# Feign

## Que es Feign Client

Es un servicio declarativo de clientes rest. Permite crear de manera sencilla clientes Rest.

## Como incluir feign en el proyecto

Hay que incluir la dependencia en Maven

<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-openfeign</artifactId>  
 <version>2.2.2.RELEASE</version>  
</dependency>

Luego hay que habilitar es uso de Feign con

@SpringBootApplication  
@EnableFeignClients  
public class Application {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(Application.class, args);  
 }  
  
}

Con la anotación **@FeignClient** tenemos varias propiedades de configuración

1. **Value**: Indica el identificador del cliente. Este nombre será usado por Ribbon (client load balancer) o por Spring Cloud LoadBalancer.
2. **url**: Dirección del servicio al que nos queremos comunicar.
3. **Qualifier**: Si no se indica, el vean adoptará el nombre de la interfaz.
4. **Configuration**: Permite agregar configuraciones personalizada. Esta configuración se gregará a la configuración que trae Feign.

@FeignClient(name = "stores", configuration = FooConfiguration.class)

* 1. Importante: La clase FooConfiguration no es necesaria que se etiquete con la anotación @Configuration. En el caso de que se haga, hay que quitar esta clase del @ComponentScan porque si no se convertirá en la clase por defecto que servirá feignDecoder, feign.Encoder, feign.Contract, etc.

1. Se pueden indicar los valores de estas propiedades en el fichero de propiedades.

@FeignClient(name = "${feign.name}", url = "${feign.url}")

# Hystrix

Es una librería de tolerancia de fallos desarrollada por Netflix. Hace uso del patrón Cortocircuito (circuit braker) para evitar el fallo en cascada de las aplicaciones.

El principio es análogo a la electrónica: Hystrix monitoriza los métodos esperando fallos en las llamadas a los servicios relacionados. Si se produce un fallo, abrirá el circuito y redirigirá la llamada a un método FallBack.

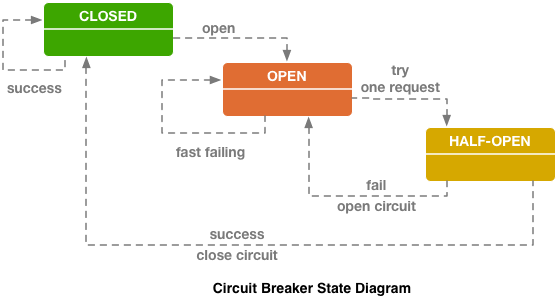
La librería tolerará fallos hasta un límite. Por encima de este límite, dejará el circuito abierto, lo que significa que desviará las llamadas a un método fallback para prevenir fallos. Esto permite establecer una ventana de tiempo para que el servicio afectado pueda recuperarse.

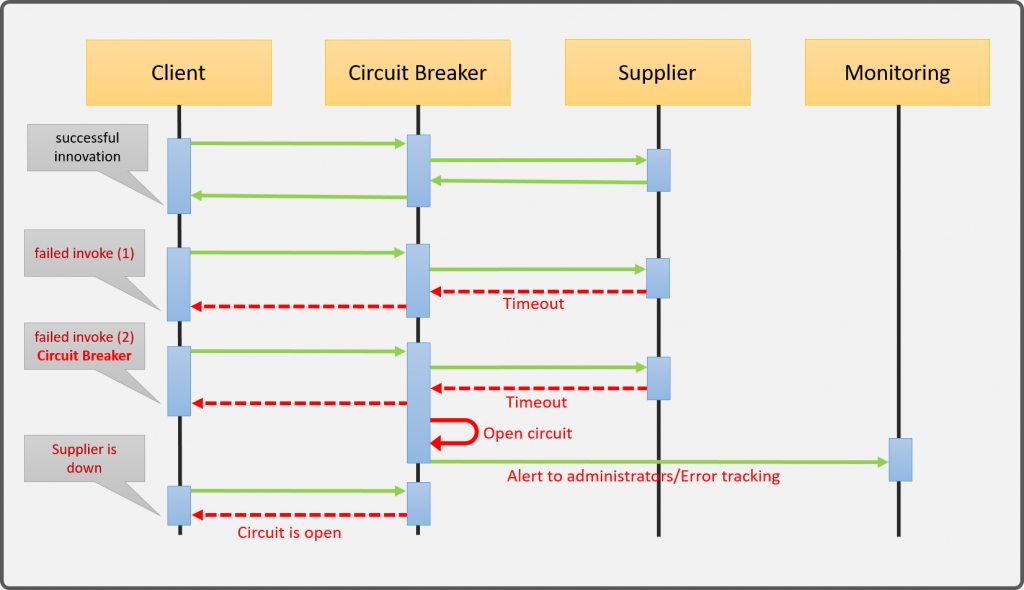
Para establecer el fallback se puede hacer de dos maneras:

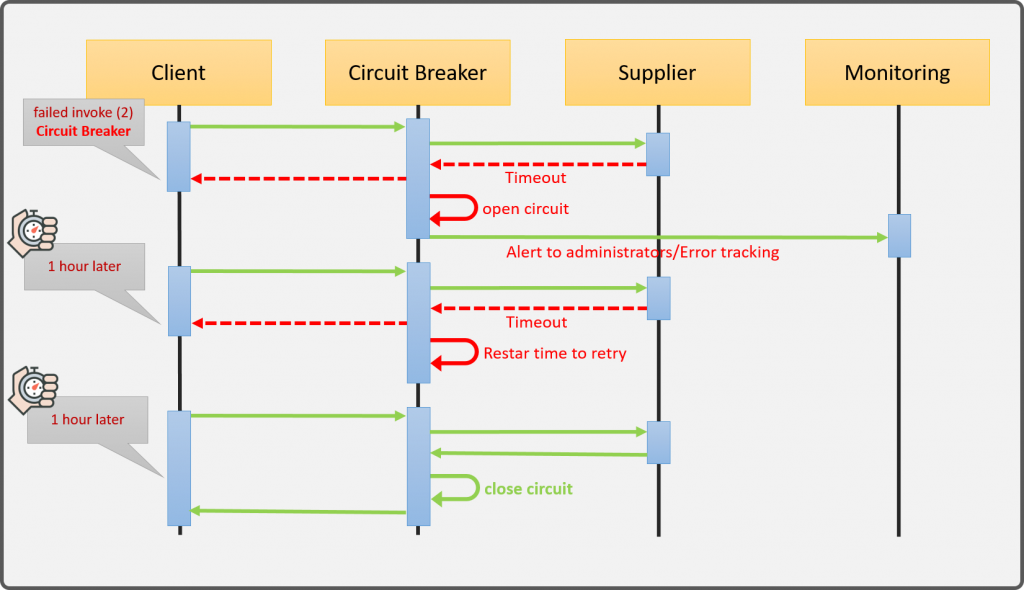
1. Crear un @Service o @Component y dentro indicar que con la anotación @HystrixCommand que método será el encargado de ejrecer de fallback.
2. Indicar en la anotación @FeignClient que clase (que implemente la interfaz a la que llama feign) hará de fallback e implementar estos métodos.

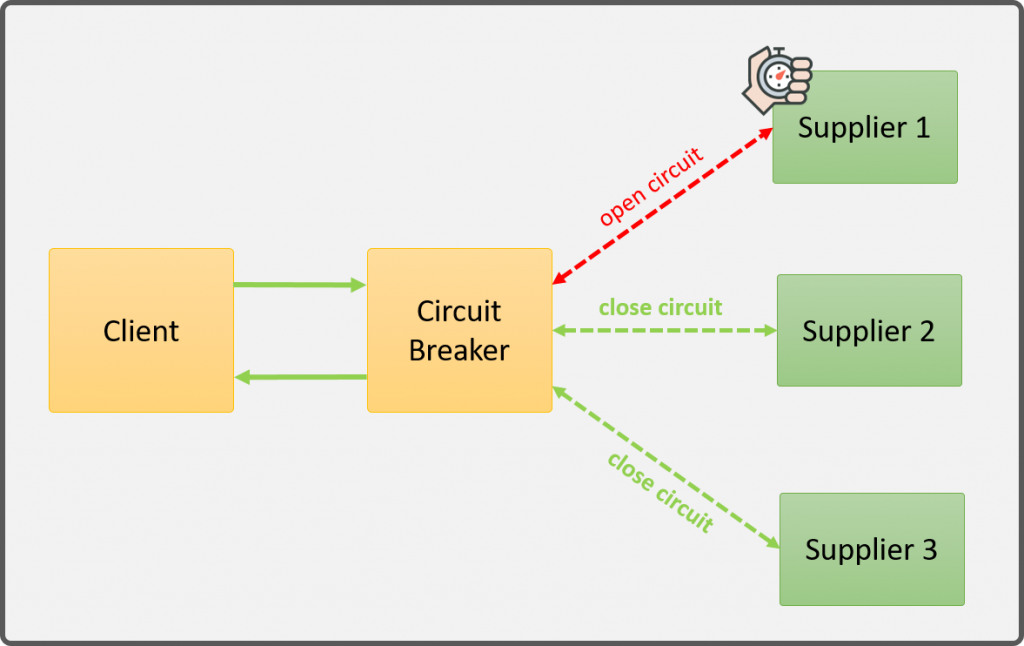
Para activar CircuitBraker, tendremos que indicar en la clase principal de la aplicación la anotación @EnableCircuitBraker.

Patrón CircuitBraker









Si el sistema funciona con normalidad, el circuit breaker permanecerá cerrado (estado **closed**). En el caso de comenzar a ser inestable y se alcance un limite de fallos, el circuito se abrirá para prevenir más errores (estado open). Cada cierto intervalo de tiempo, se pasa al estado **half open** donde Hystrix se encarga de enviar una primera petición para comprobar la disponibilidad del sistema, pasando de nuevo al estado closed (si el funcionamiento vuelve a ser el correcto) u open (si la inestabilidad del sistema continua).

Para habilitar Hystrix, podemos hacerlo desde

feign.hystrix.enabled=true

Feign encapsulará todos los métodos necesarios para aplicar el patrón de circuit braker.

Hystrix Fallbacks

Para Feign, el termino de fallback en Feign significa un trozo de código que se ejecutará cuando el circuito esté abierto o se produzca un error. Para Habilitarlo, indicaremos en la propiedad fallback que clase se hará cargo de ejecutar el código.

Spring marcará los beans de feign e hystrix como primary. Esto puede ser un problema ya que si usamos Autowired, podemos estár usando el fallback en vez del feignclient. Para esto, usaremos la propiedad primary a falso para aquellas implementaciones que no nos interece que sean inyectadas automáticamente.

Interceptors

Feign permite también agregar Interceptores para cada llamada del cliente. Para esto, basta con indicar en la clase de configuración que se va a crear el vean de tipo interceptor.

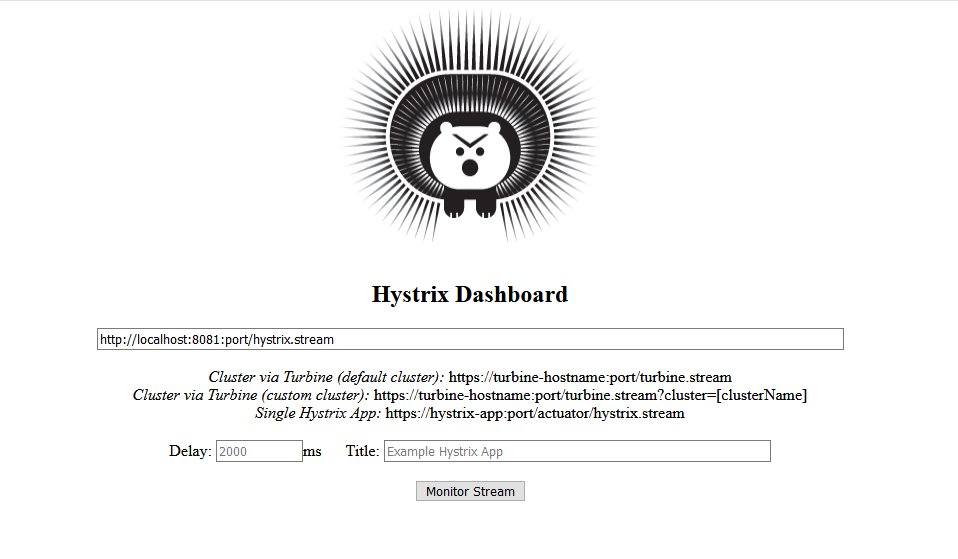
Manejo de errores

Por defecto, los errores de Feign son gestionados por Errordecoder y siempre lanzan una excepción FeignExpetion. Sin embargo, se pueden gestionar los errores como mas nos interese.

Dashboard

Existe un dashboard para poder monitorizar como funciona hystrix. Agregando las dependencias necesarias. Una vez agregada, simplemente tenemos que a colocar la anotación @EnableHystrixDashboard.

Cuando se arranca la aplicación, estas dependencias crea una interfaz en http://URL:PORT/hystrix



Aquí indicaremos el stream que queremos monitorizar (http://localhost:8081:port/hystrix.stream)

