A Study on Functioning of Selenium Automation Testing Structure

1Jyoti Devi, 2Kirti Bhatia, 3Rohini Sharma*

1, 2 Sat Kabir Institute of Technology and College Management Bahadurgarh, Haryana, India
3 Jesus and Mary Delhi University Delhi, India

DOI: 10.23956/ijarcsse/V7I5/0204

Abstract—In recent years, there is advancement in the field of software engineering, applications are getting automated. As the software is error prone, there should some easy and automated way to test the software application. It is very challenging to test such complicated web applications. Human intervention can be reduced by using automation tools. Selenium is an online web based software testing tool. In this article, we have study the design and functioning of the selenium tool and used it to test various online applications. In order to test an application, testers do need not to learn the selenium web driver tool completely. This tool is helpful for a developer and tester, they can analyse their code owing to screen shot characteristics of framework. It generates the adapted test report to the tester. It is very simple to sustain and restore the test suite for new version of the application by this tool.

Keywords—Selenium; Automated Testing; Test Cases; Report Generation

I. INTRODUCTION

The objective of software testing is to discover faults and errors in a software application. Software testing utilizes more than 50% time of software development lifecycle. Testing time depends upon the algorithm used, programming language, line of codes, function points, external and internal interfaces [1]. Testing improves the quality of an application. To ease the testing process, there are various programmed web testing tools are available in the market, a detailed survey is given in [2]. As compared to human testing, automatic testing is better, because automation testing systematizes the testing process of analysis, design and coding of various scripts [3]. Selenium [4] is an open source tool which is used for test automation of different types of online and real-time applications. Since the Selenium was proposed to automate web browser interface, therefore programming scripts can spontaneously accomplish the similar interactions that some performed manually. Although the Selenium can achieve any type of automated interaction, but was planned and initially used for automated web application testing [5]. It is licensed under Apache License 2.0 and portable automated software testing tool [6]. It has proficiencies to function through different browsers and operating systems. It is not just a single tool but a set of tools that assists testers to automate web-based applications more professionally. The selenium can’t test desktop based applications.

II. RELATED WORK

Yadav et al. have given a detailed description on the working of the selenium tool [7]. Gojare et al., have given the utility section of the selenium framework, steps to generate screenshot, different types of test suite, way of customizing test reports and finally a result on report customization [8].

III. ARCHITECTURE OF THE SELENIUM

Figure 1 describes the architecture of the Selenium web tool. It has two basic components: Selenium Client and Server. The client includes the WebDriver API, which is used to create test scripts to make interaction with web page and other application elements. It also includes the remote Web Driver class, which communicates with a remote Selenium server.

Fig. 1 Selenium Architecture
The Selenium server comprises of a server component which is used to accept requests from Selenium Client's Remote Web Driver class. It also comprises of the Web Driver API which is used to execute tests for web browsers on a server machine. The last component is the Selenium Grid which is implemented by Selenium Server in command-line options for grid characteristics, containing a central hub and nodes for numerous situations and preferred browser capabilities. Grid is a tool expended to run parallel tests through dissimilar machines and dissimilar browsers concurrently which effects the minimized execution time.

IV. SELENIUM INTEGRATED DEVELOPMENT ENVIRONMENT (IDE)

The IDE is a Firefox plugin that permits testers to record their actions as they track the work process flow that they want to test. The main components of the Selenium IDE are as follows. Selenium IDE has Menu bar, Base URL text box, Tool bar, Test Case pane, Command window and test Run or Failure status section.

Fig. 2 shows the File menu which contains options for Test Case and Test Suite as redrawn form [9]. Fig. 3 shows actions menu which permits users to record, play entire test suite, play current test case, play or pause, play fast or slow, set break point, and execute current test step. Fig. 4 shows the Help menu which permits testers to analysis documentation, UI-Element documentation and traverse to Selenium websites. Fig. 5 shows The Edit menu which permits the following options like copy, paste, delete, undo, and select all operations for editing the commands in the test case.
Step 1: First of all search selenium IDE with the help of Mozilla Firefox page, after search; download the Add-on IDE using Firefox. Next restart the browser and see the Selenium IDE in browser as shown in Fig. 6.
Step 2: After click on Selenium IDE, IDE pop up as shown in Fig 7. The red colour circle icon is used to record the test case. By default, it is in recording mode; we can pause and stop it.

Fig 7: Selenium IDE pops up in FireFox.

Step 3: Generating Selenium IDE Script
1) Application to be tested: Open a new tab and enter the address of application under test (e.g. www.easyonlineconverter.com [10]), open text converter, and type some text in the text box, then change it to uppercase, lowercase, alternate case, sentence case etc.
2) Recording: All this processing will be recorded into Selenium IDE as shown in Fig. 8. Selenium IDE supports the tester to record user interactions with the browser and therefore the complete recorded schedule is designated as Selenium IDE script.

Fig. 8: Recording of the Selenium IDE Script.

3) Save and Run Process– Once a script has been created, it can be saved for future runs as shown in Fig. 9. There is no failure in the current test run. We can slow the process of running by control button of fast and slow as shown in Fig. 9.

Figure 9: Run the Selenium IDE Script.
VI. EXAMINATION OF RESULT

After test run, its result can be checked in the log field as shown in Fig. 10. It shows that Test case has been passed and there is no failure. When the test case is executed, error and information messages which display the development in the log pane repeatedly, though the log tab has not been selected. These messages are frequently valuable for test case debugging. The Clear button is used for clearing the Log and the Info button is a drop-down which allow the selection of diverse levels of information to log. The Reference tab is the default selection whenever the Selenium commands and parameters are entered or modified in Table mode. In Table mode, it will show documentation on the current command. When entering or modifying commands, whether from Table or Source mode, it is very significant to confirm that the parameters indicated in the Target and Value fields should be equal to those identified in the parameter list in the Reference pane. The Reference tab has been shown in the Fig. 11.

VII. EXPORTING THE TEST RESULTS

The test case can be exported in either a Web Drive or Remote-Control Unit for future use, the language used for test case can be C#, JAVA, Python or Ruby, as shown in Fig. 12.
VIII. TEST CASE CODING

```csharp
using System;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading;
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Support.UI;

namespace SeleniumTests
{
    [TestFixture]
    public class Rs
    {
        private IWebDriver driver;
        private StringBuilder verificationErrors;
        private string baseURL;
        private bool acceptNextAlert = true;

        [SetUp]
        public void SetupTest()
        {
            driver = new FirefoxDriver();
            baseURL = "http://easyonlineconverter.com/";
            verificationErrors = new StringBuilder();
        }

        [TearDown]
        public void TeardownTest()
        {
            try
            {
                driver.Quit();
            }
            catch (Exception)
            {
                // Ignore errors if unable to close the browser
            }
            Assert.AreEqual("", verificationErrors.ToString());
        }

        [Test]
        public void TheRsTest()
        {
            driver.Navigate().GoToUrl(baseURL + "/converters/case_converter.html");
            driver.FindElement(By.Id("content")).Clear();
            driver.FindElement(By.Id("content")).SendKeys("jyotikirtirohini");
            driver.FindElement(By.Id("upper")).Click();
            driver.FindElement(By.Id("lower")).Click();
            driver.FindElement(By.Id("capitalized")).Click();
            driver.FindElement(By.Id("sentence")).Click();
            driver.FindElement(By.Id("alternating")).Click();
        }
    }
}
```
try
{
    driver.FindElement(by);
    return true;
}
catch (NoSuchElementException)
{
    return false;
}

private bool IsAlertPresent()
{
    try
    {
        driver.SwitchTo().Alert();
        return true;
    }
    catch (NoAlertPresentException)
    {
        return false;
    }
}

private string CloseAlertAndGetItsText()
{
    try
    {
        IAlert alert = driver.SwitchTo().Alert();
        string alertText = alert.Text;
        if (acceptNextAlert) {
            alert.Accept();
        } else {
            alert.Dismiss();
        }
        return alertText;
    } finally {
        acceptNextAlert = true;
    }
}

IX. CONCLUSION

In this article, we have tested an online application with the help of web based automation testing tool Selenium. The selenium tool runs successfully one component of the application with no failure. In the setup phase, address of the baseURL has been give, i.e. the address of the application under test. The name of the file is Rs, in the RsTest() function, text conversion operations (upper, lower, capitalized, sentence and alternate) have been applied on the keys (Jyoti, Kirti, Rohini). The text converter component of the online application has successfully convert the keys. There is no failure in the site’s operation and this has been tested and proved by the Selenium testing tool.

ACKNOWLEDGMENT

The heading of the Acknowledgment section and the References section must not be numbered.

Causal Productions wishes to acknowledge Michael Shell and other contributors for developing and maintaining the IEEE LaTeX style files which have been used in the preparation of this template. To see the list of contributors, please refer to the top of file IEEETran.cls in the IEEE LaTeX distribution.

REFERENCES


