**Exercises on Microservices with Spring Boot 3.0**

**1.Build a User and Order Management System Problem**

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0" ...>

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>user-service</artifactId>

<version>1.0</version>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

</dependency>

</dependencies>

</project>

**application.properties**

server.port=8081

spring.datasource.url=jdbc:mysql://localhost:3306/userdb

spring.datasource.username=root

spring.datasource.password=yourpassword

spring.jpa.hibernate.ddl-auto=update

**User.java**

package com.example.userservice.entity;

import jakarta.persistence.\*;

@Entity

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

// Getters and Setters

}

**UserRepository.java**

package com.example.userservice.repository;

import com.example.userservice.entity.User;

import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {}

**UserService.java**

package com.example.userservice.service;

import com.example.userservice.entity.User;

import com.example.userservice.repository.UserRepository;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class UserService {

private final UserRepository repo;

public UserService(UserRepository repo) {

this.repo = repo;

}

public List<User> getAllUsers() {

return repo.findAll();

}

public User getUserById(Long id) {

return repo.findById(id).orElse(null);

}

public User saveUser(User user) {

return repo.save(user);

}

}

**UserController.java**

package com.example.userservice.controller;

import com.example.userservice.entity.User;

import com.example.userservice.service.UserService;

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

import java.util.List;

@RestController

@RequestMapping("/users")

public class UserController {

private final UserService service;

public UserController(UserService service) {

this.service = service;

}

@GetMapping

public List<User> getAll() {

return service.getAllUsers();

}

@PostMapping

public User create(@RequestBody User user) {

return service.saveUser(user);

}

@GetMapping("/{id}")

public User getUser(@PathVariable Long id) {

return service.getUserById(id);

}

}

**OrderService (port 8082)**

<project xmlns="http://maven.apache.org/POM/4.0.0" ...>

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>order-service</artifactId>

<version>1.0</version>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

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

<artifactId>spring-cloud-starter-openfeign</artifactId>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</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>

</project>

**application.properties**

server.port=8082

spring.datasource.url=jdbc:mysql://localhost:3306/orderdb

spring.datasource.username=root

spring.datasource.password=yourpassword

spring.jpa.hibernate.ddl-auto=update

**Order.java**

package com.example.orderservice.entity;

import jakarta.persistence.\*;

@Entity

public class Order {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String product;

private Long userId; // FK to User

// Getters and Setters

}

**OrderRepository.java**

package com.example.orderservice.repository;

import com.example.orderservice.entity.Order;

import org.springframework.data.jpa.repository.JpaRepository;

public interface OrderRepository extends JpaRepository<Order, Long> {}

**UserClient.java**

package com.example.orderservice.client;

import org.springframework.cloud.openfeign.FeignClient;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

@FeignClient(name = "userClient", url = "http://localhost:8081/users")

public interface UserClient {

@GetMapping("/{id}")

Object getUserById(@PathVariable Long id);

}

**OrderService.java**

package com.example.orderservice.service;

import com.example.orderservice.client.UserClient;

import com.example.orderservice.entity.Order;

import com.example.orderservice.repository.OrderRepository;

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class OrderService {

private final OrderRepository orderRepo;

private final UserClient userClient;

public OrderService(OrderRepository orderRepo, UserClient userClient) {

this.orderRepo = orderRepo;

this.userClient = userClient;

}

public List<Order> getAll() {

return orderRepo.findAll();

}

public Object placeOrder(Order order) {

Object user = userClient.getUserById(order.getUserId());

if (user != null) {

return orderRepo.save(order);

}

throw new RuntimeException("User not found");

}

}

**OrderController.java**

package com.example.orderservice.controller;

import com.example.orderservice.entity.Order;

import com.example.orderservice.service.OrderService;

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

import java.util.List;

@RestController

@RequestMapping("/orders")

public class OrderController {

private final OrderService service;

public OrderController(OrderService service) {

this.service = service;

}

@GetMapping

public List<Order> getAll() {

return service.getAll();

}

@PostMapping

public Object placeOrder(@RequestBody Order order) {

return service.placeOrder(order);

}

}

**OrderServiceApplication.java**

package com.example.orderservice;

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) {

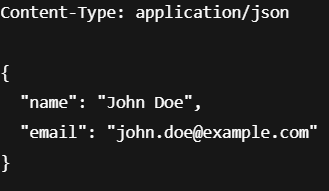
SpringApplication.run(OrderServiceApplication.class, args);

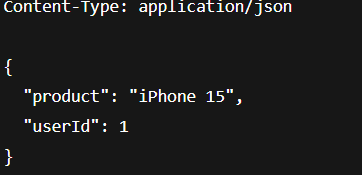
}

}

Output:

**UserServiceApplication.java**



****

**2. Inventory Management System with Service Discovery**

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0" ...>

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>config-server</artifactId>

<version>1.0.0</version>

<dependencies>

<dependency>

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

<artifactId>spring-cloud-config-server</artifactId>

</dependency>

<dependency>

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

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

</project>

**application.properties**

server.port=8888

spring.cloud.config.server.git.uri=https://github.com/YOUR\_USERNAME/inventory-config

**ConfigServerApplication.java**

@SpringBootApplication

@EnableConfigServer

public class ConfigServerApplication {

public static void main(String[] args) {

SpringApplication.run(ConfigServerApplication.class, args);

}

}

Eureka Server(Port: 8761)

**pom.xml**

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

</dependencies>

**application.properties**

server.port=8761

eureka.client.register-with-eureka=false

eureka.client.fetch-registry=false

**EurekaServerApplication.java**

@SpringBootApplication

@EnableEurekaServer

public class EurekaServerApplication {

public static void main(String[] args) {

SpringApplication.run(EurekaServerApplication.class, args);

}

}

**ProductService**

**pom.xml**

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

</dependency>

<dependency>

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

<artifactId>spring-cloud-starter-config</artifactId>

</dependency>

<dependency>

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

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

</dependency>

</dependencies>

**ProductServiceApplication.java**

@SpringBootApplication

@EnableDiscoveryClient

public class ProductServiceApplication {

public static void main(String[] args) {

SpringApplication.run(ProductServiceApplication.class, args);

}

}

**Product.java**

@Entity

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String description;

}

**ProductRepository.java**

@RestController

@RequestMapping("/products")

public class ProductController {

private final ProductRepository repo;

public ProductController(ProductRepository repo) {

this.repo = repo;

}

@PostMapping

public Product save(@RequestBody Product p) {

return repo.save(p);

}

@GetMapping

public List<Product> all() {

return repo.findAll();

}

}

**InventoryService**

**InventoryServiceApplication.java**

@SpringBootApplication

@EnableDiscoveryClient

public class InventoryServiceApplication {

public static void main(String[] args) {

SpringApplication.run(InventoryServiceApplication.class, args);

}

}

**Inventory.java**

@Entity

public class Inventory {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private Long productId;

private Integer quantity;

}

**InventoryRepository.java**

public interface InventoryRepository extends JpaRepository<Inventory, Long> {

Inventory findByProductId(Long productId);

}

**InventoryController.java**

@RestController

@RequestMapping("/inventory")

public class InventoryController {

private final InventoryRepository repo;

public InventoryController(InventoryRepository repo) {

this.repo = repo;

}

@PostMapping

public Inventory add(@RequestBody Inventory i) {

return repo.save(i);

}

@GetMapping("/{productId}")

public Inventory get(@PathVariable Long productId) {

return repo.findByProductId(productId);

}

}

**3. Implement an API Gateway**

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0" ...>

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>api-gateway</artifactId>

<version>1.0.0</version>

<dependencies>

<dependency>

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

<artifactId>spring-cloud-starter-gateway</artifactId>

</dependency>

<dependency>

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

<artifactId>spring-boot-starter-actuator</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-starter-cache</artifactId>

</dependency>

<dependency>

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

<artifactId>spring-boot-starter-data-redis</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>

</project>

**application.yml**

server:

port: 8080

spring:

application:

name: api-gateway

cloud:

gateway:

routes:

- id: customer-service

uri: http://localhost:8081

predicates:

- Path=/customers/\*\*

filters:

- RewritePath=/customers/(?<segment>.\*), /$\{segment}

- name: RequestRateLimiter

args:

redis-rate-limiter.replenishRate: 5

redis-rate-limiter.burstCapacity: 10

- id: billing-service

uri: http://localhost:8082

predicates:

- Path=/billing/\*\*

filters:

- RewritePath=/billing/(?<segment>.\*), /$\{segment}

redis:

host: localhost

port: 6379

eureka:

client:

service-url:

defaultZone: <http://localhost:8761/eureka>

**ApiGatewayApplication.java**

@SpringBootApplication

@EnableDiscoveryClient

public class ApiGatewayApplication {

public static void main(String[] args) {

SpringApplication.run(ApiGatewayApplication.class, args);

}

}

**CustomerService**

**CustomerServiceApplication.java**

@SpringBootApplication

@RestController

public class CustomerServiceApplication {

public static void main(String[] args) {

SpringApplication.run(CustomerServiceApplication.class, args);

}

@GetMapping("/hello")

public String hello() {

return "Hello from Customer Service!";

}

}

**BillingServiceApplication.java**

@SpringBootApplication

@RestController

public class BillingServiceApplication {

public static void main(String[] args) {

SpringApplication.run(BillingServiceApplication.class, args);

}

@GetMapping("/pay")

public String pay() {

return "Billing successful from Billing Service!";

}

}

**4. Resilient Microservices with Circuit Breaker**

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>payment-service</artifactId>

<version>1.0.0</version>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>io.github.resilience4j</groupId>

<artifactId>resilience4j-spring-boot3</artifactId>

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

<groupId>io.github.resilience4j</groupId>

<artifactId>resilience4j-bom</artifactId>

<version>2.1.0</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

</project>

**application.yml**

server:

port: 8083

spring:

application:

name: payment-service

resilience4j:

circuitbreaker:

instances:

paymentAPI:

registerHealthIndicator: true

slidingWindowSize: 5

failureRateThreshold: 50

waitDurationInOpenState: 10s

permittedNumberOfCallsInHalfOpenState: 2

minimumNumberOfCalls: 3

management:

endpoints:

web:

exposure:

include: "\*"

**PaymentServiceApplication.java**

@SpringBootApplication

public class PaymentServiceApplication {

public static void main(String[] args) {

SpringApplication.run(PaymentServiceApplication.class, args);

}

}

**PaymentController.java**

package com.example.paymentservice.controller;

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

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

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

import reactor.core.publisher.Mono;

@RestController

@RequestMapping("/payment")

public class PaymentController {

private static final Logger log = LoggerFactory.getLogger(PaymentController.class);

private final WebClient webClient;

public PaymentController(WebClient.Builder builder) {

this.webClient = builder.baseUrl("https://httpstat.us/200?sleep=5000").build(); // Simulates a slow API

}

@GetMapping("/process")

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

public Mono<String> makePayment() {

log.info("Calling external payment API...");

return webClient

.get()

.retrieve()

.bodyToMono(String.class)

.doOnNext(res -> log.info("Response received: {}", res));

}

public Mono<String> fallbackPayment(Throwable t) {

log.error("Fallback triggered due to: {}", t.getMessage());

return Mono.just("Payment service is currently unavailable. Please try again later.");

}

}