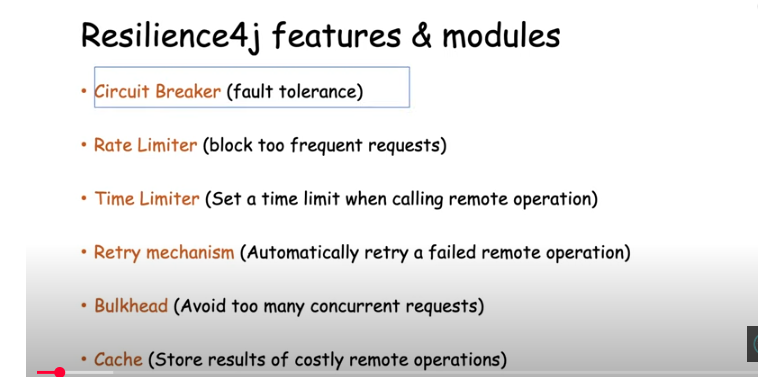
To implement Fault tolerance we implement Circuit Breaker Pattern With Resilience4j:



500 internal server error

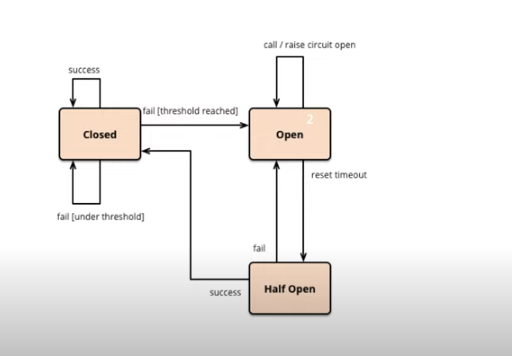
CurrencyConversion

CurrencyExchange

Here CurrencyConversion microservice is calling currencyexchange microservicce.

Lets assume dueto infrastructure issue , currencyExhange Service is down.

We can set some threshold for calls to failing services.



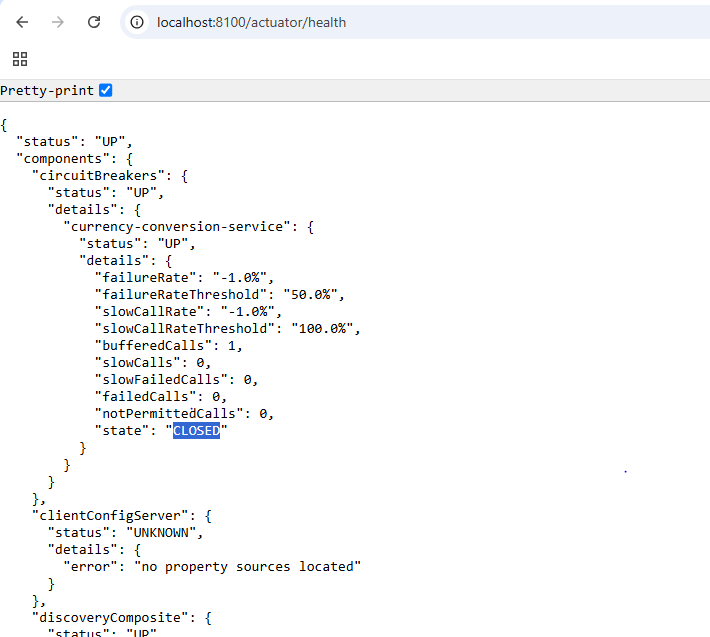
In closed state both the services in up and running

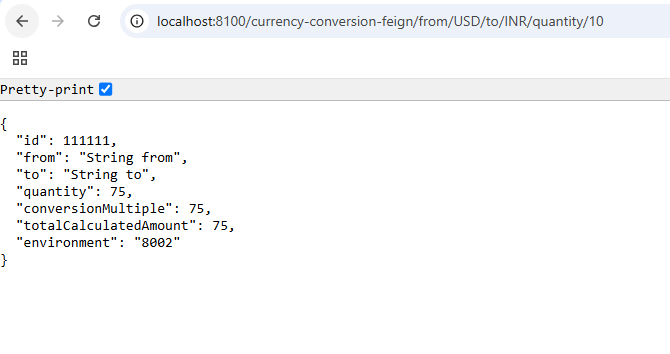
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-actuator</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-config</artifactId>  
</dependency>  
<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>

Actuator : to check health of your microservice

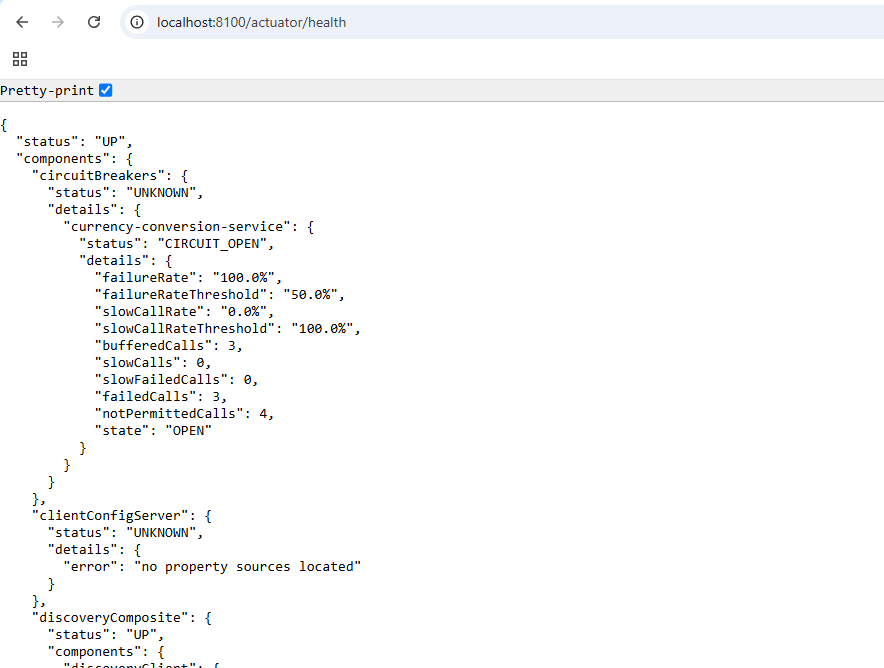
AOP : will send the metrics to the Actuator

Resilience : to implement circuit breaker









# Microservice | Resilience4J Circuit Breaker Implementation on Spring Boot | JavaTechie

https://www.youtube.com/watch?v=b6R4dElDtRc&t=1267s

**Microservice | Resilience4J Retry Module Implementation With Spring Boot | JavaTechie**

https://www.youtube.com/watch?v=Z4CSGsOLb1c