WebClient

## WebClient là gì ?

* WebClient đơn giản là một interface đại diện cho một entry point để thực hiện các request.
* Nó được tạo ra như là một phần của module Spring Web Reactive và thay thế cho RestTemplate.
* Interface này chỉ có duy nhất một triển khai đó là class DefaultWebClient.

## Cấu hình thư viện

### Sử dụng Maven

Thêm các thư viện sau vào trong file *porm.xml*

*<dependency>*

*<groupId>org.springframework.boot</groupId>*

*<artifactId>spring-boot-starter-webflux</artifactId>*

*</dependency>*

### Sử dụng Gradle

Thêm thư viện vào trong file *build.gradle*

*dependencies {*

*compile 'org.springframework.boot:spring-boot-starter-webflux'*

*}*

## Sử dụng WebClient

Sử dụng thông qua 3 bước:

* Tạo ra một thể hiện
* Tạo ra một request
* Xử lý một response

### Tạo một thể hiện của WebClient

* Tạo ra một đối tượng WebClient với cấu hình mặc định:

*WebClient client = WebClient.create();*

* Tạo một đối tượng WebClient với tham số URI:

*WebClient client = WebClient.create("*[*http://localhost:8080*](http://localhost:8080)*");*

* Tạo ra một đối tượng WebClient bằng cách sử dụng class DefaultWebClientBuilder:

*WebClient client = WebClient.builder()*

*.baseUrl("http://localhost:8080")*

*.defaultCookie("cookieKey", "cookieValue")*

*.defaultHeader(HttpHeaders.CONTENT\_TYPE, MediaType.APPLICATION\_JSON\_VALUE)*

*.defaultUriVariables(Collections.singletonMap("url", "http://localhost:8080"))*

*.build();*

* Tạo một thể hiện của WebClient với Timeout

*HttpClient httpClient = HttpClient.create()*

*.option(ChannelOption.CONNECT\_TIMEOUT\_MILLIS, 5000)*

*.responseTimeout(Duration.ofMillis(5000))*

*.doOnConnected(conn ->*

*conn.addHandlerLast(new ReadTimeoutHandler(5000, TimeUnit.MILLISECONDS))*

*.addHandlerLast(new WriteTimeoutHandler(5000, TimeUnit.MILLISECONDS)));*

*WebClient client = WebClient.builder()*

*.clientConnector(new ReactorClientHttpConnector(httpClient))*

*.build();*

Thời gian timeout mặc định là 30s, chúng ta có thể set lại thông qua *ChannelOption.CONNECT\_TIMEOUT\_MILLIS* hoặc đọc và ghi dữ liệu timeout qua *ReadTimeoutHandler, WriteTimeoutHandler*

### Chuẩn bị một request

* Đầu tiên, chúng ta sẽ cần chỉ định ra một phương thức HTTP của một request bằng cách gọi phương thức HttpMethod:

*UriSpec<RequestBodySpec> request = client.method(HttpMethod.POST);*

Hoặc đơn giản hơn:

*UriSpec<RequestBodySpec> request = client.post();*

* Cung cấp đường dẫn URL:

*RequestBodySpec bodySpec = uriSpec.uri("/resource");*

OR

*RequestBodySpec bodySpec = uriSpec.uri(uriBuilder -> uriBuilder.pathSegment("/resource").build());*

OR

*RequestBodySpec bodySpec = uriSpec.uri(URI.create("/resource"));*

* Sau đó cài đặt phần response body, content type, header, …

*RequestHeadersSpec<?> headersSpec = bodySpec.bodyValue("data");*

OR

*RequestHeadersSpec<?> headersSpec = bodySpec.body( Mono.just(new Foo("name")), Foo.class);*

OR

*RequestHeadersSpec<?> headersSpec = bodySpec.body( BodyInserters.fromValue("data"));*

Hoặc sử dụng thông qua Reactor instance:

*RequestHeadersSpec headersSpec = bodySpec.body( BodyInserters.fromPublisher(Mono.just("data")), String.class);*

*LinkedMultiValueMap map = new LinkedMultiValueMap();*

*map.add("key1", "value1");*

*map.add("key2", "value2");*

*RequestHeadersSpec<?> headersSpec = bodySpec.body(*

*BodyInserters.fromMultipartData(map));*

* Sau khi cài đặt xong phần body, chúng ta có thể cài đặt phần header, cookies, …

*ResponseSpec responseSpec = headersSpec.header(HttpHeaders.CONTENT\_TYPE, MediaType.APPLICATION\_JSON\_VALUE)*

*.accept(MediaType.APPLICATION\_JSON, MediaType.APPLICATION\_XML)*

*.acceptCharset(StandardCharsets.UTF\_8)*

*.ifNoneMatch("\*")*

*.ifModifiedSince(ZonedDateTime.now())*

*.retrieve();*

### Nhận về một response

Nhận về response thông qua 2 phương thức là *exchange()* hoặc *retrieve()*

*Mono<String> response = headersSpec.exchangeToMono(response -> {*

*if (response.statusCode()*

*.equals(HttpStatus.OK)) {*

*return response.bodyToMono(String.class);*

*} else if (response.statusCode()*

*.is4xxClientError()) {*

*return Mono.just("Error response");*

*} else {*

*return response.createException()*

*.flatMap(Mono::error);*

*}*

*});*

*Mono<String> response = headersSpec.retrieve()*

*.bodyToMono(String.class);*