Automation Testing: An Introduction to Selenium

Himanshi
Assistant Professor
CT Group of Institutions, Punjab

Nitin Umesh
Assistant Professor
Lovely Professional University, Punjab

Amar Saraswat
Assistant Professor
Dronacharya College of Engineering, Gurgaon

ABSTRACT
A good tool for testing is the one which has the ability to augment the testing performance and its competences as well as that can be a part of automation. When we chooses to build and host a Web application; automated regression testing becomes a responsibility for accomplishment. As the need of Web applications are growing, with each subsequent release, testing needs to grow tremendously. In this paper we will discuss the introduction to regression testing for Web applications and the Selenium testing framework. Selenium is open source and portable software available for different windows.

Keywords
Acceptance Testing, Manual Testing, Selenium, Web Application Testing

1. INTRODUCTION
Apart from accomplishing a variety of things testing also measures the quality of the application. Testing presumes that there are faults in the software that has to be discovered and this is rarely refuted.

Manual testing is the ancient and most demanding way of testing the software which necessitates a tester to accomplish manual test processes to test the software. However it is a lengthy activity that requires the tester to possess a certain set of qualities like to be tolerant, Vigilant, Dicey, Imaginative, Inventive, Progressive, Creative and Dexterous.

Nevertheless repetitive manual testing is difficult to perform on large software applications or applications having very large dataset coverage. So In software testing, test automation is the easiest way to automate some repetitive and necessary tasks in a formalized testing process that are otherwise difficult to perform manually.

After the development of automated tests, they can be run rapidly and frequently. This can be a cost-effective method for regression testing of software products that have a long maintenance life. Even minor changes over the lifetime of the application can cause features to break which were working at an earlier point in time. So test automation has become very crucial for the software now days.

There are two common methodologies for test automation:

Code-driven testing. The interfaces to classes, modules or libraries are tested with a range of input arguments to validate that the results that are reverted are precise.

Graphical user interface testing. A testing framework creates user interface events such as mouse clicks and keystrokes and perceives the variations that result in the user interface, to validate that the visible behaviour of the program is precise. [1]

2. BACKGROUND

2.1 Acceptance Testing
A Formal testing with respect to user requirements, necessities, and business processes directed to determine whether or not a system fulfills the acceptance benchmarks and to enable the user or customers to determine whether or not to receive the system. Unlike unit tests, acceptance tests take the form of a step-by-step script that acceptance testers walk through while sitting in front of the application under test. Tests show their true power only when run as automatic regression tests. For Java environment, GUI testing tools comprise Jemmy, SWTBot, and Abbot, and tools for testing Web applications comprise HtmlUnit or HttpUnit . In general, these tools come in two flavours: capture and-replay and programmatic.

Capture-and replay tools are great for confirming that a certain scenario leads to the identical results that it ensured previously. We perform this activity by recording a user's actions with the application and replaying those actions in the application. We can readily specify tests that way, but the problem begins if the scenario or the software changes

Programmatic tests take more time to expertise, but making amendments doesn't cause major troubles. These tests are more flexible and can be easily written without the application under test being really there.

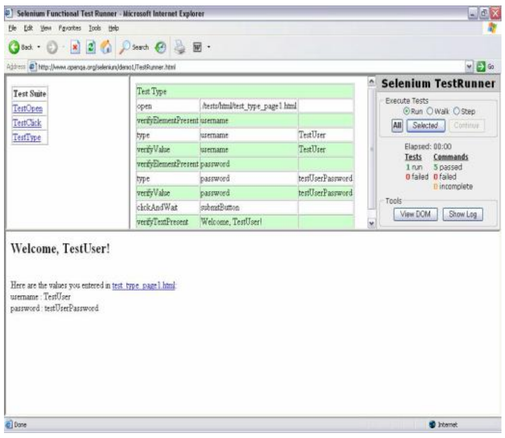
2.2 Test Automation
Many test automation tools provide record and playback features that let the users to interactively record actions and replay them back several times, comparing actual results with the expected results. The benefit of this methodology is that it needs minute or almost no development of the software. This methodology is applicable to any GUI application. However, dependence on these features poses major problems w.r.t. reliability and maintainability. Tests need to be re-recorded even for the minor changes in the application like relabeling a button or moving it to another part of the window. Record and playback also frequently adds inappropriate activities or erroneously records some activities. Most of the software applications today are web-based applications that require Internet browser to run. The efficacy of testing these applications differs widely among companies and organizations. Test automation has become a requirement for software projects that are highly interactive and responsive. Software processes in many organizations are using some form of agile methodology so test automation is frequently becoming a requirement for software projects. Test automation is time and again the only solution. Test automation means using an automated tool to run the tests for the application to be tested.

There are many advantages of test automation. Most are related to the repeatability of the tests and the speed at which the tests can be executed. There are a number of viable and open source tools available for supporting the test automation and selenium is regarded as the efficient and the most widely-used open source solution.

Test automation has specific advantages for improving the long-term efficacy of a software team's testing processes. Automation of testing provides:

- Repeated regression testing
- Quick feedback to the developers
- Indefinite repetitions of test case execution
- Support for the development methodologies like Agile and extreme development
- Orderly documentation of test cases
- Customized reporting of the defects
- Finding defects neglected during manual testing

For a web application, a functional test could be simply that a user manually navigates through the application to verify the application behaves as expected. But since automating a test is the best way to make sure it is run often, we should automate our functional acceptance tests whenever we can. We studied the open source tools like Canoo WebTest and HttpUnit, but found them insufficient, as they could not handle most instances of in-page JavaScript. The JavaScript problem is solved by the proprietary tool quickTestProfessional, a Mercury Interactive tool, which offer record-and-play of test scripts and also runs tests directly in a browser. However, we found that most of the recorded scripts in QTPro would break after small page changes, so a significant amount of script maintenance was necessary. As a result, we turned to Selenium – with which it was easy-to-write scripts that were relatively easy to maintain and which could be written before the code. To provide automated acceptance testing of Web applications, particularly those using Ajax, the set of open source Selenium tools are there (see Table 1). With these tools we can easily run tests in the Web browsers.


Selenium Functional Test Runner - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Address http://www.openqa.org/selenium/demo1/TestRunner.html
Go
Selenium TestRunner
Execute Tests
Run Walk Step
All Selected Continue
Tools
View DOM Show Log
Done
Internet

Test Suite
TestOpen
TestClick
TestType

Test Type		
open	/tests/html/test_type_page1.html	
verifyElementPresent	username	
type	username	TestUser
verifyValue	username	TestUser
verifyElementPresent	password	
type	password	testUserPassword
verifyValue	password	testUserPassword
clickAndWait	submitButton	
verifyTextPresent	Welcome, TestUser!	

Elapsed: 00:00

Tests	Commands
1 run	5 passed
0 failed	0 failed
	0 incomplete

Welcome, TestUser!

Here are the values you entered in test_type_page1.html:
username : TestUser
password : testUserPassword

Fig. 1. Selenium in Internet Explorer.

Tool	Purpose
Selenium Core	Modify and check an Ajax application using commands in Selenese, Selenium's control language.
Selenium RC	Remotely control Selenium Core using a common programming language.
Selenium Grid	Use several remote controls in parallel to expedite testing.
Selenium IDE	Capture and replay tests from within Firefox.

Table 1: Tools Required

2.3 Selenium: A testing Tool

Selenium is an automation tool for web based applications, which is generally used for functional regression testing. Selenium is a web testing tool which uses simple scripts to run tests directly within a browser. It uses JavaScript and iframes to embed the test automation engine into the browser. [3] This allows the same test scripts to be used to test multiple browsers on multiple platforms.

Selenium gives the user a standard set of commands such as open (a URL), click (on an element), or type (into an input box); it also provides a set of verification commands to allow the user to specify expected values or behaviour. The tests are written as HTML tables and run directly in the browser, with passing tests turning green and failing tests turning red as the user watches the tests run. Because Selenium is JavaScript-based and runs directly in the browser (the user can see the test running), it overcomes some of the problems encountered by users of HttpUnit or Canoo WebTest, particularly problems related to testing JavaScript functionality.

It uses java script and can embed test automation engine in your browser. It is basically a set of different software tools each with a different approach to supporting test automation. Most Selenium Quality Assurance Engineers focus on the one or two tools that mostly meet the necessities of their project, nevertheless learning all the tools will give many diverse options for approaching different solution to different test automation problems. The complete suite of tools results in a rich set of testing functions specifically required to the needs of testing all types of web applications. These operations are very much flexible and give different options to locate User Interface elements and matching expected test results with the real application behaviour. One of Selenium's key features is that it supports execution of tests on multiple browser platforms. Selenium comprises of multiple software tools each having a specific role [2].

Selenium2 (Selenium Webdriver) is the future direction of the project and the newest addition to the Selenium toolkit. This tool provides all kind of remarkable features, including a more cohesive and objects oriented API as well as an answer to the limitations of the old implementation. Selenium 2.0 is the product of that power. It can also support the WebDriver API and underlying technology, along with the Selenium 1 technology underneath the WebDriver API for maximum flexibility in porting the tests. In addition, for backward compatibility Selenium 2 still runs Selenium 1's Selenium RC interface.

Selenium 1 (Selenium RC or Remote Control) is still actively supported (mostly in maintenance mode) and provides some features that may not be available in Selenium 2 for a while, including support for several languages (Java, Javascript, PRuby, HP, Python, Perl and C#) and support for almost every browser.

Selenium IDE (Integrated Development Environment) is a prototyping tool used to build the test scripts. It is a plug in for the Firefox browser and provides an easy-to-use interface to develop automated tests. One more important feature of Selenium IDE is its recording feature. This feature records user actions as they are implemented and then disseminates them as a reusable script in one of various programming languages that can be executed later on also. [3]

Selenium-Grid allows the Selenium RC solution to scale for large test suites and for test suites that must be run in multiple environments.[3]

2.3.1. Getting Started
Setting up Selenium is easy, although there is a catch. The basic installation of Selenium must be hosted by the same web server as the Application under Test (AUT). This restriction is due to the fact that JavaScript has built-in security against cross-site scripting. [3] After installation, the user simply needs to begin writing and running tests.

2.3.2 Writing Good Tests
Selenium tests are not difficult to write. Because Selenium allows the identification of elements using the browser's DOM object, the test can be written using specific identifiers of the necessary element, such as name, id, or xpath:

type	name=theField	Text to submit
clickAndWait	Id=SubmitButton	
assertText	Xpath=//h1/span	Success!!!

Table 2: Input Data for a Selenium Test

This test for an HTML page can be written in an easy way as:

```
<html>

<input name="theField">

<input id="SubmitButton" type="submit">

<h1><span>Success!</span></h1>

</html>
```

When the test is run, each command is highlighted as it is executed, and the asserted steps either turn red or green to indicate the success or failure. The test is marked as red or green in the Suite after the completion of the thorough test.

2.3.3. Keeping Tests Self-Contained
Selenium naturally supports a Suite of tests, and tests for a certain story or iteration can be grouped together in a Suite and run sequentially. But for flexibility and maintainability, we strove to keep our tests as independent and self-contained as possible. This allowed us to move tests around and delete them, as well as re-factor tests remorselessly. We frequently included one test inside many others to reduce code duplication, and we used SetUp and TearDown tests throughout our suites in order to get the application to the desired state for testing one piece of functionality.

In fact, we found that our test-writing style became:

1. Write the tests before development;

2. After development, get them to green; then,

3. Refactor as mercilessly as possible.

By refactoring our tests, we reduce duplication of test code and increase the maintainability of the test suite

2.3.4. Selenium is Open-Source
We began using Selenium at version 0.5. Even today, it is only at a 0.6 release, and is under active development. Thus, certain stability and completeness issues were to be expected. One of our team members became an active contributor to the user forums, and steady monitoring of the user and developer forums for patches and usable extensions paid off. Our recommendation is to remember that Selenium isn't (or at least doesn't have to be) considered a black box. By understanding, modifying, and extending its functionality, we were able to make our tests more readable and to handle multiple-step actions with one command.

2.3.5. Test first?
Because Selenium tests are easy to write, a tester or analyst can write the shell of a Selenium test very quickly without knowing what the implementation will be. Although we expected that we could code to these tests, the test would seldom turn green after development. The reasons for this were usually minor: a field wasn't naturally identified by the name the tester chose, or the test command used needed to be clickAndWait instead of just click, etc. As a result, we did not usually require that the developer code to the test, and our process of writing the test before development (for its specification value), but getting the test green immediately after development, emerge.

3. CONCLUSION
Selenium is an open-source tool for in-browser testing, originated by just thoughts and now it is being used actively by both the developers and users. One of the Selenium's objectives is to become the authentic open-source replacement for the named tools like WinRunner. The most important feature of this tool is that it also facilitates the testing of web applications along with signalling of red-green signals for both customer acceptance tests and an automated regression tests. Selenium tool is certainly an asset for anyone who is looking to add a powerful web testing tool to their toolkit.

4. REFERENCES
[1] Niranjanamurthy M, Arun Kumar R, Sahana Srinivas, Manoj RK," Research Study on Web ApplicationTesting using Selenium Testing Framework" IJCSMC, Vol 3,Issue 10, October 2014,pg121-126

[2] Test automation. From Wikipedia http://en.wikipedia.org/wiki/Test_automation

[3] Test Automation for Web Applications. http://seleniumhq.org/docs/01_introducing_selenium.html

[4] Selenium, (Web Site: http://www.openqa.org/selenium)

[5] FitNesse (Web Site: http://www.fitnesse.org)