Microservices

1. Account Microservice Implementation

---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-actuator</artifactId>

</dependency>

<dependency>

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

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

</dependency>

</dependencies>

---application.properties

server.port=8080

spring.application.name=account-service

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

---account.java

public class Account {

private String number;

private String type;

private double balance;

// Constructors, getters and setters

}

---AccountController.java

@RestController

@RequestMapping("/accounts")

public class AccountController {

@GetMapping("/{number}")

public Account getAccount(@PathVariable String number) {

return new Account("00987987973432", "savings", 234343);

}

}

1. Loan Microservice Implementation

---application.properties

server.port=8081

spring.application.name=loan-service

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

---LoanController.java

@RestController

@RequestMapping("/loans")

public class LoanController {

@GetMapping("/{number}")

public Loan getLoan(@PathVariable String number) {

return new Loan("H00987987972342", "car", 400000, 3258, 18);

}

}

1. **Eureka Discovery Server**

--pom.xml

<dependency>

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

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

</dependency>

**---application.properties**

server.port=8761

eureka.client.register-with-eureka=false

eureka.client.fetch-registry=false

---EurekServerApplication.java

@SpringBootApplication

@EnableEurekaServer

public class EurekaServerApplication {

public static void main(String[] args) {

SpringApplication.run(EurekaServerApplication.class, args);

}

}

1. API Gateway Implementation

---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>

</dependencies>

---application.properties

server.port=9090

spring.application.name=api-gateway

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

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

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

spring.cloud.gateway.routes[0].predicates[0]=Path=/accounts/\*\*

spring.cloud.gateway.routes[1].id=loan-service

spring.cloud.gateway.routes[1].uri=lb://loan-service

spring.cloud.gateway.routes[1].predicates[0]=Path=/loans/\*\*

----LogFilter.java

@Component

public class LogFilter implements GlobalFilter {

private static final Logger logger = LoggerFactory.getLogger(LogFilter.class);

@Override

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

logger.info("Path of the request received -> {}", exchange.getRequest().getPath());

return chain.filter(exchange);

}

}

1. Greet Service Example

---GreetController.java

@RestController

@RequestMapping("/greet")

public class GreetController {

@GetMapping

public String greet() {

return "Hello World";

}

}

1. Circuit Breaker Implementation

---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) {

if(Math.random() > 0.7) {

throw new RuntimeException("API Failed");

}

return "Processed: " + request;

}

private String fallbackPayment(String request) {

return "Fallback processing for: " + request;

}

}

----application.properties

resilience4j.circuitbreaker.instances.paymentService.registerHealthIndicator=true

resilience4j.circuitbreaker.instances.paymentService.slidingWindowSize=10

resilience4j.circuitbreaker.instances.paymentService.minimumNumberOfCalls=5

resilience4j.circuitbreaker.instances.paymentService.permittedNumberOfCallsInHalfOpenState=3

resilience4j.circuitbreaker.instances.paymentService.automaticTransitionFromOpenToHalfOpenEnabled=true

resilience4j.circuitbreaker.instances.paymentService.waitDurationInOpenState=5s

resilience4j.circuitbreaker.instances.paymentService.failureRateThreshold=50