Building an API Client with WebClient

Esteban Herrera
Author | Developer | Consultant

@eh3rrera   eherrera.net

Overview

WebClient

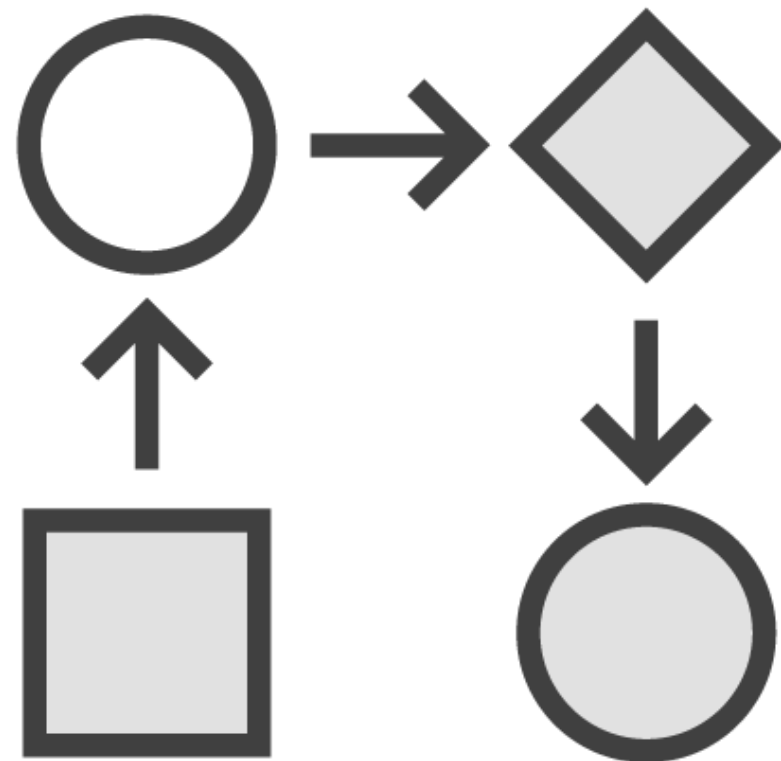Setting up the project

Building the API client

WebClient

RestTemplate Class

```java
RestTemplate template = new RestTemplate();
Product product = template.getForObject(
    "http://localhost:8080/products/1",
    Product.class
);
```

Three Steps to Working with WebClient

Creating a WebClient instance

Preparing the request

Reading the response

Creating a WebClient Instance

```java
WebClient webClient = WebClient.create();


WebClient webClient = WebClient.create("https://localhost:8080");


WebClient webClient = WebClient.builder()
        .baseUrl("https://localhost:8080")
        .defaultHeader(HttpHeaders.USER_AGENT, "Spring 5")
        .build();
```

Creating a WebClient Instance

```
WebClient webClient8081 = webClient.mutate()
        .baseUrl("https://localhost:8081")
        .build();
```

Preparing the Request

```java
webClient
  .get()  // WebClient.RequestHeadersUriSpec
  .uri("/products/{id}", id) // WebClient.UriSpec
  .accept(MediaType.APPLICATION_JSON); // WebClient.RequestHeadersSpec

webClient
  .post()  // WebClient.RequestBodyUriSpec (RequestBodySpec)
  .uri("/products") // WebClient.UriSpec
  .contentType(MediaType.APPLICATION_JSON); // WebClient.RequestHeadersSpec
```

Preparing the (POST or PUT) Request

```java
Mono<Product> productPublisher = ...
Flux<Product> productPublisher = ...

webClient
  .post()
  .uri("/products")
  .contentType(MediaType.APPLICATION_JSON)
  .body(productPublisher, Product.class);
```

Preparing the (POST or PUT) Request

```java
Product product = ...

webClient
  .post()
  .uri("/products")
  .contentType(MediaType.APPLICATION_JSON)
  .bodyValue(product);
```

Preparing the (Form Data) Request

```java
webClient
  .post()
  .uri("/products")
  .contentType(MediaType.APPLICATION_JSON)
  .body(BodyInserters.fromFormData("field1", "val1").with("field2", "val2"));
```

Preparing the (Form Data) Request

```
webClient
  .post()
  .uri("/products")
  .contentType(MediaType.APPLICATION_JSON)
  .body(BodyInserters.fromMultipartData("field1", "val1").with("file", file));
```

Preparing the (Form Data) Request

```java
MultiValueMap<String, String> form =
            new LinkedMultiValueMap<String, String>();
form.set("field1", "val1");

webClient
  .post()
  .uri("/products")
  .contentType(MediaType.APPLICATION_JSON)
  .bodyValue(form);
```

Preparing the (Form Data) Request

```java
MultipartBodyBuilder builder = new MultipartBodyBuilder();
builder.part("field", "val");
builder.part("file", new FileSystemResource("image.jpg"));
MultiValueMap<String, HttpEntity<?>> parts = builder.build();

webClient
  .post()
  .uri("/products")
  .contentType(MediaType.APPLICATION_JSON)
  .bodyValue(parts);
```

Reading the Response (retrieve)

```java
Mono<Product> mono =
  webClient
    .get()
    .uri("/products/{id}", id)
    .retrieve()
    .bodyToMono(Product.class);
```

Reading the Response (retrieve)

```java
Mono<Void> mono =
    webClient
        .delete()
        .uri("/products")
        .retrieve()
        .bodyToMono(Void.class);
```

Reading the Response (retrieve)

```java
Flux<Product> flux =
  webClient
    .get()
    .uri("/products")
    .retrieve()
    .bodyToFlux(Product.class);
```

Reading the Response (retrieve)

```java
Flux<Product> flux =
  webClient
    .get()
    .uri("/products")
    .retrieve()
    .onStatus(HttpStatus::is4xxClientError, response -> ...)
    .bodyToFlux(Product.class);
```

Reading the Response (exchange)

```java
Flux<Product> flux =
  webClient
    .get()
    .uri("/products")
    .exchangeToFlux(response -> response.bodyToFlux(Product.class));
```

Reading the Response (exchange)

```java
Mono<Product> mono =
  webClient
    .get()
    .uri("/products/{id}", id)
    .exchangeToMono(response -> response.bodyToMono(Product.class));
```

Reading the Response (exchange)

```java
Mono<ResponseEntity<Product>> mono =
  webClient
    .get()
    .uri("/products/{id}", id)
    .exchangeToMono(response -> response.toEntity(Product.class));
```

Reading the Response (exchange)

```java
Mono<ResponseEntity<java.util.List<Product>>> mono =
  webClient
    .get()
    .uri("/products/{id}", id)
    .exchangeToMono(response -> response.toEntityList(Product.class));
```

Things to Remember

WebClient
- **HTTP reactive client**
- **Asynchronous but also synchronous**
- **Also works inside Spring WebFlux**

Three steps to work with WebClient
- **Creation**
  - **create()/builder()**
- **Preparation**
  - **get()/post()**
  - **uri()/contentType()/body()**
- **Response**
  - **retrieve()**
  - **exchangeToMono()**
  - **exchangeToFlux()**