Spring 2025 – SWER007

Lab 10: API Gateway with Spring Cloud Gateway

✅ Objective

Create an **API Gateway** using Spring Cloud Gateway to route traffic to employee-service, department-service, and (later) auth-service. The gateway will serve as the **single entry point** for the entire microservices architecture.

🔧 Prerequisites

You should have:

- employee-service and department-service registered with discovery-service
- Working knowledge of service names from Eureka

1 Create API Gateway Project

Use Spring Initializr or your IDE to create:

🔹 api-gateway-service

```
spring init --dependencies=cloud-gateway,eureka-client api-gateway-service
```

Add Spring Cloud Gateway dependency (if not already included):

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-starter-gateway</artifactId>
</dependency>
```

2 Configure Gateway to Use Eureka

Update application.yml:

```
server:
  port: 8080

spring:
  application:
    name: api-gateway

  cloud:
    gateway:
      discovery:
        locator:
          enabled: true
          lower-case-service-id: true
      routes:
        - id: employee-service
          uri: lb://employee-service
          predicates:
            - Path=/employees/**
      - id: department-service
        uri: lb://department-service
        predicates:
          - Path=/departments/**

eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka
```

✅ **What This Does:**

- Enables **service discovery-based routing**
- Routes:
  - /employees/** → employee-service
  - /departments/** → department-service

3 Run and Test Gateway

1. Start the following apps:
   - discovery-service
   - employee-service
   - department-service
   - api-gateway-service
2. Visit:
   - http://localhost:8080/employees → Should route to employee service
   - http://localhost:8080/departments → Should route to department service

4 Add Logging Filter (Optional Bonus)

To observe requests passing through the gateway, add a **simple logging filter**.

Create a filter:

```
@Component
public class LoggingFilter implements GlobalFilter, Ordered {

    @Override
    public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {
        System.out.println("Incoming request: " + exchange.getRequest().getURI());
        return chain.filter(exchange);
    }

    @Override
    public int getOrder() {
        return -1;
    }
}
```

📝 Lab Tasks

1. Create api-gateway-service project
2. Enable Eureka client and register with discovery-service
3. Add routing config for employee-service and department-service
4. Run and test both routes from the gateway
5. (Optional) Add a logging filter

🎯 Expected Outcome

- API Gateway running on port 8080
- Central routing for /employees/** and /departments/**
- No more direct access to microservices from clients