Advann Microservices - Order Service (Complete Guide + Full Code)

This PDF contains the complete implementation details of the Order-Service created for the Advann E-commerce Microservices project. It includes configuration, entities, DTOs, Feign clients, repositories, service layer logic, controller APIs, exception handling, and stock reduction integration with product-service.

1. Overview

Order-Service is responsible for converting a user's cart into an order. It communicates with cart-service to fetch cart details and clear cart items, and with product-service to validate stock and reduce inventory.

2. Architecture / Flow

Flow of Place Order:
1) Fetch cart using cart-service (Feign)
2) Validate cart is not empty
3) Create Order record
4) For each cart item: validate stock using product-service
5) Reduce stock using product-service reduce-stock API
6) Save OrderItem records
7) Clear cart using cart-service
8) Return order details

3. application.yml

```
server:
port: 8084

spring:
application:
name: order-service

datasource:
url: jdbc:postgresql://localhost:5432/advann_order_db
username: postgres
password: root

jpa:
hibernate:
ddl-auto: update
show-sql: true
open-in-view: false

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

management:
endpoints:
web:
exposure:
include: health,info

endpoint:
health:
show-details: always

info:
env:
enabled: true
```

```
info:
app:
name: Order Service
description: Order Microservice for Advann
version: 1.0.0

logging:
level:
com.advann.order_service: INFO

springdoc:
api-docs:
path: /v3/api-docs

swagger-ui:
path: /swagger-ui.html
config-url: /order-service/v3/api-docs/swagger-config
url: /order-service/v3/api-docs
```

4. Main Class

```
package com.advann.order_service;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.openfeign.EnableFeignClients;

@SpringBootApplication
@EnableFeignClients
public class OrderServiceApplication {

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

5. Enums

OrderStatus:
PLACED, CONFIRMED, SHIPPED, DELIVERED, CANCELLED

PaymentStatus:
PENDING, PAID, FAILED

6. Entities

6.1 Order Entity

```
package com.advann.order_service.entity;

import com.advann.order_service.enums.OrderStatus;
import com.advann.order_service.enums.PaymentStatus;
import jakarta.persistence.*;
import lombok.*;

import java.math.BigDecimal;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

@Entity
@Table(name = "orders")
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class Order {
```

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;

@Column(nullable = false)
private Long userId;

@Enumerated(EnumType.STRING)
@Column(nullable = false)
private OrderStatus orderStatus;

@Enumerated(EnumType.STRING)
@Column(nullable = false)
private PaymentStatus paymentStatus;

@Column(nullable = false)
private BigDecimal totalAmount;

@OneToMany(mappedBy = "order", cascade = CascadeType.ALL, orphanRemoval = true)
private List orderItems = new ArrayList<>();

@Column(nullable = false)
private LocalDateTime createdAt;

@Column(nullable = false)
private LocalDateTime updatedAt;

@PrePersist
public void onCreate() {
this.createdAt = LocalDateTime.now();
this.updatedAt = LocalDateTime.now();
}

@PreUpdate
public void onUpdate() {
this.updatedAt = LocalDateTime.now();
}
}
```

6.2 OrderItem Entity

```
package com.advann.order_service.entity;

import jakarta.persistence.*;
import lombok.*;

import java.math.BigDecimal;
import java.time.LocalDateTime;

@Entity
@Table(name = "order_items")
@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class OrderItem {

@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;

@ManyToOne(fetch = FetchType.LAZY)
@JoinColumn(name = "order_id", nullable = false)
private Order order;

@Column(nullable = false)
private Long productId;

@Column(nullable = false)
private Integer quantity;

@Column(nullable = false)
private BigDecimal price;

@Column(nullable = false)
private BigDecimal totalPrice;

@Column(nullable = false)
private LocalDateTime createdAt;

@Column(nullable = false)
private LocalDateTime updatedAt;

@PrePersist
public void onCreate() {
this.createdAt = LocalDateTime.now();
this.updatedAt = LocalDateTime.now();

if (this.price != null && this.quantity != null) {
this.totalPrice = this.price.multiply(BigDecimal.valueOf(this.quantity));
}
}

@PreUpdate
public void onUpdate() {
this.updatedAt = LocalDateTime.now();

if (this.price != null && this.quantity != null) {
this.totalPrice = this.price.multiply(BigDecimal.valueOf(this.quantity));
}
}
}
```

7. DTOs (Required for Feign + Response)

7.1 CartItemResponseDto

```
package com.advann.order_service.dto;

import lombok.*;
import java.math.BigDecimal;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class CartItemResponseDto {

private Long id;
private Long productId;
private String productName;
private String productImage;
private BigDecimal price;
private Integer quantity;
private BigDecimal totalPrice;
}
```

7.2 CartResponseDto

```
package com.advann.order_service.dto;

import lombok.*;
import java.math.BigDecimal;
import java.util.List;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class CartResponseDto {

private Long cartId;
private Long userId;
private List items;
private BigDecimal grandTotal;
private Integer totalItems;
}
```

7.3 ProductResponseDto

```
package com.advann.order_service.dto;

import lombok.*;
import java.math.BigDecimal;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class ProductResponseDto {

private Long id;
private String name;
private BigDecimal price;
private Integer quantity;
private String imageUrl;
}
```

7.4 OrderItemResponseDto

```
package com.advann.order_service.dto;

import lombok.*;
import java.math.BigDecimal;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
```

```
public class OrderItemResponseDto {

private Long productId;
private String productName;
private Integer quantity;
private BigDecimal price;
private BigDecimal totalPrice;
}
```

7.5 OrderResponseDto

```
package com.advann.order_service.dto;

import com.advann.order_service.enums.OrderStatus;
import com.advann.order_service.enums.PaymentStatus;
import lombok.*;

import java.math.BigDecimal;
import java.time.LocalDateTime;
import java.util.List;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class OrderResponseDto {

private Long orderId;
private Long userId;
private OrderStatus orderStatus;
private PaymentStatus paymentStatus;
private BigDecimal totalAmount;
private List items;
private LocalDateTime createdAt;
}
```

8. ApiResponse Wrapper

```
package com.advann.order_service.payload;

import lombok.*;

@Data
@NoArgsConstructor
@AllArgsConstructor
@Builder
public class ApiResponse {

private boolean success;
private String message;
private T data;
}
```

9. Repositories

9.1 OrderRepository

```
package com.advann.order_service.repository;

import com.advann.order_service.entity.Order;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface OrderRepository extends JpaRepository {

List findByUserId(Long userId);
}
```

9.2 OrderItemRepository

```
package com.advann.order_service.repository;

import com.advann.order_service.entity.OrderItem;
import org.springframework.data.jpa.repository.JpaRepository;

import java.util.List;

public interface OrderItemRepository extends JpaRepository {

List findByOrderId(Long orderId);
}
```

10. Feign Clients

10.1 CartClient

```
package com.advann.order_service.client;

import com.advann.order_service.dto.CartResponseDto;
import com.advann.order_service.payload.ApiResponse;
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.*;

@FeignClient(name = "cart-service")
public interface CartClient {

@GetMapping("/api/cart/{userId}")
ApiResponse getCartByUserId(@PathVariable("userId") Long userId);

@DeleteMapping("/api/cart/clear/{userId}")
ApiResponse clearCart(@PathVariable("userId") Long userId);
}
```

10.2 ProductClient

```
package com.advann.order_service.client;
```

```
import com.advann.order_service.dto.ProductResponseDto;
import com.advann.order_service.payload.ApiResponse;
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.*;

@FeignClient(name = "product-service")
public interface ProductClient {

@GetMapping("/api/products/{id}")
ApiResponse getProductById(@PathVariable("id") Long id);

@PutMapping("/api/products/reduce-stock/{id}/{quantity}")
ApiResponse reduceStock(
@PathVariable("id") Long productId,
@PathVariable("quantity") Integer quantity
);
}
```

11. ModelMapper Config

```
package com.advann.order_service.config;

import org.modelmapper.ModelMapper;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class ModelMapperConfig {

@Bean
public ModelMapper modelMapper() {
return new ModelMapper();
}
}
```

12. Exception Handling

12.1 ResourceNotFoundException

```
package com.advann.order_service.exception;

public class ResourceNotFoundException extends RuntimeException {

public ResourceNotFoundException(String message) {
super(message);
}
}
```

12.2 GlobalExceptionHandler

```
package com.advann.order_service.exception;

import com.advann.order_service.payload.ApiResponse;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestControllerAdvice;

import java.util.HashMap;
import java.util.Map;

@RestControllerAdvice
public class GlobalExceptionHandler {

@ExceptionHandler(ResourceNotFoundException.class)
public ResponseEntity> handleResourceNotFound(ResourceNotFoundException ex) {

ApiResponse response = ApiResponse.builder()
.success(false)
.message(ex.getMessage())
.data(null)
.build();

return new ResponseEntity<>(response, HttpStatus.NOT_FOUND);
}

@ExceptionHandler(MethodArgumentNotValidException.class)
public ResponseEntity>> handleValidationErrors(MethodArgumentNotValidException ex) {

Map errors = new HashMap<>();

ex.getBindingResult().getFieldErrors().forEach(error -> {
errors.put(error.getField(), error.getDefaultMessage());
});

ApiResponse> response = ApiResponse.>builder()
.success(false)
.message("Validation Failed")
.data(errors)
.build();

return new ResponseEntity<>(response, HttpStatus.BAD_REQUEST);
}
@ExceptionHandler(Exception.class)
public ResponseEntity> handleGenericException(Exception ex) {

ApiResponse response = ApiResponse.builder()
.success(false)
.message("Something went wrong: " + ex.getMessage())
.data(null)
.build();

return new ResponseEntity<>(response, HttpStatus.INTERNAL_SERVER_ERROR);
}
}
```

13. Service Layer

13.1 OrderService Interface

```
package com.advann.order_service.service.services;

import com.advann.order_service.dto.OrderResponseDto;

import java.util.List;

public interface OrderService {

OrderResponseDto placeOrder(Long userId);

OrderResponseDto getOrderById(Long orderId);

List getOrdersByUserId(Long userId);

OrderResponseDto cancelOrder(Long orderId);
}
```

13.2 OrderServiceImpl

```
package com.advann.order_service.service.serviceImpl;

import com.advann.order_service.client.CartClient;
import com.advann.order_service.client.ProductClient;
import com.advann.order_service.dto.*;
import com.advann.order_service.entity.Order;
import com.advann.order_service.entity.OrderItem;
import com.advann.order_service.enums.OrderStatus;
import com.advann.order_service.enums.PaymentStatus;
import com.advann.order_service.exception.ResourceNotFoundException;
import com.advann.order_service.payload.ApiResponse;
import com.advann.order_service.repository.OrderItemRepository;
import com.advann.order_service.repository.OrderRepository;
import com.advann.order_service.service.services.OrderService;
import lombok.RequiredArgsConstructor;
import org.modelmapper.ModelMapper;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
@RequiredArgsConstructor
public class OrderServiceImpl implements OrderService {

private final OrderRepository orderRepository;
private final OrderItemRepository orderItemRepository;
private final CartClient cartClient;
private final ProductClient productClient;
private final ModelMapper modelMapper;

@Override
public OrderResponseDto placeOrder(Long userId) {

ApiResponse cartResponse = cartClient.getCartByUserId(userId);

if (cartResponse == null || cartResponse.getData() == null || cartResponse.getData().getItems().isEmpty()) {
throw new ResourceNotFoundException("Cart is empty for userId: " + userId);
}

CartResponseDto cart = cartResponse.getData();

Order order = Order.builder()
.userId(userId)
.orderStatus(OrderStatus.PLACED)
.paymentStatus(PaymentStatus.PENDING)
.totalAmount(cart.getGrandTotal())
.build();

order = orderRepository.save(order);

for (CartItemResponseDto cartItem : cart.getItems()) {

ApiResponse productResponse =
productClient.getProductById(cartItem.getProductId());
if (productResponse == null || productResponse.getData() == null) {
throw new ResourceNotFoundException("Product not found with id: " + cartItem.getProductId());
}

ProductResponseDto product = productResponse.getData();

if (product.getQuantity() < cartItem.getQuantity()) {
throw new RuntimeException("Insufficient stock for product: " + product.getName());
}

productClient.reduceStock(cartItem.getProductId(), cartItem.getQuantity());

OrderItem orderItem = OrderItem.builder()
.order(order)
.productId(cartItem.getProductId())
.quantity(cartItem.getQuantity())
.price(cartItem.getPrice())
.build();

orderItemRepository.save(orderItem);
}

cartClient.clearCart(userId);

return getOrderById(order.getId());
}

@Override
public OrderResponseDto getOrderById(Long orderId) {

Order order = orderRepository.findById(orderId)
.orElseThrow(() -> new ResourceNotFoundException("Order not found with id: " + orderId));

List orderItems = orderItemRepository.findByOrderId(orderId);

List responseItems = orderItems.stream()
.map(item -> {

ApiResponse productResponse =
productClient.getProductById(item.getProductId());

ProductResponseDto product = productResponse.getData();

return OrderItemResponseDto.builder()
.productId(item.getProductId())
.productName(product != null ? product.getName() : null)
.quantity(item.getQuantity())
.price(item.getPrice())
.totalPrice(item.getTotalPrice())
.build();
})
.toList();

return OrderResponseDto.builder()
.orderId(order.getId())
.userId(order.getUserId())
.orderStatus(order.getOrderStatus())
.paymentStatus(order.getPaymentStatus())
.totalAmount(order.getTotalAmount())
.items(responseItems)
.createdAt(order.getCreatedAt())
.build();
}

@Override
public List getOrdersByUserId(Long userId) {

List orders = orderRepository.findByUserId(userId);

return orders.stream()
.map(order -> getOrderById(order.getId()))
.toList();
}

@Override
public OrderResponseDto cancelOrder(Long orderId) {

Order order = orderRepository.findById(orderId)
```

```
.orElseThrow(() -> new ResourceNotFoundException("Order not found with id: " + orderId));

order.setOrderStatus(OrderStatus.CANCELLED);

orderRepository.save(order);

return getOrderById(orderId);
}
}
```

14. Controller

```
package com.advann.order_service.controller;

import com.advann.order_service.dto.OrderResponseDto;
import com.advann.order_service.payload.ApiResponse;
import com.advann.order_service.service.services.OrderService;
import lombok.RequiredArgsConstructor;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/api/orders")
@RequiredArgsConstructor
@CrossOrigin(origins = "*")
public class OrderController {

private final OrderService orderService;

@PostMapping("/place/{userId}")
public ResponseEntity> placeOrder(@PathVariable Long userId) {

OrderResponseDto order = orderService.placeOrder(userId);

return ResponseEntity.ok(
ApiResponse.builder()
.success(true)
.message("Order placed successfully")
.data(order)
.build()
);
}

@GetMapping("/{orderId}")
public ResponseEntity> getOrderById(@PathVariable Long orderId) {

OrderResponseDto order = orderService.getOrderById(orderId);

return ResponseEntity.ok(
ApiResponse.builder()
.success(true)
.message("Order fetched successfully")
.data(order)
.build()
);
}

@GetMapping("/user/{userId}")
public ResponseEntity>> getOrdersByUserId(@PathVariable Long userId) {

List orders = orderService.getOrdersByUserId(userId);

return ResponseEntity.ok(
ApiResponse.>builder()
.success(true)
.message("Orders fetched successfully")
.data(orders)
.build()
);
}

@PutMapping("/cancel/{orderId}")
public ResponseEntity> cancelOrder(@PathVariable Long orderId) {

OrderResponseDto order = orderService.cancelOrder(orderId);

return ResponseEntity.ok(
ApiResponse.builder()
.success(true)
.message("Order cancelled successfully")
.data(order)
.build()
);
}
}
```

15. API Summary

Method	Endpoint	Description
POST	/api/orders/place/{userId}	Place order from cart
GET	/api/orders/{orderId}	Fetch order details
GET	/api/orders/user/{userId}	Fetch all user orders
PUT	/api/orders/cancel/{orderId}	Cancel order

16. Postman Testing Steps

1) Add to Cart:
POST http://localhost:8082/api/cart/add

2) Get Cart:
GET http://localhost:8082/api/cart/{userId}

3) Place Order:
POST http://localhost:8084/api/orders/place/{userId}

4) Get Order:
GET http://localhost:8084/api/orders/{orderId}

5) Get Orders by User:
GET http://localhost:8084/api/orders/user/{userId}

6) Cancel Order:
PUT http://localhost:8084/api/orders/cancel/{orderId}