MICROSERVICES WITH SPRING BOOT 3.0

Superset ID : 6393540

Name : Antony Praveen E

E-mail : antonypraveen.2205009@srec.ac.in

Exercises:

1) Exercise 1: Build a User and Order Management System

Solution:

UserService

//pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-openfeign</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>
```

//User.java

```
@Entity
public class User {
```

```
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private String email;
}
```

//UserRepository.java

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

//UserController.java

```
@RestController
@RequestMapping("/users")
public class UserController {
    @Autowired
    private UserRepository userRepository;
    @PostMapping
    public User saveUser(@RequestBody User user) {
        return userRepository.save(user);
    }
    @GetMapping
    public List<User> getUsers() {
        return userRepository.findAll();
    }
}
```

//application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/userdb
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.hibernate.ddl-auto=update
```

OrderService

//pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-openfeign</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>
```

//Order.java

```
@Entity
public class Order {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private Long userId;
    private String product;
    private Double price;
}
```

//OrderRepository.java

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

//UserClient.java

```
@FeignClient(name = "user-service", url = "http://localhost:8080")
public interface UserClient {
    @GetMapping("/users")
    List<User> getAllUsers();
}
```

//OrderController.java

```
@RestController
@RequestMapping("/orders")
public class OrderController {
    @Autowired
    private OrderRepository orderRepository;
    @Autowired
    private UserClient userClient;
    @PostMapping
    public Order placeOrder(@RequestBody Order order) {
        return orderRepository.save(order);
    }
    @GetMapping("/users")
    public List<User> getAllUsers() {
        return userClient.getAllUsers();
    }
}
```

//application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/orderdb
spring.datasource.username=root
spring.datasource.password=root
```

```
spring.jpa.hibernate.ddl-auto=update
```

2) Exercise 2: Inventory Management System with Service Discovery

Solution:

Eureka Server

//pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-server</artifactId>
    </dependency>
</dependencies>
```

//MainApp.java

```
@SpringBootApplication
@EnableEurekaServer
public class EurekaServerApp {
    public static void main(String[] args) {
        SpringApplication.run(EurekaServerApp.class, args);
    }
}
```

//application.properties

```
server.port=8761
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
```

Config Server

//pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-config-server</artifactId>
```

```
    </dependency>
</dependencies>
// MainApp.java
@SpringBootApplication
@EnableConfigServer
public class ConfigServerApp {
    public static void main(String[] args) {
        SpringApplication.run(ConfigServerApp.class, args);
    }
}
//application.properties
server.port=8888
spring.cloud.config.server.git.uri=https://github.com/your-repo/config-files
```

Product Service

```
//application.properties
spring.application.name=product-service
server.port=8081
eureka.client.service-url.defaultZone=http://localhost:8761/eureka
spring.config.import=optional:configserver:http://localhost:8888
//Product.java
@Entity
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String name;
    private Double price;
}
```

//ProductController.java

```
@RestController
@RequestMapping("/products")
public class ProductController {
    @Autowired
    private ProductRepository productRepository;
    @PostMapping
    public Product add(@RequestBody Product product) {
        return productRepository.save(product);
    }
    @GetMapping
    public List<Product> list() {
        return productRepository.findAll();
    }
}
```

Inventory Service

//application.properties

```
spring.application.name=inventory-service
server.port=8082
eureka.client.service-url.defaultZone=http://localhost:8761/eureka
spring.config.import=optional:configserver:http://localhost:8888
```

//Inventory.java

```
@Entity
public class Inventory {
    @Id
    private Long productId;
    private Integer stock;
}
```

//InventoryController.java

```
@RestController
```

```
@RequestMapping("/inventory")
public class InventoryController {
    @Autowired
    private InventoryRepository inventoryRepository;
    @PostMapping
    public Inventory add(@RequestBody Inventory inventory) {
        return inventoryRepository.save(inventory);
    }
    @GetMapping("/{productId}")
    public Inventory checkStock(@PathVariable Long productId) {
        return inventoryRepository.findById(productId).orElse(null);
    }
}
```

3) Exercise 3: Implement an API Gateway

Solution:

API Gateway

//pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-gateway</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.cloud</groupId>
        <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
    </dependency>
    <dependency>
        <groupId>io.github.resilience4j</groupId>
        <artifactId>resilience4j-spring-boot3</artifactId>
```

```
    </dependency>
</dependencies>
```

// application.yml

```
spring:
  application:
    name: api-gateway
  cloud:
    gateway:
      routes:
        - id: customer-service
          uri: lb://customer-service
          predicates:
            - Path=/customers/**
        - id: billing-service
          uri: lb://billing-service
          predicates:
            - Path=/billing/**
      default-filters:
        - name: RequestRateLimiter
          args:
            redis-rate-limiter.replenishRate:  2
            redis-rate-limiter.burstCapacity:  4


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

Customer and Billing Service

//CustomerController.java

```
@RestController
```

```
@RequestMapping("/customers")
public class CustomerController {
    @GetMapping
    public String getCustomers() {
        return "Customer list";
    }
}
```

//BillingController.java

```
@RestController
@RequestMapping("/billing")
public class BillingController {
    @GetMapping
    public String getBilling() {
        return "Billing info";
    }
}
```

4) Exercise 4: Resilient Microservices with Circuit Breaker

Solution:

PaymentService

//pom.xml

```
<dependency>
    <groupId>io.github.resilience4j</groupId>
    <artifactId>resilience4j-spring-boot3</artifactId>
</dependency>
```

//application.yml

```
resilience4j:
circuitbreaker:
    instances:
      paymentAPI:
```

```
        registerHealthIndicator: true
        failureRateThreshold: 50
        waitDurationInOpenState: 5s
        permittedNumberOfCallsInHalfOpenState: 3
        slidingWindowSize: 10
```

//PaymentService.java

```
@Service
public class PaymentService {
    @CircuitBreaker(name = "paymentAPI", fallbackMethod = "fallbackPayment")
    public String processPayment() {
        // simulate slow API
        try {
            Thread.sleep(5000);
        } catch (InterruptedException e) {
            throw new RuntimeException("API timeout");
        }
        return "Payment processed";
    }
    public String fallbackPayment(Throwable t) {
        return "Payment service is down. Please try again later.";
    }
}
```

//PaymentController.java

```
@RestController
@RequestMapping("/payment")
public class PaymentController {
    @Autowired
    private PaymentService paymentService;
    @GetMapping
    public String pay() {
```

```
        return paymentService.processPayment();
    }
}
```