jFAT: An Automation Framework for Web Application Testing

Hanh Phuc Nguyen[1,2(✉)], Hong Anh Le[3], and Ninh Thuan Truong[1]

[1] VNU - University of Engineering and Technology,
144 Xuan Thuy, Cau Giay, Ha Noi, Vietnam
thuantn@vnu.edu.vn
[2] VMU - Vietnam Maritime University,
484 Lach Tray, Le Chan, Hai Phong, Vietnam
phucnh@vimaru.edu.vn
[3] Hanoi University of Mining and Geology,
18 Pho Vien, Bac Tu Liem, Hanoi, Vietnam
lehonganh@humg.edu.vn

Abstract. Web technologies have developed rapidly because web application is currently leading the trends of software development. A web-based application is a program that is accessed over a network connection, rather than existing within a device's memory, hence detecting its failures is different from other software systems. Many approaches and tools have been proposed for web testing, however, introducing new frameworks is still an emerging topic in this field. This paper proposes an automation framework running in Java platform for web testing, called jFAT, which integrates with Selenium and TestNG. The paper also illustrates the use of framework with the Bank application case study.

Keywords: Automation testing · Web applications · POM

1 Introduction

Software applications nowadays are built and worked in web environments, they are also called web-based applications. A web-based application is a program that is accessed over a network connection, rather than existing within a devices memory, it often runs inside a web browser. Their advantage is the ability to share functions, data in the system; support multiple environments working. Therefore, providing quality assurance solutions for web-based applications is becoming more and more important.

Software testing is a technique used widely in software quality assurance. It is a process of executing a program or application with the intent of finding the software bugs. It can also be stated as the process of validating and verifying that a software program or application meets technical requirements that guided its design and development.

P. Cong Vinh and V. Alagar (Eds.): ICCASA 2018/ICTCC 2018, LNICST 266, pp. 48–57, 2019.
https://doi.org/10.1007/978-3-030-06152-4_5

In reality, two methods may be applied to test a software application: manual and automation testing. However, manual testing can become boring and hence error-prone. To avoid these disadvantages and to reduce time and money consuming in testing process, one uses automation testing, in other words an automation tool is used to execute a test case suite.

This paper proposes a Java-based automation testing framework of web-based applications, which integrates with Selenium and TestNG providing features to perform test cases automatically. The contribution of the paper includes (1) module-driven approach to web testing; (2) a high performance and secure framework which is able to execute multiple test scripts at a time.

The remainder of the paper is structured as follows. Section 2 provides some backgrounds of web automation testing. Main work of the paper is presented in Sect. 3 which presents the approach and detailed design of the proposed framework. Followed by Sect. 4, a case study of Bank management system is shown for illustration purposes. Section 5 summarizes the work that relates to this paper research topic. Finally, Sect. 6 concludes the paper and gives some future research directions.

2 Background

Automation testing is a method of software testing that uses software tools to perform tests and then compares actual test results with expected results. All of this is done automatically with little or no human intervention. Automation testing includes the following main phases:

- Test tool selection: One of the most important steps before starting automation in any organization is test automation tool selection. Test Tool selection is heavily dependent on the technology which the Application Under Test is built on. Our paper brings the benefit to this phase by providing an automated testing tool for web applications.
- Define the scope of automation: The scope of automation is the area of the Application Under Test will be automated. In order to determine the scope, pay attention to matters such as: scenarios which have a large amount of data; common functionalities across applications; the complexity of test cases; etc
- Planning, Design, and Development: In this phase, you build the strategy and plan for automation. Planning includes establishing an approach to the automation testing framework that consists of the hardware, network, models, tools, and analysis methods. You need to perform the following tasks: Automation tools selected; Framework design and its features; In-Scope and Out-of-scope items of automation; Automation testbed preparation; etc.
- Test execution: This phase involves actual execution of automated tests or integration of the automation framework with built systems. Execution can be performed by means of the automation tool directly or by utilizing the Test Management tool which will trigger the automation tool. This phase also involves verifying the reports generated by automated tests, analyzing the failures and logging appropriate failures as defects in defect tracking system.

– Maintenance: This phase is made to ensure that all automated tests are updated regularly to accommodate changes in application under test so that they remain on purpose and give reliable results.

Testing of Web-based applications is difficult due to its nature of execution environment such as heterogeneity, multi-platform support, autonomy, cooperation, and distribution, etc. Many approaches are proposed to test web applications, these approaches concentrated on the following aspects:

– Functionality Testing: This is used to verify if your product is as per the specifications you intended for it as well as the functional requirements you planned for it in your developmental documentation. Functional testing requires that the tester perform a test of all the links in the site, the format used in the site to send and receive the necessary information from the user. There are also database connections, cookies checks and HTML/CSS verification.
– Usability testing: To verify how the application is easy to use with.
– Interface testing: Performed to verify the interface and the data flow from one system to others. There are three areas that need testing - Application, Web and Database Server.
– Database Testing: Database is one crucial component of your web application and it must be tested comprehensively.
– Compatibility testing: Compatibility testing is performed based on the context of the application.
– Performance Testing: This will guarantee your site's ability to handle all loads.
– Security testing: Performed to verify if the application is secured as data theft and unauthorized access are common issues on web environments.

Our approach will work with the functionality aspect and improve the performance of testing process.

3 An Automation Framework for Web Application Testing

In this section, we first introduce the architecture of the proposed automation framework. After that, its detailed design is presented. The proposed testing approach, which is modularity-driven and POM (Page Object Model)-based allows us to develop and maintain test cases more easily. It includes the main steps: (1) Generating POM from web applications; (2) Developing test scripts; (3) Executing test scripts; and (4) Logging the test results.

Following the proposed idea, we construct the jFAT automation framework with the main characteristics as follows

– Web elements are defined at once and reused for all test cases.
– Test cases are constructed in correspondence to certain functionalities of web applications.

Figure 1 depicts the architecture of the proposed framework. Pages and objects are automatically generated from web applications using Selenium Page Object Generator. Two core components of the jFAT are constructing test scripts from POM model and executing test cases using TestNG. First, users develop test cases using POM model. The framework creates test scripts that correspond to test cases based on these models. Then, it converts the test plan defined by users to an XML-based file, which is executable by TestNG. The framework runs the test cases on the browsers by using TestNG and Selenium API. The test results are captured and stored in the file.

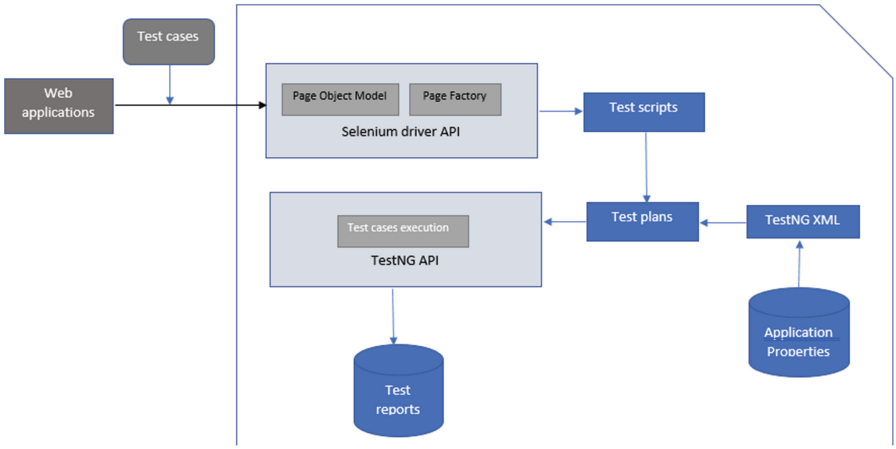

Fig. 1. Architecture of jFAT framework

The proposed framework provides security policies to prevent testing scripts from unintentionally accessing resources. The secure model is based on sandbox mechanism deployed in Java platform. Users also can change the security policies by customizing the configuration file, whose format is similar to a policy file of Java.

TestNG and Selenium are integrated into the framework and can be customized by using an XML-based configuration as follows.

```
<?xml version="1.0" encoding="UTF-8"?>
<dependency>
    <groupId>org.testng</groupId>
    <artifactId>testng</artifactId>
<version>6.13.1</version>
</dependency>
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.8.1</version>
</dependency>
```

4 Case Study: Banking Application

In the case study, we take an existing website from Guru99 [1] that simulates banking transactions for managers. The information of customers include full name, birthday, city, email, phone number, and pin code. One customer has several banking accounts. One manager is in charge of many customers and is able to do the following tasks.

- Create, update, delete customers from the banking system.
- Based on the existing information, managers can create accounts for customers. A banking account has the information such as customer code, account type, and initial balance.
- Managers can deposit/withdraw money to/from a certain account by providing an account number, amount and the reasons. Transactions have to make sure that deposited money is greater than 0 and withdrew money is less than the current balance.
- Transferring between accounts is performed similarly with source and destination accounts.
- Managers also can check the current balance and transaction history of all accounts that they manage.

To illustrate the use of the proposed framework, we define 10 test cases as follows (Table 1).

Table 1. Test cases of the application from guru website

No	TC Name	Description
1	TC01	Create new manager account from guru-bank site
2	TC02	Login to guru-bank page
3	TC03	Change manager password
4	TC04	Create new customer
5	TC05	Create new account for customers
6	TC06	Change customer account type
7	TC07	Deposit to an account
8	TC08	Withdraw from an account
9	TC09	Transfer between two accounts
10	TC10	Delete an account

Follow the proposed approach, jFAT users need to create classes for pages and objects of home, log in, and register pages. The page classes defined methods which correspond to the functional features of the pages, whilst the object classes defined attributes for web element identifiers.

Listing 1. A part of Home page class

```
public void clickFundTransferMenu(){
    waitForElement(hpo.MN_FUND_TRANSFER).click();
}

public void enterAccountID(String payer,String payee){
    waitForElement(hpo.TF_PAYER_ACCOUNT).sendKeys(payer);
    waitForElement(hpo.TF_PAYEE_ACCOUNT).sendKeys(payee);
}

public void enterAmountAndDesc(int amount,String desc){
    waitForElement(hpo.NF_FUND_AMOUNT).sendKeys(String.valueOf(amount));
    waitForElement(hpo.TF_DESCRIPTION).sendKeys(desc);
}
```

Listing 2. A part of Home page object class

```
@FindBy(xpath = "//a[contains(text(),'Change Password')]")
public WebElement MN_CHANGE_PASSWORD;

@FindBy(xpath = "//a[contains(text(),'New Customer')]")
public WebElement MN_NEW_CUSTOMER;
```

In order to execute test cases with jFAT, users need to create a test case class whose method shows the steps of a test case. The snippet below show the content of test case TC08.

Listing 3. A part of Home page object class

```
 WebDriver driver = drivers.get();
driver.get("http://demo.guru99.com/v4/");
LoginPage loginPage = new LoginPage(driver);
loginPage.enterUserName(un);
loginPage.enterPassword(pw);
HomePage homePage = loginPage.clickSubmit();
homePage.checkManagerID(un);
homePage.clickBalanceEnquiryMenu();
String accid =DataStorage.getData(2).toArray()[0].toString();
homePage.enterAccountID(accid);
homePage.clickSubmitEditForm();
int currentAmount = homePage.saveCurrentAmount();
homePage.clickWithdrawalMenu();
homePage.enterAccountID(accid);
int withdraw = 1200;
homePage.enterAmount(withdraw);
homePage.enterDescription("withdraw");
homePage.clickSubmitEditForm();
homePage.checkRemainingAmount(currentAmount,withdraw);
```

jFAT allows to define a test plan in a text file that indicates which test cases to run and the number of runs. Listing 4 means that the test plan executes *TC01*, *TC02*, *TC03*, *TC09* and *TC02* will be performed twice.

Listing 4. A test plan

```
Testsuite|tc01|1
Testsuite|tc02|2
Testsuite|tc03|1
Testsuite|tc09|1
```

In order to run all test cases, the test plan just need to define *Testsuite—All—1*. Figure 2 shows the result of the execution of the test plan 4. It graphically reports that 4 test cases are successful and 01 test case is failed.

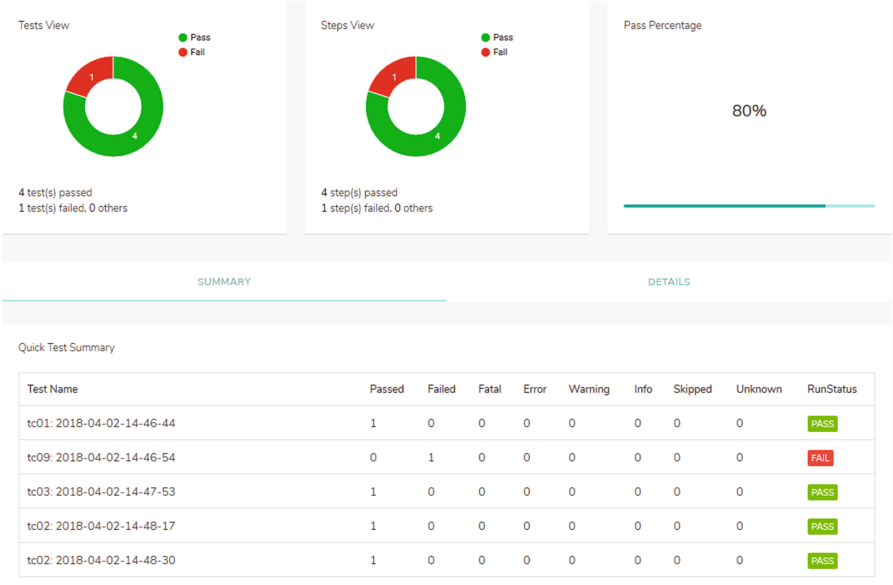

Fig. 2. Test report of the test plan

5 Related Work

Many approaches and tools have been proposed to automation testing. Some of them are applied in industry projects.

Since tests need to be executed repeatedly, such manual tests then have to go through test automation to create scripts or programs out of them. The paper [11] describes a technique to automate test automation. The input to their technique is a sequence of steps written in natural language, and the output is a sequence

of procedure calls with accompanying parameters that can drive the application without human intervention.

The paper [6] has proposed to design an automatic software testing framework for web applications based on the Selenium and JMeter. With the use of the software framework, they efficiently improve the extensibility and reusability of automated test. The results of the paper show that the software framework improves software products quality and develop efficiency.

The paper [9] exploits an object-oriented model of a web application as a test model, and proposes a definition of the unit level for testing web application. Based on this model, a method to test the single units of a web application and for the integration testing is proposed. An approach to web application functional testing is provided in the paper.

In recent years, automated web application testing tools have grown to the maturity stage in both quantity and quality. Hower [7] has presented more than 200 both commercial, and free-on-the Web testing tools for Web applications which fall into six categories. The tools are comprised of the last category that is used to assist the functionality testing of Web applications. Therefore, from a general perspective, the tools for Web application functionality testing are classified into three major categories as mentioned here.

Romano *et al.* [10] developed the tool for the non-functional requirements testing. They were interested in testing the aspects of (i) *Load and performance testing* and (ii) *Navigability testing*. The result of this research may be employed to assess the hosting infrastructure of Web Applications in improving quality, reliability and acceptance of Web applications.

In the line of the web application functional testing, Di Lucca *et al.* [9] exploited an object-oriented model of a web application as a test model as well as proposed a definition of the unit level for testing web application. A method to test the single units of a web application and for the integration testing is proposed based on this model.

Huang *et al.* [8] introduced a software tool, named WASATT *(Web Application Scenario Automated Testing Tool)* to support the automated testing for scenario of web-based applications.

The paper [6] has proposed to design an automatic software testing framework for web applications based on the Selenium and JMeter. With the use of the software framework, they efficiently improve the extensibility and re-usability of automated test. The results of the paper show that the software framework improves software products quality and develops efficiency.

Beside approaches presented in the papers, many automation testing tools are also available in market. Test automation tools can be expensive, and are usually employed in combination with manual testing.

Selenium [2] is an open source, popular testing tool that provides playback and recording amenity for regression testing. Recorded script in other languages like Java, Ruby, RSpec, Python, C♯, etc. can be exported using Selenium.

QTP [3] is commonly used for functional and regression testing and accommodates for every major software application and environment. Test creation and

maintenance can be simplified by QTP utilising the concept of keyword driven testing. The tester is thus able to build test cases directly from the application. Defects can be fixed faster by comprehensively documenting and replicating defects for developer.

Rational Functional Tester (RFT) [4] is an Object-Oriented automated functional testing tool that is able to deliver automated functional, regression, data-driven testing and GUI testing. It supports a wide variety of protocols and applications like Java, HTML, NET, Windows, SAP, Visual Basic, etc. It is capable of recording and replaying the actions on demand, developers can also be allowed to devise keyword associated script so that it can be re-used.

Silk Test [5] is designed for performing functional and regression testing. Silk test is the primary functional testing product for e-business application. From an object-oriented language perspective, Silk Test uses the concept of an object, classes, and inheritance. Script commands can be converted into GUI commands by Silk Test. Commands can be operated on a remote or host machine on the similar machine. Silk Test can be implemented to recognise the movement of the mouse along with keystrokes. It can be useful for both playback and record method or descriptive programming methods to get the dialogs. All controls and windows of the application under test as objects are identified and all of the attributes and properties of each window are determined by Silk Test.

6 Conclusions and Future Work

This paper proposes a module-driven and high performance framework for web testing. The proposed framework, integrated with Selenium and TestNG, allows users to easily develop and perform test plans. The framework also provides a rich and friendly graphical test result reports of all test cases. It has just been done in the initial stage.

In the future, the framework needs to automate some steps to assist users in the development such as automatic extraction of web elements and test script generation. Moreover, we will adopt the data-driven testing approach to separate data and scripts that makes the jFAT more flexible.

Acknowledgments. This work has been supported by VNU University of Engineering and Technology under Project QG.16.32.

References

1. https://www.guru99.com
2. https://www.seleniumhq.org
3. https://www.tutorialspoint.com/qtp/
4. https://www.automation-consultants.com/products/ibm-products/rational-functional-tester/
5. https://www.microfocus.com/products/silk-portfolio/silk-tes

6. Fei, W., Wencai, D.: A test automation framework based on web. In: IEEE/ACIS 11th International Conference on Computer and Information Science, ICIS. IEEE (2012)
7. Hower, R.: Software QA and testing resource center (2006). www.softwareqatest.com
8. Huang, C.-H., Chen, H.Y.: A tool to support automated testing for web application scenario. In: IEEE International Conference on Systems, Man and Cybernetics, SMC 2006, vol. 3, pp. 2179–2184. IEEE (2006)
9. Lucca, G.A.D., Fasolino, A.R., Faralli, F., Carlini, U.D.: Testing web applications. In: International Conference on Software Maintenance, Proceedings, pp. 310–319 (2002)
10. Romano, B.L., et al.: Software testing for web-applications non-functional requirements. In: 2009 Sixth International Conference on Information Technology: New Generations, pp. 1674–1675. IEEE (2009)
11. Suresh, T., Saurabh, S., Nimit, S., Satish, C.: Automating test automation. In 34th International Conference on Software Engineering (ICSE), ICSE, IEEE (2012)