TEXT AND KEYWORD DRIVEN AUTOMATION TESTING USING SELENIUM WEB DRIVER

Mr. Dashrath Mane[1]; Gaurav Bhadekar[2] & Santosh Salunkhe[3]
Department of MCA
Vivekanand Education Society's Institute of Technology
Chembur, Mumbai 400074.

Abstract *- Nowadays in Software industry manual testing is replaced by automation testing at very large scale. Software testing using automation software tools can increase depth and scope of testing process to improve software product quality. Testers have to write a code in automated software tools like 'Selenium Web Driver' to automate whole software test process. By using automation tools, testing is become more efficient and repetitive.*

Selenium Web-Driver is widely used automation software tool basically used to test web elements present on the screen. Testers have to write a code in selenium web driver to locate web element on a screen and perform testing on them according to the test scenario. But, still some problems faced in selenium web driver for locating web element on the screen. In this paper will be focused on Automation testing using Selenium web driver, problems faced in selenium web driver and its solution using our research.

Key Words: Manual Testing, Automation Testing, Selenium Web Driver, etc.

1. INTRODUCTION

The software testing is a process of evaluating system software and its components to find out whether they are meeting actual requirements of system or not. The main objective of software testing is that to create such a test scenarios or that verifies whether the system software satisfies System /Business requirement specifications.

For the process of software testing, testers have to follow some process to achieve this process. Figure defines the various stages in STLC (Software Testing Life Cycle). One of most important steps in this is to plan various test conditional scenarios and write down various test cases. Test cases are nothing but set of documents with test data, preconditions, expected results, various conditions developed for a particular test scenario to verify specific requirements. Executions of test cases are depending on number of times test cycles are going to repeat.

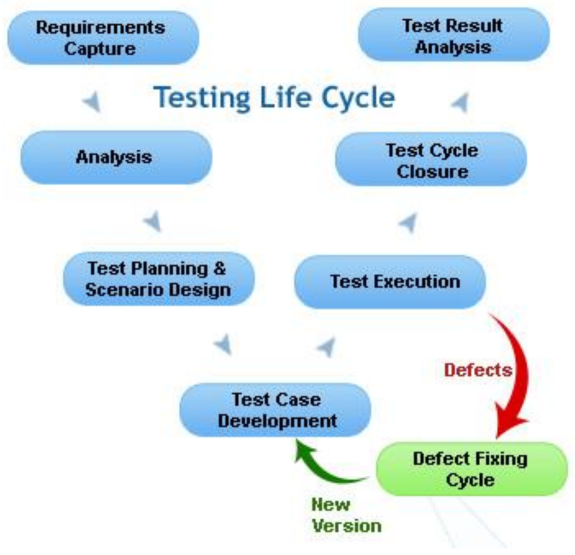

Fig -1: Software Testing Life Cycle

2. AUTOMATION TESTING

Automation testing is a process of testing software and its components using automated tools. By using automated tools, we can take control whole test process and can easily compare actual outcome with predicted outcomes. Automated software testing can increase depth and scope of test to help improve software quality. Lengthy tests can be executed easily by using automated testing. They can be run on multiple computers of different configurations at a same time. Automated software testing method can test an application and see memory contents, data tables, file contents, and internal program current status to determine if the product is behaving as expected. Test automation can easily execute thousands of various test cases during every test-case run providing coverage that is impossible with manual tests. There are various tools are available in market for automation testing with multiple programming language compatibility. Following are list of automated software tools widely used:

- Selenium Web Driver, QF-Test
- Windmill, Rational,
- Functional Tester
- Tellurium, Ranorex

3. SELENIUM WEB DRIVER

Selenium is automation testing framework that used for web applications allows us to write tests scenarios in many programming languages like Java, C#, Groovy, Perl, PHP, Python and Ruby. WebDriver is a web automation framework that allows you to execute your tests scenarios against different browsers. Selenium-WebDriver makes direct calls to browser using every browser's local support for automation. You are now able to make powerful tests because WebDriver can use in any programming language which helps in designing your tests. Selenium web driver basically locates the web element on the screen by its position. After locating element on the screen testing is done accordingly on those web elements.

Following are the ways to locate elements on the screen in selenium web driver.

- className
- cssSelector
- id
- linkText
- name
- partialLinkText
- tagName
- xpath

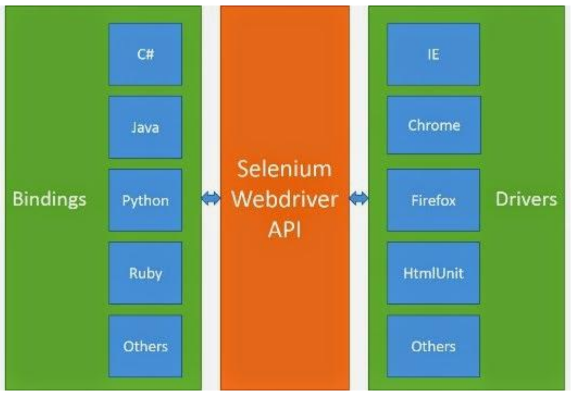

Fig -2: Architecture of Selenium Web Driver

The above figure illustrates the architecture of selenium web driver. Here we are writing our test in any programming language using common selenium API and that language binding is sending commands across this common interface. Now other side is listening a driver, it interprets those commands and runs them on the actual browser and return result backup using the API. Basically in programming terms Web driver is the name of key interface against which there are many methods are present. These methods are executed as various action steps on the web elements to perform test operations using selenium web driver. Following are some methods in selenium web driver:

findElement(), getCurrentUrl() getTitle(), getPageSource(), close(), quit()

Let us see simple Example with code snippet how testing is done in Selenium web driver. Here we will Login to Gmail account and will automate the below scenarios.

1. Open A Browser.
2. Navigate to the URL.
3. Enter the User Name.
4. Click On Next Button
5. Enter Password
6. Sign-In to the Gmail Account.
7. Click on compose button.
8. Sign-Out from the Gmail account.
9. Close the browser.

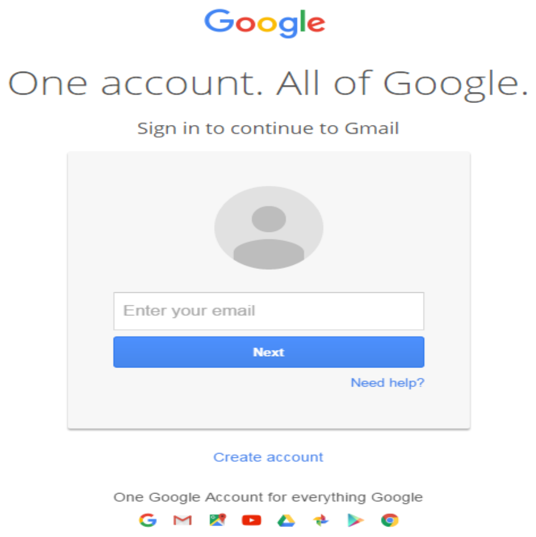
Google
One account. All of Google.
Sign in to continue to Gmail
Enter your email
Next
Need help?
Create account
One Google Account for everything Google

Fig -3: Gmail Login Page

```
WebDriver driver = new FirefoxDriver();
//NaviGate URL
driver.get("https://mail.google.com/");
//Enter UserName,Click On Next Button
driver.findElement(By.id("Email")).sendKeys("YOUR
USER NAME");
driver.findElement(By.id("next")).click();
//Enter Password,Click On SignIn Button
driver.findElement(By.id("Passwd")).sendKeys("YOUR
PASSWORD");
driver.findElement(By.id("signIn")).click();
//Click On Compose Button.
driver.findElement(By.xpath("//*[@id=":gr"]/div/div")).c
lick();
//Logout
driver.findElement(By.xpath("//*[@id="gb"]/div[1]/div[1
]/div[2]/div[4]/div[1]/a/span")).click();
driver.findElement(By.linkText("Sign out")).click();
//Close the browser.
driver.close();
}}
```

4. PROBLEMS FACED WHILE USING SELENIUM WEB DRIVER

The basic work of selenium web driver is to locate web element on a screen and perform testing on to that web elements. We have to write code in such a way that first it locates element and then performs various test scenarios on them. But following problems we are faced during use of selenium web driver:
1] Problem with this technology is that small change done in a web element designing code causes rewrite the whole code in selenium the code is not reusable.
2] If web element changes its position slightly on the screen we have to change our selenium web driver code for that web element.
3] In some cases, web elements cannot directly have located using selenium web driver because in that case web designer not use id or xpath to develop that web element so it's difficult to locate web elements. 4] In case of dynamic grids or say dynamic web elements it is difficult to locate them using id, class or xpath because their id, class, xpath are also dynamic so it changes frequently.
5] Nowadays web-site designers try to design their web-sites using responsive web designing techniques using Bootstrap and WordPress. In responsive web designing size of the website is changed automatically according to different screen sizes. So, web elements also relocate their position accordingly to the screen size. In this scenario it is difficult to write a code to locate that web elements on the screen.
6] In new programming languages like Angular JS it's difficult to locate web elements directly.
7] Every time new Web element is added to screen new set of code have to add in web driver. For every new test cases testers have to write code in background.

5. TEXT DRIVEN APPROACH TO LOCATE WEB ELEMENT

The problem faced by us in selenium web driver to locate the web element. This is because of changes in id, class, xpath and other locating methods of web elements. So, to overcome to that problem we are try to replace id, xpath, class by using text driven approach.

In text driven approach we have try to locate web elements on the screen on the basis of text. Every web element displayed on the screen by using some text. Ex Username, Password, etc. So, using text driven approach web elements are located by text present on the screen. Following techniques that we are used in text driven approach:
1] Contains: By using 'contains' function in XPath, we can extract all the elements which matches a particular text value.
Example: Here we are searching a web element displayed as UserName

```
"[contains(text(),'UserName')]"
```

2] Following: Using following keyword, we can fetch a web element on the which is next to some other element.
Example: Here we locate textbox on the basis of following keyword.

```
"//ul/li[contains(text(),'UserName')]/
following-sibling::input"
```

3] Preceding: Using Preceding keyword, we can fetch a web element on the which is preceding siblings to some other element.
Example: Here we locate Gmail image on the basis of Preceding keyword.

```
"//ul/li[contains(text(),'UserName')]/
preceding-sibling::input"
```

4] Title: Using title tag it is very easy to locate web element using that particular title
Example: Here we are searching a web element whose title is UserName.

```
"//*[@title = '"UserName"']"
```

These techniques we are using in text driven approach to replace id, class. Also, getAbsolutexpath(), ancestor() are helpful to locate web elements on the screen. This approach solves the problem of locating web elements.
Following are some advantages of text driven approach:
1] The main benefit of keyword driven approach is that it is easy to locate web element on the screen because search of the web elements is done on the basis of new techniques.
2] Change of position of web element on the screen, hidden web elements, code change of web elements does not affect test process using automating automation.
3] No additional code has to write to locate web element on the screen and it is guaranteed method to locate element on the screen.
4] This technique is work with web driver code written in java, C#, python and many more programming languages.

6. KEYWORD DRIVEN APPROACH TO OPTIMIZE AUTOMATION PROCESS

The idea behind the Keyword Driven approach in automation testing is to separate the coding from the test case & test step. This method helps a non-technical person to understand the automation very well. With this method without changing code in background we can add new test cases on web element as per different test scenarios. In keyword driven test framework, all the operations and instructions are written in some external file like CSV file. We have to just add different test scenarios on web elements by using this CSV files. So, by using this method no need to change code in background and code reusability is also achieved.

The main concept behind this approach is web elements and actions performed on them. The common web elements used in selenium web driver are

- Button
- Link
- Navigation
- Text Box
- Radio Button
- Check Box
- Drop Down
- Wait

Actions that also called keywords performed on them are

- Click – Perform the click option
- Navigate – Navigate to URL
- Select – Select the value from input
- Sendkeys – Provide the input to the web element
- waitForElement – Dynamic wait for web element

So, by locating web element on the by using various locators, actions are performed on them as per test scenario in this test case. Following is the architecture of keyword driven approach

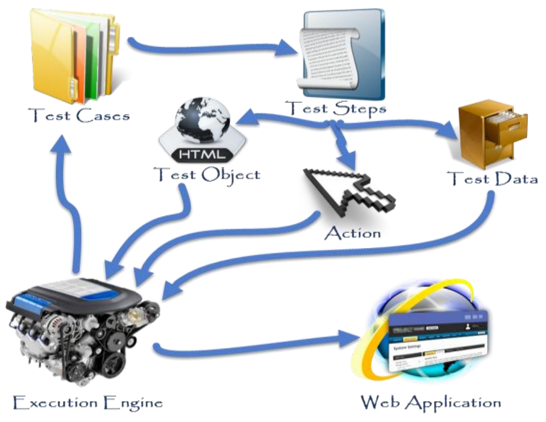

Fig -4: Architecture of Keyword Driven Approach

Let us see example of Gmail login using keyword driven techniques.

The external CSV file is passed to the selenium web driver code is as follows:

Table -1: Example of External CSV file

Keyword	Locator	LocatorValue	Parameter
Navigate			https: //mail. google.com/
SendKeys	xpath	[contains(text(), 'Enter your email')]	YOUR USER NAME
Click	xpath	[contains(text(),'Next')]	
SendKeys	id	Passwd	YOUR PASS WORD
Click	xpath	Sign in	
Click	xpath	[contains(text(), 'COMPOSE')]	
Click	linkText	Google Account: User (username@gmail.com)	
Click	linkText	Sign out	

After Passing CSV external file to java code java following steps are done in Keyword driven approach:

- Read the test steps from the CSV file one row at a time
- Execute the keyword corresponding to the current step in the test case
- Record the results in another CSV file.

Resulted Output file looks like follows:

Table -2: Example of Generated Result CSV file

Test Case ID	Result	Error Log
1	Pass	
2	Pass	
3	Pass	
4	Pass	
5	Pass	
6	Pass	
7	Pass	
8	Pass	

Java uses CSV reader for read the CSV file line by line. Each line is considered as one test case and java code executes that test case. We have to add just add keywords and locators with parameters as test new test cases adds in scenario.

Resulted output file contains test results with error log for the reference. So, it is difficult to know the tester which test case is passed and which is failed so tester can able to make changes in CSV file.

6.1 Advantages of keyword driven approach

1] Keyword driven technique doesn't require the user to acquire programming knowledge.
2] Code reusability is achieved at large scale; no new code has to written for new web element.
3] A single keyword can be used across multiple test scripts.
4] Keyword driven techniques can be uses as a testing framework to a multiple project.
5] Large number of test cases are designed and executed in less time rather than manual testing.
6] Output files are created with the logs so it is easy to know test results.

6.2 Disadvantages of keyword driven approach

1] The user should be well known with the Keyword creation mechanism to be able to efficiently use this approach as a framework.
2] This method becomes complicated gradually as it grows and a number of new keywords are introduced.

7. CONCLUSIONS

Automated Software testing is the best technique to improve the effectiveness, efficiency and coverage of software testing and Selenium is a framework comprises of many tools used for testing web applications. In this research paper we are focused on some new approaches used in the automation testing to improve its efficiency and overcome some problems. The main aim of Automating automation is mostly some efforts which we are taking while automation testing is reduced and automation testing done in more optimal ways.

Text driven approach and keyword driven approach are finds very helpful in automation testing. These two approaches help in speed up the testing process and improve efficiency in selenium web driver tool.

8. REFERENCES

[1]. http://seleniumhq.org/

[2]. http://guru99.com/automation-testing.html

[3]. http://softwarequalitymethods.com/papers

[4]. https://en.wikipedia.org/wiki/Test_automation

[5]. http://www.origsoft.com/whitepapers/software-testing-glossary/glossary_of_terms.pdf

[6]. Sherry Singla, Harpreet Kaur, "Selenium Keyword Driven Automation Testing Framework", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 6, June 2014.

[7]. Chandraprabha, Ajeet Kumar, Sajal Saxena, "Data Driven Testing Framework using Selenium WebDriver", International Journal of Computer Applications (0975 – 8887), Volume 118 – No. 18, May 2015.

[8]. Vishawjyoti, Sachin Sharma, "Study and Analysis of Automation Testing Techniques", Journal of Global Research in Computer Science, Volume 3, No. 12, December 2012.

[9]. R.S. Pressman, "Software Engineering a Practitioner's Approach", McGraw-Hill International Edition, ISBN 007-124083-7.

9. BIOGRAPHIES

Mr. Dashrath Mane – Assistant Professor, Department of Master in Computer Application, Vivekanand Education Society's
Institute of Technology - Mumbai, Maharashtra.
Email: dashrath.mane@ves.ac.in

Mr. Gaurav Bhadekar – Student,
Department of Master in Computer Application, Vivekanand Education Society's
Institute of Technology - Mumbai, Maharashtra.
Email: gaurav.bhadekar@ves.ac.in

Mr. Santosh Salunkhe – Student,
Department of Master in Computer Application, Vivekanand Education Society's
Institute of Technology - Mumbai, Maharashtra.
Email: santosh.salunke@ves.ac.in