Exploring Web Testing Tools For Use In A Classroom

Brian Valerius and Elena Machkasova (Advisor)
Computer Science Discipline
University of Minnesota Morris
Morris MN, 56267
valer028@umn.edu, elenam@umn.edu

Abstract

Testing frameworks for web applications are becoming more and more common. In this paper I will describe a new testing system called Selenium (an open source project hosted by googlecode.com) that makes web application testing faster, more convenient, and also repeatable. Selenium is a suite of tools used to automate web application testing that can potentially be used in a classroom environment. It has many great features that are applicable to students in web development classes. It can also be used by a variety of levels of web development courses for students. Students with little, if any, web development experience, and also students with substantial amounts of web development experience could find Selenium useful.
My main research into Selenium is directed towards finding a tool to get students familiar with a resourceful testing framework, and also helping them learn in the process. Selenium-IDE, the main focus of my research, is a Firefox plug-in that allows the user to create and play back tests in the web browser. Tests can also be created by hand and exported to multiple programming languages. They can be written in the IDE and used by a teaching assistant or a professor to provide an easier experience for lower-level web development classes. In this paper I give examples and illustrations of uses of Selenium-IDE that are suitable for different levels of student backgrounds.

1 Introduction and Background

1.1 Motivation

In recent years there have been numerous tools developed for testing web applications. The main idea behind my research was to find a tool to help varying levels of students with testing their web applications in a classroom environment. If students were able to spend less time on web testing they would have more time to focus on actually understanding and implementing the code.

At the University of Minnesota, Morris we have two groups of web development classes. There are entry-level web development classes that are accessible to non-majors and there are upper-level classes that are for computer science majors with more programming experience. In an entry-level course, pre-written test cases could be less time consuming for students and the instructor. Spending less time on web testing could give students more time to figure out the given problem, and also help them decide which methods need to be implemented, therefore reducing trial and error. Upper-level students could use automated web testing tools to quickly make test cases for any given problem. With entry-level courses, pre-written tests could potentially make grading easier for an instructor or a teaching assistant (T.A.). Even though upper-level course students may be writing their own tests, having a pre-defined test suite to grade off of could help the instructor or T.A. with grading. Being able to quickly automate web testing would give the students extra time to create a more enriched web experience.

For both entry-level and upper-level web development courses, using automated web testing tools could promote consistency. Being able to constantly automate tests would mean more consistent results, and a better understanding of how important testing is. Consistent testing could also lead to a more powerful learning environment for students. It is possible that through tests, students might be able to better understand various other aspects of web development, such as XPath.

Through my research into web testing tools I found Selenium [1] to be one of the most practical tools for classroom use. Selenium is a suite of tools that supports test automation for web-based applications. It has a large number of testing functions designed for the needs of web developers. Selenium allows for many options for locating web elements, and for comparing expected test results against actual developed results. It can be used to make some aspects of web testing easier and less time consuming.

1.2 Overview of Selenium

Selenium is made up of three main components. One component is Selenium-IDE, the one I have been researching for use in a classroom environment. It operates as a Firefox add-on and has an easy-to-use interface for developing and running tests for web applications. The other two Selenium components are Selenium-RC (Remote Control) and Selenium-Grid. Selenium-RC allows the developer to use a high-level programming language for developing test logic. Selenium-Grid allows for multiple instances of Selenium-RC to be run on various operating systems and browser configurations. Selenium-RC and Selenium-Grid turned out to be more complex than the IDE and required a better understanding of servers and client drivers for students. If they were set up and configured properly they may be beneficial in different ways from the IDE. In this paper, my main focus will be on the IDE.

2 Detailed Description of Selenium-IDE

2.1 Installation

The installation procedure for Selenium-IDE is quite simple. It consists of going to the SeleniumHQ website and downloading the IDE. It automatically installs the IDE as a Firefox plug-in and is configured to be used immediately after a restart of the web browser. The IDE can be installed on any operating system that Firefox is installed on.

2.2 Main Features

Selenium-IDE is the Integrated Development Environment for building Selenium test cases within the Firefox browser. Although the IDE is only a Firefox add-on, tests developed in it can be run against other browsers using Selenium-RC. The test cases can be used in the IDE, or exported to multiple programming languages. It uses specific Selenium commands called *selenese*. The developer can use and modify tests, and also put them into an existing testing framework. Selenium-IDE can export tests to many programming languages, including Ruby, PHP, Java, Perl, C#, and Python. Most exported tests need to be run in Selenium-RC, but Selenium does provide other ways to use the exported tests, such as Selenium on Rails, Selenium on Ruby, and CubicTest (for Eclipse).

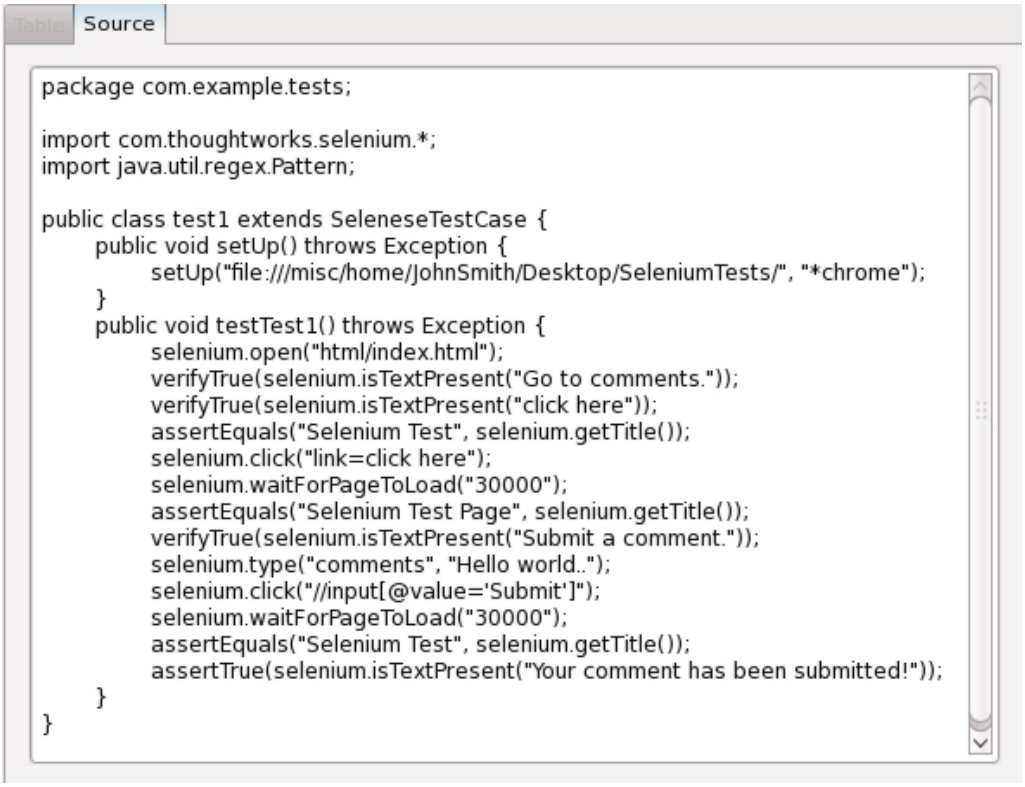

Figure 1: A test formatted to Java.

Selenium can compare expected test results to actual ones. This works well for most test

cases, such as verifying if certain text is present on a page, or asserting that a certain element is present. A `verify` command in Selenium is a boolean statement that compares an expected test result with an actual test result, and displays whether the comparison was true or false. The `assert` command works the same way except that if an `assert` command returns false the entire testing framework halts, whereas the `verify` command will allow the framework to continue on with the rest of the tests but displays that the a certain test failed. The IDE has a simple and easy-to-understand interface that can make developing tests and test suites quite easy.

2.3 Advantages of Selenium

As previously mentioned (Section 1.1), the idea behind my research into a web application testing tool was to satisfy the needs of both entry-level students and upper-level students. Selenium has a powerful enough suite of tools to get the job done. The IDE is practical enough to be used by varying levels of web development students and appears to be the correct choice for an easy-to-use web application testing tool. The easy installation (see Section 2.1) also helps to make Selenium-IDE a good choice for web development classes.

2.4 What Selenium-IDE Can Do

Selenium-IDE provides multiple easy ways to create tests and test suites. One useful feature for students is the record and playback feature. On start-up of the Firefox plugin, the record feature is automatically turned on, allowing the user to record any action done inside the web page.

Figure 2: Selenium-IDE toolbar. The button farthest to the right is the record button.

This works well for actions such as clicking a link or button with the mouse, typing text into a text field with the keyboard, or submitting a form. The user can then play back any recorded action in the web browser. It can also record actions such as selecting options from a drop-down list, and clicking checkboxes or radio buttons. With Selenium-IDE recording, the user can right-click anywhere on a web page and see a context menu showing verify and/or assert commands. This works well for quick commands such as `verifyTextPresent`, `assertTitle`, and `assertElementPresent`. Clicking on any of these items in the context menu will automatically insert the command into the IDE. This kind of simple testing could be easy to understand by entry-level web development students.

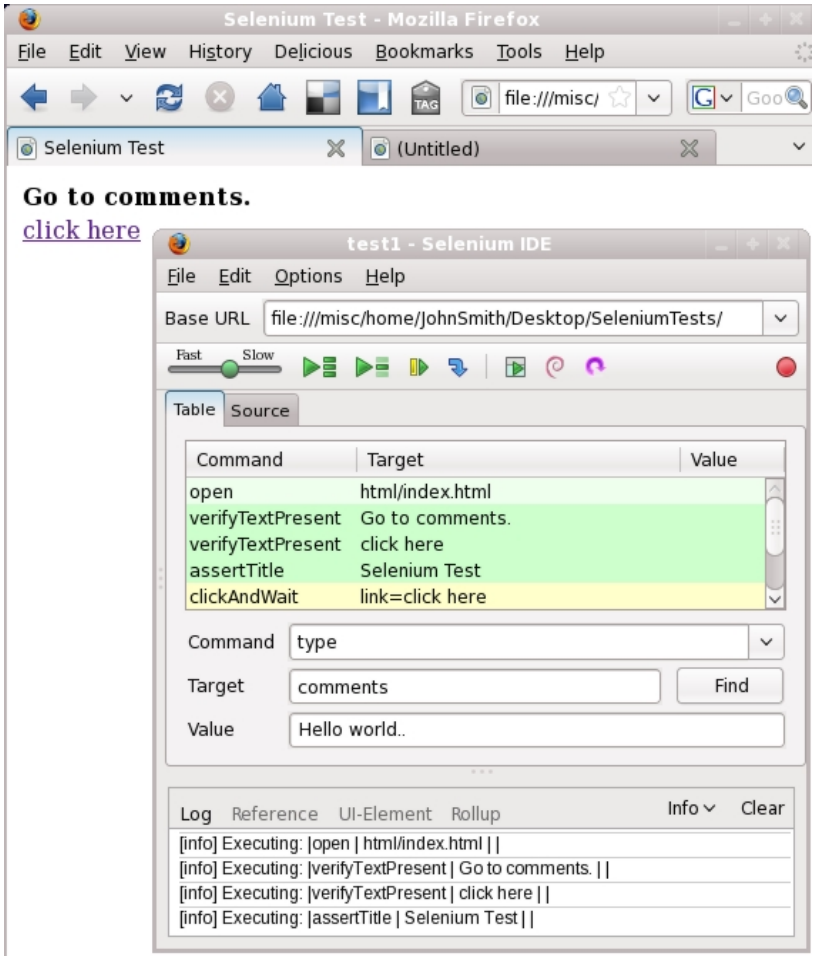

Figure 3: A test running to verify text is present, and also to assert the title on the webpage is correct. It was paused before clicking on the link.

One helpful part of the IDE, for both experienced users and unexperienced users, is the autocomplete feature for all common Selenium commands (see Figure 4). Selenium-IDE also provides a way to debug and set breakpoints to make error handling easier. In addition, the user can simply walk through tests step-by-step to help with debugging.

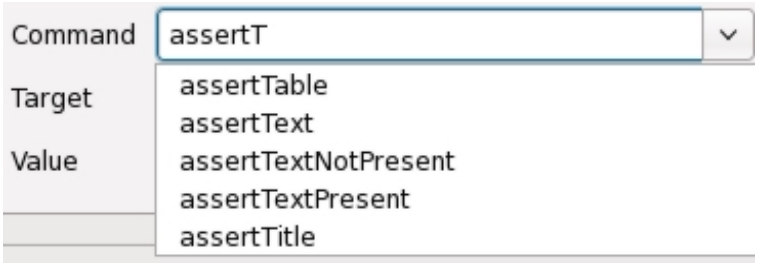

Figure 4: The autocomplete feature.

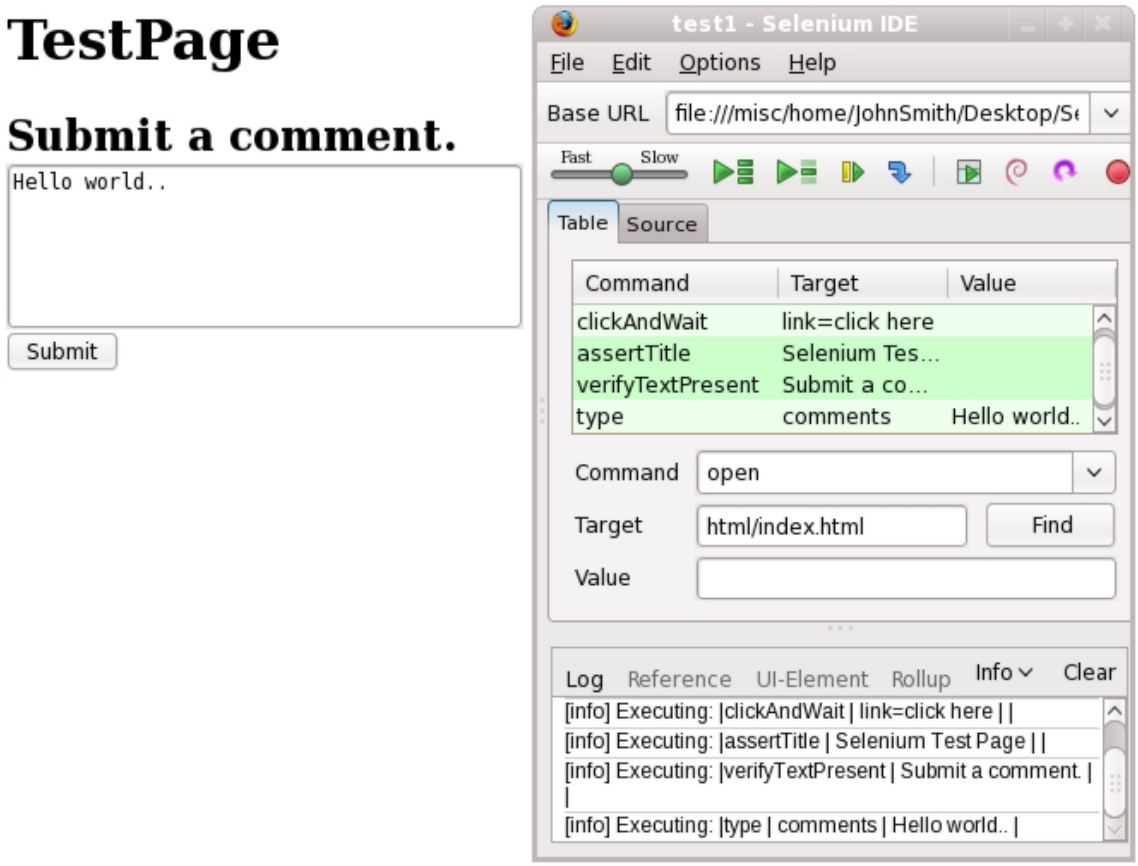

Figure 5: A test running that clicks on a link, asserts that the title of the new page is correct, verifies the correct text is present on the page, and also types in the textbox ”Hello world..”.

Another nice feature of Selenium is the ability to store constants into variables at the beginning of a script. These variables can be used any number of times throughout the test case. Selenium can store text, and also store the boolean value corresponding to a `verifyElementPresent` command.

Test cases can be created, opened, and saved through the File menu. One great part of the IDE is the ability to save several test cases into test suites. This allows the user to separate test cases out into groups of test cases to promote organization. This can work well for students who have multiple assignments that they are testing.

One interesting feature under the Options menu is to have the IDE automatically insert an `assertTitle` command to every new test case created which promotes consistency. Another feature under the Options menu is the tab Formats. It is designed to allow the user to add their own formats for exporting tests to the programming language of their choice. This is more advanced and probably would not be used by a student in a web development course, but it is still an interesting feature.

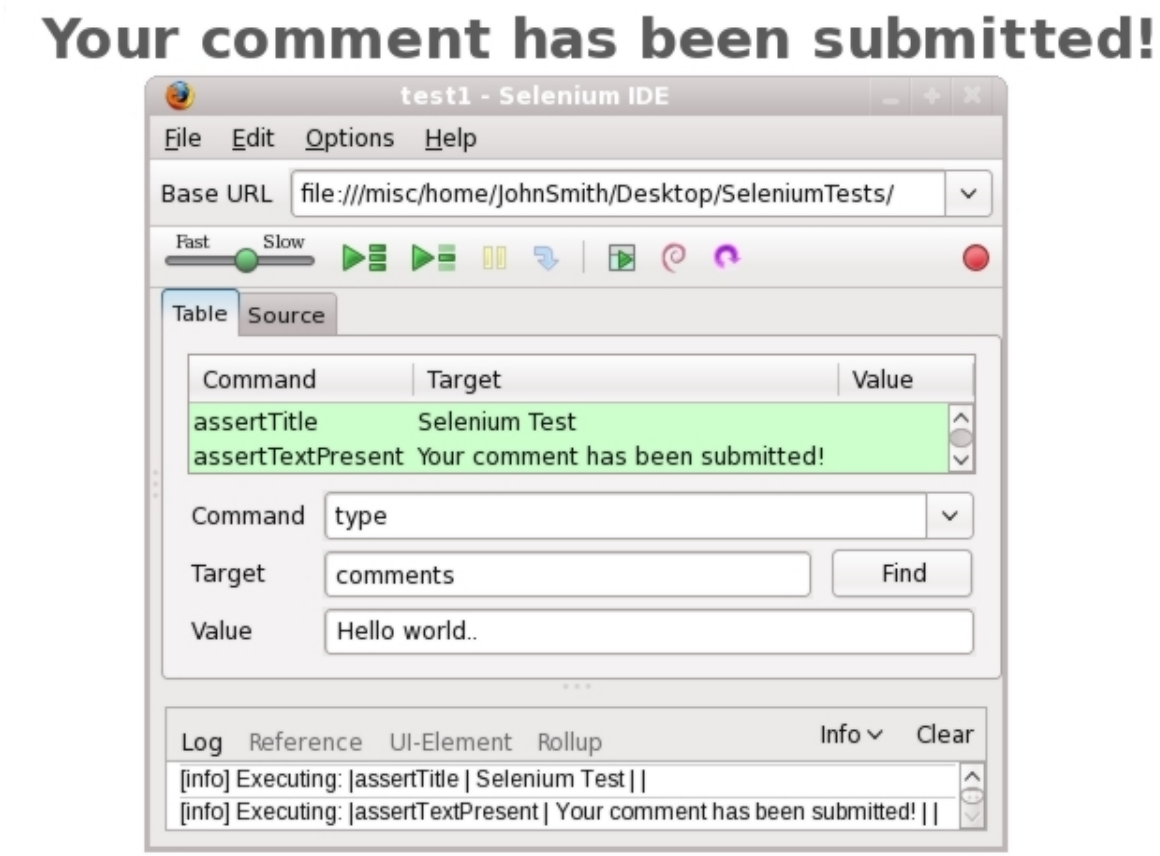

Figure 6: A test running that utilizes the record assertTitle automatically feature, and asserts that the correct text is displayed on the page.

2.5 Finding Elements in a Page

There are a few ways to find an element in a web page using Selenium-IDE. When the IDE records a locator-type argument, it also stores additional information which can allow the user to view other possible locator-type arguments that could be used instead. Elements can be located through the IDE with `id` tags, but this is only useful when an `id` is actually set. An `id` is an attribute that specifies a unique name for a HTML element on a web page. An example of an `id` for the second header from Figure 5 would look like this:

```
<h2 id="submitHeader">Submit a comment.</h2>
```

One really neat way to locate an element in Selenium-IDE, even when an `id` is not set, is through XPath. XPath is a language for finding information in several types of web documents. It can be thought of as a regular expression for web pages. For example, the XPath of the text field from the web page in Figure 5 would look like:

```
/html/body/form/textarea
```

This line of XPath looks through the html, down the body, into the form, and at the text field. An example of XPath getting an element with a particular `id` would look like:

```
//*[@id='elementID']
```

This line of XPath looks through all elements until it finds one with the `id` elementID.
A great tool to help with XPath is the Firefox extension XPath Checker written by Brian Slesinsky [2]. This helpful tool allows the user to right-click anywhere on a webpage and get the XPath of that element. Entry-level students can use XPath Checker to quickly find the XPath of an element, and also learn how XPath works in the process. Upper-level students who may already understand XPath can use the tool to simply save time. Regular expressions can be used in Selenium-IDE in conjunction with XPath for a more powerful experience.

3 Conclusions and Future Work

Through my research into Selenium-IDE I have found the tool to be practical and applicable to web development courses. The tool can not only be used as an easy-to-understand testing framework, but it also serves as a learning tool for students. It can be used by all ranges of skill levels. It helps promote consistency and a better understanding of web application testing frameworks. The IDE has a powerful enough language to get testing done, and also has a set of unique features. It not only benefits students, but also has the potential to help instructors or teaching assistants as well.
The next step in this research with Selenium is to see how well the IDE will work in a web development class. This can be accomplished by having an instructor of a suitable course utilize the tool. In an entry-level course this might work as a way to assist grading of student work. Pre-written test cases can be developed and run against student code to check if it satisfies the given test. Students could also run these pre-written test cases to check if their own code satisfies the requirements. Upper-level course students could be given a brief explanation of Selenium-IDE and give feedback if or when they use it.
Using Selenium in a classroom will require measuring if the tool is actually beneficial. Measuring the amount of time it takes students to complete assignments will be useful. Looking at how much upper-level students understand about testing frameworks will measure if they have gained any knowledge from using the tool. All of this and more will be needed to figure out whether or not using Selenium-IDE will benefit a web development course.

References

[1] OPENQA. *Selenium*. http://seleniumhq.org.

[2] SLESINSKY, B. *XPath Checker*. http://code.google.com/p/xpathchecker/.