**1. Microservices using Spring Boot 3 exercises.pdf**

**Exercise- 1 Build a User and Order Management System**

**UserController.java**

package com.example.userservice.controller;

import org.springframework.web.bind.annotation.\*;

import java.util.\*;

@RestController

@RequestMapping("/users")

public class UserController {

private static final Map<Integer, Map<String, Object>> users = new HashMap<>();

static {

Map<String, Object> user1 = new HashMap<>();

user1.put("id", 1);

user1.put("name", "Alice");

users.put(1, user1);

Map<String, Object> user2 = new HashMap<>();

user2.put("id", 2);

user2.put("name", "Bob");

users.put(2, user2);

}

@GetMapping("/{id}")

public Map<String, Object> getUserById(@PathVariable int id) {

return users.getOrDefault(id, Map.of("error", "User not found"));

}

@GetMapping

public Collection<Map<String, Object>> getAllUsers() {

return users.values();

}

}

**OrderController.java**

package com.example.orderservice.controller;

import org.springframework.web.bind.annotation.\*;

import org.springframework.web.reactive.function.client.WebClient;

import java.util.\*;

@RestController

@RequestMapping("/orders")

public class OrderController {

private static final List<Map<String, Object>> orders = new ArrayList<>();

private final WebClient webClient = WebClient.create("http://localhost:8081");

static {

Map<String, Object> order1 = new HashMap<>();

order1.put("id", 1);

order1.put("userId", 1);

order1.put("item", "Laptop");

orders.add(order1);

Map<String, Object> order2 = new HashMap<>();

order2.put("id", 2);

order2.put("userId", 2);

order2.put("item", "Phone");

orders.add(order2);

}

@GetMapping

public List<Map<String, Object>> getOrdersWithUserDetails() {

List<Map<String, Object>> detailedOrders = new ArrayList<>();

for (Map<String, Object> order : orders) {

int userId = (int) order.get("userId");

Map user = webClient.get()

.uri("/users/{id}", userId)

.retrieve()

.bodyToMono(Map.class)

.block();

order.put("user", user);

detailedOrders.add(order);

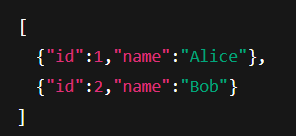
}

return detailedOrders;

}

}

**Output:**

****

****

**Exercise 2: Inventory Management System with Service Discovery**

**ConfigServerApplication.java**

package com.example.configserver;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.config.server.EnableConfigServer;

@SpringBootApplication

@EnableConfigServer

public class ConfigServerApplication {

public static void main(String[] args) {

SpringApplication.run(ConfigServerApplication.class, args);

}

}

**EurekaServerApplication.java**

package com.example.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);

}

}

**ProductController.java**

package com.example.productservice.controller;

import org.springframework.web.bind.annotation.\*;

import java.util.\*;

@RestController

@RequestMapping("/products")

public class ProductController {

static final Map<String, Map<String, Object>> products = new HashMap<>();

static {

products.put("P101", Map.of("id", "P101", "name", "Pen", "price", 10));

products.put("P102", Map.of("id", "P102", "name", "Notebook", "price", 50));

}

@GetMapping

public Collection<Map<String, Object>> getProducts() {

return products.values();

}

@GetMapping("/{id}")

public Map<String, Object> getProduct(@PathVariable String id) {

return products.getOrDefault(id, Map.of("error", "Product not found"));

}

}

**InventoryController.java**

package com.example.inventoryservice.controller;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.web.bind.annotation.\*;

import org.springframework.web.reactive.function.client.WebClient;

import java.util.\*;

@RestController

@RequestMapping("/inventory")

public class InventoryController {

@Autowired

private WebClient.Builder webClientBuilder;

static final Map<String, Integer> stockMap = new HashMap<>();

static {

stockMap.put("P101", 100);

stockMap.put("P102", 50);

}

@GetMapping("/{productId}")

public Map<String, Object> getStockDetails(@PathVariable String productId) {

Map<String, Object> product = webClientBuilder.build()

.get()

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

.retrieve()

.bodyToMono(Map.class)

.block();

int stock = stockMap.getOrDefault(productId, 0);

product.put("stock", stock);

return product;

}

}

**InventoryServiceApplication.java**

@SpringBootApplication

@EnableDiscoveryClient

public class InventoryServiceApplication {

public static void main(String[] args) {

SpringApplication.run(InventoryServiceApplication.class, args);

}

@Bean

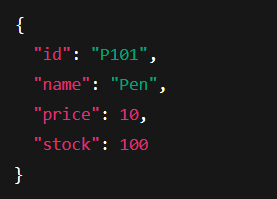
public WebClient.Builder webClientBuilder() {

return WebClient.builder();

}

}

**Output:**

****

**Exercise-3: Implement an API Gateway**

**CustomerController.java**

package com.example.customerservice.controller;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/customers")

public class CustomerController {

@GetMapping("/{id}")

public String getCustomer(@PathVariable String id) {

return "Customer info for ID: " + id;

}

}

**BillingController.java**

package com.example.billingservice.controller;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/billing")

public class BillingController {

@GetMapping("/{id}")

public String getBilling(@PathVariable String id) {

return "Billing info for ID: " + id;

}

}

**ApiGatewayApplication.java**

package com.example.apigateway;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class ApiGatewayApplication {

public static void main(String[] args) {

SpringApplication.run(ApiGatewayApplication.class, args);

}

}

**Exercise 4: Resilient Microservices with Circuit Breaker**

**SlowController.java**

package com.example.mockthirdparty.controller;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/thirdparty")

public class SlowController {

@GetMapping("/process")

public String slowApi() throws InterruptedException {

Thread.sleep(4000); // Simulate delay

return "Third-party payment processed";

}

}

**PaymentService.java**

package com.example.paymentservice.service;

import io.github.resilience4j.circuitbreaker.annotation.CircuitBreaker;

import org.springframework.stereotype.Service;

import org.springframework.web.reactive.function.client.WebClient;

@Service

public class PaymentService {

private final WebClient webClient;

public PaymentService(WebClient.Builder builder) {

this.webClient = builder.baseUrl("http://localhost:8083").build();

}

@CircuitBreaker(name = "thirdPartyCB", fallbackMethod = "fallbackPayment")

public String makePayment() {

return webClient.get()

.uri("/thirdparty/process")

.retrieve()

.bodyToMono(String.class)

.block();

}

public String fallbackPayment(Throwable t) {

System.out.println("Fallback triggered due to: " + t.getMessage());

return "Fallback: Payment system is currently unavailable. Try again later.";

}

}

**PaymentController.java**

package com.example.paymentservice.controller;

import com.example.paymentservice.service.PaymentService;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/payment")

public class PaymentController {

private final PaymentService paymentService;

public PaymentController(PaymentService paymentService) {

this.paymentService = paymentService;

}

@GetMapping("/make")

public String makePayment() {

return paymentService.makePayment();

}

}

**PaymentServiceApplication.java**

@SpringBootApplication

public class PaymentServiceApplication {

public static void main(String[] args) {

SpringApplication.run(PaymentServiceApplication.class, args);

}

@Bean

public WebClient.Builder webClientBuilder() {

return WebClient.builder();

}

}

**Output:**

****

****

**2. Microservices with API Gateway.pdf**

**GreetController.java**

package com.example.greetservice.controller;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/greet")

public class GreetController {

@GetMapping

public String greet() {

return "Hello, world!";

}

}

**application.properties**

server.port=8081

spring.application.name=greet-service

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

**API Gateway**

**application.properties**

server.port=9090

spring.application.name=api-gateway

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

spring.cloud.gateway.discovery.locator.enabled=true

spring.cloud.gateway.discovery.locator.lower-case-service-id=true

spring.cloud.gateway.routes[0].id=greet-service

spring.cloud.gateway.routes[0].uri=lb://greet-service

spring.cloud.gateway.routes[0].predicates[0]=Path=/greet-service/\*\*

spring.cloud.gateway.routes[0].filters[0]=RewritePath=/greet-service/(?<segment>.\*), /greet/${segment}

**ApiGatewayApplication.java**

@SpringBootApplication

public class ApiGatewayApplication {

public static void main(String[] args) {

SpringApplication.run(ApiGatewayApplication.class, args);

}

}

**LogFilter.java**

package com.example.apigateway.filter;

import org.springframework.cloud.gateway.filter.GlobalFilter;

import org.springframework.cloud.gateway.filter.GatewayFilterChain;

import org.springframework.core.Ordered;

import org.springframework.stereotype.Component;

import org.springframework.web.server.ServerWebExchange;

import reactor.core.publisher.Mono;

@Component

public class LogFilter implements GlobalFilter, Ordered {

@Override

public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {

System.out.println("Logging request path: " + exchange.getRequest().getPath());

return chain.filter(exchange);

}

@Override

public int getOrder() {

return -1;

}

}

**Output:**

****