Day15_Class_Assignmnet

Microservices Workflow (E-Commerce Example)

1. Product Service (8081) – Product Catalog Management

Entity: Product Responsibilities:

•Add new products to the catalog.

• Update product details (price, stock, etc.).

• View product details by ID or list all products.

Example Workflow:

Admin adds a product: Laptop, $1200, stock=10.

Customers can view all products or search for specific products.

Stock decreases when an order is successfully placed (through order-service)

2. Order Service (8082) – Order Management
   Entity: Order
   Responsibilities:
   • Accept order requests from customers.
   • Validate product availability via product-service.
   • Calculate total price based on product details.
   • Forward payment request to payment-service.
   • Update order status after payment confirmation.

Example Workflow

Customer places an order for Product ID P101 (Quantity = 2).

Order Service → Product Service:

◦ Checks if Product P101 exists and stock is ≥ 2.

If available, the order is tentatively created with status "PENDING_PAYMENT".

Order Service → Payment Service: Sends payment request for the total amount.

If payment is successful, order status is updated to "CONFIRMED".

Order Service → Product Service: Reduces stock count by the quantity ordered.

3. Payment Service (8083) – Payment Processing
   Entity: Payment Responsibilities:
   Receive payment requests from order-service.
   • Validate payment details (amount, order ID).

• Simulate/execute payment transaction (e.g., with a payment gateway).
• Send payment confirmation back to order-service

Eureka

Pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>

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

        <modelVersion>4.0.0</modelVersion>

        <parent>

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

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

                <version>3.5.4</version>

                <relativePath/> <!-- lookup parent from repository -->

        </parent>

        <groupId>com.example</groupId>

        <artifactId>eureka</artifactId>

        <version>0.0.1-SNAPSHOT</version>

        <name>eureka</name>

        <description>Demo project for Eureka Server</description>

        <url/>

        <licenses>

                <license/>

        </licenses>

        <developers>

                <developer/>

        </developers>

        <scm>
```

```xml
		<connection/>
		<developerConnection/>
		<tag/>
		<url/>
	</scm>
	<properties>
		<java.version>17</java.version>
		<spring-cloud.version>2025.0.0</spring-cloud.version>
	</properties>
	<dependencies>
		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-starter-actuator</artifactId>
		</dependency>
		<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-server</artifactId>
		</dependency>

		<dependency>
			<groupId>org.springframework.boot</groupId>
			<artifactId>spring-boot-devtools</artifactId>
			<scope>runtime</scope>
			<optional>true</optional>
```

```xml
        </dependency>
        <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>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
```

```
</project>
```

Applicationproperties:

```
server.port=8761
spring.application.name=eureka-server

eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
```

EurekaExampleApplication:

```
package com.example.Eureka.Example;

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

@SpringBootApplication
@EnableEurekaServer
public class EurekaExampleApplication {

	public static void main(String[] args) {
		SpringApplication.run(EurekaExampleApplication.class, args);
	}

}
```

Productmanagement:

Entity:

Product.java:

```
package com.example.product.management.entity;


import jakarta.persistence.Entity;
import jakarta.persistence.Id;


@Entity
public class Product {
	@Id
	private String id;
	private String name;
	private double price;
	private int stock;


	public String getId() {
		return id;
	}


	public void setId(String id) {
		this.id = id;
	}


	public String getName() {
		return name;
	}


	public void setName(String name) {
		this.name = name;
	}

	public double getPrice() {
		return price;
	}

	public void setPrice(double price) {
		this.price = price;
	}

	public int getStock() {
		return stock;
	}

	public void setStock(int stock) {
		this.stock = stock;
	}

}
```

Controller:

ProductController:

```
package com.example.product.management.controller;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.example.product.management.entity.Product;
import com.example.product.management.repository.ProductRepository;


@RestController
@RequestMapping("/products")
public class ProductController {

   @Autowired
   private ProductRepository repo;

   @PostMapping
   public Product addProduct(@RequestBody Product p) {
      return repo.save(p);
   }

   @GetMapping("/{id}")
   public Product getProduct(@PathVariable String id) {
      return repo.findById(id).orElse(null);
   }
```

```
    @GetMapping
    public List<Product> getAll() {
        return repo.findAll();
    }

    @PutMapping("/{id}/reduceStock")
    public ResponseEntity<String> reduceStock(@PathVariable String id, @RequestParam int
qty) {
        Product product = repo.findById(id).orElse(null);
        if (product != null && product.getStock() >= qty) {
            product.setStock(product.getStock() - qty);
            repo.save(product);
            return ResponseEntity.ok("Stock reduced.");
        }
        return ResponseEntity.badRequest().body("Insufficient stock.");
    }
}
```

Repository:

ProductRepository:

```
package com.example.product.management.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.example.product.management.entity.Product;
@Repository
public interface ProductRepository extends JpaRepository<Product, String> {
}
```

Applicationproperties:

```
spring.application.name=product-service

server.port=8081

spring.datasource.url=jdbc:h2:mem:productdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.h2.console.enabled=true

eureka.client.service-url.defaultZone=http://localhost:8761/eureka

management.endpoints.web.exposure.include=*

management.endpoint.health.show-details=always

info.app.name=Product Service

info.app.version=1.0.0
```

pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>

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

	<modelVersion>4.0.0</modelVersion>

	<parent>

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

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

		<version>3.5.4</version>

		<relativePath/> <!-- lookup parent from repository -->

	</parent>
```

```xml
	<groupId>com.example</groupId>
	<artifactId>productservice1</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<name>product-service1</name>
	<description>Demo project for Spring Boot</description>
	<url/>
	<licenses>
		<license/>
	</licenses>
	<developers>
		<developer/>
	</developers>
	<scm>
		<connection/>
		<developerConnection/>
		<tag/>
		<url/>
	</scm>
	<properties>
		<java.version>17</java.version>
		<spring-cloud.version>2025.0.0</spring-cloud.version>
	</properties>
	<dependencies>
	<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<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>

<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-devtools</artifactId>
        <scope>runtime</scope>
        <optional>true</optional>
</dependency>
<dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
        <scope>runtime</scope>
</dependency>
<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>
                        <plugin>
                                <groupId>org.springframework.boot</groupId>
                                <artifactId>spring-boot-maven-plugin</artifactId>
                        </plugin>
                </plugins>
        </build>

</project>
```

Ordermanage:

Ordermanageapplication:

```
package com.example.ordermanage;
```

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.client.discovery.EnableDiscoveryClient;
import org.springframework.cloud.openfeign.EnableFeignClients;

@SpringBootApplication
@EnableDiscoveryClient
@EnableFeignClients
public class OrdermanageApplication {

	public static void main(String[] args) {
		SpringApplication.run(OrdermanageApplication.class, args);
	}

}
```

Applicationproperties:

```
server.port=8082
spring.application.name=order-service
spring.datasource.url=jdbc:h2:mem:orderdb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.h2.console.enabled=true
eureka.client.service-url.defaultZone=http://localhost:8761/eureka
```

```
management.endpoints.web.exposure.include=*

management.endpoint.health.show-details=always

info.app.name=Order Service

info.app.version=1.0.0
```

Entiry:

Order.java:

```java
package com.example.ordermanage.entity;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;


@Entity

@Table(name = "orders")

public class Order {

   @Id

   private String id;

   private String productId;

   private int quantity;

   private double totalAmount;

   private String status;


   public String getId() {

      return id;

   }

   public void setId(String id) {

      this.id = id;

   }
```

```
    public String getProductId() {
        return productId;
    }
    public void setProductId(String productId) {
        this.productId = productId;
    }
    public int getQuantity() {
        return quantity;
    }
    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
    public double getTotalAmount() {
        return totalAmount;
    }
    public void setTotalAmount(double totalAmount) {
        this.totalAmount = totalAmount;
    }
    public String getStatus() {
        return status;
    }
    public void setStatus(String status) {
        this.status = status;
    }
}
```

Repository:

OrderRepository:

```
package com.example.ordermanage.repository;
```

```java
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.example.ordermanage.entity.Order;
@Repository
public interface OrderRepository extends JpaRepository<Order, String> {
}
```

Feign:

PaymentClient:

```java
package com.example.ordermanage.feign;
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;

import com.example.ordermanage.DTO.PaymentRequest;
import com.example.ordermanage.DTO.PaymentResponse;

@FeignClient(name = "payment-service")
public interface PaymentClient {
   @PostMapping("/payments")
   PaymentResponse processPayment(@RequestBody PaymentRequest paymentRequest);
}
```

ProductClient:

```java
package com.example.ordermanage.feign;
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestParam;

import com.example.ordermanage.DTO.ProductResponse;

@FeignClient(name = "product-service")
public interface ProductClient {
   @GetMapping("/products/{id}")
   ProductResponse getProductById(@PathVariable String id);

   @PutMapping("/products/{id}/reduceStock")
   ResponseEntity<String> reduceStock(@PathVariable String id, @RequestParam int qty);
}
```

DTo:

OrderRequest:

```
package com.example.ordermanage.DTO;

public class OrderRequest {
   private String productId;
   private int quantity;
   private String paymentMethod;

   public String getProductId() {
      return productId;
   }
```

```
    public void setProductId(String productId) {
        this.productId = productId;
    }
    public int getQuantity() {
        return quantity;
    }
    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
    public String getPaymentMethod() {
        return paymentMethod;
    }
    public void setPaymentMethod(String paymentMethod) {
        this.paymentMethod = paymentMethod;
    }
}
```

PaymentRequest:

```
package com.example.ordermanage.DTO;

public class PaymentRequest {
    private String orderId;
    private double amount;
    private String paymentMethod;

    public String getOrderId() {
        return orderId;
    }
    public void setOrderId(String orderId) {
        this.orderId = orderId;
    }
    public double getAmount() {
        return amount;
    }
    public void setAmount(double amount) {
        this.amount = amount;
    }
    public String getPaymentMethod() {
        return paymentMethod;
    }
    public void setPaymentMethod(String paymentMethod) {
        this.paymentMethod = paymentMethod;
    }
}
```

PaymentResponse:

```
package com.example.ordermanage.DTO;

public class PaymentResponse {
    private String status;

    public String getStatus() {
        return status;
    }
    public void setStatus(String status) {
        this.status = status;
    }
}
```

ProductResponse:

```java
package com.example.ordermanage.DTO;


public class ProductResponse {
    private String id;
    private String name;
    private double price;
    private int stock;


    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public double getPrice() {
        return price;
    }
    public void setPrice(double price) {
        this.price = price;
    }
    public int getStock() {
        return stock;
    }
    public void setStock(int stock) {
        this.stock = stock;
    }
}
```

PaymentProcessing:

Entity:

Payment.java:

```
package com.example.payment.processing.dto.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;

@Entity
public class Payment {
    @Id
    private String id;
    private String orderId;
    private double amount;
    private String paymentMethod;
    private String status;

    public String getId() { return id; }
    public void setId(String id) { this.id = id; }

    public String getOrderId() { return orderId; }
    public void setOrderId(String orderId) { this.orderId = orderId; }

    public double getAmount() { return amount; }
    public void setAmount(double amount) { this.amount = amount; }

    public String getPaymentMethod() { return paymentMethod; }
    public void setPaymentMethod(String paymentMethod) { this.paymentMethod =
paymentMethod; }

    public String getStatus() { return status; }
    public void setStatus(String status) { this.status = status; }
}
```

Repository:

PaymentRepository:

```
package com.example.payment.processing.dto.entity.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.example.payment.processing.dto.entity.Payment;
@Repository
public interface PaymentRepository extends JpaRepository<Payment, String> {
}
```

Controller

PaymentController:

```
package com.example.payment.processing.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import com.example.payment.processing.dto.PaymentRequest;
import com.example.payment.processing.dto.PaymentResponse;
import com.example.payment.processing.dto.entity.Payment;
import com.example.payment.processing.dto.entity.repository.PaymentRepository;

import java.util.UUID;

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

    @Autowired
    private PaymentRepository paymentRepository;

    @PostMapping
    public PaymentResponse processPayment(@RequestBody PaymentRequest request) {
        Payment payment = new Payment();
        payment.setId(UUID.randomUUID().toString());
        payment.setOrderId(request.getOrderId());
        payment.setAmount(request.getAmount());
        payment.setPaymentMethod(request.getPaymentMethod());

        String status = request.getAmount() > 0 ? "SUCCESS" : "FAILED";
        payment.setStatus(status);

        paymentRepository.save(payment);
        PaymentResponse response = new PaymentResponse();
        response.setStatus(status);
        return response;
    }
}
```

Dto:

PaymentRequest:

```
package com.example.payment.processing.dto;

public class PaymentRequest {
    private String orderId;
    private double amount;
    private String paymentMethod;

    public String getOrderId() { return orderId; }
    public void setOrderId(String orderId) { this.orderId = orderId; }

    public double getAmount() { return amount; }
    public void setAmount(double amount) { this.amount = amount; }

    public String getPaymentMethod() { return paymentMethod; }
    public void setPaymentMethod(String paymentMethod) { this.paymentMethod =
paymentMethod; }
}
```

PaymentResponse:

```
package com.example.payment.processing.dto;


public class PaymentResponse {
    private String status;


    public String getStatus() { return status; }
    public void setStatus(String status) { this.status = status; }
}
```

Applicationproperties:

```
spring.application.name=payment-service
server.port=8083
eureka.client.service-url.defaultZone=http://localhost:8761/eureka
eureka.client.register-with-eureka=true
eureka.client.fetch-registry=true
management.endpoints.web.exposure.include=*
management.endpoint.health.show-details=always
info.app.name=Payment Service
info.app.version=1.0.0
```

pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<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
https://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>
        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>3.5.4</version>
                <relativePath/> <!-- lookup parent from repository -->
        </parent>
        <groupId>com.example</groupId>
        <artifactId>paymentservice</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <name>paymentservice</name>
        <description>Payment Service for E-commerce</description>
        <url/>
        <licenses>
                <license/>
        </licenses>
        <developers>
                <developer/>
        </developers>
        <scm>
                <connection/>
                <developerConnection/>
                <tag/>
                <url/>
        </scm>
        <properties>
                <java.version>17</java.version>
                <spring-cloud.version>2025.0.0</spring-cloud.version>
        </properties>
        <dependencies>
                <dependency>
```

```
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-actuator</artifactId>
	</dependency>
	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-starter-data-jpa</artifactId>
	</dependency>
	<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>

	<dependency>
		<groupId>org.springframework.boot</groupId>
		<artifactId>spring-boot-devtools</artifactId>
		<scope>runtime</scope>
		<optional>true</optional>
	</dependency>
	<dependency>
		<groupId>com.h2database</groupId>
		<artifactId>h2</artifactId>
		<scope>runtime</scope>
	</dependency>
	<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>

                <plugin>

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

                    <artifactId>spring-boot-maven-plugin</artifactId>

                </plugin>

            </plugins>

        </build>


    </project>
```