Sahi–Web Automation and Testing Tool: A White Paper

By

Mahesh Kurapati-Test Engineer
Vineela Sukhavasi-Test Engineer
Zen Quality Assurance Pvt. Ltd.
Contents

1. Abstract.................................................................................................................. 01

2. Introduction............................................................................................................. 01

3. Related work.......................................................................................................... 02
   i) AJAX loading cannot be handled by selenium commands.............................. 02
   ii) Working with XPath is brittle and complex.................................................. 03
   iii) Selenium doesn’t provide inbuilt reporting functionality........................... 04
   iv) May not run the same script in multiple browsers without any modification 06
   v) Record and playback is limited to Firefox only.............................................. 06

4. How Sahi works?.................................................................................................... 07

5. Features of sahi...................................................................................................... 07

6. Conclusion............................................................................................................. 08

References
1. Abstract

Web testing is the process of testing the applications hosted on the web in which the application interfaces and functionalities are tested. Testing the application before going to live will resolve many issues. Generally, more than 80% of the verification process are repetitive, with testers performing the same verification steps manually from release to release.

Manual testing is a time consuming process but cannot be overlooked because each time a software doesn’t perform well according to the specifications. Test automation tests the application using special software. The primary objective of test automation is to reduce repetitive manual tasks. There are many commercial tools and open source frameworks available for automated testing of web applications.

This paper introduces the new automation tool named Sahi and also shows the comparison with Selenium. Both the mentioned tools are used to automate web applications and the comparison shows the better choice of automation tool.

2. Introduction

Sahi is an automation and testing tool for web applications. It is available in both open-source and proprietary versions. The open-source version includes record and playback on all browsers, HTML reports, suites and batch run and parallel playback, etc. Pro version includes some of the enhanced features like test distribution and report customization. While considering the technical details of Sahi, it runs as a proxy server and the proxy settings are configured to point to a Sahi’s proxy and then injects JavaScript event handlers into web pages.

Fig. 1.1 Sahi Logo
Selenium is a testing framework used to automate the web applications and has several components like Selenium IDE, Selenium Remote control, Selenium WebDriver and Selenium Grid, etc. Each one of these components has its own specialties, out of this Selenium RC considered to be depreciated one and its successor is WebDriver.

Every tool has its own limitations, this paper will discuss about Sahi and how it can overcome the challenges faced with Selenium.

3. Related work

The objective of research is to find the tool which can overcome the challenges faced with Selenium. Performance of any tool depends on overcoming limitations of other tool and its own features those are not already implemented ever. Sahi has unique features and it can overcome the limitations of Selenium, those limitations are as follows:

i. AJAX loading cannot be handled by Selenium commands
ii. Working with XPath is brittle and complex
iii. Selenium doesn’t provide inbuilt reporting functionality
iv. May not run the same script in multiple browsers without any modifications
v. Record and playback is limited to Firefox only

i) AJAX loading cannot be handled by Selenium commands-

Consider an application which has AJAX loading part in it.

Scenario:

It’s a calendar application, It will display the day, month and year after AJAX loading on clicking on the date in the calendar. Let’s see how the Selenium and Sahi behaves on clicking on the three dates on the calendar.
**Selenium:**

Selenium uses about 20 lines of code with explicit waits to handle the AJAX loading. By executing the script, 3 dates in the calendar are clicked and continuous spinner is displayed and no selected date information is displayed.

![Selenium Example](image1.png)

*Fig. 1.3 Spinner is displayed in AJAX loading example using Selenium*

**Sahi:**

Sahi uses 4 lines of code with no waits to handle the AJAX loading. By executing the script, 1 date is clicked and its corresponding information is displayed after the AJAX loading. Like this, 3 dates are handled.

![Sahi Example](image2.png)

*Fig. 1.4 Sahi is handled AJAX loading*

**ii) Working with XPath is brittle and complex**

XPath is used with Selenium to locate the elements on a web page.
Selenium:

Execution time is the crucial part of automation. Working with XPath will lead to increase the time for the execution. XPath can be written in different ways, depending upon the user it may vary i.e. long or short.

XPath is used to traverse XML structures which represent data. An HTML page is not data. It is a visual representation of data and is prone to modification. Further XPath is not natively available in browsers. One may need to use external libraries for it. Browsers also have differences in their implementation of XPath.

The time taken for the long and short xpath for the same script in the different browsers are:

Using Firefox browser-
Long XPath lookup took 11.27337114sec.
Short XPath lookup took 0.087291036sec.

Using Internet Explorer browser-
Long XPath lookup took 22.27337114sec.
Short XPath lookup took 4.087291036sec.

By the above comparison, we observe that there is no best practice to use locators in Selenium. Using XPath leads to brittle, unmaintainable and slow tests. We may use different XPath’s for a different purpose in the whole automation script. But we don’t know the exact time for script execution. This uncertainty of time taking will increase the execution time.

Sahi:

There is no such problem with the locators and unnecessary consumption of time. Because, it uses its own wrappers around the DOM to locate the elements.

iii) Selenium doesn’t have inbuilt reporting functionality-

Selenium:

Selenium consider to be an automation framework and it is not having the capability of providing the reports for executed script. We need to use some third party tools to generate the reports in Selenium.

Sahi:

Sahi has capability of generating the reports by its own for the executed script.
Below are the reports those are generated after executing the following script using Sahi:

```javascript
#include("Sahi_lib.sah");
_navigateTo("http://Sahiitest.com/demo/training/");
_login("test1", "secret");
_assertVisible(_div("Invalid username or password"));
_login("test", "secret");
_highlight(_textbox("q"), _near(_cell("Core Java")));
_setValue(_textbox("q"), _near(_cell("Core Java")), "3");
_setValue(_textbox("q[1]"), _near(_cell("Core Java"))), "2");
_setValue(_textbox("q[2]"), _near(_cell("Core Java"))), "1");
_click(_button("Add"), _near(_cell("Core Java")));
_verifyFunction("1650");
_click(_button("Logout"), _near(_cell("Core Java")));
```

Fig. 1.5 Script report

Fig. 1.6 Test Case report with execution time
iv) **May not run the same script in multiple browsers without any modification** -

**Selenium:**

Selenium can only work very efficiently with Firefox and most probably with chrome. But working with IE is a nightmare. We may not run the same script written for any web application for Firefox in another browser. The very first cause is the locating elements, so it may not run the same script in all browsers.

**Sahi:**

Sahi is able to run the same script in multiple browsers without any modification to the script. Sahi’s JavaScript and proxy support is guaranteed to work with any browser and its APIs are normalized to work across browsers. Sahi’s APIs is fairly robust and can withstand peripheral UI changes. The script written using Sahi is very precise and can be executable in any browser that is configured with Sahi. Sahi doesn’t use any XPath to locate the elements, Hence there is no modification of script is required.

v) **Record and playback is limited to Firefox only** -

We don’t have the facility to record & playback in Selenium other than Firefox. We can play back in another(other than Firefox) browser, but its workaround. In Sahi, we can record & playback from any browser and in any OS. It can also provide the reports to those playbacked scripts.
4. How Sahi works?

Sahi has simple and straightforward architecture and it is a proxy between Java and the browser.

Java -> Sahi -> JavaScript -> Browser
Java <-> Sahi <-> JavaScript <-> Browser

Sahi proxy server is written in Java. It uses an HTTP proxy as its core to inject JavaScript into web pages. The injected JavaScript uses custom code to identify elements in the browser and simulate actions like click, type, etc., on them. HTML responses that pass through the proxy are modified such that JavaScript is injected at the start and the end of the response. Sahi Script uses the same constructs of JavaScript, that adds the capability to interact with the browser efficiently.

5. Features of Sahi

1. **Simple & Powerful Scripting**
   Sahi Script is based on JavaScript. Interact with your File-System, Databases, Excel sheets, CSV files with ease. Call any Java code or library from Sahi Script to get added power.

2. **Inbuilt Excel Framework**
Use the inbuilt Excel Framework to let your business analysts and non-technical testers contribute to testing. Easily test from the Controller. Get detailed inbuilt reports.

3. Intuitive and simple APIs
4. JavaScript based scripts for good programming control
5. Command line and ant support for integration into build processes
6. Supports external proxy, HTTPS, 401 & NTLM authentications
7. Supports browser popups and modal dialogs
8. Can directly invoke Java code from scripts. This is used to access databases, read pdf files
9. Supports data-driven testing.

6. Conclusion

Sahi is focused towards saving time for testers and developers. Sahi takes care of reporting, handling suites, distributed playback across multiple machines, etc. with minimal effort from the tester. In this paper, limitations of Selenium and uses of Sahi over Selenium are shown. On the basis of analysis performed by considering the aspects of Selenium limitations like AJAX loading, working with XPath, multibrowser support and reporting functionality kind of things shows that automation of web application can be better performed using Sahi over Selenium. While choosing a good tool, performance is the most important factor to take into consideration rather than the cost. We conclude that Selenium may be right for certain specific situation, but SAHI can be the best choice in many more situations.

References & Appendix:

Sahi: Automation Testing Tool For Web Applications – official site
http://Sahipro

Sahi - Effective Web Testing (MT)
http://www.slideshare.net/Sahi.co.in/Sahi-effect?fr

[Review]: Web Automation Testing Tool Sahi Pro
http://appaloud.com/review-web-automation-testing-tool-Sahi-pro/

Sahi - Functional testing tool for web application
https://www.youtube.com/watch?v=JxZMoQZF8A

Principles and Architecture of Sahi:Web Automation Simplified by Narayan Raman
https://www.youtube.com/watch?v=Fue5unKjuCM
THANK YOU