Spring Boot-Microservices-Case Studies

1. Product Service

# Product.java

*@Entity*

public class Product {

*@Id*

private String id;

private String name;

private double price;

private int stock;

}

#Productrepository.java

public interface Productrepository extends JpaRepository<Product,String>{

}

#ProcuctController.java

*@RestController*

*@RequestMapping*("/products")

public class ProductController {

*@Autowired*

private Productrepository repo;

*@PostMapping*

public Product add(*@RequestBody* Product p) {

return repo.save(p);

}

*@GetMapping*

public List<Product> getAllProducts() {

return repo.findAll();

}

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

public Product get(*@PathVariable* String id) {

return repo.findById(id).orElseThrow();

}

*@PutMapping*("/{id}/reduceStock")

public Product reduceStock(*@PathVariable* String id, *@RequestParam* int qty) {

Product p = repo.findById(id).orElse(null);

if (p != null && p.getStock() >= qty) {

p.setStock(p.getStock() - qty);

repo.save(p);

}

return p;

}

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

public Product updateProduct(*@PathVariable* String id, *@RequestBody* Product product) {

return repo.save(product);

}

}

#ProductService.java

*@Configuration*

public class swaggerConfig {

*@Bean*

public SwaggerTransportClientFactories SwaggerTransportClientFactories() {

return new SwaggerTransportClientFactories();

}

}

#ProductRepository

#ProductServiceApplication

*@SpringBootApplication*

public class ProductServiceApplication {

public static void main(String[] args) {

SpringApplication.*run*(ProductServiceApplication.class, args);

}

}

#Application.Properties

spring.application.name=product-service

server.port=8081

# H2 Database settings

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

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.datasource.platform=h2

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

# Eureka client configuration

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

eureka.client.register-with-eureka=true

eureka.client.fetch-registry=true

1. Order Service

# Order.java

*@Entity*

*@Table*(name = "orders")

public class Order {

*@Id*

private String id;

private String productId;

private int quantity;

private double totalAmount;

private String status;

}

#OrderController

*@RestController*

*@RequestMapping*("/orders")

public class OrderController {

*@Autowired*

private OrderRepository orderRepo;

*@Autowired*

private ProductClient productClient;

*@Autowired*

private PaymentClient paymentClient;

*@PostMapping*

public Order placeOrder(*@RequestBody* Order order) {

ProductDto product = productClient.getProduct(order.getProductId());

if (product == null || product.getStock() < order.getQuantity()) {

order.setStatus("FAILED");

return order;

}

double totalPrice = product.getPrice() \* order.getQuantity();

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

order.setTotalAmount(totalPrice);

order.setStatus("PENDING\_PAYMENT");

orderRepo.save(order);

PaymentRequest paymentRequest = new PaymentRequest();

paymentRequest.setOrderId(order.getId());

paymentRequest.setAmount(totalPrice);

paymentRequest.setPaymentMethod("Credit Card");

PaymentResponse paymentResponse = paymentClient.makePayment(paymentRequest);

if ("SUCCESS".equals(paymentResponse.getStatus())) {

order.setStatus("CONFIRMED");

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

}

orderRepo.save(order);

return order;

}

*@GetMapping*

public List<Order> getAllOrders() {

return orderRepo.findAll();

}

}

#OrderServiceApplication

*@SpringBootApplication*

*@EnableDiscoveryClient*

*@EnableFeignClients*

public class OrderServiceApplication {

public static void main(String[] args) {

SpringApplication.*run*(OrderServiceApplication.class, args);

}

}

# Swaggerconfig

*@Configuration*

public class SwaggerConfig {

*@Bean*

public SwaggerTransportClientFactories SwaggerTransportClientFactories() {

return new SwaggerTransportClientFactories();

}

}

1. Payment Service

# Payment.java

public class Payment {

private String orderId;

private double amount;

private String paymentMethod;

private String status;

}

# PaymentController

*@RestController*

*@RequestMapping*("/payments")

public class PaymentController {

*@PostMapping*

public Payment processPayment(*@RequestBody* Payment payment) {

// Simulate payment success for simplicity

payment.setStatus("SUCCESS");

return payment;

}

}

#PatmentServiceApplication

*@SpringBootApplication*

*@EnableDiscoveryClient*

public class PaymentServiceApplication {

public static void main(String[] args) {

SpringApplication.*run*(PaymentServiceApplication.class, args);

}

}

# Application Properties

spring.application.name=payment-service

server.port=8083

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