Alright, let's jump right in and continue exploring the WebClient, from the first part of the lesson.

In the first example, we used *retrieve()* as the simplest way to get response body and decode it.

But, for more flexibility, we can also use *exchange* to access the full *ClientResponse*.

Let's add another test that creates a new *Privilege* by simply doing a POST. Then, let's use the *exchange* API and get just the status code from the response.

Finally, like last time, let's subscribe to the result and print it out.

```
JAVA
@Test
public void createNewPrivilege() throws InterruptedException {
    Mono<HttpStatus> result = webClient.post()
      .uri("/privileges")
      .syncBody(getAPrivilegeForPost())
      .exchange()
      .map(response ->response.statusCode());

    result.subscribe(httpStatus -> System.out.println("Http Status code:" + httpStatus.value()));
}

private Privilege getAPrivilegeForPost() {
    Privilege privilege = new Privilege();
    privilege.setName(RandomStringUtils.random(5, 'A', 'Z'));
    return privilege;
}
```

And we're done: we created a new resource, extracted the status code from the response and printed it.

2.4. *WebTestClient*

Finally, we'll have a look at *WebTestClient*, the test-focused version of the *WebClient*.

We can also use this class for testing WebFlux endpoints.

Let's define a test client bean in our test class and write a quick test.

This test retrieves a *Privilege* from the live server and asserts that correct privilege is retrieved.

JAVA

```java
@Test
public void whenTheGivenPrivilegeIsRetrived_thenItIsRetrievedCorrecly() {
    webTestClient.get()
      .uri("/privileges/1").accept(MediaType.APPLICATION_JSON)
      .exchange()
      .expectBody(Privilege.class).isEqualTo(getPrivilege());
}
```

The use of the *WebTestClient* class is very similar to *WebClient*. It delegates most of the work to an internal *WebClient* instance focused on providing a test context.

The client for testing can be bound to a real server or work with specific controllers or functions.

3. Resources

- Spring Framework Reference - WebClient

- Baeldung - Spring 5 WebClient