Microservices

Exercise 1

1-Dependencies:

<dependencies>

<dependency>

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

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

</dependency>

</dependencies>

2-Configuration:

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-Logging Filter:

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;

@Component

public class LoggingFilter implements GlobalFilter {

@Override

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

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

return chain.filter(exchange);

}

}

Exercise 2

1-Dependencies:

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</d

ependency>

</dependencies>

2-Configuration:

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/\*\*

3-Load balancer:

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.cloud.client.ServiceInstance;

import org.springframework.cloud.client.loadbalancer.LoadBalancerClientFactory;

import org.springframework.cloud.loadbalancer.core.RandomLoadBalancer;

import org.springframework.cloud.loadbalancer.core.ReactorLoadBalancer;

import org.springframework.cloud.loadbalancer.support.LoadBalancerClientFactory;

import org.springframework.cloud.loadbalancer.support.ServiceInstanceListSupplier;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.core.env.Environment;

@Configuration

public 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

);

}

}

Exercise 3

1-Dependencies

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

<artifactId>resilience4j-spring-boot2</artifactId>

</dependency>

</dependencies>

2-Configuration:

resilience4j.circuitbreaker.instances.exampleCircuitBreaker.registerHealthIndicator=true

resilience4j.circuitbreaker.instances.exampleCircuitBreaker.slidingWindowSize=10

resilience4j.circuitbreaker.instances.exampleCircuitBreaker.failureRateThreshold=50

3-Resilliance Config:

import io.github.resilience4j.circuitbreaker.CircuitBreakerConfig;

import io.github.resilience4j.timelimiter.TimeLimiterConfig;

import org.springframework.cloud.circuitbreaker.resilience4j.Resilience4JCircuitBreakerFactory;

import org.springframework.cloud.circuitbreaker.resilience4j.ReactiveResilience4JCircuitBreakerFactory;

import org.springframework.cloud.client.circuitbreaker.Customizer;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import java.time.Duration;

@Configuration

public class ResilienceConfiguration {

@Bean

public Customizer<ReactiveResilience4JCircuitBreakerFactory> defaultCustomizer() {

return factory -> factory.configureDefault(id ->

new Resilience4JCircuitBreakerFactory.Resilience4JConfigBuilder(id)

.circuitBreakerConfig(CircuitBreakerConfig.custom()

.failureRateThreshold(50)

.slidingWindowSize(10)

.build())

.timeLimiterConfig(TimeLimiterConfig.custom()

.timeoutDuration(Duration.ofSeconds(4))

.build())

.build()

);

}

}