}")

public Customer getCustomer(@PathVariable Long id) {

return new Customer(id, "Customer-" + id);

}

}

**CustomerServiceApplication.java**

@SpringBootApplication

public class CustomerServiceApplication {

public static void main(String[] args) {

SpringApplication.run(CustomerServiceApplication.class, args);

}

}

**BILLING SERVICE**

**application.properties**

server.port=8082

spring.application.name=billing-service

**Bill.java**

public class Bill {

private Long id;

private Double amount;

public Bill() {}

public Bill(Long id, Double amount) {

this.id = id;

this.amount = amount;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public Double getAmount() {

return amount;

}

public void setAmount(Double amount) {

this.amount = amount;

}

}

**BillingController.java**

@RestController

public class BillingController {

@GetMapping("/")

public String home() {

return "Billing Service Home";

}

@GetMapping("/{id}")

public Bill getBill(@PathVariable Long id) {

return new Bill(id, 199.99);

}

}

**BillingServiceApplication.java**

@SpringBootApplication

public class BillingServiceApplication {

public static void main(String[] args) {

SpringApplication.run(BillingServiceApplication.class, args);

}

}

**Output:**

curl http://localhost:8080/customers/1

{"id":1,"name":"Customer-1"}

curl http://localhost:8080/billing/2

{"id":2,"amount":199.99}

**Resilient Microservices with Circuit Breaker**

**application.properties**

server.port=8083

spring.application.name=payment-service

**third-party-service/src/main/resources/application.properties**

server.port=8084

spring.application.name=third-party-service

**PaymentServiceApplication.java**

@SpringBootApplication

public class PaymentServiceApplication {

public static void main(String[] args) {

SpringApplication.run(PaymentServiceApplication.class, args);

}

}

**PaymentController.java**

@RestController

@RequestMapping("/payment")

public class PaymentController {

@Autowired

private WebClient.Builder webClientBuilder;

@GetMapping("/{id}")

@CircuitBreaker(name = "thirdPartyAPI", fallbackMethod = "fallbackResponse")

public ResponseEntity<String> getPaymentStatus(@PathVariable Long id) {

return webClientBuilder.build()

.get()

.uri("http://localhost:8084/thirdparty/" + id)

.retrieve()

.toEntity(String.class)

.block();

}

public ResponseEntity<String> fallbackResponse(Long id, Throwable t) {

System.out.println("Fallback triggered due to: " + t.getMessage());

return ResponseEntity.ok("Payment status for ID " + id + " is temporarily unavailable. Please try again later.");

}

}

**PaymentConfig.java**

@Configuration

public class PaymentConfig {

@Bean

public WebClient.Builder webClientBuilder() {

return WebClient.builder();

}

}

**ThirdPartyServiceApplication.java**

@SpringBootApplication

public class ThirdPartyServiceApplication {

public static void main(String[] args) {

SpringApplication.run(ThirdPartyServiceApplication.class, args);

}

}

**ThirdPartyController.java**

@RestController

@RequestMapping("/thirdparty")

public class ThirdPartyController {

@GetMapping("/{id}")

public ResponseEntity<String> getStatus(@PathVariable Long id) throws InterruptedException {

Thread.sleep(5000); // Simulate delay

return ResponseEntity.ok("Processed payment for ID: " + id);

}

}

**Payment.java**

public class Payment {

private Long id;

private String status;

public Payment() {}

public Payment(Long id, String status) {

this.id = id;

this.status = status;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getStatus() {

return status;

}

public void setStatus(String status) {

this.status = status;

}

}

**Output:**

1. Before delay triggers fallback:

curl http://localhost:8083/payment/1

Output:

Processed payment for ID: 1

2. After delay triggers circuit breaker fallback:

Payment status for ID 1 is temporarily unavailable. Please try again later.

GET /payment/1

Status: 200 OK

Body:

Processed payment for ID: 1

After CircuitBreaker opens

Status: 200 OK

Body:

Payment status for ID 1 is temporarily unavailable. Please try again later.