УДК 004

Тестирование кода с использованием Spring WebClient

Еровлева Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье будут рассмотрены принципы тестирования написанного кода с помощью Spring WebClient. Работа будет происходить в среде разработки IntelijIdea.
Ключевые слова: WebClient, Spring, Spring boot

Testing your code using Spring WebClient

Erovleva Regina Viktorovna
Sholom-Aleichem Priamursky State University
Student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article will cover the principles of testing written code using the Spring WebClient. The work will take place in the IntelijIdea development environment.
Keywords: WebClient, Spring, Spring boot

WebClient - это новый клиент для отправки веб-запросов, включая вызовы REST. Он входит в состав реактивного фреймворка и, таким образом, поддерживает асинхронную связь.

Цель данной работы – проверить, как работает WebClient и протестировать с его помощью написанный код.

В своей работе А.Б.Джемалетдинов, А.А.Шевченко рассмотрели вопросы создания тестов для Spring Boot mvc контроллеров [1]. В.И.Зарайский провел обзор на разработку модуля автоматизации работы с конференциями в кафедральном приложении [2]. Р.И.Ибраимов продемонстрировал процесс создания Docker-образа для Spring Boot проекта и развернул его на платформе AWS EC2[3]. Е.О.Кабардинский, А.Г.Ивашко провели сравнительный анализ сервисных шин предприятия, а так же

сравнили некоторые ESB, одна из которых Spring Boot [4]. Так же Р.И.Ибраимов, А.Р.Зайчик, Н.С.Минзатров разработали генеалогическое дерево на языке Java с использованием фреймворка Spring Boot и библиотеки gedcom4j[5].

Если нужно написать тесты для части логики, одним из вариантов может быть «издевательство» над файлом WebClient. Однако у этого подхода есть два недостатка. Во-первых, получаются очень некрасивые тесты. Нужно будет написать несколько макетов, чтобы правильно смоделировать весь свободный API. Вторая проблема заключается в том, чтобы вызвать, WebClient.

К примеру, необходимо проверить, что вызывается API: http://example.org/api/sum?value1=3&value2=5.

Есть несколько способов WebClient вызова этого API. Можно было бы построить полный путь самостоятельно и передать его в uri() или использовать UriBuilder. Но даже если использовать UriBuilder можно было бы добавить параметры запроса напрямую, либо использовать переменные шаблона.

Второе решение - имитировать сам API. Если использовать RestTemplate с расширением от MockRestServiceServer класса, то этот класс позволяет писать ожидания о самом запросе, а не о том, как он создается.

Теперь настроим MockWebServer в тесте (рис.1).

```
class MathServiceTest {
    private MathService service;
    private MockWebServer server;

    @BeforeEach
    void setUp() throws IOException {
        server = new MockWebServer();
        server.start();
        String rootUrl = server.url("/api/").toString();
        service = new MathService(WebClient.create(rootUrl));
    }


}
```

Рисунок 1 – Настройка класса

В данном примере используется JUnit5, поскольку это среда тестирования по умолчанию, которая поставляется с Spring. Самый простой способ настроить MockWebServer - воссоздавать его перед каждым тестом и уничтожать после каждого теста.

Итак, в этом setUp() методе настраиваем MockWebServer и передаем URL-адрес объекту WebClient, чтобы он использовал этот URL-адрес в качестве базового URL-адреса для вызовов API.

Кроме того, также напишем tearDown() способ выключения сервера (рис.2).

```
@AfterEach
void tearDown() throws IOException {
    server.shutdown();
}
```

Рисунок 2 – Метод отключения

Теперь попробуем написать тест (рис.3).

```
@Test
void sum_usesSumAPI() throws InterruptedException {
    MockResponse response = new MockResponse();
    server.enqueue(response);
    StepVerifier
        .create(service.sum(5, 3))

        .verifyComplete();
    RecordedRequest request = server.takeRequest();
    assertThat(request.getMethod()).isEqualTo("GET");
    assertThat(request.getPath()).startsWith("/api/sum");
    assertThat(request.getRequestUrl().queryParameter("value1")).isEqualTo("5");
    assertThat(request.getRequestUrl().queryParameter("value3")).isEqualTo("3");
}
```

Рисунок 3 – Написание теста

Если необходимо отправить ответ, то можно сделать это путем вызова addHeader() и setBody() метода MockResponse. Это позволяет написать утверждение в тесте на основе возвращаемого значения.

Если нужно отправить более сложный ответ, то можно использовать отдельный файл и загружать его следующим образом (рис.4).

```
Path responseFile = Paths.get(getClass().getResource("response.json").toURI());
String responseBody = Files.readString(responseFile, defaultCharset());
MockResponse response = new MockResponse()
    .addHeader("Content-Type", "application/json")
    .setBody(responseBody);
```

Рисунок 4 – Отправка json

Независимо от того, какой тип клиента используется, следует тестировать, какой API вызывается, а не какие методы используются. Это применимо при использовании фреймворка, который дает несколько вариантов достижения одного и того же.

В данной работе были рассмотрены примеры тестирования кода с использованием Spring WebClient.

Библиографический список

1. Джемалетдинов А.Б., Шевченко А.А. Spring boot: создание тестов для spring mvc контроллеров // Информационно-компьютерные технологии в экономике, образовании и социальной сфере. 2017. №4(18). С. 104-111.
2. Зарайский В.И. Разработка модуля автоматизации работы с конференциями в кафедральном приложении // Вестник Ульяновского государственного технического университета. 2019. №3. С. 74-82.
3. Ибраимов Р.И. Развертывание spring приложения с помощью сервиса aws ec2 и docker-контейнеров // Информационно-компьютерные технологии в экономике, образовании и социальной сфере. 2020. №1(27). С. 138-147.
4. Кабардинский Е.О., Ивашко А.Г. Сравнительный анализ сервисных шин предприятия (esb) // Математическое и информационное моделирование. 2017. №10. С. 177-185.
5. Ибраимов Р.И., Зайчик А.Р., Минзатров Н.С. Разработка генеалогического дерева средствами фреймвока spring boot // Информационно-компьютерные технологии в экономике, образовании и социальной сфере. 2017. №4(18). С. 18-23.