Spring Cloud Tutorial- Netflix Hystrix Circuit Breaker Simple Example

javainuse.com/spring/spring_hystrix_circuitbreaker

Spring Cloud- Circuit Breaker using Hystrix

In a previous post we had implemented Fallback method using Hystrix. In this post we implement the Circuit Breaker using Hystrix

Spring Cloud - Table Of Contents

What is the Netflix Hystrix Circuit Breaker Feature? Need for it?

In previous posts we had two services- employee-consumer consuming the service exposed by the employee-producer.
Due to some reason the employee-producer exposed service throws an exception. In this case using Hystrix we defined a fallback method. In case of exception in the exposed service the fallback method returned some default value.

If the exceptions keep on occuring in the firstPage method() then the Hystrix circuit will break and the employee consumer will skip the firtsPage method all together and directly call the fallback method.

The purpose of circuit breaker is to give time to the first page method or other methods that the firstpage method might be calling and is causing the exception to recover. It might happen that on less load the issue causing the exceptions have better chance of recovering

Video

This tutorial is explained in the below Youtube Video.

Lets Begin-

The employee-producer had the following firstpage method annotated with the hystrix annotation.

```
package com.javainuse.controllers;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import com.javainuse.model.Employee;
import
com.netflix.hystrix.contrib.javanica.annotation.HystrixCommand;

@RestController
public class TestController {

 @RequestMapping(value = "/employee", method = RequestMethod.GET)
 @HystrixCommand(fallbackMethod = "getDataFallBack")
 public Employee firstPage() {

  System.out.println("Inside firstPage");

  Employee emp = new Employee();
  emp.setName("emp1");
  emp.setDesignation("manager");
  emp.setEmpId("1");
  emp.setSalary(3000);

  if(emp.getName().equalsIgnoreCase("emp1"))
   throw new RuntimeException();

  return emp;
 }

 public Employee getDataFallBack() {

  System.out.println("Inside fallback");


  Employee emp = new Employee();
  emp.setName("fallback-emp1");
  emp.setDesignation("fallback-manager");
  emp.setEmpId("fallback-1");
  emp.setSalary(3000);

  return emp;

 }

}
```

Previously using employee-consumer we were calling the employee producer only once. Now using for loop we will call it multiple times and check if the circuit trips and the fallback method gets called directly.

```
package com.javainuse;

import java.io.IOException;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.Bean;
import org.springframework.web.client.RestClientException;

import com.javainuse.controllers.ConsumerControllerClient;

@SpringBootApplication
public class SpringBootHelloWorldApplication {

 public static void main(String[] args) throws RestClientException, IOException {
  ApplicationContext ctx = SpringApplication.run(
    SpringBootHelloWorldApplication.class, args);

  ConsumerControllerClient
consumerControllerClient=ctx.getBean(ConsumerControllerClient.class);
  System.out.println(consumerControllerClient);
  for(int i=0;i<100;i++)
  consumerControllerClient.getEmployee();

 }

 @Bean
 public  ConsumerControllerClient  consumerControllerClient()
 {
  return  new ConsumerControllerClient();
 }
}
```

As we had done in previous posts- Start the following Spring Boot Applications-

- eureka-server
- employee-producer
- employee-consumer

On running the employee consumer we get the output in employee-producer module as follows-

We can see that after some exceptions the **fallback method getting called directly and the hystrix annotated method skipped. So the hystrix circuit is open.**