

Microservices Workflow (E-Commerce Example)

Case Study

Eureka Server(8761)

1.pom.xml

2.application.yml

3.eureka server.application

POM.XML

```
<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 http://maven.apache.org/xsd/maven-4.0.0.xsd">
```

```
<modelVersion>4.0.0</modelVersion>
```

```
<groupId>com.example</groupId>
```

```
<artifactId>eureka-server</artifactId>
```

```
<version>0.0.1-SNAPSHOT</version>
```

```
<parent>
```

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

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

```
<version>3.1.3</version>
```

```
<relativePath/>
```

```
</parent>
```

```
<dependencies>
```

```
<dependency>
```

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

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

```
</dependency>
<dependency>
<groupId>org.projectlombok</groupId>
<artifactId>lombok</artifactId>
<optional>true</optional>
</dependency>
</dependencies>
</project>
```

APPLICATION.YML

```
server:
  port: 8761
spring:
  application:
    name: eureka-server
eureka:
  client:
    register-with-eureka: false
    fetch-registry: false
```

EUREKA SERVER APPLICATION.JAVA

```
package com.ecommerce.eurekaserver;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;
@SpringBootApplication
```

```

@EnableEurekaServer

public class EurekaServerApplication {

    public static void main(String[] args) {

        SpringApplication.run(EurekaServerApplication.class, args);

    }

}

```

PRODUCT SERVICE(8081)

1. pom.xml
2. application.yml
3. Product.java
4. ProductController.java
5. ProductServiceApplication.java

Pom.xml

```

<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
        http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.ecommerce</groupId>
    <artifactId>product-service</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>product-service</name>

```

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>3.2.4</version>
  <relativePath/>
</parent>
```

```
<properties>
  <java.version>17</java.version>
</properties>
```

```
<dependencies>
  <!-- Web -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
```

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

```
  <!-- Lombok -->
  <dependency>
    <groupId>org.projectlombok</groupId>
```

```
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>

    <!-- Actuator -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-actuator</artifactId>
    </dependency>
</dependencies>

<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-dependencies</artifactId>
            <version>2023.0.1</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.yml

```
server:
  port: 8081
spring:
  application:
    name: product-service
eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka
```

Product.java

```
package com.ecommerce.productservice.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Product {

    private Long id;
```

```
private String name;  
private String description;  
private double price;  
}
```

Product Controller.java

```
package com.ecommerce.productservice.controller;  
import com.ecommerce.productservice.model.Product;  
import org.springframework.web.bind.annotation.*;  
import java.util.*;  
  
@RestController  
@RequestMapping("/api/products")  
public class ProductController {  
  
    private Map<Long, Product> productMap = new HashMap<>();  
  
    public ProductController() {  
        // Sample data  
        productMap.put(1L, new Product(1L, "Laptop", "Gaming Laptop", 85000.0));  
        productMap.put(2L, new Product(2L, "Phone", "Smartphone", 25000.0));  
    }  
  
    @GetMapping  
    public List<Product> getAllProducts() {  
        return new ArrayList<>(productMap.values());  
    }  
}
```

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

```
public Product getProductById(@PathVariable Long id) {
```

```
    return productMap.get(id);
```

```
}
```

```
@PostMapping
```

```
public Product addProduct(@RequestBody Product product) {
```

```
    product.setId(new Random().nextLong(1000));
```

```
    productMap.put(product.getId(), product);
```

```
    return product;
```

```
}
```

```
@PutMapping("/{id}")
```

```
public Product updateProduct(@PathVariable Long id, @RequestBody Product product) {
```

```
    product.setId(id);
```

```
    productMap.put(id, product);
```

```
    return product;
```

```
}
```

```
@DeleteMapping("/{id}")
```

```
public String deleteProduct(@PathVariable Long id) {
```

```
    productMap.remove(id);
```

```
    return "Product deleted with ID: " + id;
```

```
}
```

```
}
```

Product Service Application.java


```
package com.ecommerce.productservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
```

```
@SpringBootApplication
```

```
@EnableEurekaClient
```

```
public class ProductServiceApplication {

    public static void main(String[] args) {

        SpringApplication.run(ProductServiceApplication.class, args);

    }

}
```

ORDER SERVICE(8082)

project structure

order-service/

├── controller/

| └── OrderController.java

├── model/

| ├── Order.java

| └── Product.java

├── config/

| └── WebClientConfig.java

└── OrderServiceApplication.java

├── resources/

| └── application.yml

└─ pom.xml

OrderServiceApplication.java

```
package com.ecommerce.orderservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;

@SpringBootApplication
@EnableEurekaClient

public class OrderServiceApplication {

    public static void main(String[] args) {

        SpringApplication.run(OrderServiceApplication.class, args);

    }

}
```

Order.java

```
package com.ecommerce.orderservice.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@NoArgsConstructor
public class Order {

    private String id;

    private String productId;

    private String productName;
```

```
    private double productPrice;
    private String status;
}
```

Product.java

```
package com.ecommerce.orderservice.model;
import lombok.Data;
@Data
public class Product {
    private String id;
    private String name;
    private String description;
    private double price;
}
```

OrderController.java

```
package com.ecommerce.orderservice.controller;
import com.ecommerce.orderservice.model.Order;
import com.ecommerce.orderservice.model.Product;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.reactive.function.client.WebClient;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
@RestController
@RequestMapping("/api/orders")
```

@RequiredArgsConstructor

public class OrderController {

private final WebClient.Builder webClientBuilder;

private List<Order> orderList = new ArrayList<>();

@PostMapping("/{productId}")

public Order placeOrder(@PathVariable String productId) {

Product product = webClientBuilder.build()

.get()

.uri("http://product-service/api/products/" + productId)

.retrieve()

.bodyToMono(Product.class)

.block();

if (product == null) {

throw new RuntimeException("Product not found: " + productId);

}

Order order = new Order();

order.setId(UUID.randomUUID().toString());

order.setProductId(product.getId());

order.setProductName(product.getName());

order.setProductPrice(product.getPrice());

order.setStatus("PLACED");

orderList.add(order);

```
        return order;
    }

    @GetMapping
    public List<Order> getAllOrders() {
        return orderList;
    }
}
```

WebClientConfig.java

```
package com.ecommerce.orderservice.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.reactive.function.client.WebClient;

@Configuration
public class WebClientConfig {

    @Bean
    public WebClient.Builder webClientBuilder() {
        return WebClient.builder();
    }
}
```

application.yml

```
server:
  port: 8082

spring:
```

application:

name: order-service

eureka:

client:

service-url:

defaultZone: http://localhost:8761/eureka

pom.xml

```
<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
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
```

```
<modelVersion>4.0.0</modelVersion>
```

```
<groupId>com.ecommerce</groupId>
```

```
<artifactId>order-service</artifactId>
```

```
<version>0.0.1-SNAPSHOT</version>
```

```
<name>order-service</name>
```

```
<description>Order Microservice</description>
```

```
<packaging>jar</packaging>
```

```
<properties>
```

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

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

```
</properties>
```

```
<dependencies>
```

<!-- Spring Boot Starter Web -->

<dependency>

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

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

</dependency>

<!-- WebClient -->

<dependency>

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

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

</dependency>

<!-- Eureka Client -->

<dependency>

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

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

</dependency>

<!-- Lombok -->

<dependency>

 <groupId>org.projectlombok</groupId>

 <artifactId>lombok</artifactId>

 <scope>provided</scope>

</dependency>

<!-- Testing -->

<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>
        <!-- Spring Boot Plugin -->
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>

```


</project>

PAYMENT SERVICE(8083)

Project Structure

payment-service/

```
├── src/
│   ├── main/
│   │   ├── java/
│   │   │   ├── com/
│   │   │   │   ├── ecommerce/
│   │   │   │   │   ├── paymentservice/
│   │   │   │   │   │   ├── PaymentServiceApplication.java
│   │   │   │   │   │   ├── controller/
│   │   │   │   │   │   │   ├── PaymentController.java
│   │   │   │   │   │   │   ├── model/
│   │   │   │   │   │   │   │   ├── Payment.java
│   │   │   │   │   │   └── resources/
│   │   │   │   │   └── application.yml
│   │   └── pom.xml
```

PaymentServiceApplication.java

```
package com.ecommerce.paymentservice;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;

@SpringBootApplication
```

```
@EnableEurekaClient

public class PaymentServiceApplication {

    public static void main(String[] args) {

        SpringApplication.run(PaymentServiceApplication.class, args);

    }

}
```

Payment.java

```
package com.ecommerce.paymentservice.model;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor

public class Payment {

    private String paymentId;

    private String orderId;

    private double amount;

    private String paymentStatus;

}
```

PaymentController.java

```
package com.ecommerce.paymentservice.controller;

import com.ecommerce.paymentservice.model.Payment;
import org.springframework.web.bind.annotation.*;
import java.util.*;
```

```

@RestController
@RequestMapping("/api/payments")
public class PaymentController {

    private final Map<String, Payment> payments = new HashMap<>();

    @PostMapping
    public Payment makePayment(@RequestBody Payment payment) {
        String paymentId = UUID.randomUUID().toString();
        payment.setPaymentId(paymentId);
        payment.setPaymentStatus("SUCCESS");
        payments.put(paymentId, payment);
        return payment;
    }

    @GetMapping
    public Collection<Payment> getAllPayments() {
        return payments.values();
    }

    @GetMapping("/{paymentId}")
    public Payment getPaymentById(@PathVariable String paymentId) {
        return payments.get(paymentId);
    }
}

```

application.yml

server:

port: 8083

spring:

application:

name: payment-service

eureka:

client:

service-url:

defaultZone: http://localhost:8761/eureka

pom.xml

```
<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
    http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.ecommerce</groupId>
  <artifactId>payment-service</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>payment-service</name>
  <description>Payment Microservice</description>

  <properties>
    <java.version>17</java.version>
    <spring-cloud.version>2022.0.4</spring-cloud.version>
  </properties>
```

```
<dependencies>
```

```
<!-- Spring Boot Starter Web -->
```

```
<dependency>
```

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

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

```
</dependency>
```

```
<!-- Eureka Client -->
```

```
<dependency>
```

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

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

```
</dependency>
```

```
<!-- Lombok -->
```

```
<dependency>
```

```
<groupId>org.projectlombok</groupId>
```

```
<artifactId>lombok</artifactId>
```

```
<scope>provided</scope>
```

```
</dependency>
```

```
<!-- Spring Boot Test -->
```

```
<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>
    <!-- Spring Boot Maven Plugin -->
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
</project>
```

Postman Testing

Eureka Server at <http://localhost:8761>

Test Product Service

Add Product

POST <http://localhost:8081/api/products>

Body (JSON):

```
{  
  "name": "iPhone 15",  
  "description": "Latest Apple iPhone",  
  "price": 79999.0  
}
```

Get All Products

GET <http://localhost:8081/api/products>

Test Order Service

Place Order

POST <http://localhost:8082/api/orders/{productId}>

<http://localhost:8082/api/orders/1>

Get All Orders

GET <http://localhost:8082/api/orders>

Get All Payments

GET <http://localhost:8083/api/payments>