Study of Browser Based Automated Test Tools WATIR and Selenium

Nisha Gogna

***Abstract*—Web testing is the name given to software testing that focuses on web applications. Complete testing of a web based system before going live can help address several issues. Manually testing is a time consuming practice and is difficult to repeat but can't be overlooked. Each time a software does not perform to specifications; the program will record and report the exact command that caused the problem. Once the problem is identified and the bug is fixed, one can execute the very same set of commands to verify the success. There are a number of commercial and open source tools available for assisting with the development of test automation. In this paper, automated test tools named WATIR (Web Application Testing in Ruby) and Selenium are proposed to support the automated test scenario for web based applications.**

***Index Terms*—WATIR, ruby, selenium, selenium – IDE, RC, grid, selenium commands.**

I. Introduction

In software testing, test automation is the use of special software (separate from the software being tested) to control the execution of tests and the comparison of actual outcomes to predicted outcomes. Test automation can automate some repetitive but necessary tasks in a formalized testing process already in place, or add additional testing that would be difficult to perform manually [1].

With the development of internet technology, Web application becomes more and more complex and the scale of it changes more and more great. The quality and reliability of it get much more attentions. The Web application programs testing, especially the regression testing is much more difficult than the traditional [2], [3]. Web developers tend to build "beautiful" pages but do not care about accessibility. In Italy where Microsoft Internet Explorer(IE) is used by the vast majority of the common people webmasters tend to program so that the pages are viewed only by this browser(IE), ignoring that in other browsers their pages look badly, or (in some cases) some features are not showed [4].

Consequently research is required to help focussing on Browser compatibility for effectively debugging and testing web applications. Therefore it becomes important to carry out research by providing tools and mechanisms that would help towards Browser based testing. This paper intends to give a contribution in this direction by proposing browser based testing tools named WATIR and Selenium.

Manuscript received October 19, 2013; revised December 19, 2013.

N. Gogna is with the Australian Computer Society (ACS) Australia, India (e-mail: peaceindia86@yahoo.in).

II. WATIR (4.0)

Watir (Web Application Testing in Ruby, pronounced water), is an open-source (BSD) family of Ruby libraries for automating web browsers as specified in Table I.

It drives Internet Explorer, Firefox, Chrome, Opera and Safari, and is available as a RubyGems gem. Watir was primarily developed by Bret Pettichord and Paul Rogers.

Watir is an open-source (BSD) library for automating web browsers. It allows writing tests that are easy to read and maintain in a simple and flexible manner.

Like other programming languages, Ruby gives the power to connect to databases, read data files and spreadsheets, export XML, and structure the code as reusable libraries. Unlike other programming languages, Ruby is concise and often a joy to read [5].

TABLE I: WATIR BASICS

Developer(s)	Bret Pettichord, Charley Baker, Angrez Singh
Stable release	4.0 / September 30, 2012
Development Status	Active
Written in	Ruby (Programming Language)
Operating system	Cross– platform
Type	Software testing framework for web applications
License	BSD license

A. Functionality

Watir makes use of the fact that Ruby has built in OLE capabilities. As such it is possible to drive the Internet Explorer programmatically. Watir operates differently than HTTP based test tools, which operate by simulating a browser. Instead Watir directly drives the browser through the Object Linking and Embedding protocol, which is implemented over the Component Object Model (COM) architecture.

The COM permits interprocess communication (such as between Ruby and Internet Explorer) and dynamic object creation and manipulation (which is what the Ruby program does to the Internet Explorer).

Microsoft calls this OLE automation, and calls the manipulating program an automation controller. Technically, the Internet Explorer process is the server and serves the automation objects, exposing their methods; while the Ruby

program then becomes the client which manipulates the automation objects [5].

B. Ruby

Ruby is a dynamic, reflective, general purpose object-oriented programming language that combines syntax inspired by Perl with Smalltalk-like features. Ruby originated in Japan during the mid-1990s and was first developed and designed by Yukihiro"Matz". It was influenced primarily by Perl, Smalltalk, Eiffel and Lisp.

As given in Table II Ruby supports multiple programming paradigms, including functional, object oriented, imperative and reflective. It also has a dynamic type system and automatic memory management.

The standard 1.8.7 implementation is written in C, as a single-pass interpreted language. There is currently no specification of the Ruby language, so the original implementation is considered to be the *de facto* reference. As of 2010, there are a number of complete or upcoming alternative implementations of the Ruby language, including YARV, JRuby, Rubinius, IronRuby, MacRuby, and HotRuby, each of which takes a different approach, with IronRuby, JRuby and MacRuby providing just-in-time compilation and MacRuby also providing ahead-of-time compilation [6].

TABLE II: RUBY

Paradigm	multi-paradigm
Appeared in	1995
Developer	Yukihiro Matsumoto, et al.
Stable release	2.0.0-p247 (June 27, 2013)
Typing discipline	duck, dynamic, strong
Influenced by	Smalltalk, Perl, Lisp, Scheme, Python, CLU, Eiffel, Ada, Dylan
File extensions	.rb, .rbw

C. Examples

Including Watir gem to drive Internet Explorer on Windows

```
require 'watir'
```

Including Watir-WebDriver gem to drive Firefox/Chrome on Windows/Mac/Linux

```
require 'watir-webdriver'
```

Starting a new browser & and going to the site

```
browser = Watir::Browser.new
browser.goto 'http://bit.ly/watir-example'
```

Setting a text field

Text Field

```
browser.text_field (: name => 'entry.0.single').set 'Watir'
```

Setting and clearing a radio button

Radio Buttons

```
browser.radio (: value => 'Watir').set
browser.radio (: value => 'Watir').clear
```

Checkboxes

```
browser.checkbox (: value => 'Ruby').set
browser.checkbox (: value => 'Python').set
browser.checkbox (: value => 'Python').clear
```

Clicking a button

Button

```
browser.button (: name => 'submit').click
```

Checking for text in a page

```
puts browser.text.include? 'Your response has been recorded.'
```

Checking the title of a page

```
puts browser. Title == 'Thanks!'
```

III. SELENIUM

Selenium is a robust set of tools that supports rapid development of test automation for web-based applications.

TABLE III: SELENIUM BASICS

Stable release	2.33 / May 22, 2013
Development Status	Active
Written in	Java(programming language)
Operating system	Cross platform
Type	Software testing framework for web applications
License	Apache license 2.0
Website	seleniumhq.org

TABLE IV: BROWSER COMPATIBILITY

Supported Operating system	**Supported Languages**	**Supported Browsers**
Windows	C#	Firefox 3
Linux	Java	Firefox
Macintosh	Ruby	IE 8
	Groovy	IE 7
	Python	Safari 3
	PHP	Safari 2
	Perl	Opera 9
		Opera 8
		Others…

Selenium provides a rich set of testing functions specifically geared to the needs of testing of a web application. These operations are highly flexible, allowing many options for locating UI elements and comparing expected test results against actual application behavior. One

of Selenium's key features is the support for executing one's tests on multiple browser platforms [7]. Basics of Selenium are shown in Table III.

A. Supported Browsers

In Selenium 2.0, the supported browsers vary depending on whether we are using Selenium-WebDriver or Selenium-RC as shown in Table IV [8].

B. Selenium Components

Selenium is composed of three major tools. Each one has a specific role in aiding the development of web application test automation.

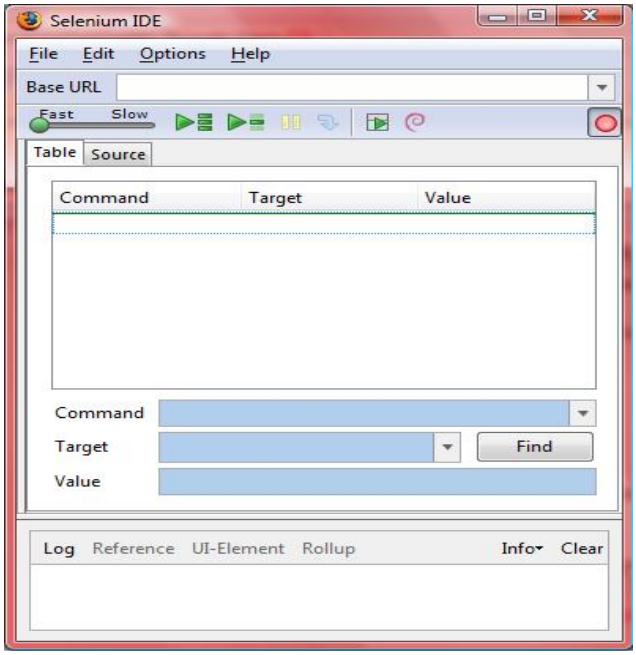

Fig. 1. Selenium IDE.

1) Selenium-IDE

Selenium-IDE is the Integrated Development Environment for building Selenium test cases. It operates as a Firefox add-on and provides an easy-to-use interface for developing and running individual test cases or entire test suites. Selenium-IDE has a recording feature, which will keep account of user actions as they are performed and store them as a reusable script to play back (Fig. 1). It also has a context menu (right-click) integrated with the Firefox browser, which allows the user to pick from a list of assertions and verifications for the selected location. Selenium-IDE also offers full editing of test cases for more precision and control.

Although Selenium-IDE is a Firefox only add-on, tests created in it can also be run against other browsers by using Selenium-RC and specifying the name of the test suite on the command line.

2) Selenium-RC (remote control)

Selenium-RC allows the test automation developer to use a programming language for maximum flexibility and extensibility in developing test logic. For instance, if the application under test returns a result set, and if the automated test program needs to run tests on each element in the result set, the programming language's iteration support can be used to iterate through the result set, calling Selenium commands to run tests on each item as illustrated in Fig. 2.

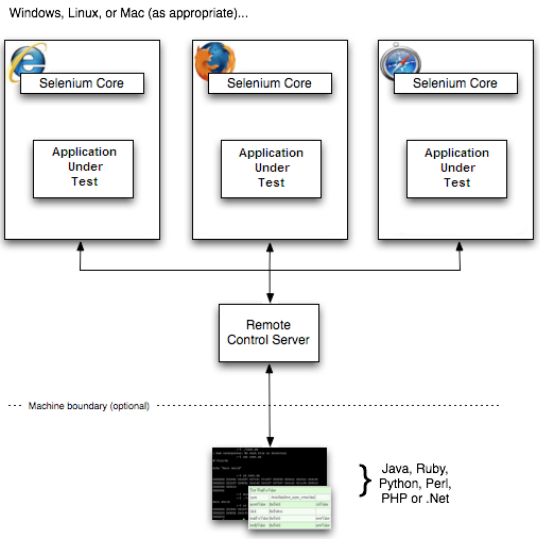

Fig. 2. Architecture of selenium-RC.

3) Selenium-grid

Selenium-Grid allows the Selenium-RC solution to scale for large test suites or test suites that must be run in multiple environments. With Selenium-Grid, multiple instances of Selenium-RC are running on various operating system and browser configurations. Each of these when launching register with a hub. When tests are sent to the hub they are then redirected to an available Selenium-RC, which will launch the browser and run the test. This allows for running tests in parallel, with the entire test suite theoretically taking only as long to run as the longest individual test.

C. Selenium Commands

Selenium provides a rich set of commands for fully testing a web-app in virtually any way. The command set is often called selenese. These commands essentially create a testing language. In selenese, one can test the existence of UI elements based on their HTML tags, test for specific content, test for broken links, input fields, selection list options, submitting forms, and table data among other things. In addition Selenium commands support testing of window size, mouse position, alerts, Ajax functionality, pop-up windows, event handling, and many other web-application features [7].

A command is what tells Selenium what to do. Selenium commands come in three "flavors": Actions, Accessors and Assertions.

- *Actions* are commands that generally manipulate the state of the application. They do things like "click this link" and "select that option". If an Action fails, or has an error, the execution of the current test is stopped.
- *Accessors* examine the state of the application and store the results in variables, e.g. "storeTitle".
- *Assertions* are like Accessors, but they verify that the state of the application conforms to what is expected. Examples include "make sure the page title is X" and "verify that this checkbox is checked".

D. Script Syntax

Selenium commands are simple; they consist of the command and two parameters as given below. For example:

Verify text	//div//a[2]	login

The parameters are not always required; it depends on the command. In some cases both are required, in others one parameter is required, and in still others the command may take no parameters at all.

Parameters vary, however they are typically

- a locator for identifying a UI element within a page
- a text pattern for verifying or asserting expected page content
- a text pattern or a selenium variable for entering text in an input field or for selecting an option from an option list

E. Test Suites

A test suite is a collection of tests. Often one will run all the tests in a test suite as one continuous batch-job. When using Selenium-IDE, test suites also can be defined using a simple HTML file. The syntax again is simple. An HTML table defines a list of tests where each row defines the filesystem path to each test.

An example tells it all.

```
<html>
<head>
<title>Test Suite Function Tests - Priority 1</title>
</head>
<body>
<table>
<tr><td><b>Suite Of Tests</b></td></tr>
<tr><td><a href= "./Login.html" >Login</a></td></tr>
<tr><td><a href= "./SearchValues.html" >Test Searching for Values</a></td></tr>
<tr><td><a href= "./SaveValues.html" >Test Save</a></td></tr>
</table>
</body>
</html>
```

A file similar to this would allow running the tests all at once, one after another, from the Selenium-IDE.

F. Commonly Used Selenium Commands

- **open**: opens a page using a URL.
- **click/clickAndWait**: performs a click operation, and optionally waits for a new page to load.
- **verifyTitle/assertTitle**: verifies an expected page title.
- **verifyTextPresent**: verifies expected text is somewhere on the page.
- **verifyElementPresent**: verifies an expected UI element, as defined by its HTML tag, is present on the page.
- **verifyText**: verifies expected text and it's corresponding HTML tag are present on the page.
- **verifyTable**: verifies a table's expected contents.
- **waitForPageToLoad**: pauses execution until an expected new page loads. Called automatically when clickAndWait is used.
- **waitForElementPresent**: pauses execution until an expected UI element, as defined by its HTML tag, is present on the page.

IV. Conclusion

This paper presented the most basic features of Browser based automated test tools Watir and Selenium for a web application.

In all one have to learn Ruby (unless you choose Watij or Watin) first for Watir and every browser requires a different library for running test cases while Selenium has its own IDE and can record and playback tests. But for Selenium one has to learn a vendorscript first that is Selenese unless tests have to be written in any other language and has trouble recording IFrames, Frames and popup windows while Frames and pop-ups are accessible using API in WATIR.

Also, selenium has Deep learning curve to switch from Selenium IDE to Selenium RC as compared to Watir. To tackle this problem, future work is required to be done to cover the limitations of one tool with respect to another.

References

[1] A. Kolawa, D. Huizinga. (2007). Automated Defect Prevention: Best Practices in Software Management. *IEEE Computer Society Press*. ISBN 0-470-04212-5. [Online]. pp. 74. Available: http://www.en.wikipedia.org/wiki/Test automation.

[2] F. Coda, C. Ghezzi, G. Vigna *et al*., "Towards a Software Engineering Approach to Web Site Development," in *Proc. the Ninth International Workshop on Software Specification and Design*, Ise-Shirna Japan: IEEE Computer Society Press, April 16-18, 1998, pp. 8-17.

[3] J. T. Yang, J. L. Huang, F. J. Wang, and W. C. Chu, "An Object-Oriented Architecture Supporting Web Application Testing," in *Proc. IEEE 23rd Annual International Computer Software and Application Conference (COMPSAC2000)*, Phoenix Arizona USA: IEEE Computer Society Press, Oct 2000, pp. 122-127.

[4] M. Kirchner, "A Benchmark for Testing the Evaluation Tools for Web Pages Accessibility," in *Proc. the Fifth IEEE International Workshop on Web Site Evolution (WSE'03)*, 2003.

[5] Watir home page. Watir web site. (11 October 2012). [Online]. Available: http://www.Watir - Wikipedia, the free encyclopedia.htm

[6] About Ruby. Ruby-lang.org. (Nov. 29, 2001). [Online]. Available http://www.Ruby (programming language)-Wikipedia, the free encyclopedia.htm.

[7] Selenium Documentation Release 1.0, Selenium Project. (February 24, 2010). [Online]. pp. 3-164. Available: http://www.seleniumhq.orgs/docs.

[8] Selenium Tutorial for Beginners (c), Appvance, as PushToTest Inc.. (2012). Frank Cohen. [Online]. Available: http://www.SeleniumTutorial for Beginners - Tutorial 4.htm

Nisha Gogna is currently an associate of Australian Computer Society (ACS). She was born in India on 24 April 1987. She received her bachelor of technology – with Distinction (information technology) Punjab Technical University, Punjab, India (2004-2008). She obtained her master of engineering (information Technology) Panjab University, Chandigarh, India (2009-2011). She has completed her 6 months of research on the topic, "Comparative Study of Browser Based Open Source Testing Tools Watir And Wet".

Being an assistant professor she holds an adequate experience in the field of Engineering alongside the publication of following research papers: 1) Study of Browser Based Automated Test Tools Watir and WET," International Journal of Information Sciences and Application, ISSN 0974-2255 vol. 2, no. 2 (2010), pp. 289-293. 2) "Advancing Towards Automated Testing through Effective Manual Testing," Engineering Sciences, 5th Chandigarh Science Congress, CHASCON 2011. 3) "Comparative Study Of Browser Based Open Source Testing Tools Watir And Wet," International Journal on Computer Science and Engineering (IJCSE11-03-05-128), vol. 3 no. 5 May 2011, ISSN : 0975-3397, pp. 1910-1923.

Ms. Nisha Gogna is a current Associate of Australian Computer Society (ACS), Australia.