SELENIUM WEBDRIVER КАК ИНСТРУМЕНТ ФУНКЦИОНАЛЬНОГО ТЕСТИРОВАНИЯ ВЕБ-ПРИЛОЖЕНИЙ

Альтшулер И.О.
студент 4 курса БрГУ имени А.С. Пушкина,
г. Брест, Республика Беларусь
Научный руководитель – Кондратюк А.П., старший преподаватель

Тестирование является в настоящее время быстро развивающейся отраслью IT и одним из наиболее устоявшихся способов повышения качества разработки программного обеспечения (далее – ПО). В широком значении под тестированием ПО понимают процесс его исследования с целью получения информации о качестве продукта. Тестирование занимает важное место в жизненном цикле ПО и включает активности по планированию работ, проектированию тестов, выполнению тестирования и анализу полученных результатов. Следует отметить, что современный этап развития тестирования характеризуется глубокой интеграцией с процессом разработки, широким использованием автоматизации, большим набором технологий. Автоматизированное тестирование представляет собой набор техник, подходов и инструментальных средств, позволяющих исключить человека из выполнения отдельных задач в процессе тестирования [1].

Веб-приложения являются одним из наиболее распространенных видов ПО. К основным базовым проверкам веб-приложений относится функциональное тестирование, включающее разработку и выполнение тестов по оценке функций веб-приложений. Практически все из них имеют формы с полями для ввода пользователем определенной информации (формы регистрации, авторизации, изменения пароля, отправки отзывов, платежные данные и др.). Они должны соответствовать определенным требованиям. При функциональном тестировании проверяется работоспособность и взаимодействие элементов веб-приложений. Установлено, что автоматизация наиболее применима при тестировании валидационных форм, базовых операций, часто используемой функциональности приложений и др.

Вышесказанное позволяет сделать вывод об актуальности проблемы автоматизированного функционального тестирования веб-приложений, которая является предметом нашего изучения в рамках дипломного проекта. В связи с этим, цель данной статьи – раскрыть возможности Selenium WebDriver как инструмента автоматизированного функционального тестирования веб-приложений. WebDriver вместе с Selenium IDE, Selenium Grid образует фреймворк для автоматизированного тестирования веб-приложений.

В ходе дипломного проектирования с целью осуществления функционального тестирования веб-сайтов, например, тестирования форм регистрации, наряду с WebDriver нами использовались инструменты рабочего окружения (Git, Maven, Jenkins, IntelliJ IDEA), необходимые для реализации автоматизированного тестирования веб-приложений. Кроме того, работа с WebDriver предполагала его установку для браузеров, настройку среды разработки (создание файловой структуры проекта и базовой конфигурации); создание maven-проекта, подключение библиотеки WebDriver, освоение ее функционала (WebDriver API Core). При написании тестовых сценариев также необходимо было решать проблемы синхронизации с применением возможностей WebDriver API по управлению таймаутами и ожиданиями, а также использовать локаторы и селекторы WebDriver.

Полезной функцией WebDriver является возможность его интеграции с другими инструментами автоматизации. В нашем проекте тестовые сценарии, выполненные с помощью Selenium IDE, экспортировались в WebDriver. Приведем пример автоматизированного теста для формы авторизации на сайте Национальной библиотеки Беларуси.

Наиболее распространенным методом создания тестовых сценариев в Selenium IDE является запись. Затем сценарий выполняется с помощью функции воспроизведения и при необходимости корректируется. Ниже приводится часть тест-кейса по авторизации на сайте с введением невалидных данных в поле «Логин» после экспорта его в WebDriver (с выбором Java JUnit).

```
@Test
public void invaledName()
{ driver.get("https://nlb.by/");
driver.manage().window().setSize(new Dimension(1600, 860));
driver.findElement(By.cssSelector(".header-logo-phone > .col-xs-12 .svg-logo-ani-
mate")).click();
driver.findElement(By.cssSelector(".hide-1280")).click();
driver.findElement(By.name("USER_PASSWORD")).sendKeys("i12345");
driver.findElement(By.name("USER_LOGIN")).sendKeys("InvalidN");
driver.findElement(By.cssSelector(".button_wide")).click();
driver.findElement(By.name("USER_LOGIN")).click(); }
```

Вместе с тем, следует отметить, что одного WebDriver API недостаточно, чтобы протестировать веб-приложение. WebDriver является библиотекой программирования, набором методов, позволяющим автоматизировать поведение приложения в рамках веб-браузера. Тесты же имеют свою структуру, предварительные условия, механизм выполнения и т.п. Тестирование предполагает также сбор и представление результатов (отчетов об ошибках). Кроме того, требуется устанавливать те или иные атрибуты запуска тестов или их наборов. Эти возможности предоставляют unit test фреймворки, которые являются набором Java-методов и сопутствующих аннотаций, специально разработанных для выполнения задач модульного тестирования. Примерами unit test фреймворков являются JUnit и TestNG. В нашем исследовании нашел применение фреймворк TestNG, который является в настоящее время устоявшейся инженерной практикой.

После базовых настроек TestNG позволяет задавать структуру теста, осуществлять проверки, реализуя при этом концепции Page Object и Page Factory. Как подчеркивается в специальной литературе, Page Object один из наиболее полезных и применяемых архитектурных решений в автоматизации. Данный шаблон проектирования помогает инкапсулировать работу с отдельными элементами страницы, что позволяет уменьшить количество кода и его поддержку. К его основным преимуществам можно отнести разделение кода тестов и описания страниц, а также объединение всех действий по работе с веб-страницей в одном месте [2]. Приведем пример (фрагмент кода) использования шаблона Page Object. Создадим класс с описанием страницы логина, которая имеется почти на всех сайтах.

```
public class LoginPage {
By usernameLocator = By.id("username");
By passwordLocator = By.id("passwd");
By loginButtonLocator = By.id("login");
private final WebDriver driver;
public LoginPage(WebDriver driver) {
        this.driver = driver;
        if (!"Login".equals(driver.getTitle())) {
                throw new IllegalStateException("This is not the login page");
    } } …}
```

Таким образом, Selenium поддерживает автоматизацию всех основных браузеров на рынке с помощью WebDriver. WebDriver – это API и протокол, определяющий не зависящий от языка интерфейс для управления поведением веб-браузеров [3]. Selenium

Webdriver является инструментом для автоматизации тестирования веб-приложений. Код, написанный средствами WebDriver API с помощью фреймворка TestNG, превращается в автоматизированный тестовый сценарий. Важнейшее преимущество Seleni-um WebDriver как инструмента и библиотеки методов автоматизации тестирования веб-приложений заключается в его интеграции с другими инструментами автоматизации.

Литература

1. Куликов, С.С. Тестирование программного обеспечения. Базовый курс / С. С. Куликов. – Минск: Четыре четверти, 2017. – 312 с.

2. Selenium Webdriver [Электронный ресурс]. – Режим доступа: https://drive.google.com/file/d/0B7TBmsv_w76nQ1FfQVdyWWRXYk0/view. – Дата доступа: 02.04.2021.

3. Getting started with WebDriver [Электронный ресурс]. – Режим доступа: https://www.selenium.dev/documentation/en/getting_started_with_webdriver/. – Дата доступа: 03.04.2021.

РАЗРАБОТКА ПРОГРАММЫ «SIMBEVIR»

Бачун Р.В.

учащийся 2 курса Оршанского колледжа ВГУ имени П.М. Машерова,
г. Орша, Республика Беларусь

Научный руководитель – Алейников М.А., магистр педагогических наук

Пандемия коронавируса показала беспомощность и неподготовленность людей в подобных ситуациях. Хоть и научно-технический прогресс стремительно развивается в последние годы, человечеству все равно нужно время на исследование вируса, создание и тестирование вакцины. При чем за это время могут умереть миллионы людей, или чего хуже – появление другого штамма, на исследование которого так же нужно время.

Небольшая информированность людей или же нехватка информации – еще большая проблема. Из-за игнорирования мер безопасности многие люди, не подозревая этого могут быть носителями вируса, а как показывает статистика, в среднем, ежедневно человек контактирует как минимум с 15 людьми. Не сложно предположить с какой скоростью таким образом распространяется вирус.

Цель исследования – создать приложение и реализовать математическую модель для предсказания появления новых вирусов или их штаммов, а также реализовать математическую модель для получения информации о вирусе по их признакам или на основе генома.

Материал и методы. Для реализации программного продукта был выбран язык программирования Python, а также математические и графические библиотеки, такие как: NumPy, PyQt и др. Реализация модели для поиска закономерностей в геноме вирусов. Реализация алгоритма для генерации генома вирусов.

Результаты и их обсуждение. В результате исследования разработана математическая модель для поиска закономерностей в геноме вирусов. Вирус – неклеточный инфекционный агент, который может воспроизводиться только внутри клеток, поражают все типы организмов, от растений и животных до бактерий и архей. Нейронная сеть – математическая модель, а также её программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей – сетей нервных клеток живого организма. На данный момент нейронные сети используются в многочисленных областях машинного обучения и решают проблемы различной сложности. В основе перцептрона (простейшая модель нейронной сети) лежит математическая модель восприятия информации мозгом. Разные исследователи по-разному его определяют. В самом общем своем виде (как его описывал Розенблатт) он представляет