**MANDATORY HANDS-ON**

**(includes all other hands-on of week-5)**

**WEEK-5**

Microservices with Spring Boot 3 and Spring Cloud

2.Microservices with API gateway

**Creating Microservices for account and loan**

**Implement AccountController**

package com.cognizant.account.controller;

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

import java.util.HashMap;

import java.util.Map;

@RestController

@RequestMapping("/accounts")

public class AccountController {

@GetMapping("/{number}")

public Map<String, Object> getAccountByNumber(@PathVariable String number) {

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

response.put("number", number);

response.put("type", "savings");

response.put("balance", 234343);

return response;

}

}

**Implement LoanController**

package com.cognizant.loan.controller;

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

import java.util.HashMap;

import java.util.Map;

@RestController

@RequestMapping("/loans")

public class LoanController {

@GetMapping("/{number}")

public Map<String, Object> getLoanByNumber(@PathVariable String number) {

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

response.put("number", number);

response.put("type", "car");

response.put("loan", 400000);

response.put("emi", 3258);

response.put("tenure", 18);

return response;

}

}

**Create Eureka Discovery Server and register microservices**

Create Eureka Discovery Server and register microservices

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

logging.level.com.netflix.eureka=OFF

logging.level.com.netflix.discovery=OFF

Main Class

package com.cognizant.eurekaserver;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;

@SpringBootApplication

@EnableEurekaServer

public class EurekaDiscoveryServerApplication {

public static void main(String[] args) {

SpringApplication.run(EurekaDiscoveryServerApplication.class, args);

}

}

Account Microservice

pom.xml

<dependency>

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

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

</dependency>

**application.properties**

spring.application.name=account-service

server.port=8080

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

Main Class

package com.cognizant.account;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication

@EnableDiscoveryClient

public class AccountApplication {

public static void main(String[] args) {

SpringApplication.run(AccountApplication.class, args);

}

}

Controller

package com.cognizant.account.controller;

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

import java.util.\*;

@RestController

@RequestMapping("/accounts")

public class AccountController {

@GetMapping("/{number}")

public Map<String, Object> getAccountByNumber(@PathVariable String number) {

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

account.put("number", number);

account.put("type", "savings");

account.put("balance", 234343);

return account;

}

}

Loan Microservice

application.properties

spring.application.name=loan-service

server.port=8081

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

Main Class

package com.cognizant.loan;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication

@EnableDiscoveryClient

public class LoanApplication {

public static void main(String[] args) {

SpringApplication.run(LoanApplication.class, args);

}

}

**Controller**

package com.cognizant.loan.controller;

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

import java.util.\*;

@RestController

@RequestMapping("/loans")

public class LoanController {

@GetMapping("/{number}")

public Map<String, Object> getLoanByNumber(@PathVariable String number) {

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

loan.put("number", number);

loan.put("type", "car");

loan.put("loan", 400000);

loan.put("emi", 3258);

loan.put("tenure", 18);

return loan;

}

}

Greet Microservice

application.properties

spring.application.name=greet-service

server.port=8082

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

Main Class

package com.cognizant.greet;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication

@EnableDiscoveryClient

public class GreetServiceApplication {

public static void main(String[] args) {

SpringApplication.run(GreetServiceApplication.class, args);

}

}

Controller

package com.cognizant.greet.controller;

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

@RestController

public class GreetController {

@GetMapping("/greet")

public String greet() {

return "Hello World";

}

}

API Gateway with Logging

**application.properties**

spring.application.name=api-gateway

server.port=9090

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

Main Class

package com.cognizant.gateway;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.client.discovery.EnableDiscoveryClient;

@SpringBootApplication

@EnableDiscoveryClient

public class ApiGatewayApplication {

public static void main(String[] args) {

SpringApplication.run(ApiGatewayApplication.class, args);

}

}

Log Filter Class

package com.cognizant.gateway.filter;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.core.annotation.Order;

import org.springframework.stereotype.Component;

import org.springframework.web.server.ServerWebExchange;

import reactor.core.publisher.Mono;

@Component

@Order(1)

public class LogFilter implements GlobalFilter {

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

@Override

public Mono<Void> filter(ServerWebExchange exchange, org.springframework.cloud.gateway.filter.GatewayFilterChain chain) {

logger.info("Request URI: {}", exchange.getRequest().getURI());

return chain.filter(exchange);

}

}

Exercises on Microservices with Spring Boot 3.0

1. Build a User and Order Management System

application.yml

server:

port: 8081

spring:

datasource:

url: jdbc:mysql://localhost:3306/userdb

username: root

password: password

jpa:

hibernate:

ddl-auto: update

show-sql: true

User.java

@Entity

@Data

@NoArgsConstructor

@AllArgsConstructor

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> {}

UserService.java

@Service

public class UserService {

@Autowired

private UserRepository repo;

public User saveUser(User user) {

return repo.save(user);

}

public User getUser(Long id) {

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

}

}

**UserController.java**

@RestController

@RequestMapping("/users")

public class UserController {

@Autowired

private UserService service;

@PostMapping

public ResponseEntity<User> saveUser(@RequestBody User user) {

return ResponseEntity.ok(service.saveUser(user));

}

@GetMapping("/{id}")

public ResponseEntity<User> getUser(@PathVariable Long id) {

return ResponseEntity.ok(service.getUser(id));

}

}

Order Service (Port: 8082)

application.yml

server:

port: 8082

spring:

datasource:

url: jdbc:mysql://localhost:3306/orderdb

username: root

password: password

jpa:

hibernate:

ddl-auto: update

show-sql: true

Order.java

@Entity

@Data

@NoArgsConstructor

@AllArgsConstructor

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> {}

User.java

@Data

public class User {

private Long id;

private String name;

private String email;

}

OrderService.java

@Service

public class OrderService {

@Autowired

private OrderRepository repo;

@Autowired

private WebClient.Builder webClient;

public Order placeOrder(Order order) {

return repo.save(order);

}

public User getUserInfo(Long userId) {

return webClient.build()

.get()

.uri("http://localhost:8081/users/" + userId)

.retrieve()

.bodyToMono(User.class)

.block();

}

}

Config: WebClientConfig.java

@Configuration

public class WebClientConfig {

@Bean

public WebClient.Builder webClientBuilder() {

return WebClient.builder();

}

}

Controller: OrderController.java

@RestController

@RequestMapping("/orders")

public class OrderController {

@Autowired

private OrderService service;

@PostMapping

public ResponseEntity<Order> placeOrder(@RequestBody Order order) {

return ResponseEntity.ok(service.placeOrder(order));

}

@GetMapping("/user/{userId}")

public ResponseEntity<User> getUser(@PathVariable Long userId) {

return ResponseEntity.ok(service.getUserInfo(userId));

}

}

1. Inventory Management System with Service Discovery

config-server/pom.xml

<dependencies>

<dependency>

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

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

</dependency>

</dependencies>

ConfigServerApplication.java

@SpringBootApplication

@EnableConfigServer

public class ConfigServerApplication {

public static void main(String[] args) {

SpringApplication.run(ConfigServerApplication.class, args);

}

}

Create Eureka Discovery Server

discovery-server/pom.xml

<dependencies>

<dependency>

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

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

</dependency>

</dependencies>

**DiscoveryServerApplication.java**

@SpringBootApplication

@EnableEurekaServer

public class DiscoveryServerApplication {

public static void main(String[] args) {

SpringApplication.run(DiscoveryServerApplication.class, args);

}

}

application.yml

server:

port: 8761

eureka:

client:

register-with-eureka: false

fetch-registry: false

Create product-service

product-service/pom.xml

<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-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

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

</dependency>

ProductServiceApplication.java

@SpringBootApplication

@EnableDiscoveryClient

public class ProductServiceApplication {

public static void main(String[] args) {

SpringApplication.run(ProductServiceApplication.class, args);

}

}

@Entity

public class Product {

@Id @GeneratedValue

private Long id;

private String name;

private int stock;

}

@Repository

public interface ProductRepository extends JpaRepository<Product, Long> {}

@RestController

@RequestMapping("/products")

public class ProductController {

@Autowired private ProductRepository repo;

@GetMapping

public List<Product> getAll() {

return repo.findAll();

}

@PostMapping

public Product create(@RequestBody Product product) {

return repo.save(product);

}

}

Create inventory-service

@SpringBootApplication

@EnableDiscoveryClient

@EnableFeignClients

public class InventoryServiceApplication {

public static void main(String[] args) {

SpringApplication.run(InventoryServiceApplication.class, args);

}

}

Example Feign Client

@FeignClient(name = "product-service")

public interface ProductClient {

@GetMapping("/products")

List<Product> getAllProducts();

}

Controller

@RestController

@RequestMapping("/inventory")

public class InventoryController {

@Autowired

private ProductClient productClient;

@GetMapping("/stocks")

public List<Product> getAllStocks() {

return productClient.getAllProducts();

}

}

Centralized Configuration

product-service.yml

spring:

application:

name: product-service

datasource:

url: jdbc:mysql://localhost:3306/productdb

username: root

password: password

jpa:

hibernate:

ddl-auto: update

bootstrap.yml

spring:

application:

name: product-service

cloud:

config:

uri: http://localhost:8888

eureka:

client:

service-url:

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

1. Implement an API Gateway

Create customer-service and billing-service

**Example customer-service Controller:**

@RestController

@RequestMapping("/customer")

public class CustomerController {

@GetMapping("/info")

public String getInfo() {

return "Customer Info";

}

}

**Example billing-service Controller:**

@RestController

@RequestMapping("/billing")

public class BillingController {

@GetMapping("/info")

public String getInfo() {

return "Billing Info";

}

}

Create api-gateway Project

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>org.springframework.boot</groupId>

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

</dependency>

</dependencies>

ApiGatewayApplication.java

@SpringBootApplication

@EnableDiscoveryClient

public class ApiGatewayApplication {

public static void main(String[] args) {

SpringApplication.run(ApiGatewayApplication.class, args);

}

}

application.yml

server:

port: 8080

spring:

application:

name: api-gateway

cloud:

gateway:

discovery:

locator:

enabled: true

lower-case-service-id: true

routes:

- id: customer-service

uri: lb://customer-service

predicates:

- Path=/api/customer/\*\*

filters:

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

- name: RequestRateLimiter

args:

redis-rate-limiter.replenishRate: 5

redis-rate-limiter.burstCapacity: 10

- id: billing-service

uri: lb://billing-service

predicates:

- Path=/api/billing/\*\*

filters:

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

- name: RequestRateLimiter

args:

redis-rate-limiter.replenishRate: 3

redis-rate-limiter.burstCapacity: 5

eureka:

client:

service-url:

defaultZone: http://localhost:8761/eureka

# Use Redis only if rate limiting with Redis

spring:

redis:

host: localhost

port: 6379

1. Resilient Microservices with Circuit Breaker

pom.xml

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>io.micrometer</groupId>

<artifactId>micrometer-core</artifactId>

</dependency>

<dependency>

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

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

</dependency>

</dependencies>

PaymentService.java

@Service

public class PaymentService {

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

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

public String callThirdPartyPaymentAPI() {

try {

Thread.sleep(3000);

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

}

throw new RuntimeException("Third-party API failed!");

}

public String fallbackPayment(Throwable t) {

logger.warn("Fallback called due to: {}", t.toString());

return "Payment service is currently unavailable. Please try later.";

}

}

PaymentController.java

@RestController

@RequestMapping("/payment")

public class PaymentController {

@Autowired

private PaymentService paymentService;

@GetMapping("/process")

public ResponseEntity<String> makePayment() {

return ResponseEntity.ok(paymentService.callThirdPartyPaymentAPI());

}

}

application.yml

server:

port: 8082

resilience4j:

circuitbreaker:

instances:

paymentAPI:

registerHealthIndicator: true

slidingWindowSize: 5

minimumNumberOfCalls: 3

failureRateThreshold: 50

waitDurationInOpenState: 10s

permittedNumberOfCallsInHalfOpenState: 2

automaticTransitionFromOpenToHalfOpenEnabled: true