**Sample Hands-on Exercises on Edge Services and API Gateway with Spring Boot 3 and Spring Cloud**

**Exercise 1: Implementing Edge Services for Routing and Filtering**

### 1. pom.xml

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

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>edge-service</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<properties>

<java.version>17</java.version>

<spring-cloud.version>2022.0.3</spring-cloud.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

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

</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>

<build>

<plugins>

<plugin>

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

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build></project>

2. application.properties

server.port=8080

spring.cloud.gateway.routes[0].id=example\_route

spring.cloud.gateway.routes[0].uri=http://example.org

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

3. LoggingFilter.java

package com.example.gateway;

import org.springframework.cloud.gateway.filter.GlobalFilter;import org.springframework.cloud.gateway.filter.GatewayFilterChain;import org.springframework.stereotype.Component;import org.springframework.web.server.ServerWebExchange;import reactor.core.publisher.Mono;

@Componentpublic class LoggingFilter implements GlobalFilter {

@Override

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

System.out.println("Request: " + exchange.getRequest().getURI());

return chain.filter(exchange);

}

}

4. EdgeServiceApplication.java

package com.example.gateway;

import org.springframework.boot.SpringApplication;import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplicationpublic class EdgeServiceApplication {

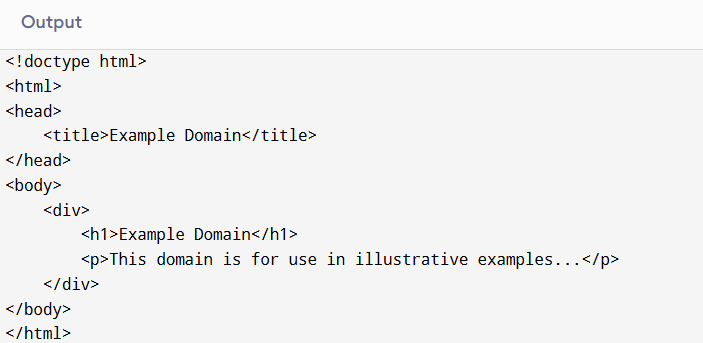
public static void main(String[] args) {

SpringApplication.run(EdgeServiceApplication.class, args);

}

}

**OUTPUT:**



**Exercise 2: Load Balancing in an API Gateway**

**1.** pom.xml **dependencies**

<dependencies>

<!-- Spring Boot Starter Web (Optional if not writing controller) -->

<dependency>

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

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

</dependency>

<!-- Spring Cloud Gateway -->

<dependency>

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

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

</dependency>

<!-- Spring Cloud LoadBalancer -->

<dependency>

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

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

</dependency>

<!-- Spring Boot Actuator (Optional for health check) -->

<dependency>

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

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

</dependency></dependencies>

<dependencyManagement>

<dependencies>

<dependency>

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

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

<version>2023.0.0</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies></dependencyManagement>

**2.** application.properties

spring.application.name=api-gateway

# Route with LoadBalancer URI

spring.cloud.gateway.routes[0].id=load\_balanced\_route

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

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

# For logging and debugging

logging.level.org.springframework.cloud.gateway=DEBUG

**3. Custom Load Balancer Configuration**

package com.example.gateway.config;

import org.springframework.cloud.client.ServiceInstance;import org.springframework.cloud.loadbalancer.core.RandomLoadBalancer;import org.springframework.cloud.loadbalancer.core.ReactorLoadBalancer;import org.springframework.cloud.loadbalancer.support.LoadBalancerClientFactory;import org.springframework.context.annotation.Bean;import org.springframework.context.annotation.Configuration;import org.springframework.core.env.Environment;

@Configurationpublic class LoadBalancerConfiguration {

@Bean

public ReactorLoadBalancer<ServiceInstance> randomLoadBalancer(

Environment environment,

LoadBalancerClientFactory loadBalancerClientFactory) {

String name = environment.getProperty(LoadBalancerClientFactory.PROPERTY\_NAME);

return new RandomLoadBalancer(

loadBalancerClientFactory.getLazyProvider(name, ServiceInstanceListSupplier.class),

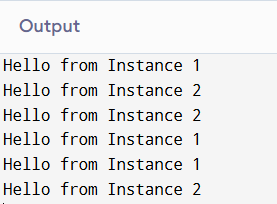
name

);

}

}

**OUTPUT:**



**Exercise 3: Resilience Patterns in an API Gateway**

### 1. pom.xml Dependencies

<dependencies>

<!-- Spring Boot Gateway -->

<dependency>

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

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

</dependency>

<!-- Resilience4j for circuit breaker -->

<dependency>

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

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

</dependency>

<!-- Spring Cloud Resilience4j Gateway integration -->

<dependency>

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

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

</dependency>

<!-- (Optional) Actuator for metrics -->

<dependency>

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

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

</dependency></dependencies>

<dependencyManagement>

<dependencies>

<dependency>

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

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

<version>2023.0.0</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies></dependencyManagement>

2. application.properties

# Circuit Breaker Settings

resilience4j.circuitbreaker.instances.exampleCircuitBreaker.registerHealthIndicator=true

resilience4j.circuitbreaker.instances.exampleCircuitBreaker.slidingWindowSize=10

resilience4j.circuitbreaker.instances.exampleCircuitBreaker.failureRateThreshold=50

# Gateway route using circuit breaker

spring.cloud.gateway.routes[0].id=resilient\_route

spring.cloud.gateway.routes[0].uri=http://localhost:8081/fail-me

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

spring.cloud.gateway.routes[0].filters[0]=CircuitBreaker=name=exampleCircuitBreaker,fallbackUri=forward:/fallback

3. ResilienceConfiguration.java

package com.example.gateway.config;

import io.github.resilience4j.circuitbreaker.CircuitBreakerConfig;import io.github.resilience4j.timelimiter.TimeLimiterConfig;import org.springframework.cloud.circuitbreaker.resilience4j.ReactiveResilience4JCircuitBreakerFactory;import org.springframework.cloud.circuitbreaker.resilience4j.Resilience4JConfigBuilder;import org.springframework.context.annotation.Bean;import org.springframework.context.annotation.Configuration;import org.springframework.cloud.gateway.filter.factory.rewrite.ModifyResponseBodyGatewayFilterFactory;import java.time.Duration;import org.springframework.context.annotation.Customizer;

@Configurationpublic class ResilienceConfiguration {

@Bean

public Customizer<ReactiveResilience4JCircuitBreakerFactory> defaultCustomizer() {

return factory -> factory.configureDefault(id ->

new Resilience4JConfigBuilder(id)

.circuitBreakerConfig(CircuitBreakerConfig.ofDefaults())

.timeLimiterConfig(TimeLimiterConfig.custom()

.timeoutDuration(Duration.ofSeconds(2))

.build())

.build());

}

}

4. FallbackController.java

package com.example.gateway.controller;

import org.springframework.web.bind.annotation.GetMapping;import org.springframework.web.bind.annotation.RestController;

@RestControllerpublic class FallbackController {

@GetMapping("/fallback")

public String fallback() {

return "Fallback response: Service is temporarily unavailable. Please try again later.";

}

}

**OUTPUT:**

