

Relazione WebClient

Introduzione

WebClient[1] è un'interfaccia messa a disposizione dalla libreria Spring WebFlux[2], che permette di effettuare chiamate asincrone a servizi REST, che quindi si differenzia rispetto a RestTemplate in quanto quest'ultima permette di effettuare solamente chiamate sincrone. Web Flux nello specifico è uno stack che consente di realizzare applicazioni web reattive.

Implementazione

L'implementazione parte con il richiamare all'interno del file *pom.xml* le dipendenze necessarie:

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-webflux</artifactId>
</dependency>
```

Dopodichè si utilizzano i builder preconfigurati di spring boot per consentire l'utilizzo di *WebClient* all'interno del progetto.

```
@Configuration
public class WebClientConfig {

    @Bean
    public WebClient webClient(){
        return WebClient.builder().build();
    }
}
```

Questo è un caso pratico dove viene effettuata un'operazione di tipo *GET* su un servizio *REST*.

```
public class Service{

    @Autowired WebClient webClient; 1)

    public Mono<String> webClientGet(Uri uri) { 2)
        return webClient 3)
            .get() 4)
            .uri(uri) 5)
            .retrieve() 6)
            .bodyToMono(String.class); 7)
    }
}
```

Nello specifico accade:

1. Viene effettuato la *dependency injection* all'interno del codice
2. viene definito un metodo dove viene passato in input un parametro di tipo *URI*, che corrisponde all'indirizzo dove viene effettuata la chiamata *GET*, e in *output* un oggetto *Mono* di tipo *String* che rappresenta uno *stream reattivo*.
3. Dopodichè vengono effettuate in cascata differenti operazioni partendo da *webClient* con l'aiuto del pattern *builder*
4. viene definito il tipo di chiamata *Http* che in questo caso é *GET*
5. viene passato in input l'indirizzo *URI*
6. viene applicato un metodo che precede il tipo di risposta che si vuole ottenere successivamente, semplicemente passando tutti i dati, a differenza della sua alternativa *exchange()* con la quale si possono controllare meglio gli oggetti di response, gli headers o le eccezioni
7. attraverso questo metodo indichiamo che il metodo *GET* restituirà un oggetto di tipo stringa.

Riferimenti

1. <https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/web/reactive/function/client/WebClient.html>
2. <https://docs.spring.io/spring-framework/docs/current/reference/html/web-reactive.html>

Autori

Vito Malato, Michelangelo Di Gennaro, Olga Chistiakova