Automation Testing using Selenium Webdriver

Pooja Yadav#1, Mamta Yadav*2
#Pooja Yadav, Mamta Yadav(HOD ,CSE Department)
Yaduvanshi college of Engg. & Technology, Narnaul(Haryana)

Abstract

In Automation testing, selenium works as a web application tools, and is also an open source software. In a very short span of time, selenium tools achieved wide acceptance. Automation testing using automation tools to reduce human intervention and repeatable tasks. To reduce human intervention and repeatable tasks automation testing uses automation tools i.e. (selenium). For testing the web application, in this paper we design and implement the automation testing framework. Here, tester doesn't need to study this tool in details, but due to screen shot property of framework the tester can easily write their test cases efficiently in less time. In this paper we study on various component of selenium IDE, selenium RC, selenium web driver. Automated Software testing is the best way to increase the effectiveness, efficiency and coverage of software testing and Selenium is a framework comprises of many tools used for testing web applications .With the help of the case study, we analyse and find the testing of a web application using automation testing tool "Selenium IDE". Using this approach, test cases are automatically recorded in background while tester is entering the data in a web application screen and these test cases are reusable and best suited in the Regression Testing environment.

Keywords

Web browser, Selenium IDE,JRE,

INTRODUCTION

Any software, no matter what language it is implemented in, has some bugs. Some of them are detected and removed while coding. The rest are found and fixed while integrating software modules into a system during formal testing. However, it is known to all manufacturers that bugs that remained in the software can be fixed later. At this point, testing is required for a successful implementation of the software. Based on the project requirements, testing will fix all the bugs before releasing the software. However, for a most complicated project test automation is needed. Now-a-days, there are many commercial software testing tools. A powerful and flexible software-testing tool with more features to test a large scale project should be chosen so that it can deal with browsers' rich content api along with dynamic web applications. On the other hand, a tester can focus on fixing bugs in high risk areas with such a tool in a complicated web application.

Web Driver is a new technology with a suite of various tools to automate browser-based web application. The purpose of the thesis was to automate Gmail on multiple browsers and Selenium Web Driver was chosen. The web browser can be driven by Selenium Web Driver in order to execute different types of automation tests on the web application as if an end user can navigate through them. Selenium Web Driver can emulate actions like clicking on the image, context or entering text, dragging and dropping a Web Element anywhere in the DOM, submitting forms, opening and closing a new tab or window, downloading a file from the internet and saving it to a particular location in a local machine, and reporting the results back to the end user so that end

User knows that it works as expected. Using this tool, four test scripts were written to perform functional testing on gmail and results were documented as well. All test cases were executed on chrome and firefox browsers. All test cases were tested successfully. The analysed result of the total execution time taken by chrome and firefox browsers is given in this thesis.

In practice, it is unrealistic to automate everything in a web application like gmail by web driver.

I. .

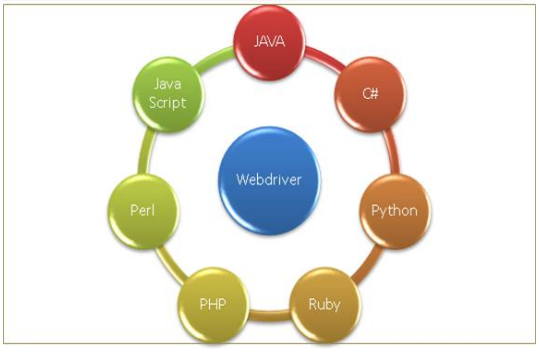
A.

B. What is Automation & Why:

C. Automation is a way to make sure that after fixing issues, all other functionalities of an application should be working fine or not. Automating tests also means that testercan develop a test script during the program is written by core developers. This is the ideal process because the tester can then confirm as soon as the program is written that it does what was expected of it with the click of a button. Each time you write an automated test, you might have missed the chance to perform n number of manual tests. In general, creating the original automated test script with encoded verifications that test certain elements of the program can often be more intricate than simply clicking the links and verifying with one look that everything has loaded and working properly. That being said, with the knowledge that these scripted tests can be run over and over again, it is best to trust automation to do the job. Consider the fact that the tester, who may or may not be involved in development, is not always able to pick up on the finer details of the program that might have changed. Instead of having to remember every aspect of the program and make sure everything is still in order, the tester can trust their previously constructed script to do so. In future cases, the need to manually test the program would no longer be necessary. Automated testing comes in especially handy with web software. If you were testing a shrink-wrap product whose product direction and code base has changed wildly in the last few months you may not even have time to try all the obvious tests once. In the time you would spend automating your tests, you could find at least one completely new bug. In this case, the cost of automation is high. There are a few questions one should ask oneself when determining whether or not to automate tests. Is the feature is core or critical ? Is the test tedious and error prone? Will my test script verify results via a fragile method (screen capture) or a sturdy method? Is the feature I am trying to automate undergoing a lot of churn? When this script fails, how easy will it be for me to investigate the failure? One thing that will cause test scripts to fail just about faster than anything else is the product changing. This is why refactoring tests is so important.

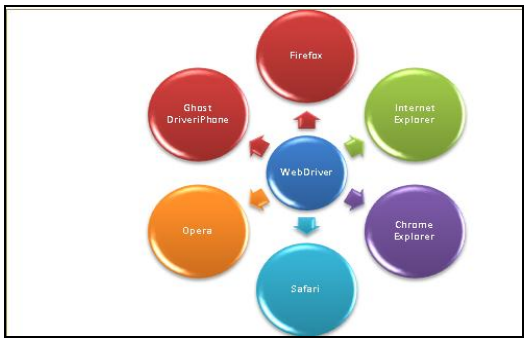
- Fig. Multiple Languages Binding *SUPPORT*

D.



E. *Fig. 1 Multiple browser Support*

F.



Selenium We bDriver Architecture:

G. Selenium Web Driver implemented on Layered Design, ideology behind this implementation is to more and more usage of tool for automation and this could be possible by fitting best fit programming languages.

H. Implementation of Web Driver is that each browser has a language that is most natural to use when attempting to drive it. Browser drivers are built as per the best fit language and we can just see the wrapper around them.

I. We can say that for any of browser driver if any of the features works there in one binding language then it should be easy to add support to other binding languages also.

J. Selenium Web Driver is a compact Object Oriented API which can directly interacts with the Application under tests.

K. Selenium Web Driver utilizes the browser native compatibility to automation without using any peripheral entity

L. ***References***

HTTPS://WWW.GURU99.COM/SELENIUM-TUTORIAL.HTML

HTTP://TOOLSQA.COM/SELENIUM-WEBDRIVER/SELENIUM-INTRODUCTION/

http://www.seleniumeasy.com/selenium-webdriver-tutorials

CONCLUSIONS

Core goal of a developer and a tester is to make sure they develop and deliver a user friendly and error free application & that is only possible by thoroughly testing of that application before go live. More often than not, using automated tools like selenium &test scripts is the best way to be sure that this goal is accomplished. In particular, the tester wants to be sure to create maintainable test scripts that will last through the many changes that applications undergo. If modifying or refactoring the test script does become necessary, there are ways to

make sure this job is done quickly and correctly. Always avoid duplicate cases. By keeping specific tests self-contained, they can be reused in several places and only one modification would be necessary for all instances. Selenium Web-driver has many advantages, including grabbing evidences of errors in the form of screenshots and logs, which can be helpful for developer to find out location of issue in code & module.

ACKNOWLEDGMENT

It is a great pleasure for me to EXPRSS MY sincere gratitude to my supervisor, Ms. Mamta Yadav Assistant Professor, Department of Computer Science & engineering of Yaduvanshi College of Engineering and Technology, Narnaul Haryana Delhi for his valuable guidance,

timely advice and constant encouragement during the project work.

I am very much thankful to Ms. Mamta Yadav, HOD, Department of Computer Science & Engineering, Yaduvanshi College of Engineering and Technology, Narnaul, for his expert advice and inspiration from time to time which helped me to concentrate more on our project work and execute it on time. I would like to take this opportunity to acknowledge the valuable help and guidance received from Mr. Pardeep Kumar(Director-Principal, YCET) in spite of his extremely busy schedule. He not only gave me his valuable time but also provided the details regarding the project. I am thankful to the staff members of Department of Computer Engineering at Yaduvanshi College of Engineering and Technology, Narnaul for their valuable suggestions. Although it is not possible to name individual, I cannot forget my well-wishers for their persistent support and cooperation. I am always in debited to my parents and family for their endless support being it a moral, financial or emotional, which enabled me to reach at this level of knowledge

References

[1] PRESSMAN Roger, Software Engineering, McGraw Hill, 1997

[2] Richardson, Alan (2012). Selenium Simplified, Second Edition, Compendium Developments

[3] HERZUM Peter, SIMS Oliver, Business Component Factory, Wiley, 1999

[4] HUIZING M., Component Based Development, http://www.win.tue.nl/xootic

[5] /magazine/jan-1999/huizing.pdf, last access:01.12.2005

[6] WANG JuAn, TowardsComponent-BasedSoftwareEngineering,wang.pdf?key1=357729&key2=7337455011&coll=GUIDE&dl=GUIDE&CFID=36365381&CFTOKEN=91792709, last access:01.12.2005

[7] HILL, Bennett, McROBB, Farmer, Object Oriented System Analysis and

[8] Design (using UML), 2nd Edition, McGraw Hill, 2002

[9] ATKINSON Colin, BAYER Joachim, LAITENBERGER Oliver, ZETTEL Jörg,Component-Based Software Engineering:TheKobrA Approach, http://se2c.uni.lu/tiki/se2c-bib_download.php?id=700, last access:01.12.2005

[10] DOGRU Ali H., TANIK Murat M., A Process Model for Component- Oriented Software Engineering, IEEE Software, March/April 2003

[11] SCHACH Stephen R., Classical and Object Oriented Software Engineering, 4th Edition, McGraw Hill, 1999

[12] RUMBAUGH James, BLAHA Michael, PREMERLANI William, EDDY Frederick, LORENSEN William, Object Oriented Modelling and Design, Prentice Hall, 1991

[13] HEINEMAN George T., COUNCILL William T., Component-Based Software Engineering – Putting the Pieces together, Addison Wesley, May 2001

[14] STOJANOVIC Zoran, An Integrated Component-Oriented Framework for Effective and Flexible Enterprise DistributedSystems Development, http://www.betade.tudelft.nl/publications/Stojanovic_CAISE2002.pdf, last access:01.12.2005

[15] BAYAR Vedat, A Process Model For Component Oriented Software Development, Master Thesis, 2001