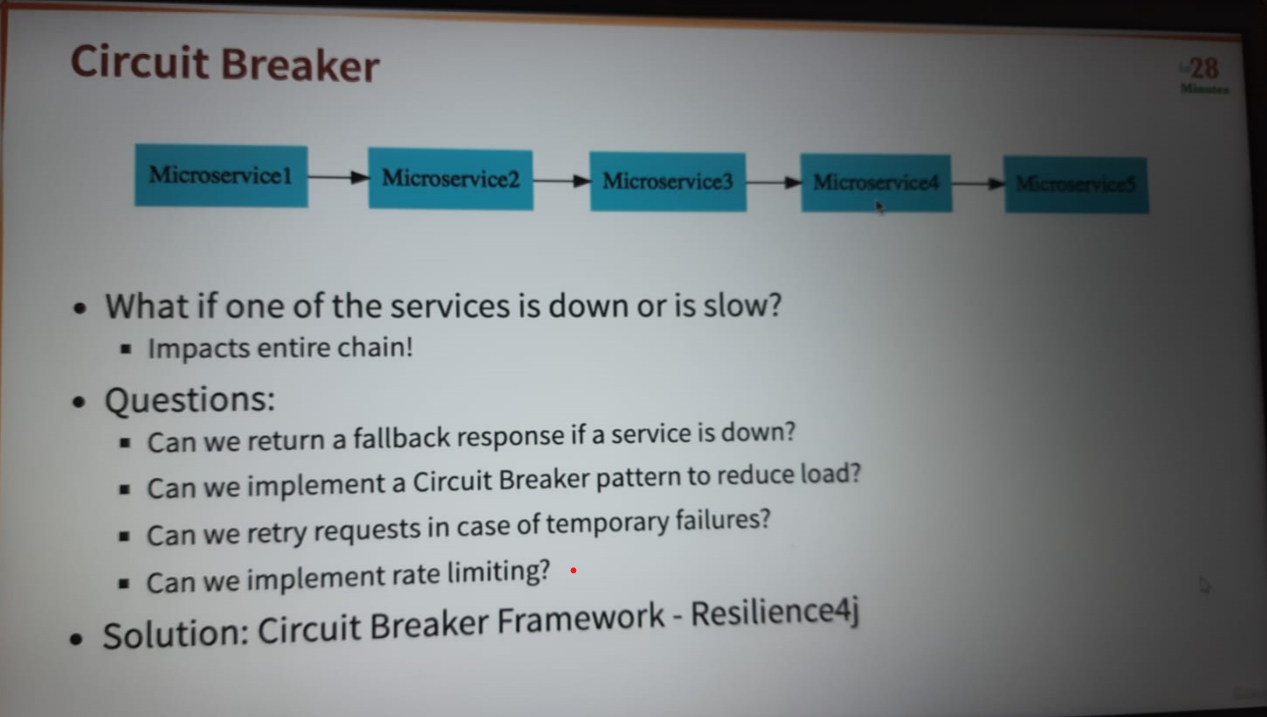
In microservices arch. Few microservices are very complex and form like chain

Microservice1 🡪 Microservice2 🡪 Microservice3 🡪 Microservice4 🡪 Microservice5

Here m1 depends on m2, m2 depends on m3 and so on.

What if one microservice fails or very slow ?

-entire chain will get impacted.



Rate limit is limited no. of request per unit time.

Fallback response is default response.

**Resilience4j** is a lightweight fault tolerance library designed for functional programming. Resilience4j provides higher-order functions (decorators) to enhance any functional interface, lambda expression or method reference with a Circuit Breaker, Rate Limiter, Retry or Bulkhead. You can stack more than one decorator on any functional interface, lambda expression or method reference. The advantage is that you have the choice to select the decorators you need and nothing else.

Now, implementation :-

In currency-exchange service which is source service n provide response to client we need to do changes.

Need to add both below dependencies: -

<dependency>

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

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

</dependency>

<dependency>

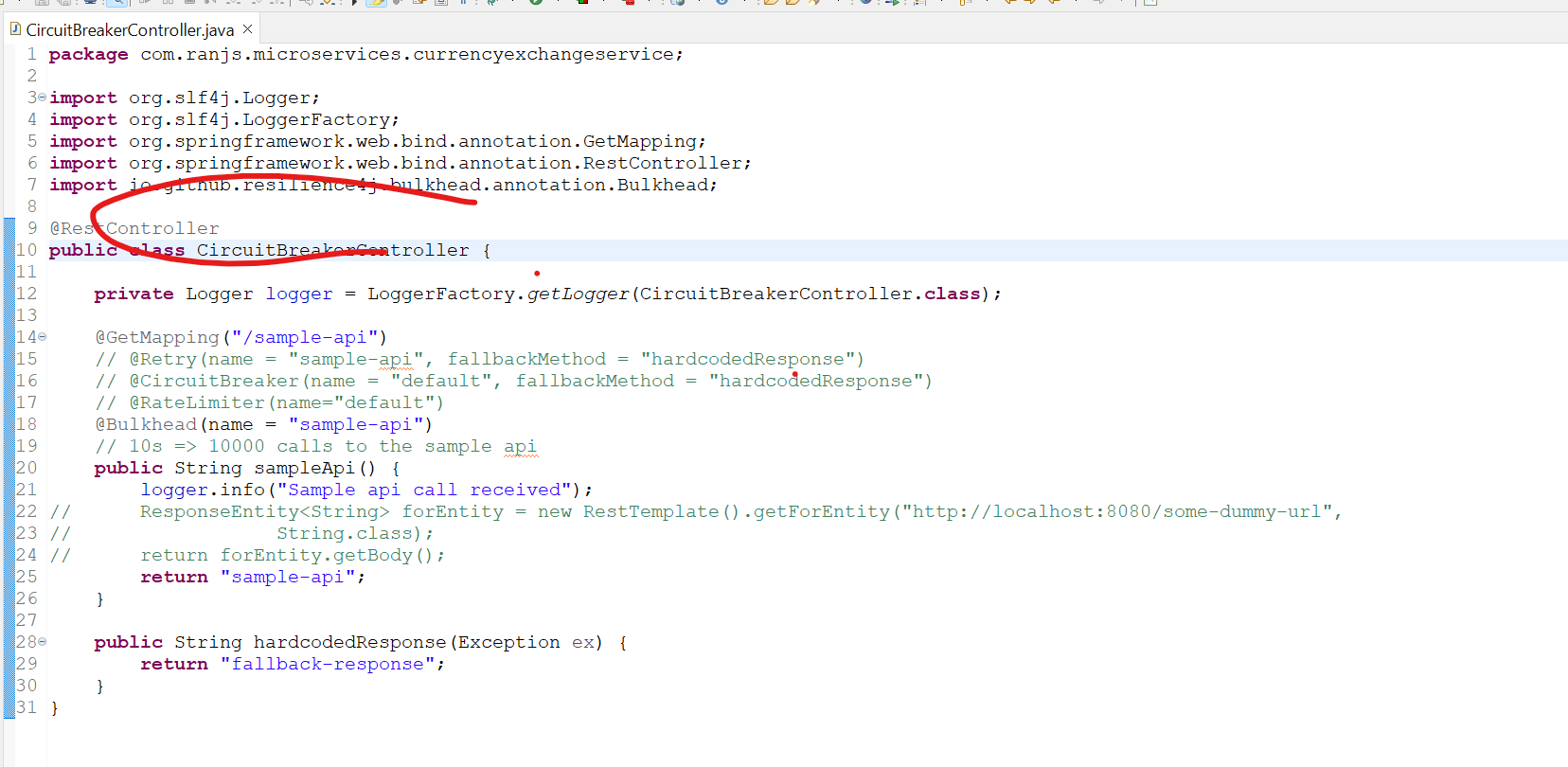
<groupId>io.github.resilience4j</groupId>

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

</dependency>

Create new controller: -

CircuitBreakerController.



In app.properties

resilience4j.retry.instances.sample-api.maxRetryAttempts=5

resilience4j.retry.instances.sample-api.waitDuration=1s

resilience4j.retry.instances.sample-api.enableExponentialBackoff=true

#resilience4j.circuitbreaker.instances.default.failureRateThreshold=90

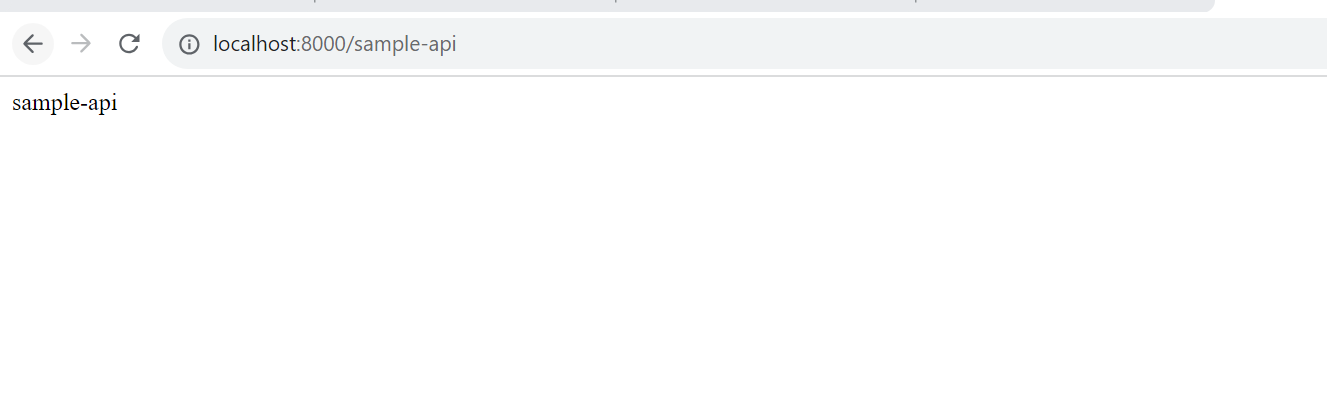
resilience4j.ratelimiter.instances.default.limitForPeriod=2

resilience4j.ratelimiter.instances.default.limitRefreshPeriod=10s

resilience4j.bulkhead.instances.default.maxConcurrentCalls=10

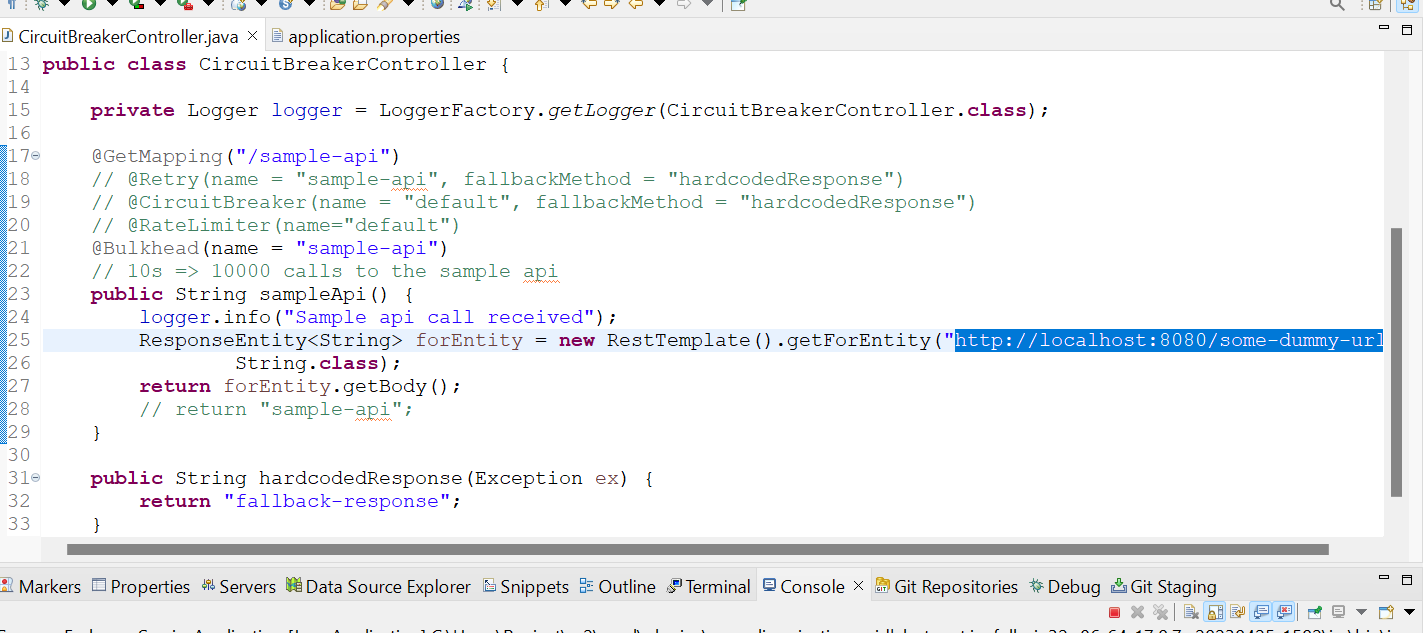
resilience4j.bulkhead.instances.sample-api.maxConcurrentCalls=10

<http://localhost:8000/sample-api>

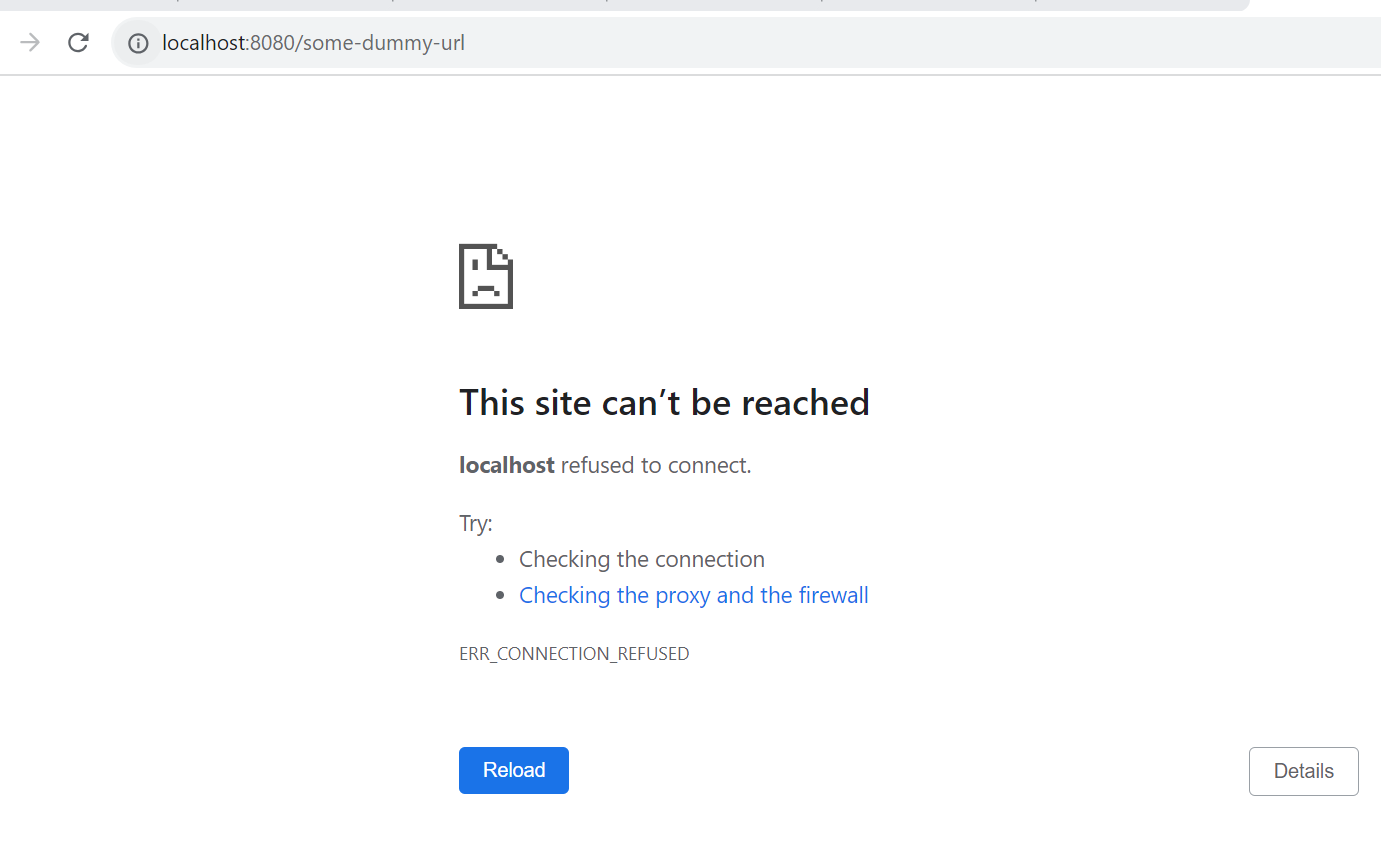


Now, to test lets uncomment the comment in controller n try to access

http://localhost:8080/some-dummy-url



– so, it will fail



Now we will use @Retry(name = "sample-api")

By adding this it will retry 3 times after that only it will throw error. resilience4j.retry.instances.sample-api.maxRetryAttempts=5 🡪 for custom no of retry.

@Retry(name = "sample-api", fallbackMethod = "hardcodedResponse")

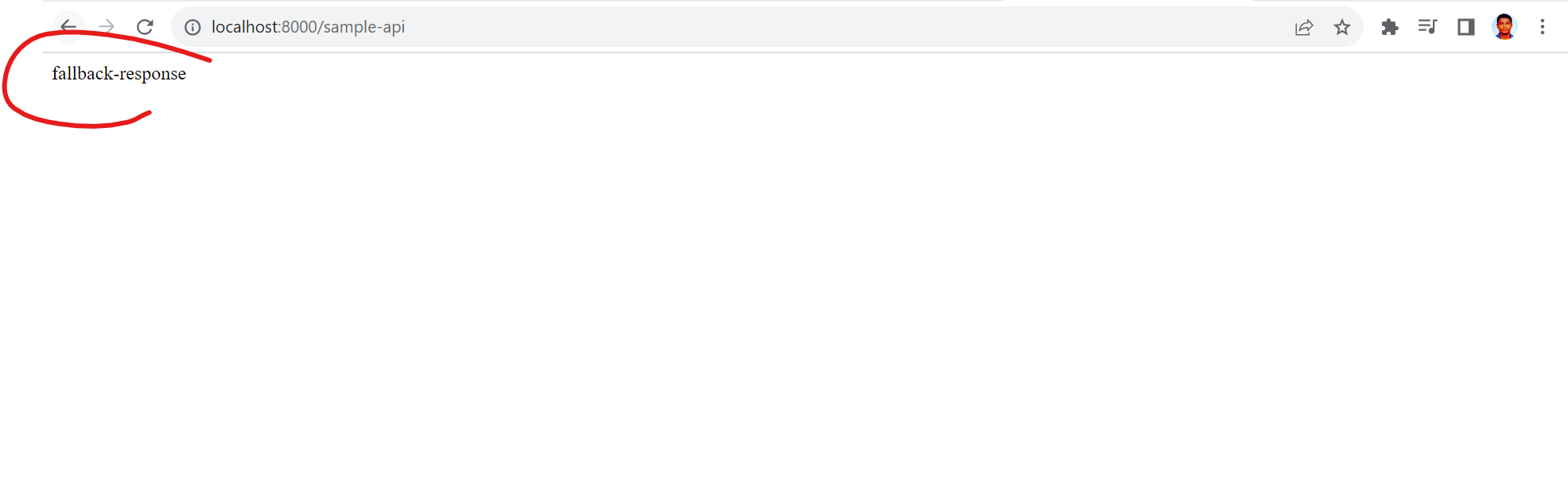
To throw default error after no of retry



And resilience4j.retry.instances.sample-api.maxRetryAttempts=5

In app.properties to retry for 5 attempts before posting the response on browser

<http://localhost:8000/sample-api>



Default response

Some more properties :-

resilience4j.retry.instances.sample-api.waitDuration=1s

resilience4j.retry.instances.sample-api.enableExponentialBackoff=true

#resilience4j.circuitbreaker.instances.default.failureRateThreshold=90

resilience4j.ratelimiter.instances.default.limitForPeriod=2

resilience4j.ratelimiter.instances.default.limitRefreshPeriod=10s

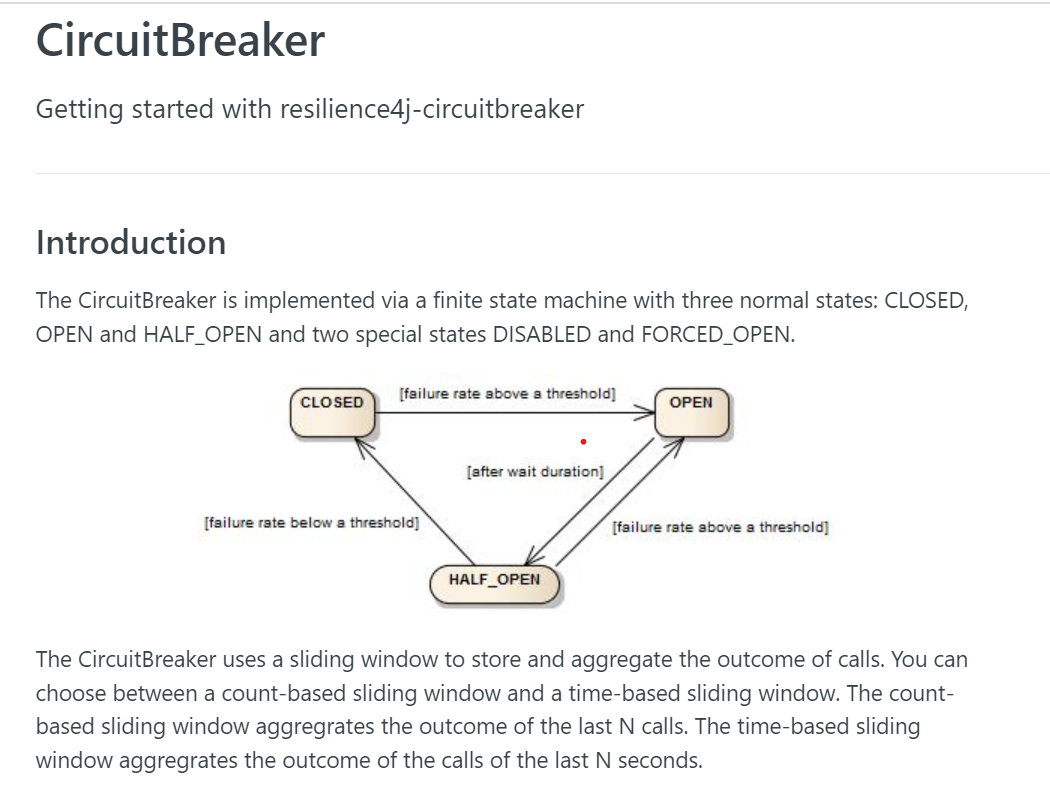
resilience4j.bulkhead.instances.default.maxConcurrentCalls=10

resilience4j.bulkhead.instances.sample-api.maxConcurrentCalls=10

now , comment @Retry

and use @CircuitBreaker

@CircuitBreaker(name = "default", fallbackMethod = "hardcodedResponse")



<https://resilience4j.readme.io/docs/circuitbreaker>

now use @RateLimiter(name="default")

its abt limit the no of requests per specific time period

resilience4j.ratelimiter.instances.default.limitForPeriod=2

resilience4j.ratelimiter.instances.default.limitRefreshPeriod=10s

and finally, @Bulkhead(name = "sample-api")

addition to rate limiter –we can use bulkhead for concurrent calls

resilience4j.bulkhead.instances.default.maxConcurrentCalls=10

resilience4j.bulkhead.instances.sample-api.maxConcurrentCalls=10