08 - Service Discovery and Communication (Part 1)

Table of Contents

- Agenda
  - Service Discovery with Eureka
    - Introduction to Eureka Server
  - What is Load Balancing?
    - Implementing Load Balancing using Spring Load Balancer
  - Hands-On - Configuring Eureka Server and Eureka Clients
  - Hands-On Configuring Load Balancing

Agenda

- Service Discovery with Eureka
  - Introduction to Eureka Server
  - Setting up Eureka Server for service discovery
  - Registering services with Eureka
- What is Load Balancing?
  - Implementing load balancing using Spring Load Balancer

Service Discovery with Eureka

Introduction to Eureka Server

- What is Eureka?
  - Eureka is a service discovery tool from Netflix, part of the Netflix OSS stack.
  - It acts as a registry for services, enabling automatic detection and communication between services in a microservices architecture.
- Why Service Discovery?
  - In microservices, services are dynamic and can scale up or down. Eureka helps manage the dynamic nature by registering services and enabling clients to discover them without hard-coding service locations.
- Eureka Server and Eureka Client
  - Eureka Server: The registry where services (clients) register themselves and retrieve information about other registered services.
  - Eureka Client: A service that registers itself with the Eureka Server and uses it to discover other services.

What is Load Balancing?

- **Load Balancing Overview:**
  - Load balancing distributes incoming network traffic across multiple instances of a service to ensure no single instance becomes overwhelmed. This improves reliability and availability.

Implementing Load Balancing using Spring Load Balancer

- **Spring Load Balancer:**
  - Spring Cloud provides a client-side load balancing solution that helps balance the load among service instances registered with Eureka.
- **Dependencies:**
  - Ensure that the `spring-cloud-starter-loadbalancer` dependency is included in the service's `pom.xml`.

Hands-On - Configuring Eureka Server and Eureka Clients

1. Download `ServiceRegistry` from Spring Initialiser. Use Dependencies:
   1. Eureka Server - Spring Cloud Discovery
   2. Spring Web
2. Configure pom.xml for other microservices which are required as client.

Add Dependency Management for Spring Cloud

```xml
<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>
```

In the properties add spring-cloud-version

```xml
<properties>
    <java.version>21</java.version>
    <spring-cloud.version>2023.0.3</spring-cloud.version>
  </properties>
```

Add Eureka Discovery Client - Spring Cloud Discovery dependency

```
 <dependency>
      <groupId>org.springframework.cloud</groupId>
      <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
</dependency>
```

3. In application.properties configure

In server

```
spring.application.name=ServiceRegistry

server.port=8761

#disable as client
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
```

In clients

```
eureka.instance.client.serverUrl.defaultZone=http://localhost:8761
```

Now your eureka clients will appear in the eureka server

Hands-On Configuring Load Balancing

1. Add Load Balancer dependency to the service which you want to balance the load of

In this case ProductService

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

2. Go to ProductClient in UserService and edit the URL to name

```
//@FeignClient(url = "http://localhost:8082", value = "Product-Client")

TO

@FeignClient(name="ProductService")
```

3. Change the port to run another instance.
