Selenium: An Automated Testing Tool For Web Applications and its Locating Strategies

P.Shravani[1], K.Sreeveda[2]
[1](Assistant Professor, CSE Dept, Kamala Institute of Technology & Science, Singapuram
Email: shravanipulluri123@gmail.com)
[2](Assistant Professor, CSE Dept, Kamala Institute of Technology & Science, Singapuram
Email: sreeveda.karrepu@gmail.com)

ABSTRACT

Software testing is the most important process to find bugs and improves the quality of the software product. Software testing also helps to identify errors, gaps or missing requirements in contrary to the actual requirements. It can be either done manually or using automated tools. Testing is a very expensive process. Manual testing involves a lot of efforts, time and cost and these efforts can be reduced by using the automated testing with specific tools. At present majority of the applications are based on web and that are executed in a web browser and these web applications are more complex to test manually. Manual testing process can't provide accurate results and this can be avoided by using automation testing process. The main objective of this paper is to make automation testing process for web based applications using software testing tool Selenium. Selenium is a open source automated testing suite for web applications across different browsers and platforms.

Keywords: *Automated Testing, Selenium Grid, Selenium IDE, Selenium RC, Software Testing, Selenium Web Driver.*

I. INTRODUCTION

Software testing is a process to identify all bugs that are exist in a software product. Software testing is an activity to check whether the actual results match the expected results and to ensure that the software system is defect free. Software testing is also performed to achieve quality by using the software with applicable test cases. Testing can be integrated at various points in the development process depending upon the tools and methodologies used. Software Testing process can be performed by two ways that are manual testing or automation testing.

Manual Testing
Automated Testing
Software testing

Fig 1: Software Testing

1.1 Manual Testing:

Manual testing is the oldest and most rigorous type of software testing. Manual testing is a process to test the software manually to find out the bugs it requires a tester to perform manual test operations on the test software without the help of Test automation[1]. Any new application must be manually tested before its testing can be automated. Manual testing requires more effort, but is necessary to check automation feasibility. Manual Testing does not require knowledge of any testing tool. One of the Software Testing Fundamental is "100% Automation is not possible"**.** Manual testing is not suitable for large projects as it requires more resources and time. Manual testing is performed by a human sitting in front of a computer carefully executing the test cases.

1.2 Automated Testing:

Automated testing is a method in software testing that makes use of special software tools to control the execution of tests , compares actual test results with expected results and generates detailed test reports . All of this is done automatically with little or no intervention from the test engineer. Automation testing [3]saves time, money by making testing more efficient and it also improves testing accuracy compared to testing directed by human. Using automated testing tool[1] it is possible to record the test suite and re-play it as when we require. Once the test suite is automated, no human intervention is required. The automation software testing process consists of a sequence of activities and tools that are processed in order to execute the test on software and to keep the record of the result of tests. A general testing process is depicted in figure 2.

The following activities are include in testing process:

- Test Tool Selection
- Define Scope of Automation
- Test Planning
- Test Analysis
- Test Design & Implementation
- Test Execution
- Test Evaluation

1.2.1 Test tool selection:

Test Tool selection[4] largely depends on the technology the Application Under Test is built on. The following factors are important during tool selection:

- Assessment of the organization's maturity (e.g. readiness for change).
- Identification of the areas within the organization where tool support will help to improve testing processes.
- Evaluation of tools against clear requirements and objective criteria.
- Proof-of-concept to see whether the product works as desired and meets the requirements and objectives defined for it.
- Evaluation of the vendor (training, support and other commercial aspects) or open-source network of support.
- Identifying and planning internal implementation.

1.2.2 Scope of Automation:

Scope of automation is the area of our application under test which will be automated. Following points help determine scope:

- Scenarios which have large amount of data.
- Common functionalities across applications.
- Technical feasibility.
- Complexity of test cases.
- Ability to use the same test cases for cross browser testing.

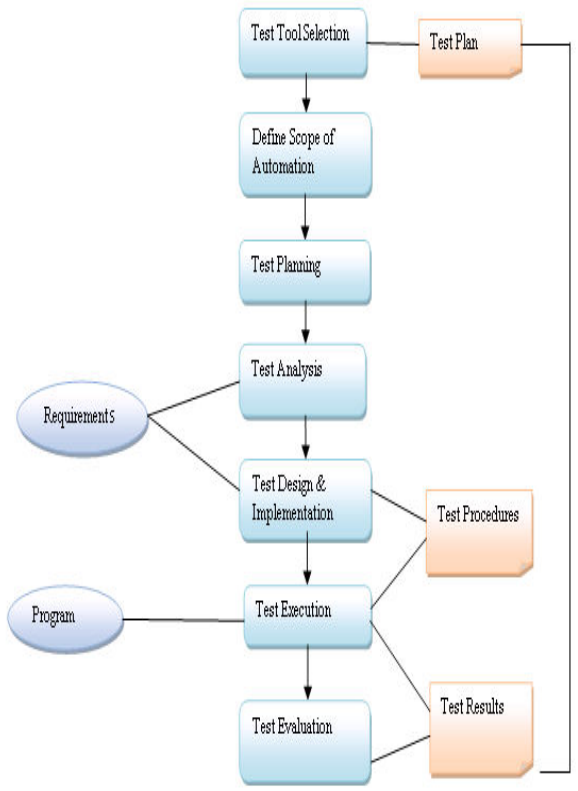

Fig 2: Testing Process

1.2.3 Test Planning:

Test planning has following major tasks:

- To determine the scope and risks and identify the objectives of testing.
- To determine the test approach.
- To implement the test policy and/or the test strategy.
- To determine the required test resources like People, test environments, PCs, etc.

1.2.4 Test Analysis and Test Design:

Test analysis and Test Design has the following major tasks:

- To review the test basis**.**
- To design the tests.
- To evaluate testability of the requirements and system.
- To design the test environment set-up and identify and required infrastructure and tools.
- To identify test conditions.

1.2.5 Test Implementation:

Test implementation has the following major task:

- To develop and prioritize our testcases by using techniques and create test data for those tests.
- To create test suites from the test cases for efficient test execution.
- To implement and verify the environment.

  During this phase we can create Automation strategy & plan, which contains following details:
- Automation tools selected
- Framework design and its features
- In-Scope and Out-of-scope items of automation
- Schedule and Timeline of scripting and execution
- Deliverables of automation testing

1.2.6 Test Execution:

Automation Scripts are executed during this phase. Execution can be performed using the automation tool directly or through the Test Management tool which will invoke the automation tool. Test execution has the following major task:

- To execute test suites and individual test cases following the test procedures.
- To re-execute the tests that previously failed in order to confirm a fix. This is known as confirmation testing or re-testing.
- To log the outcome of the test execution and record the identities and versions of the software under tests. The test log is used for the audit trial.
- To Compare actual results with expected results.

1.2.6 Test Evaluation:

Test Evaluation is the process by which a system or components are compared against requirements and specifications through testing. During evaluation we must check the results and evaluate the software under test and the completion criteria, which helps us to decide whether we have finished testing and whether the software product has passed the tests.

2. AUTOMATION TESTING TOOL: SELENIUM

Selenium is a software testing tool [2] used for regression testing. It is an open source testing tool that provides playback and recording facility for regression testing. The Selenium IDE only supports Mozilla Firebox web browser.

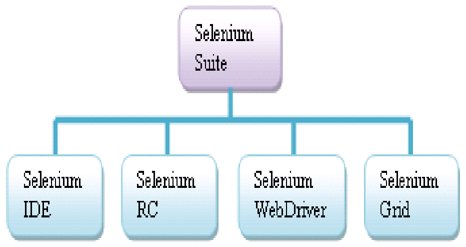

Fig 3 : Selenium Suite

- It provides the provision to export recorded script in other languages like Java, Ruby, RSpec, Python, C#, JUnit and TestNG.
- It can execute multiple tests at a time.
- Auto complete for Selenium commands that are common Walkthrough tests.
- Identifies the element using id, name, X-path, etc.
- Store tests as Ruby Script, HTML, and any other format.
- It provides an option to assert the title for every page
- It supports selenium user-extensions.js file.
- It allows to insert comments in the middle of the script for better understanding and debugging

2.1 Selenium IDE:

Selenium IDE (Integrated Development Environment) is a tool to develop Selenium test cases. Selenium IDE was originally created by Shinya Kasatani and donated to Selenium project in 2006. It is implemented as a Firefox Plug-in that allows recording, editing and debugging the selenium test cases. Selenium name comes from Selenium Recorder. On start-up of the Firefox, the recording option is automatically turned on. This option allows user to record any action done inside the web page. In Selenium IDE [5]scripts are recorded in Selenese, a special test scripting language which is a set of Selenium commands. It is used to test web application. Actions, Accessors, Assertions are the classification of selenium.

2.1.1 Features:

- It is simple and easy record and playback.
- Selenium IDE supports intellectual field selection options like ID's, XPath and Names.
- It saves test scripts in several formats like Selenese, Ruby etc.
- IDE allow to customization through plugins.
- Selenium IDE having an option for adding different asserts options in scripts.
- It allows setting breakpoints and debugging the scripts.

- It also supports auto complete commands.

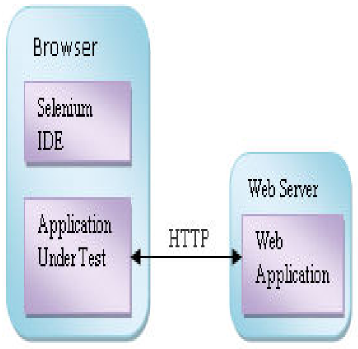

Fig 4: Architecture of IDE

2.1.2 Limitations:

- Selenium IDE works only in Mozilla Firefox and it cannot be used with other browsers.
- There is no option to verify images.
- It can execute scripts created in selenese only.
- It is difficult for checking complex test cases involving dynamic contents.

2.2 Selenium RC:

To overcome the Selenium IDE limitations, ThoghtWork's engineer Paul Hammant decided to create a server that will act as HTTP proxy to "trick" the browser into believing that Selenium Core and the web application being tested come from the same domain. This system known as Selenium Remote Control[7]. It is possible to run tests inside every JavaScript compatible browser using a wide range of programming language. Selenium RC has two components

Selenium RC has two parts:
Selenium Server: It uses Selenium core and browser's built-in JavaScript interpreter to process selenese commands (such as click, type) and report back results**.**
Selenium Client Libraries: Client libraries are the API's for the programming languages to communicate with Selenium server.
Selenium RC components are:

- The Selenium Server which launches and kills browsers, interprets and runs the Selenese commands passed from the test program, and acts as an HTTP proxy, intercepting and verifying HTTP messages passed between the browser and the AUT(Application Under Test).
- Client libraries which provide the interface between each programming language and the Selenium RC Server.

2.2.1 Architecture of RC:

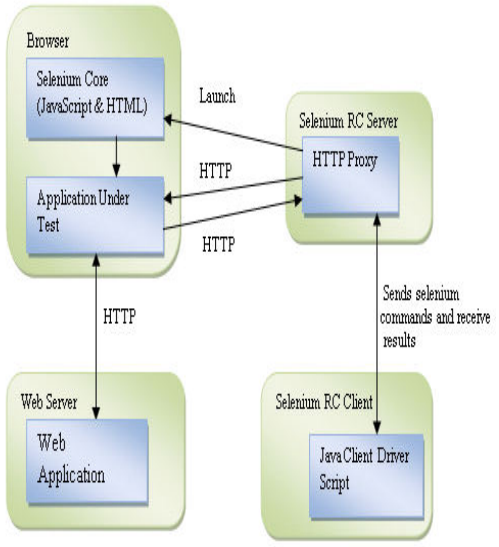

Fig 5: Architecture of RC

2.2.2 Features:

- It faster execution speed than IDE.
- Cross browser and cross platform.
- Have matured and complete API.
- It can readily support new browsers.
- Selenium can run tests automatically as many times as we want.
- Selenium can support data driven testing.
- It allows the user to use programming language.

2.2.3 Limitations:

- Selenium RC is slow.
- It has limited features of drag and drop of Objects.
- It struggles when running concurrent tests.
- It does not allow simultaneously tests across different OS and browsers.

2.3 Selenium Web Driver:

Simon Stewart created Web Driver 2006 when browsers and web applications were becoming more powerful and more restrictive with JavaScript programs like Selenium Core. It was the first cross platform testing framework that could control the browser. To provide a simpler, more concise programming interface. It supports dynamic web pages where elements of a page may change without the page itself being reloaded. Web Driver is the name of the key interface against which tests should be written in Java. Selenium Web Driver[6] is the successor to Selenium RC. It does not need a special server to execute tests. It directly starts a browser instance and controls it. Selenium Grid can be used with Web Driver to execute tests on remote systems.

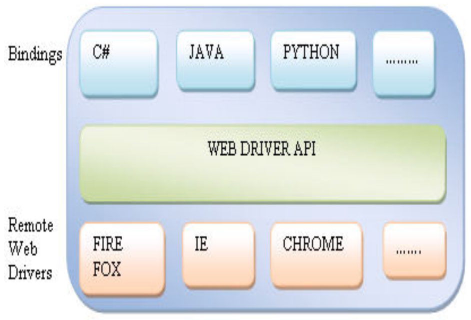

Fig 6: Architecture of Web Driver

Selenium Web Driver makes directly calls to the browser using each browser’s native support for automation. There are so many browsers and many programming languages there is need for common specification provided by Web Driver API. Remote Web driver means each browser has to implement this API. Language bindings will send the commands to the common driver API, on the other end there is going to be a driver listening to those commands and they will be executed in browser using remote Web Driver and it’s going to return the result/response via API to the code/Binding. Web Driver API that communicates with the use a common wire protocol which is named as JSON Wired Protocol which is a RESTFUL web service using JSON over HTTP.

2.3.1 Features:

- It allows us to execute the tests against different browsers.
- Use a programming language of our own choice for creating test scripts
- This architecture is simpler than Selenium RC's architecture.
- It directly run with the browser by using the browser's own engine to control it.
- Support the headless Html Unit browser.

2.3.2 Limitations:

- Selenium Web Driver cannot support new browsers because it operates on the OS level and also different browsers communicate differently with the Operating System.
- Built-in commands are not available.

2.4 Selenium GRID:

A test of different machines against different browsers in parallel can be run by using Selenium Grid. It runs on multiple tests at the same time against different machines running different browsers and operating systems. Selenium Grid support distributed test execution. It is a server that allows tests to use web browser instances running on remote machines. One server acts as the Hub. Tests contact the hub to obtain access to browser instances. The hub offers list of servers that provide access to browser instances, and let’s tests use of these instances. The tests will run parallel on multiple machines, and to manage different browser versions. Selenium Grid has 2 versions - the older Grid 1 and the newer Grid 2.

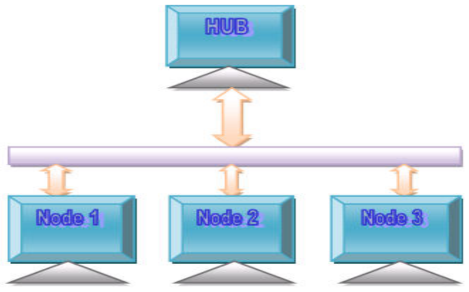

Fig 7.Selenium Grid

Selenium Grid uses a hub-node concept. It only run the test on a single machine called a hub, but the execution will be done by different machines called nodes.

2.4.1 Features:

- It can be extended by distributing tests on a number of machines. Executions can be done parallel.
- It manages multiple environments from a central point and make test to run easily against a huge combination of browsers as well as Operating System.
- Maintenance time will be reduced for the grid by allowing us to implement regular hooks to influence virtual infrastructure for instance.

2.4.2 Limitations:

- Selenium grid by itself cannot run multiple tests in parallel, the framework like TestNG or JUnit are used to provide multiple tests to the grid.

III. Locating Strategies in Selenium

Locating strategies are used by Selenium to find and match the elements of our AUT(Application Under Test) with which it needs to perform some action(like clicking, typing, selecting, verifying).It is used in Target column of Selenium IDE. In simple words it tells Selenium which HTML element of our application a command has to perform the action.

For many Selenium commands, a target is required. This target identifies an element in the content of the web application, and consists of the location strategy followed by the location in the format locator Type=location. The various locator types are explained below with examples for each.

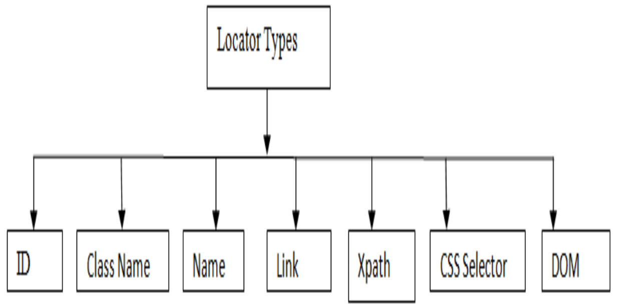

Fig 8.Locating Strategies

Locating Principle:-
locator Type=location
Locator Types:-
Every object(control) visible on a webpage is a "Web Element".
Selenium has different ways of locating controls (web elements).

- Id
- Class Name
- Name
- Link
- XPath
- CSS Selector
- DOM

ID:-
This approach is considered superior compared to other locators Unfortunately there are many cases when an element does not have an id (or the id is somehow dynamically generated and unpredictable). In these cases we will need to use an alternative locator strategy.
Target Format: id=id of the element
Example:- id = "userName"

Class Name : There may be multiple elements with the same name, if we just use find Element By Class Name.
Target Format: class=class name of the element
Example:- class="D(ib) Py(0) Zoom Va(t) uhBtn Ff(ss)! Fw(40) Bxz(bb) Td(n) D(ib) Zoom Va(m) Ta(c) Bgr(rx) Bdrs(3px) Bdw(1px) M(0)! C(#fff) uh-ignore-rapid Cur(p)"

Name:-
Locating elements by name are very similar to locating by ID, except that we use the "name=" prefix instead.
Note:-If multiple elements have the same value for a name attribute, then we can use filters to further refine our location strategy.
Target Format: name=name of the element
Example :- name="p"

Link:- This approach uses a hyperlink in the web page to locate the element by using the text of the link.we can find elements of "**a**" tags(**Link**) with the link names. Use this when we know link text used within an anchor tag. If two links with the same text are present, then the first match will be used.

Target Format: link=link_text
Example:- link=Sign in

XPath:-
XPath is the language used when locating XML (Extensible Markup Language) nodes. While DOM is the recognized standard for navigation through an HTML element tree, XPath is the standard navigation tool for XML; and an HTML document is also an XML document (xHTML).
It can access almost any element, even those without class, name, or id attributes.

Target : xpath=//tagname[@attribute="value"]
Example : xpath=//input[@id="username"]
xpath=//form[@name="loginForm"]
xpath=//*[@name="loginForm"]

CSS:-
CSS is a language which is used for beautification of HTML controls like button should have green color. CSS uses Selectors for binding style properties to elements in the document. These Selectors can be used by Selenium as another locating strategy. Locating by CSS Selector is more complicated than the previous methods, but it is the most common locating strategy of advanced Selenium users because it can access even those elements that have no ID or name.(Like X-Path)

Syntax:
Id: css=tag#id

- Tag=The HTML tag of the element being accessed
- #=The hash sign. This should always be present when using a CSS selector with ID
- Id=The Id of the element being accessed.

Example:- css=form#loginForm

Class: css=tagname.classname
Example: css= input.passfield

DOM:-
The Document Object Model (DOM), in simple terms, is the way by which HTML elements are structured.
Selenium is able to use the DOM structure in accessing page elements.
There are four basic ways to locate an element through DOM:

- getElementById
- getElementsByName
- dom:name (applies only to elements within a named form)
- dom:index

Since selenium core is able to interpret the dom as document so we generally remove the locator type and directly use the location value.(Thus we can omit DOM as locator Type and use directly the location value) value.(Thus we can omit DOM as locator Type and use directly the location value.

Example :-
We can locate password object by its id, using getElementById
We can either use
dom=document.getElementById("password") or direct value

IV. CONCLUSION

The main objective of automated testing is to reduce the testing efforts. In this paper we have discussed about the Web automation testing tool Selenium and its components. Now a days Selenium is used as a best testing tool for web applications. Future enhancements of selenium is to test the window based applications.

REFERENCES

[1] Neha Bhateja, "A Study on Various Software Automation Testing Tools", International Journal of Advanced Research in Computer Science and Software Engineering Research Paper, Volume 5, Issue 6, June 2015, ISSN: 2277 128X.

[2] Chandraprabha, Ajeet Kumar, Sajal Saxena," Systematic study of a web testing tool: selenium " International Journal Of Advance Research In Science and Engineering, IJARSE, Vol. No.2, Issue No.11, November 2013.

[3] Fei Wang, Wencai Du-"A Test Automation Framework Based on WEB" 11th International Conference on Computer and Information Science (ICIS), ISBN:978-1-4673-1536-4 IEEE/ACIS May 30 2012-June 2012.

[4] Monika Sharma and Rigzin Angmo,"Web Based Automation Testing and Tools" , international journal of Computer Science And Information Technology (IJCSIT), Vol.5(1),2014, ISSN:0975-9646.

[5]Sherry single,Harpreet kaur,"Selenium keyword automation testing framework", International Journal of Advanced Research In Computer Science and Software Engineering, Vol.4, 2014.

[6] Nidhika Uppal, Vinay Chopra,"Design and Implementation in Selenium IDE with Web Driver" International Journal of Computer Applications (0975 – 8887) Volume 46–No.12May 2012.

[7] Y.C.Kulkarni, "Automating the web applications using the selenium RC", ASM's International Journal of Ongoing Research in Management and IT e-ISSN-2320-0065, 2011.