UDC 004.075

RESEARCH AND ANALYSIS OF APPLICATION OF AUTOMATED TESTING IN WEB APPLICATIONS

Antonova A.[1], Shanovskiy B.[2]

[1,2]Odessa National Academy of Food Technologies, Odessa
E-mail: [1]allaantonova62@gmail.com, [2]shanovskyi.b@gmail.com

Анотація: У статті розглядаються питання технології та методології для автоматизації процесу тестування веб-додатків. В даний час розробники і фахівці автоматизації переходять на популярні розвиваються середовища розробки. В процесі розробки і тестування програмного забезпечення в команді працюють не тільки розробники і тестувальники, і бізнес-фахівці, які визначають набір змін при випуску нових версій програмних продуктів. При оновленні продукту невід'ємною частиною випуску є створення автоматизованого тестування. Засоби забезпечення автоматизації - це використання об'єктно-орієнтованого підходу в реалізації проекту, стандартний набір інструментів для забезпечення процесів складання і налагодження програм, підключення допоміжних бібліотек для застосовується мови програмування (C++, C#, Java і т.д.); перевірки версій системи, при зберіганні проекту автоматизації та програмного продукту в одному місці розташування.

Основна мета даного дослідження полягає у виявленні та подальшому аналізі слабких і сильних сторін методів автоматизації тестування веб-додатків, які в даний час знаходяться в стадії бурхливого розвитку. Для виконання самого дослідження була виконана інформаційна опрацювання (статті, доповіді, самі засоби автоматизації) етапів розвитку даної теми. В результаті авторами зроблені висновки про подальші напрямки розвитку в тестуванні веб-додатків, виявлені переваги існуючих рішень і визначені області, які недостатньо опрацьовані.

Abstract: The article discusses the issues of technology and methodology for automating the process of testing Web applications. Currently, developers and automation professionals are moving to popular developing development environments. In the process of developing and testing software in the team, not only developers and testers work, but also business professionals who define a set of changes when new versions of software products are released. When you update the product, an integral part of the release is the creation of automated testing. Means of providing automation is the use of an object-oriented approach to project implementation, a standard set of tools for providing processes for building and debugging programs, connecting auxiliary libraries for the programming language used (C++, C#, Java, etc.); checking the system versions, storing the automation project and the software product in one location.

The main goal of this study is to identify and then analyze the weaknesses and strengths of methods for automating the testing of web applications, which are currently in the stage of rapid development. To carry out the research itself, information work was carried out (articles, reports, automation tools themselves) of the development stages of this topic. As a result, the authors made conclusions about further directions in the development of web application testing, identified the merits of existing solutions and identified areas that are not sufficiently developed.

Ключові слова: Тести, програмне забеспечення, тестирование, автоматичне тестування, фронт-энд, Платформа JS, Selenium, веб драйвер.

Keywords: Tests, software, testing, Web Application, automation testing, Front-End, PhantomJS, Selenium, Web Driver

Introduction

If the developer writes the code, then he certainly tests it. If it's a function, the developer can call it with different arguments, and see what it returns. If the developer drilled the site, he opens it in the browser, checks the links, buttons, and checks if everything is done in accordance with the technical assignment. This process is called manual testing - a person checks the work of the program.

If you count all desktop and mobile browsers, it turns out that you need to support more than 15 browsers, and each of them has a lot of versions significantly different from each other. Until recently, we tested manually. The tester just passed all the states of the site pages in all browsers and checked to see if it works as intended. This led to the fact that the process of release

was very delayed. Up to the point that development and testing took approximately the same time. Many bugs have escaped the eyes of the tester or were detected at later stages of operation.

There was a question: Can this task be shifted to the shoulders of robots? Usually it is possible, and this task is performed by automated testing.

Solutions variants

To test a web application (site), you need to simulate the operation of the browser. There are different approaches for this.

The earliest and the simplest tools in the history of Internet development can only send HTTP requests to the server and analyze the resulting HTML code. Such tools are easy to write, but they are of little use. The maximum that allows you to analyze this approach is the correctness of the structure of the document, and the text content of the page [1].

Obviously, this was not enough for a full-fledged testing of web applications. In order to solve this problem, PhantomJS appeared - it's a browser engine (Webkit is used - the same one used in Safari, Opera, Yandex-browser and old versions of Chrome), which can be managed using scripts on Javascript. This is a headless browser, that is, it does not display any windows (and does not require a video card and display at all), but works as a command line application. It is cross-platform and can be run, for example, automatically on a remote server.

This server can navigate through the pages, load CSS / JS (optionally, it is possible to load pictures), take screenshots, execute arbitrary JS code in the context of the page [2].

The engine PhantomJS opened tremendous opportunities for working with the browser, and therefore for testing, in fact, it opened several new directions in testing sites.

First of all, this is a **functional test**, based on the ability to control the execution of scripts in the browser. This type of testing covers a weighty set of tasks previously faced by testers:

- check the operation of all mandatory functions of the site;
- testing the working capacity of user's forms on the site (for example, feedback, adding a comment to the blog);
- checking the work of the search (including the relevance of the results);
- checking hyperlinks, searching for non-working links;
- checking the uploading of files to the server;
- viewing the content of the site pages to match the original content provided by the customer.

Another stipulated direction was **testing of layout**. The fundamental point of this type of testing is the verification of the location of the elements, the correspondence of their positions to the provided layouts. PhantomJS allows to start the browser, to open the page and to make a screenshot of the entire page or a specific element on the page. This screenshot will be saved as a basic image for future use. After making any changes on the site, it is possible to restart the engine. PhantomJS will make one more screenshot and compare it with the original one. If no differences are found, the test is passed. If the screenshots do not match, the test is considered failed, a new image is created showing the differences. Such tool is ideal for testing changes in CSS. Of course, it is not difficult to automate this process to open multiple pages of the site and compare each of them with its previous version. Thus, the possibility of automatic regression testing of interface mapping was achieved [2].

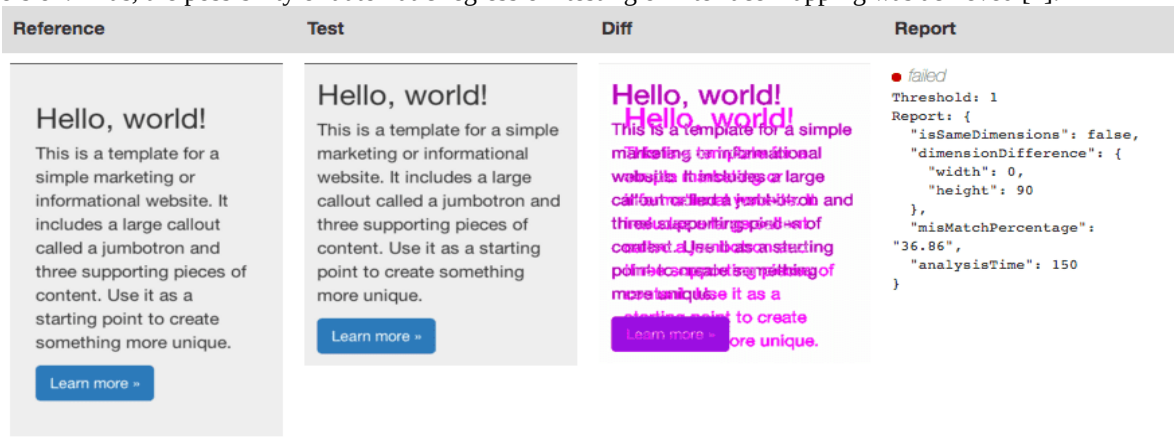

Fig.1 – Comparison of the element's display with its previous version

At first glance, the basic needs for testing sites were satisfied, but this is not the case. One of the main difficulties in the development of sites is the presence of many browsers, on which this site should be displayed. It is necessary to make sure that in each of them everything functions correctly. But PhantomJS is only one browser, testing in it did not allow to be sure that in all other browsers everything will be the same.

Browsers developers understood the need to provide tools for automatically testing sites in them. Thus, there appeared Headless Chrome and Headless Mode in Firefox, but each of them had its own methods for interacting with the site. In fact, developers had to write tests for each of the browsers separately, and this is a huge time cost.

Cross-browser solution

Seeing this problem, the World Wide Web Consortium (W3C, an organization that develops and implements technological standards for the World Wide Web) developed and adopted a WebDriver Internet standard specification. This standard describes the API for the implementation of browsers, which, using the browser on the network, can connect your application

and send commands in JSON-format via HTTP-protocol, and can receive data on the reaction of the browser. W3C were able to negotiate with leading browsers about the implementation of this interaction mechanism. To date, this Internet standard is implemented by most browsers and allows cross-browser automated testing [3].

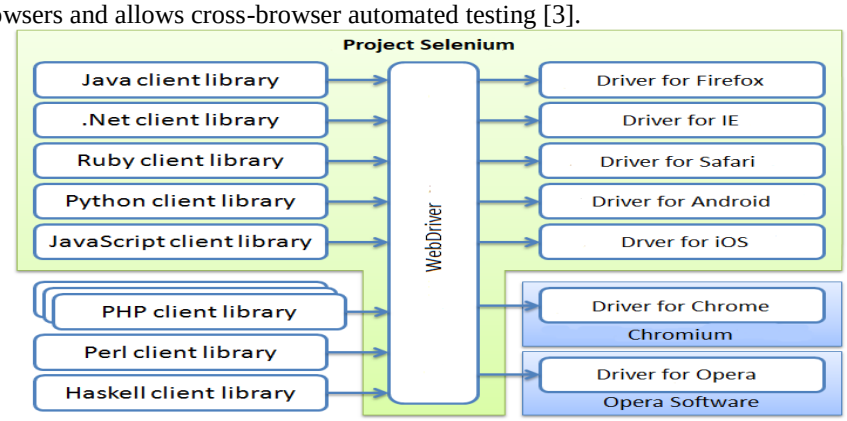

Fig.2 – Scheme of work Selenium WebDriver

It is interesting that the WebDriver specification was based on the already existing solution of one of the automated testing tools - Selenium. Selenium is an open source project that includes several software products, mostly written in Java, that allow you to automate the testing of the web applications interface. Selenium opens the web application being tested in an instance of a browser and, by sending various commands (opening URLs, moving mouse, pressing buttons, etc.), registers the response of the web page to these commands.

The WebDriver standard was compiled in the semblance of Selenium WebDriver, a software library for managing browsers, which aggregated the interaction interfaces with the various browsers necessary for automatic testing.

Selenium is still one of the most popular solutions in the field of automated testing of web-applications, thanks to a number of modules:

- Selenium Server is a server that allows you to manage the browser from a remote computer, over the network;
- Selenium Grid is a cluster consisting of several Selenium servers. It is designed for the organization of a distributed network that allows you to run many browsers on a large number of computers in parallel;
- Selenium IDE is a plug-in to the Firefox browser that can record user actions, can play them, and generate code for WebDriver or Selenium RC, in which the same actions are performed.

Testers who can not (or do not want)to program, use the Selenium IDE as a stand-alone product, without converting the recorded scripts into program code. This, of course, does not allow to develop sufficiently complex test sets, but there are enough simple linear scripts for some programs.

Conclusion

Writing tests may seem like a waste of time. But over time, when the project grows, a new functionality is added or the existing one is reworked, automatic tests (autotests) save very much time for testing, help to detect errors in time, give confidence in the stable operation of the site.

Thanks to the work of the browser community, automated testing has become really effective and convenient to use - it has become cross-browser. Now it is worthwhile to concentrate on the test methods themselves - both creating new ones and improving the algorithms of the old ones. In my opinion, one of the possible areas of improvement is testing the appearance of the layout. Existing solutions rely on the percentage of pixels resemblance in the resulting blocks, and this is a rather questionable characteristic for identifying bugs - it allows a large number of false positives and finding bugs where they do not exist. One of the goals that I am now setting myself will be the solution to this problem. One of the ways I consider is the creation of algorithms that determine the shifts between the internal blocks of the element in question and their relative position.

Литература

[1] Святослав Куликов, Тестирование программного обеспечения. Базовый курс, EPAM Systems, 2017.-314с.
[2] Candea, G. Automated software testing as a service [Text]: / Proceedings of the1st ACM Symposium on Cloud Computing (Indianapolis, USA, 10-11 June 2010) / G. Candea, S. Bucur, C. Zamfir, 2010. - сс. 155–160.
[3] Кудрявцева Е. Автоматизированное тестирование веб-интерфейсов. Горный информационно-аналитический бюллетень (научно-технический журнал), ГИАБ, ISSN 0236-1493, 2014. - сс. 353-356.
[4] “WebDriver”, [Online], Available: https://www.w3.org/TR/webdriver/#compatibility, [Accessed: 16-Jan-2018]
[5] “Selenium WebDriver”, [Online], Available: https://kreisfahrer.gitbooks.io/selenium-webdriver/content/index.html, [Accessed: 16-Jan-2018]

References

[1] Svyatoslav Kulikov. Testirovanie programmnogo obespecheniya. Bazoviy kurs. [Software testing. Basic course], EPAM Systems, 2017. – 314 p. [In Russion]

[2] Candea, G. Automated software testing as a service [Text]: / Proceedings of the1st ACM Symposium on Cloud Computing (Indianapolis, USA, 10-11 June 2010) / G. Candea, S. Bucur, C. Zamfir, 2010. - pp. 155–160.
[3] Kudryavceva E. Avtomatizirovannoe testirovanie web-interfaisov. [Automated testing in web applications], Gorny informatsionno-analiticheskiy byulleten (nauchno-teknicheskii zhurnal) /Mining informational and analytical bulletin(scientific and technical journal)/ , GIAB, ISSN 0236-1493, 2014. - pp. 353-356 [In Russion]
[4] "Visual Regression Testing with PhantomCSS", [Online], Available: https://css-tricks.com/visual-regression-testing-with-phantomcss/, [Accessed: 14-Jan-2018]
[5] "Selenium WebDriver", [Online], Available: https://kreisfahrer.gitbooks.io/selenium-webdriver/content/index.html, [Accessed: 16-Jan-2018]

УДК 664.724:005.591.6:005.936.41

МОДЕЛЮВАННЯ ДИНАМІКИ ЗАПАСІВ ЗЕРНА НА ХЛІБОПРИЙМАЛЬНОМУ ПІДПРИЄМСТВІ: КОНЦЕПТУАЛЬНА, МАТЕМАТИЧНА ТА ІМІТАЦІЙНА МОДЕЛІ

Світий І.М.

Одеська національна академія харчових технологій, Одеса, Україна
ORCID: orcid.org/0000-0002-6181-0475
E-mail: switiy@yahoo.com

Анотація: Проаналізовано сучасні перспективи України як зернової держави в контексті зовнішньої та внутрішньої торгівлі. Означено неефективне використання наявних потужностей зернових підприємств як основу нестачі потужностей. Обґрунтовано необхідність підвищення ефективності рішень, що мають прийматися персоналом, за рахунок інтелектуальної підтримки прийняття рішень, як спосіб підвищення ефективності роботи зернових підприємств. Для вирішення задачі підтримки прийняття рішень та для попередньої оцінки ефективності запропонованих рішень означено задачу побудови моделі процесів накопичення та витрачання запасів зерна, що має стати складовою моделі зберігання запасів зерна на підприємстві. Запропоновано концептуальну модель створення та витрачання запасів зерна. Основу моделі склали основні положення теорії черг. При цьому основні етапи технологічного процесу накопичення та витрачання запасів зерна подано як систему масового обслуговування. Основними параметрами концептуальної моделі означено рівень запасів зерна, кількість обслужених транспортних засобів. Основними факторами, що впливають на означені параметри системи, є інтенсивність вантажопотоків, розмір та час обслуговування заявки. Запропонована математична модель динаміки рівню запасів зерна, що залежить від інтенсивності вхідного та вихідного вантажопотоку. Інтенсивність вантажопотоку напряму корелює з продуктивністю поточно-транспортної системи підприємства. Імітаційну модель запасів зерна було отримано для прикладу зернового терміналу. При цьому поточно-транспортна система терміналу розглянута як багатофазна одно канальна система масового обслуговування. З отриманою моделлю проведено серію машинних експериментів. Також було окреслено основні перспективи розвитку та використання моделі для вирішення задач удосконалення алгоритмів керування запасами зерна.

Abstract: The current prospects of Ukraine as a grain state in the context of foreign and domestic trade are analyzed. The inefficient use of available capacities of grain enterprises as a basis for lack of capacity is indicated. The necessity of increasing the efficiency of the decisions to be taken by the personnel, due to intellectual support of decision-making, as a way