09 - Service Discovery and Communication (Part 2)

Table of Contents

Agenda

- Overview of API Gateway
- Route requests through API Gateway

Overview of API Gateway

1. What is an API Gateway?

- **Definition:**
  - An API Gateway is a server that acts as an intermediary for requests from clients seeking access to backend services. It consolidates multiple service requests into a single request, providing a unified entry point for a microservices architecture.
- **Purpose of an API Gateway:**
  - **Request Routing:** Directs client requests to the appropriate backend services.
  - **Authentication and Security:** Manages authentication, authorization, and encryption of requests.
  - **Load Balancing:** Distributes incoming requests across multiple service instances.
  - **Caching:** Reduces the load on services by caching responses.
  - **Rate Limiting:** Controls the rate at which clients can make requests, preventing overuse of resources.
  - **Service Aggregation:** Combines data from multiple services into a single response, reducing the number of client calls.
- **Benefits:**
  - Simplifies client interactions with microservices.
  - Centralizes cross-cutting concerns like security, logging, and monitoring.
  - Enhances scalability and reliability.

- **Popular API Gateway Solutions:**
    - **Spring Cloud Gateway:** A popular API Gateway framework in the Spring ecosystem, built on top of the Spring Framework.
    - **Netflix Zuul:** Another gateway solution, although now mostly replaced by Spring Cloud Gateway.
    - **NGINX, Kong, and AWS API Gateway:** Other well-known API Gateway solutions outside the Spring ecosystem.

2. Key Features of Spring Cloud Gateway:

- **Routing:** Directs incoming requests to appropriate microservices based on the request path, headers, etc.
- **Filters:** Pre and post-processing of requests and responses (e.g., adding headers, logging, modifying request/response).
- **Predicate Factories:** Built-in mechanisms to match incoming requests (e.g., by path, method, headers).

Hands-On

- Create a new microservice -> API GATEWAY. And add dependencies
    1. Gateway - Spring Cloud Routing
    2. Eureka Discovery Client - Spring Cloud Discovery
    3. Spring Boot Actuator - OPS

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
<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>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
```

- In application.properties:

```
spring.application.name=ApiGateway

server.port=8083

# Routes configurations


spring.cloud.gateway.routes[0].id=UserService
spring.cloud.gateway.routes[0].uri=lb://UserService
spring.cloud.gateway.routes[0].predicates[0]=Path=/user/**

spring.cloud.gateway.routes[1].id=ProductsService
spring.cloud.gateway.routes[1].uri=lb://ProductsService
spring.cloud.gateway.routes[1].predicates[0]=Path=/products/**

management.endpoints.web.exposure.include=*
management.endpoint.health.show-details=always
```