ISSN:0975-9646

Novel Framework for Browser Compatibility Testing of a Web Application using Selenium

Prof. Anand Motwani[#1], Amber Agrawal[*2], Dr. P. N. Singh[#3], Prof. Anurag Shrivastava[#4]

[#]*Assistant Professor, Department of Computer Science and Engineering*
NRI Institute of Research and Technology, Bhopal

[*]*Student, Department of Computer Science and Engineering*
NRI Institute of Research and Technology, Bhopal

***Abstract*— With a wide range of web browsers available, end users can use a variety of web browsers to access a web application. It is now become essential to review and compare website functionality and styles on multiple browser platforms. For different browsers requests may be handled and processed in a different manner based on the user agents received from the client browser. So just testing on a single browser may cause the application functionality to behave differently or incorrectly on another browser. You need to make sure that your web application works fine across multiple browsers.**

Compatibility testing is a non-functional testing in which the application is verified to run on different browsers, various resolution, various operating systems, etc. But testing across multiple browsers can become very time consuming and costly.

In this paper, our focus will be on cross-browser testing which simply means to test a web application in multiple browsers, and making sure that it works consistently and as intended and does compromise in quality or features. The aim of this paper is to create a prototype framework for browser compatibility testing. It deals with automation testing for a web application on different web browsers using Selenium WebDriver. It briefly describes the tools and frameworks used for automation testing along with the advantages and disadvantages of automated testing with the focus on testing user interfaces of web-based applications to achieve high quality software.

Keywords: Test automation, Browser compatibility, Cross-browser testing, user interface, web application, Selenium, WebDriver and TestNG.

***Keywords*— Test automation, Browser compatibility, Cross-browser testing, user interface, web application, Selenium, WebDriver and TestNG.**

I. INTRODUCTION

Any web application or piece of software which has been developed or will be developed, has or will have some bugs. One of the basic functions of software testing is to detect these errors and to detect them as fast as possible. With the increase in importance of the quality of software applications, it has become essential that we have efficient methods for software testing.

Organizations are now focusing on agile methodologies, which has a shorter software lifecycle. In the attempt to deliver quickly but an error-free product, organizations are turning to automated testing. Automated testing is executing tests with the help of some tools that can be run repeatedly at any time of the day. The idea is to combine related test cases and create a test suite. Software automation tools execute these large and complicated tests or test suites and generate reports. We can also compare these results with the tests run earlier which can allow the developers to analyse the difference. As compared to manual testing, automation testing is more reliable, maintainable, re-usable and comprehensive. It not only saves time as it has less human interactions but also decreases cost and has greater code coverage.

With the launch of various browsers and different versions of those browsers, it becomes essential for a web application designer to ensure that the application is compatible with all the browsers and support if not all at least most of the latest versions. The goal of our work in this paper is to create a framework through which we can easily test a web application on all browsers and their different versions. Finally, create a report template displaying compatibility results across browsers and different versions.

A test automation framework is an integrated system which defines rules of automation specific to a product. The framework includes function libraries, test data sources and other reusable modules. All these components are the building blocks which when assembled will represent a business process.

There are several test automation tools available in the market. This paper will focus on one such tool namely, Selenium. Selenium is an open source tool used for automation testing of web based applications. The tool directly runs on the browser and supports almost all available browsers like Google Chrome, Mozilla Firefox, Internet Explorer, Safari, etc. It runs on all platforms such as Windows, Linux and Macintosh. It is mainly used for writing end to end tests for web applications and is a very useful tool for system functional testing and browser compatibility testing. A tool like Selenium used for browser automation is expected to do exactly what you would expect: automate the control of the browser so that repetitive tasks like clicking, editing text field etc., can be automated. But only the availability and use of the tool does not solve our problem. Hence, we design a framework to help us analyse other aspects.

This paper focuses on one such approach where we propose a medium-scale framework to be able to perform automated browser compatibility test execution and reporting for a web application on different browsers. The concept and its underlying requirements were tested by

implementing a prototype of the framework and executing a few automated tests for different browsers. Based on the results of the prototype, the overall framework concept was found to be feasible. There were certain changes made to the framework while design from the original requirements. The most interesting finding is that it is possible to cover most of the cross browser testing needs to be done with the keyword-driven approach alone.

As a conclusion to this work, results are reviewed and a general analysis is presented on the basis of the results. This would generally help the developers and other stakeholders of the project to analyse and approve the usability of the web application on different versions of multiple browsers.

Besides this introduction, this paper is organized as follows: Section 2 summarizes some related works which have been done to perform compatibility testing using manual test methods and test tools. Section 3 describes the extension of the Selenium WebDriver tool to create a prototype of the framework, developed to support automated compatibility testing. Section 4 reports a case study performed to apply the proposed tool in a real software project and the comparison between two other strategies (manual and semi-automated tests execution). Finally, Section 5 presents the conclusions and future work.

II. Related Work

A Several people have highlighted the problems faced when web applications are run on different browsers. A few authors have also mentioned the problems faced in cross browser compatibility testing of modern web applications [1] as a 'functional consistency' check of web application and presented an automated solution for it. While those papers focus on capturing the behaviour of a web application on different browsers and observe the behaviour as a finite state machine navigation model, our approach is to provide a framework which will help testers test the web application on different browsers and come up with a report for analysis. In another paper [4], the authors describe common set of compatibility issues in different versions of Internet Explorer. The focus is on application compatibility with different versions of Internet explorer and the issues faced by website using legacy features. The suggested framework is used to test both static and dynamic web pages. The idea was to list down all the use cases and based on the application flows, classify the application as High/Medium/low complexity application. The testing methods are manual where the testing begins by navigating to web pages on target Internet Explorer environment and verified against legacy environment (previous Internet Explorer). The page rendering in terms of misaligning of web objects – objects on header and footer, logos, menu items, contents implemented through web tables etc. are verified. Then they documented issues found (as defects) and published them in the form of ACR (application compatibility report). Based on the issues, one can classify the web applications as Red or Amber or Green.

Green application will be those that renders flawlessly on target Internet Explorer. Amber application will be those that requires minor tweak in application configuration files, changing browser settings etc. Once these settings are done, these applications will also work fine on target Internet Explorer. However Red applications will be those that are altogether incompatible on target Internet Explorer. These applications require code remediation or code fix in order for them to work on target Internet Explorer.

The paper also mentions some online web application testing tools which help speed up testing and proves reliable while testing. Using the information from this paper, we extend the research for different browsers and their versions not limiting only to Internet Explorer.

III. PROPOSED WORK

The idea of compatibility testing of a web application for only one type of browser and its different versions was not enough to ensure that the application is stable and can be used extensively. To ensure this we would like to extend the work and propose a framework which will allow testing of a web application on different browsers like Internet Explorer, Mozilla Firefox and Google Chrome.

In this work a generic framework with the help of Selenium is proposed. Selenium is a browser automation tool [2, 8] which lets programmer to automate operations like: type, click and selection from a drop down of a web page. It provides a rich set of testing functions specifically geared to the needs of testing of a web application. One of Selenium's key features is the support for executing one's tests on multiple browser platforms. Using this feature we will test our application's compatibility with different types of browsers and create a report for the same.

The prototype framework is designed to use the Selenium WebDriver which allows support of various browsers using a special controller, using that it communicates directly with the web browser.

There are drivers available for all most common operating systems and related browsers starting with versions:- Google Chrome 12+, Internet Explorer 6+ (both 32 and 64-bit version), Opera 11.5+, Mozilla Firefox 3.6 - 18, Safari 5.1+, HtmlUnit 2.9. Support for two most prevalent mobile platforms has also been recently added and WebDriver now supports: Android 2.3 + (the physical device and emulator), iOS 3+ for smartphones and iOS 3.2+ for tablets (the physical device and emulator). Not only can we test different browsers but we can test them on different platforms like Linux, Windows, etc.

The high level architecture of the prototype is explained in the following diagram:

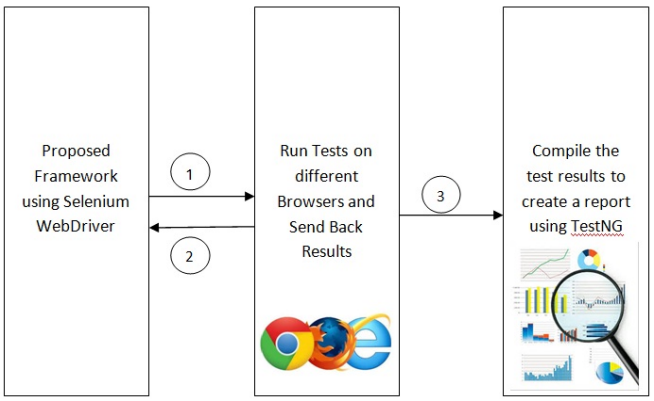

Figure 1: High level architecture

The prototype framework design consists of three base classes which include the different web drivers for each type of browser. Then, function-specific or use-case specific classes are created to test their execution. So, once the base classes are ready, these are extended and test classes are written.

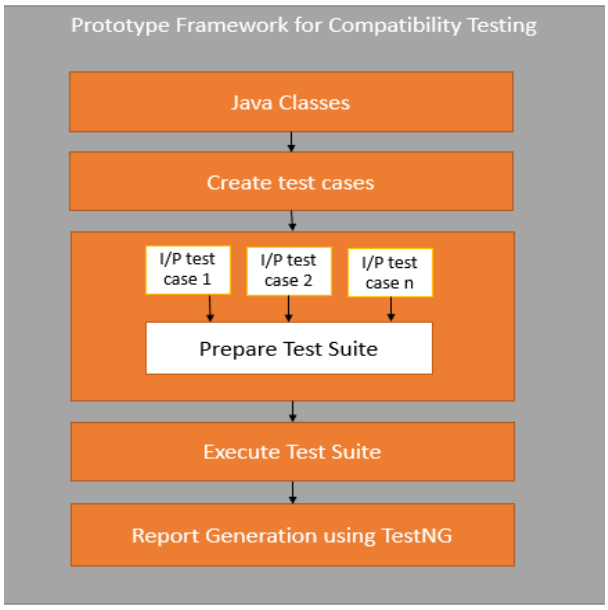

Figure 2: Flowchart for prototype framework

In this work, the prototype is created for testing login functionality of a Gmail account. Other features or use case tests can be eventually added as separate classes and they can extend the abstract base classes to use the browser-specific web drivers.

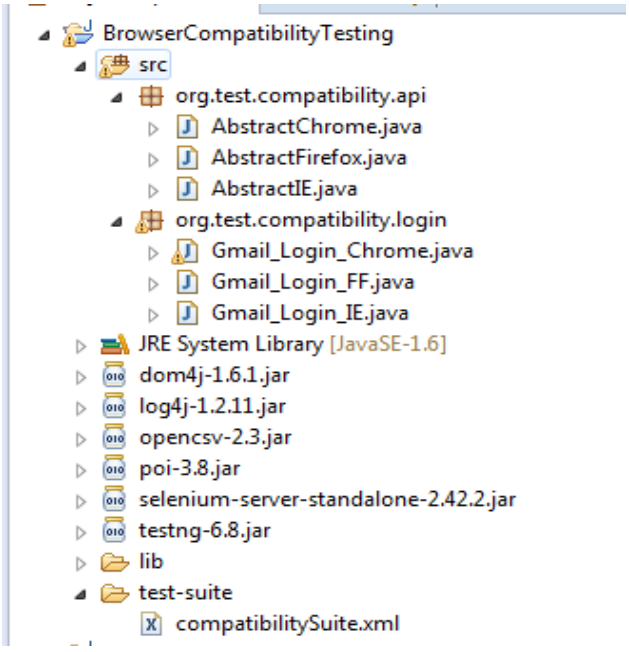

Figure 3: Project structure

Another key aspect of the prototype is keyword driven automation testing using Selenium WebDriver [5]. In Keyword driven testing, each keyword corresponds to an individual testing action like a mouse click, selection of a menu item, keystrokes, opening or closing a window or other actions. Keyword driven Testing involves a set of keywords which define a sequence of operations. These keywords are then used to create reusable functions mapped to a particular functionality of the application. With the keyword driven approach, we can automate the following test scenarios for Gmail as under:-

1. User should able to logging in its account, when we are entering correct username and password.
2. User should not be able to login in its account, when any one of username and password is incorrect.
3. User should be able to view the inbox mails, etc.

For such web applications, in each test case we usually write the methods that perform specific task. In our case, we have created one package structure, under which we have written browser specific classes for each browser, and within those classes we have added methods for the same functionality. For example, for login, we create the package – "org.compatibility.login", and under this we have 3 classes – Gmail_Login_Chrome, Gmail_Login_FF and Gmail_Login_IE. Similarly we can create the methods for each functionality. The advantage of creating use case specific methods and classes is that we can re-use these methods in multiple test cases. By providing different input data, same tests can be run again, thus speeding up the automation process and increasing productivity.

Following are screenshots from the abstract base class for all three browsers used for testing.

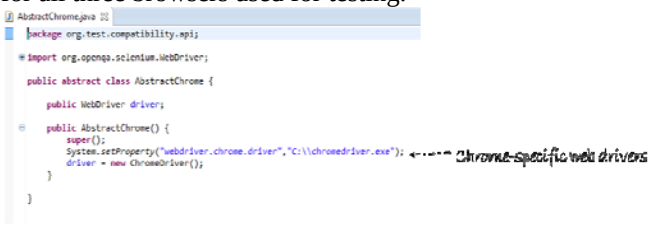

Figure 4: Abstract class for Chrome

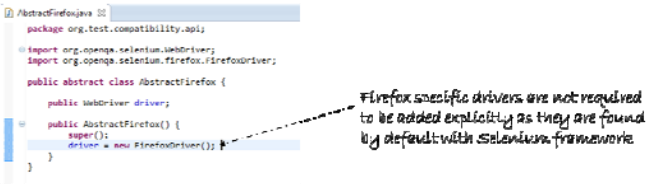

Figure 5: Abstract class for Firefox

Figure 6: Abstract class for Firefox

IV. Result And Discussion

Using this prototype framework, three test scripts were written to perform compatibility testing on Gmail and results were documented as well. All tests were executed on Chrome and Firefox browsers. All test cases were tested successfully. The result includes the total execution time taken by Chrome, Firefox and Internet Explorer browsers. In practice, it is unrealistic to automate everything in a web

application like Gmail by WebDriver. To show feasibility of the approach we have taken the example of the login page of a Gmail account.

The selenium test results are passed to TestNG test framework to create proper readable reports. The main advantage of using TestNG is that test cases can be grouped easily, which in our case can be done for each browser or for each use-case or both. Another advantage is that parallel testing is possible. TestNG is a testing framework that is capable of making easy to understand reports using selenium tests results. Following is the screenshot of the test suite created to run tests for login into a Gmail account from all three browsers:

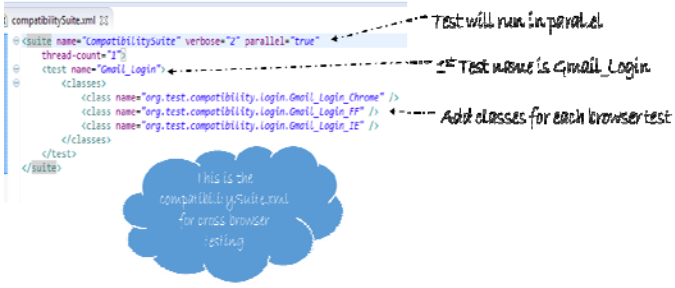

Figure 7: Login Compatibility Test Suite

Following is the html report of the test suite run for all three browsers using TestNG:

Test	# Passed	# Skipped	# Failed	Time (ms)	Included Groups	Excluded Groups
Browser_Compatibility_Testing						
GMail_Login	3	0	0	24,897		

Class	Method	Start	Time (ms)
Browser_Compatibility_Testing			
GMail_Login — passed			
org.test.compatibility.login.Gmail_Login_Chrome	Test	1446464107281	3383
org.test.compatibility.login.Gmail_Login_FF	Test	1446464110666	3719
org.test.compatibility.login.Gmail_Login_IE	Test	1446464114388	5202

GMail_Login

org.test.compatibility.login.Gmail_Login_Chrome#Test

back to summary

org.test.compatibility.login.Gmail_Login_FF#Test

back to summary

org.test.compatibility.login.Gmail_Login_IE#Test

back to summary

Figure 8: TestNG Results of Compatibility

The report shown is of a test suite which included tests for logging into a Gmail account run on all three browsers. It contain the status that is passed or failed along with the execution time taken by particular browser test case. The summary report provides details of execution, duration, test start time and end time. This helps in performing effective analysis on the execution report.

V. Conclusions

Selenium is a framework which comprises of many tools used for testing web applications. In this paper, a prototype of the Compatibility Automation Test Keyword Driven Framework has been created to perform automation testing of web applications using Selenium WebDriver.

VI. Future Scope

As a future scope of this paper, we can extend the framework to be able to test the web application functionality on mobile browsers and possibly different resolutions. Since most people have started using browsers on mobiles to open web pages, it becomes important to ensure that our web application will be compatible with them as well. Web based applications are usually designed for desktop screens with high end processors and memory as compared to mobiles. Mobile devices just don't have that space available. And hence, once are web application is optimized to be accessed on mobiles, it will naturally involve the testing to extend to mobile browsers as well.

Acknowledgment

The author(s) would like to thank Anand Motwani for supporting this research.

References

[1] http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6032495&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F6032121%2F6032438%2F06032495.pdf%3Farnumber%3D6032495
[2] http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=1667589
[3] http://www.ece.ubc.ca/~amesbah/docs/icse11.pdf
[4] http://www.ijascse.org/volume-3-issue-10/Browser_compatibility_testing.pdf
[5] http://www.ijarcsse.com/docs/papers/Volume_4/6_June2014/V4I6-0157.pdf
[6] http://www.softwaretestinghelp.com/types-of-software-testing/
[7] http://gmail.com
[8] http://www.seleniumhq.org/
[9] https://en.wikipedia.org/wiki/Web_testing