**Day3: Lab: Service Discovery Pattern – Server-side Gateway**

**Example of implementing the API Gateway Pattern using Spring Boot microservices** with **Spring Cloud Gateway** as the gateway:

**Tech Stack**

* Java 17+
* Spring Boot
* Spring Cloud Gateway (API Gateway)
* Spring Boot Microservices (Product Service, Order Service)
* Eureka Discovery Server
* IntelliJ IDEA (or any IDE)
* Maven

**Microservice Components**

1. **Eureka Discovery Server** – for service registration
2. **API Gateway** – routes requests to services
3. **Product Service** – microservice providing product data
4. **Order Service** – microservice managing orders

**Step 1: Create Eureka Discovery Server**

**pom.xml**

<dependencies>

<dependency>

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

<artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>

</dependency>

</dependencies>

**application.yml**

server:

port: 8761

eureka:

client:

register-with-eureka: false

fetch-registry: false

**EurekaServerApplication.java**

@SpringBootApplication

@EnableEurekaServer

public class EurekaServerApplication {

public static void main(String[] args) {

SpringApplication.run(EurekaServerApplication.class, args);

}

}

Run it and visit: http://localhost:8761

**Step 2: Create Product Service**

**pom.xml**

<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-client</artifactId>

</dependency>

</dependencies>

**application.yml**

server:

port: 8081

spring:

application:

name: product-service

eureka:

client:

service-url:

defaultZone: http://localhost:8761/eureka

**ProductController.java**

@RestController

@RequestMapping("/products")

public class ProductController {

@GetMapping

public List<String> getProducts() {

return List.of("TV", "Laptop", "Phone");

}

}

**Step 3: Create Order Service**

Similar setup as Product Service.

**application.yml**

server:

port: 8082

spring:

application:

name: order-service

eureka:

client:

service-url:

defaultZone: http://localhost:8761/eureka

**OrderController.java**

@RestController

@RequestMapping("/orders")

public class OrderController {

@GetMapping

public List<String> getOrders() {

return List.of("Order1", "Order2");

}

}

**Step 4: Create API Gateway**

**pom.xml**

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>

</dependency>

</dependencies>

**application.yml**

server:

port: 8080

spring:

application:

name: api-gateway

cloud:

gateway:

discovery:

locator:

enabled: true

routes:

- id: product-service

uri: lb://product-service

predicates:

- Path=/products/\*\*

- id: order-service

uri: lb://order-service

predicates:

- Path=/orders/\*\*

eureka:

client:

service-url:

defaultZone: http://localhost:8761/eureka

**Step 5: Test Everything**

1. **Run Eureka Server**
2. **Run Product & Order services**
3. **Run API Gateway**

**Test URLs via API Gateway:**

* http://localhost:8080/products → routed to product-service
* http://localhost:8080/orders → routed to order-service

**📌 Summary**

| **Component** | **Port** | **Role** |
| --- | --- | --- |
| Eureka Server | 8761 | Discovery |
| API Gateway | 8080 | Gateway/Router |
| Product Service | 8081 | Microservice |
| Order Service | 8082 | Microservice |