Circuit Breaker & Retry module & Rate Limiter & Load Balancing

***User service -----> catalog service*** (user service calling catalog service) ( so suppose catalog service is down )

**User service config**

1 . Dependency : - aop,actuators,resilience4j

2. application.properites :-

**management:**

**health:**

**circuitbreakers:**

**enabled:** true

**endpoints:**

**web:**

**exposure:**

**include:** health

**endpoint:**

**health:**

**show-details:** always

(circuit breaker config)

**resilience4j:**

**circuitbreaker:**

**instances:**

**userService:**

**registerHealthIndicator:** true

**eventConsumerBufferSize**: 10

**failureRateThreshold**: 50

**minimumNumberOfCalls**: 5

**automaticTransitionFromOpenToHalfOpenEnabled**: true

**waitDurationInOpenState**: 5s

**permittedNumberOfCallsInHalfOpenState**: 3

**slidingWindowSize**: 10

**slidingWindowType**: COUNT\_BASED

(retry module config)

**retry:**

**instances:**

**userService:**

**maxRetryAttempts**: 5 [1st time hit then every 10 sec it automatically hit ]

**waitDuration**: 10s

(rate limiter config)

**ratelimiter:**

**instances:**

**userService:**

**limit-refresh-period:** 4s [4 sec madhe every thread che 2 request allow]

**limit-for-period**: 2

**timeout-duration**: 2s

3**.Service:-**

@GetMapping("/displayOrders")

**// @CircuitBreaker(name ="userService",fallbackMethod = "getAllAvailableProducts")**

//**@Retry(name = "userService",fallbackMethod = "getAllAvailableProducts")**

//**@RateLimiter(name = "userService",fallbackMethod = "getAllAvailableProducts")**

public **List<OrderDTO>** displayOrders(@RequestParam("category") String category) {

String url = category == null ? BASEURL : BASEURL + "/" + category;

System.out.println("retry method called "+attempt++ +" times "+" at "+new Date());

return restTemplate.getForObject(url, ArrayList.class);

}

**public List<OrderDTO> getAllAvailableProducts(Exception e){**

**return Stream.of(**

**new OrderDTO(119, "LED TV", "electronics", "white", 45000),**

**new OrderDTO(345, "Headset", "electronics", "black", 7000),**

**new OrderDTO(475, "Sound bar", "electronics", "black", 13000),**

**new OrderDTO(574, "Puma Shoes", "foot wear", "black & white", 4600),**

**new OrderDTO(678, "Vegetable chopper", "kitchen", "blue", 999),**

**new OrderDTO(532, "Oven Gloves", "kitchen", "gray", 745)**

**).collect(Collectors.toList());**

**}**

**Load balancing:-**

1. Created and ran Eureka Server & Api Gateway

2 Created multiple instances of product-service

3 Created order-service that uses @LoadBalanced RestTemplate

4 Verified that each request goes to different instance

5 (Optional) Set up API Gateway for centralized routing

<dependency>

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

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

</dependency>

@Bean

@LoadBalanced

public RestTemplate restTemplate() {

return new RestTemplate();

}

@GetMapping("/order")

public String placeOrder() {

String response = restTemplate.getForObject("http://product-service/api/products", String.class);

return "Order placed. Product info: " + response;

}