Funzionamento OpenFeign

Dipendenza

```
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-openfeign</artifactId>
</dependency>
```

Tramite la dipendenza di OpenFeign possiamo crearci dei client REST dichiarativi che utilizzano le JAX-RS (Java API for RESTful Web Services) oppure le annotazioni di spring MVC.

Grazie a questa dipendenza possiamo utilizzare un annotazione (@EnableFeignClients) da inserire sul main dell'applicazione.

La precedente annotazione fa un setup di tutte le configurazioni di default necessarie che serve ai FeignClient per funzionare e scansiona automaticamente tutte quelle classi (interfacce) annotate come @FeignClient

L'annotazione @FeignClient permette la dichiarazione di N espressioni all'interno delle parentesi tonde, maggiormente usate sono:

- Value/name : identifica quel client (deve combaciare col nome dell'app sul discovery)
- url : Url su cui andra' a fare la richiesta
- configuration: Eventuale classe di configurazione per quel client
- fallback: richiama una classe in fallback per un determinato evento (e.g. Circuit Breaker)

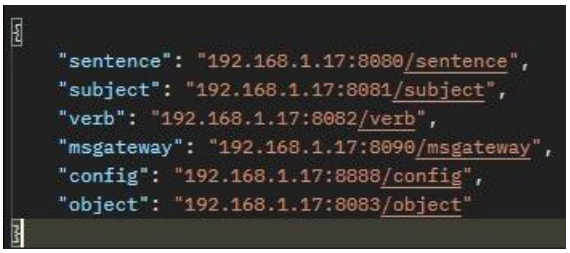
In questo progetto, abbiamo optato per non hard-codare l'url direttamente nell'espressione ma abbiamo seguito il metodo di seguito spiegato.

Seguendo il link** si verra' reindirizzati ad una repository di GitHub dove vi sara' il codice del metodo sopracitato.

Viene creata una classe annotata come @Configuration all'interno di un package config, l'iniezione della dipendenza EurekaClient permette - invece - di poter eseguire diversi metodi per ricevere le istanze registrate (e non solo) dal Discovery.

Successivamente viene creato un bean @PostConstruct che crea un HashMap nel quale vi saranno registrati come KEY il nome dell'applicazione registrata all'interno del Discovery e come value invece l'URL composto di quel microservizio.

Si ha una cosa del genere:



Cosi facendo abbiamo un Bean che ad ogni aggiornamento delle applicazioni su discovery, aggiunge o rimuove on-the-fly dall'hashmap il relativo microservizio così da avere sempre un hashmap aggiornato con i microservizi attivi.

Così facendo si puo', tramite SpEL (Spring Expression Language), iniettare l'url sul @FeignClient seguendo questa sintassi:

```
url =
"#{eurekaInstanceReceiver.getEurekaInfoByServiceName().#this.get('${eureka.service-name.object}')}"
```

In pratica SpEL permette l'accesso a dei Bean in base al nome del metodo ed inoltre anche l'accesso all'oggetto che va a creare, essendo un HashMap<X,Y> utilizzeremo un #this. (referenzia l'istanziamento di quell'oggetto).get("nome del servizio").

Infine prendiamo il nome del servizio dallo yaml:

```
eureka:
  service-name :
    subject: subject
    verb: verb
    object: object
```

Circuit Breaker

Feign ha un CircuitBreaker integrato che automaticamente capisce quando un microservizio è su oppure no.

Tramite l'aggiunta di un espressione *fallback* all'annotazione @FeignClient possiamo indirizzare ad una classe specifica che compiera' determinate azioni (e.g. l'ovverride della stringa in response con una "Service not available").

L'unica cosa da apporre nello yaml sono le seguenti diciture, che permettono l'abilitazione del CircuitBreaker:

```
feign:
  circuitbreaker:
    enabled: on
    alphanumeric-ids:
      enabled: true
```

Link**:
https://github.com/ddbdev/Sentence-Project/blob/master/ms-sentence/src/main/java/it/cgmconsulting/mssentence/config/EurekaInstanceReceiver.java