**Microservices using Spring Boot3**

**Exercise 1:**

**Implement an API Gateway**

**// pom.xml**

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

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>api-gateway</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>api-gateway</name>

<description>API Gateway with Spring Cloud Gateway</description>

<parent>

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

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

<version>3.1.3</version>

<relativePath/>

</parent>

<properties>

<java.version>21</java.version>

<spring-cloud.version>2022.0.4</spring-cloud.version>

</properties>

<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-actuator</artifactId>

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>io.github.bucket4j</groupId>

<artifactId>bucket4j-core</artifactId>

<version>8.2.0</version>

</dependency>

</dependencies>

<dependencyManagement>

<dependencies>

<dependency>

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

<artifactId>spring-cloud-dependencies</artifactId>

<version>${spring-cloud.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

</project>

**// 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=/customer/\*\*

filters:

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

- name: RequestRateLimiter

args:

redis-rate-limiter.replenishRate: 5

redis-rate-limiter.burstCapacity: 10

- id: billing-service

uri: lb://billing-service

predicates:

- Path=/billing/\*\*

filters:

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

- name: RequestRateLimiter

args:

redis-rate-limiter.replenishRate: 5

redis-rate-limiter.burstCapacity: 10

eureka:

client:

service-url:

defaultZone: http://localhost:8761/eureka

instance:

prefer-ip-address: true

**// ApiGatewayApplication.java**

package com.example.apigateway;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class ApiGatewayApplication {

public static void main(String[] args) {

SpringApplication.run(ApiGatewayApplication.class, args);

}

}

**Output:**

GET <http://localhost:8080/customer/api/info>

{

"service": "Customer Service",

"status": "UP",

"message": "Welcome to the Customer Service!"

}

GET <http://localhost:8080/billing/api/status>

{

"service": "Billing Service",

"status": "Running",

"message": "Billing operations active."

}

**Exercise 2:**

**Creating Microservices for account and loan**

// src/main/java/com/cognizant/account/controller

package com.cognizant.account.controller;

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

import java.util.Map;

@RestController

@RequestMapping("/accounts")

public class AccountController {

@GetMapping("/{number}")

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

return Map.of(

"number", number,

"type", "savings",

"balance", 234343

);

}

}

**Output:**

GET <http://localhost:8080/accounts/00987987973432>

{

"number": "00987987973432",

"type": "savings",

"balance": 234343

}

// src/main/java/com/cognizant/loan/controller

package com.cognizant.loan.controller;

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

import java.util.Map;

@RestController

@RequestMapping("/loans")

public class LoanController {

@GetMapping("/{number}")

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

return Map.of(

"number", number,

"type", "car",

"loan", 400000,

"emi", 3258,

"tenure", 18

);

}

}

**Output:**

GET <http://localhost:8081/loans/H00987987972342>

{

"number": "H00987987972342",

"type": "car",

"loan": 400000,

"emi": 3258,

"tenure": 18

}

**Exercise 3:**

**Create Eureka Discovery Server and register microservices**

**// AccountServiceApplication.java**

@SpringBootApplication

@EnableDiscoveryClient

@RestController

public class AccountServiceApplication {

public static void main(String[] args) { SpringApplication.run(...); }

@GetMapping("/accounts/{number}")

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

return Map.of("number",number,"type","savings","balance",234343);

}

}

**// application.yml**

server:

port: 8081

spring:

application:

name: account-service

eureka:

client:

service-url:

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

**// LoanServiceApplication.java**

@SpringBootApplication

@EnableDiscoveryClient

@RestController

public class LoanServiceApplication {

@GetMapping("/loans/{number}")

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

return Map.of(

"number",number,

"type","car",

"loan",400000,

"emi",3258,

"tenure",18

);

}

}

**// application.yml**

server:

port: 8082

spring:

application:

name: loan-service

eureka:

client:

service-url:

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

**// EurekaDiscoveryServerApplication.java**

@SpringBootApplication

@EnableEurekaServer

public class EurekaDiscoveryServerApplication {

public static void main(String[] args) { SpringApplication.run(...); }

}

**// application.yml**

server:

port: 8761

spring:

application:

name: eureka-server

eureka:

client:

register-with-eureka: false

fetch-registry: false

logging:

level:

com.netflix.eureka: OFF

com.netflix.discovery: OFF

**// ApiGatewayApplication.java**

@SpringBootApplication

@EnableDiscoveryClient

public class ApiGatewayApplication { public static void main(...) }

**// application.yml**

server:

port: 8080

spring:

application:

name: api-gateway

eureka:

client:

service-url:

defaultZone: http://localhost:8761/eureka

spring:

cloud:

gateway:

routes:

- id: account-route

uri: lb://account-service

predicates:

- Path=/accounts/\*\*

- id: loan-route

uri: lb://loan-service

predicates:

- Path=/loans/\*\*

//

@Component

@Order(Ordered.HIGHEST\_PRECEDENCE)

public class LoggingGlobalFilter implements GlobalFilter {

private final Logger log = LoggerFactory.getLogger(getClass());

@Override

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

log.info("PRE: {} {}", ex.getRequest().getMethod(), ex.getRequest().getURI());

return chain.filter(ex).then(Mono.fromRunnable(() ->

log.info("POST: {}", ex.getResponse().getStatusCode())));

}

}

**Output:**

GET <http://localhost:8081/accounts/00987987973432>

{

"number": "00987987973432",

"type": "savings",

"balance": 234343

}