Laboratorio 8

Paso 1: Añadir Hystrix al proyecto

Lo primero de todo será incluir Hystrix como una nueva dependencia en el fichero pom.xml del microservicio *client-greeting.*

```
<dependency>
   <groupId>org.springframework.cloud</groupId>
   <artifactId>spring-cloud-starter-hystrix</artifactId>
   <versión>1.4.4.RELEASE</versión>
</dependency>
```

Paso 2: Aplicar Circuit Breaker Pattern

Spring Cloud Netflix Hystrix buscará cualquier método con la anotación ***@HystrixCommand*,** el cual envolverá en un proxy conectado al circuit breaker para que Hystrix pueda monitorizarlo. Estos métodos deberán pertenecer a una clase anotada necesariamente con ***@Component*** o ***@Service***

```
@Service
public class ClientGreetingService {

   @HystrixCommand(fallbackMethod = "greetingDefault")
   public Greeting greeting (String name) {

      return restTemplate.getForObject(serviceUrl +
"/greeting/{name}",Greeting.class, name);
   }

   public Greeting greetingDefault(String name) {
      return new Greeting("Hello World thanks to Circuit Breaker (Hystrix)");
    }
}
```

Observa que en la anotación *@HystrixCommand* se especifica el ***fallbackMethod,*** atributo donde se indica a *Hystrix* que método llamar al **abrirse el *circuit breaker*** en el caso de que surgieran problemas en la llamada remota REST al microservicio *greeting.*

Para que Hystrix localice correctamente el método de *fallback*, es importante que éste se encuentre ubicado en la misma clase que el método anotado con *@HystrixCommand* con el que está asociado y que la signatura de ambos sea la misma (sólo cambia el nombre del método)

El código anterior muestra el ***modelo síncrono*** de *Hystrix* pero ten en cuenta que existen otros dos modelos de ejecución:

- ***Asíncrona (Future):*** Realiza el control de ruptura del circuito en modo asíncrono

```
@HystrixCommand(fallbackMethod = "greetingFutureDefault")
public Future<Greeting> greetingFuture(final String name) {
   return new AsyncResult<Greeting>() {
   public Greeting invoke() {
      return restTemplate.getForObject(serviceUrl +
"/greeting/{name}", Greeting.class, name);
   }
   };
}
```

- ***Reactiva (Observable):*** Declara un método como callback, pudiendo definir varios comportamientos dependiendo del resultado de la llamada (casos de éxito, error, etc). Funciona como un modelo de subscripción a una cola de respuesta. Por el momento, y para no hacer más complejo el ejemplo, dejaremos la implementación de este caso para más adelante

Paso 3: Habilitar Circuit Breaker

Incluir la anotación ***@EnableCircuitBreaker*** para indicar a Spring Cloud que el microservicio *clientGreeting* utiliza *circuit breaker* y habilitar la monitorización, apertura y cierre del circuito

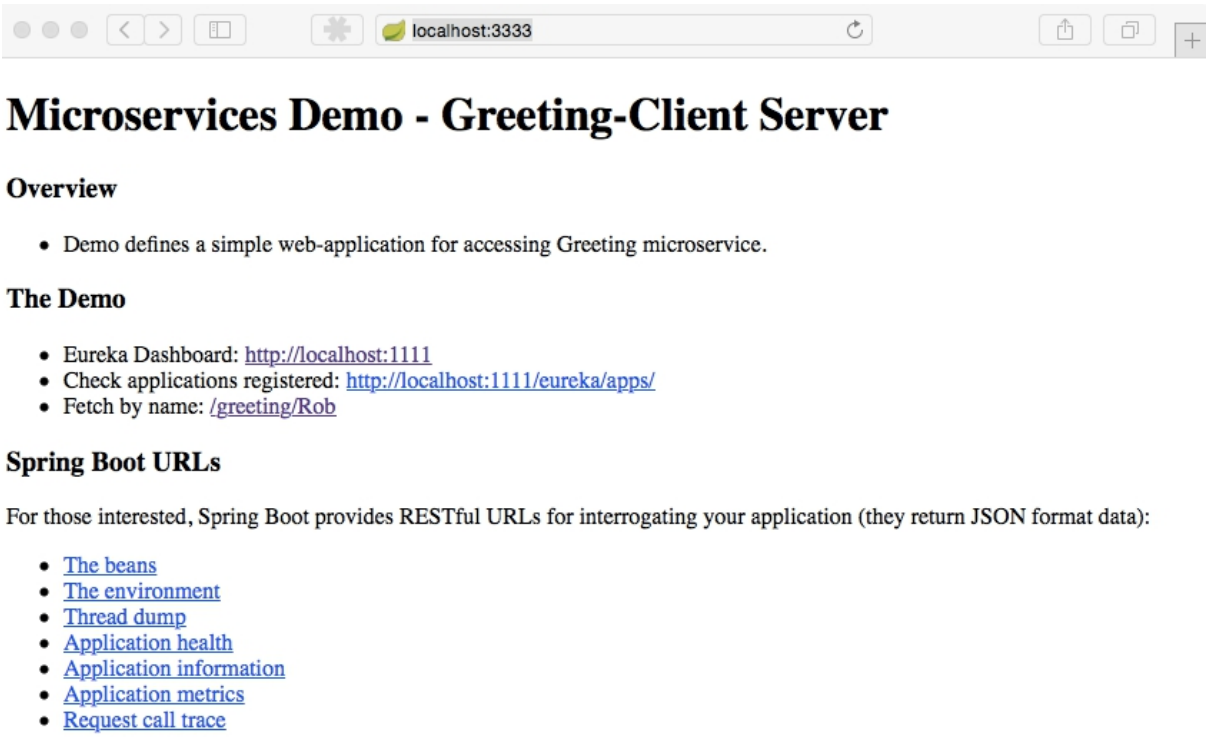
```
@SpringBootApplication
@EnableDiscoveryClient
@EnableCircuitBreaker
public class MicroserviceClientGreetingApplication {

   public static void main(String[] args) {
      // Tell server to look for web-server.properties or web-server.yml
      SpringApplication.run(MicroserviceClientGreetingApplication.class, args);
   }
   ...
}
```

Paso 4: Probar Circuit Breaker

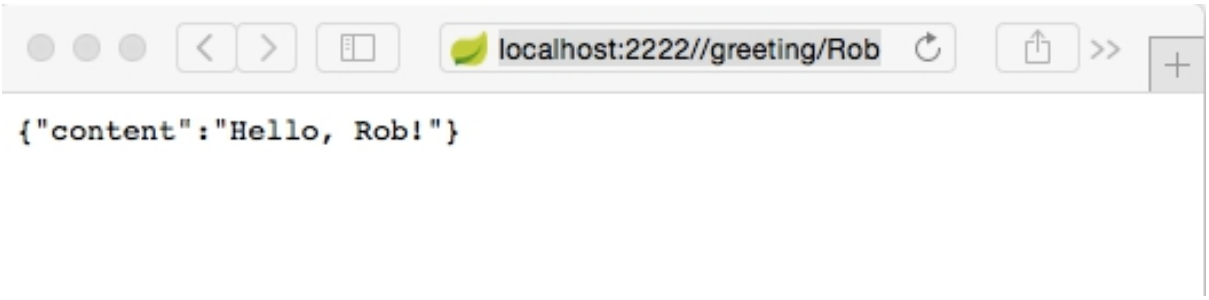
1. Inicia ***Eureka***. Accede a http://localhost:1111 y comprueba que visualizas correctamente el *dashboard* de*l servicio de registro*
2. Levanta el microservicio ***greeting*** *(MicroserviceGreetingApplication).* Comprueba que el microservicio se registra correctamente en Eureka y

que responde a peticiones REST como la siguiente: http://localhost:2222//greeting/Rob

3. Ejecuta el microservicio ***client-greeting*** *(MicroserviceClientGreetingApplication)*. Al igual que en el caso anterior, verifica que aparece registrado en Eureka y que está accesible en http://localhost:3333



Si pinchas en el link *Fetch by Name*, el microservicio *client-greeting-service* llamará a *Eureka* para localizar el *endpoint* del microservicio *greeting-service* y a continuación realizará una petición REST para llamar al método *greeting* de este servicio pasándole como parámetro un nombre ("*Rob*"). A continuación muestro el resultado



4. Para probar el circuit breaker vamos a simular que el microservicio greeting no responde correctamente a las peticiones. Para ello vamos a

detener su ejecución. A continuación, sitúate de nuevo en la página principal del microservicio client-greeting (http://localhost:3333) y vuelve a ejecutar la operación ***Fetch by Name***. En esta ocasión, al encontrarse el microservicio greeting caído y por tanto no responder a la peticiones recibidas, *Hystrix* detectará el error, abrirá el circuit breaker y derivará el flujo de la ejecución al método *greetingDefault(String name)* que devolverá un mensaje de saludo genérico como el que se muestra a continuación:

Espero te haya resultado interesante esta entrada y te animo a utilizar este patrón en los casos de uso que te resulten más apropiados.