Microservices

1. User and Order Management System

---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>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

</dependencies>

---Application.yml

spring:

datasource:

url: jdbc:mysql://localhost:3306/user\_db

username: root

password: password

jpa:

hibernate:

ddl-auto: update

----user.java

@Entity

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

// getters and setters

}

----userController.java

@RestController

@RequestMapping("/users")

public class UserController {

@Autowired

private UserRepository userRepository;

@PostMapping

public User createUser(@RequestBody User user) {

return userRepository.save(user);

}

@GetMapping("/{id}")

public User getUser(@PathVariable Long id) {

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

}

}

Order service:

----OrderController.java

@RestController

@RequestMapping("/orders")

public class OrderController {

@Autowired

private WebClient.Builder webClientBuilder;

@Autowired

private OrderRepository orderRepository;

@PostMapping

public Order createOrder(@RequestBody Order order) {

User user = webClientBuilder.build()

.get()

.uri("http://user-service/users/" + order.getUserId())

.retrieve()

.bodyToMono(User.class)

.block();

if(user == null) throw new RuntimeException("User not found");

return orderRepository.save(order);

}

}

1. Inventory Management System with Service Discovery

Eureka Server:

---pom.xml

<dependency>

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

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

</dependency>

---application.yml

server:

port: 8761

eureka:

client:

register-with-eureka: false

fetch-registry: false

----Main class

@SpringBootApplication

@EnableEurekaServer

public class EurekaServerApplication {

public static void main(String[] args) {

SpringApplication.run(EurekaServerApplication.class, args);

}

}

Product Service:

---application.yml

spring:

application:

name: product-service

eureka:

client:

service-url:

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

Inventory Service:

---application.yml

spring:

application:

name: inventory-service

eureka:

client:

service-url:

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

3 API Gateway Implementation

---pom.xml

<dependency>

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

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

</dependency>

---application.yml

spring:

cloud:

gateway:

routes:

- id: customer-service

uri: lb://customer-service

predicates:

- Path=/customers/\*\*

filters:

- name: RequestRateLimiter

args:

redis-rate-limiter.replenishRate: 10

redis-rate-limiter.burstCapacity: 20

- id: billing-service

uri: lb://billing-service

predicates:

- Path=/billing/\*\*

filters:

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

1. Resilient Microservices with Circuit Breaker

---pom.xml

<dependency>

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

<artifactId>spring-cloud-starter-circuitbreaker-resilience4j</artifactId>

</dependency>

---PaymentService.java

@Service

public class PaymentService {

@Autowired

private CircuitBreakerFactory circuitBreakerFactory;

public String processPayment(String request) {

return circuitBreakerFactory.create("paymentService")

.run(() -> callThirdPartyAPI(request),

throwable -> fallbackPayment(request));

}

private String callThirdPartyAPI(String request) {

// Simulate API call

if(Math.random() > 0.7) throw new RuntimeException("API Failed");

return "Processed: " + request;

}

private String fallbackPayment(String request) {

return "Fallback processing for: " + request;

}

}

---application.yml

resilience4j.circuitbreaker:

instances:

paymentService:

registerHealthIndicator: true

slidingWindowSize: 10

minimumNumberOfCalls: 5

permittedNumberOfCallsInHalfOpenState: 3

automaticTransitionFromOpenToHalfOpenEnabled: true

waitDurationInOpenState: 5s

failureRateThreshold: 50