© 2017 IJNRD | Volume 2, Issue 8 August 2017 | ISSN: 2456-4184

CENTRALIZED UI AUTOMATION FOR SAAS APPLICATION

[1]**Sowmya Nagaraju**
M Tech Computer networks engineering
BMS College of Engineering
(Affiliated to VTU) Bull Temple Road, Bangalore-19

Abstract - The purpose of this proposed system is to reduce the burden on the manual tester and time consumed is less. Testing the functionalities of SAAS applications at a regular rate is an overhead and a time-consuming process. To reduce the burden of manual testers, organizations have arrived at the solution called automation testing.
Currently there exists a framework which is a wrapper around Selenium Web Driver. Framework has been created in such a way as to test any SAAS application independent of the technology. It uses Google Guice for dependency injection and Page Object factory pattern. It also uses the builder pattern to perform complex operations which include multiple actions.
Using the TestNG framework, tests are run. TestNG supports many powerful features and is very easy to use. Hamcrest framework has been utilized for efficient creation of Matcher Objects. Using Page Object Model a new framework is created which can be used for each application. The framework includes location strategies of web elements and actions performed on them. Several applications under a team are created using different technologies and are separately tested. A common portal to log in to applications and test each scenario automatically is the aim of the new framework.

Keywords: Automation, TestNG, Selenium Webdriver, SAAS.

I. INTRODUCTION

In today's world, most of the applications are web based. There are several techniques used to test these applications. In many scenarios, various functionalities of the applications are to be tested regularly. Manual testers should spend a lot of time every day to perform the same tests. In such situations organizations turn to automation testing to save time used for performing the tests. Automation testing uses software tools to perform tests in a repeated manner against an application. It is basically used for regression testing and for Build Acceptance Testing. The repeatable nature of testing and the speed of testing the applications are advantages of automation testing. It is not always wise to automate tests. Certain applications have their user interface changing very often in which case creating an automated test case is a waste of time, since the code should be changed as and when the UI changes. In such cases, manual testing saves time when compared to automation testing.

Several software's are available to perform automation testing. Selenium is one of the most widely used standardized automation tool in the industry. Selenium was initially created in 2004 by Jason Huggins. He created a JavaScript library to interact with web pages and run automated tests against browsers, but it had its drawbacks like security limitations due to JavaScript. In 2006, an engineer from Google named Simon Stewart introduced WebDriver avoiding the limitations of the JavaScript environment. Finally, in 2008, both Selenium and WebDriver were merged together to create a better framework that reduced the shortcomings of both Selenium and WebDriver.

II. SYSTEM MODEL

The objectives of following an architecture with the standards are:

Performance - Provide a strategy for design and development of an efficient Automation Framework in an iterative incremental development model used for SAAS Applications Congeniality - Address issues surrounding framework compatibility with respect to the required used technologies including Java, Junit, Selenium-Webdriver, Browsers, Maven etc.

Feasibility and maintainability - Provide a fit-for-purpose automation solution that is easy to use and maintain in functionality

Extensibility - Focus on the scalability of the framework, leading to higher efficiency in activities related to enhancing its test functional base across SAAS Applications Modularity - Use modularity to assist framework maintenance and ensure consistent performance in a scenario where the application under test is constantly evolving

Regularity - Provide a structured development methodology to ensure uniformity of design across multiple applications/libraries/test scripts to reduce dependency on individual test-case.

The architecture of the automation framework can be simplified and explained by the diagram below:

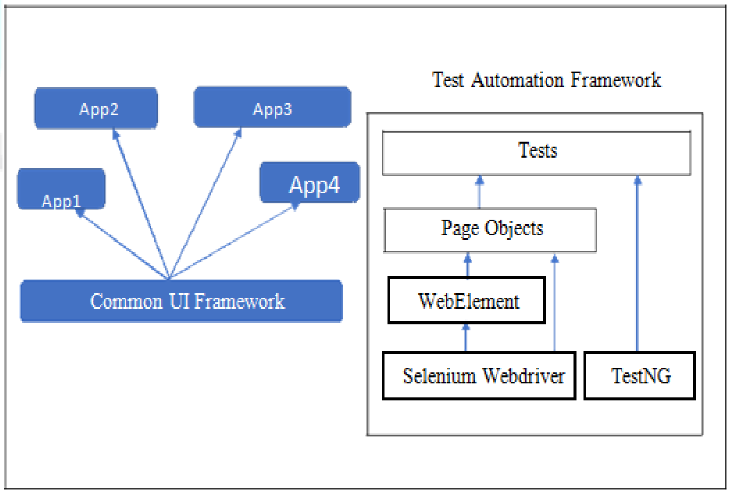

Fig 2.1. Simplified Architecture with SAAS Applications

The automation infrastructure is being created for SAAS applications. SAAS contains many applications. A few of them are Application1, Application2 and Application3. As of now, all the applications are developed using different languages and tested individually. A lot of time is invested in manual testing. On checking the amount of time spent only on manual testing for each application, automation testing was put forward as a solution. A common portal was created through which every application can be accessed by their specific logins. "Common UI Framework" in the diagram refers to the common portal for all the applications. It includes a common login page; a company selector page and further different applications have different UI. The application that has been automated as part of this project is SAAS application. It involves several retailers, and hence retailers according to their requirements make minor changes to their UI. To access the web elements on each web page and perform operations on them, Page classes are created for every web page. This is part of the automation framework that is being developed. These page objects are created with the help of an underlying framework. This framework is again a wrapper around Selenium Web Driver code. Tests are written using TestNG (Next Generation Testing) framework. TestNG allows testers to use annotations. It allows testers to run tests parallel. It also provides more flexibility for regression, functional testing etc. Tests are separately created in Test classes. The web elements are accessed from the Page objects by importing the necessary classes into the test classes. Thus, the framework code (Page Objects) are separated from the test scripts.

III. PREVIOUS WORK

This project aims to create a Centralized Automation Infrastructure to automate the tests of various internal applications of SAAS. SAAS is a collection of cloud-based applications for retailers and many more. They offer their service for both physical stores and online stores. The development team develops various features or even updates certain existing features and the testing team continuously or even sometimes on a regular basis are required to manually test the functionality of the applications. This has resulted in a huge overhead and hence they have taken up the solution of automating the test cases. The automation framework created for automating the test cases can be explained in a simple manner by the below diagram. Using WebElement, the testing framework is being created using the Page Object Design Pattern.

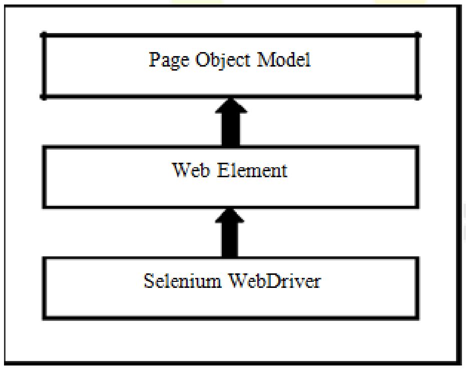

Fig 3.1. Simplified Architecture of Automation

Framework
TestNG:

In the framework that is being developed, TestNG is used for writing the test code.

Features of TestNG Framework:
TestNG supports many powerful features & it is easy to use.

- It supports the use of parameters.
- It supports testing of methods that are dependent on each other.
- The test configuration is flexible.
- It supports a powerful execution model.
- TestNG handles parameterized tests with the data-provider concept.
- Multiple tests can exist in a single TestNG test class.
- It can be used with different tools and plugins like Eclipse, Maven, etc.

Supports different Annotations like BeforeSuite, AfterSuite, BeforeClass, AfterClass, BeforeTest, AfterTest, BeforeGroups, AfterGroups, BeforeMethod, AfterMethod, DataProvider, Factory, Listeners, Parameters, Test.

Selenium can be used as a functional web testing tool. Selenium is not just a single tool. It can be used in four different forms: Selenium Grid, Selenium Remote control, Selenium IDE and Selenium Web Driver. Selenium IDE is installed as a Firefox plug-in. It can be used to develop test cases. Selenium RC (Selenium Remote control) runs tests in browsers suited for JavaScript.

Selenium Web Driver can be used on dynamic web pages. It has a well-defined framework. Selenium Grid tests can be run on different machines against different browsers.

Selenium automation testing tool can be included in testing frameworks for better results. Selenium can function with different Operating Systems, different browsers, many programming languages and other different testing frameworks. It can also follow different approaches of automation test testing for web based applications. Software testing can be an overhead and also a very tedious process when done manually.

Automating the test cases can reduce the overhead. Testing is a very important part of software development, because quality of the product really matters. Automation testing can save money and time and help business to accept the recent trends of the real world. Improper use of a testing framework can cause more damage than benefit.

SELENIUM COMPONENTS

- Selenium IDE.
- Selenium RC.
- Selenium Grid
- Selenium WebDriver.

Selenium WebDriver

Selenium WebDriver is also known as Selenium 2.0. The change brought about in the new automation framework is the integration of the WebDriver API. It is easier to use when compared to the Selenium-RC. The working of Selenium WebDriver is different when compared to that of Selenium RC. Selenium WebDriver can directly access the browser, whereas Selenium RC requires JavaScript functions to be injected into a browser when its loaded and with the JavaScript functions perform tests on the web application. WebDriver has built in support for each browser.

WebDriver is nothing but an interface, which is implemented by classes like FirefoxDriver, ChromeDriver, InternetExplorerDriver etc. Firefox driver does not require any additional installations, where as other drivers require additional installations. Their path must also be provided.

Examples of most commonly used commands using Web Driver can be explained using the default driver, FirefoxDriver and the programming language Java.

Selenium-WebDriver API Commands and Operations Fetching a page

The initial operation mostly done using a selenium web driver is opening a web page. The method used for this purpose is get. Eg: driver.get("http://www.vtu.ac.in");

Locating Web Elements

To perform operations on a web page, one must locate the web elements on the page and further use specific methods on it. We have several ways of locating a web element. It has been discussed below. Each location method uses the FindElement and also the By class.

- By Id
- By Class Name
- By Tag Name
- By Name
- By Link Text
- By Partial Link Text
- By CSS
- By XPath

Using JavaScript

Similar operations performed by web driver can be done using JavaScript. Sometimes certain operations may not be done due to complications on the UI using web driver. In such scenarios, we can use JavaScript.

Example:

WebElement signIn=driver.findElement(By.xpath("//*[@id=button"));
JavascriptExecutor jse=(JavascriptExecutor)driver;
jse.executeScript("arguments[0].click();",signIn);

Moving between Windows and Frames

Sometimes a web page can consist of multiple frames or the application can have multiple windows. In such scenarios to locate an element in a frame or another window, it is necessary to move the control to the frame or window first and then locate it by any of the above mentioned techniques. “SwitchTo” method helps to move between frames or windows.

Example:

Eg: driver.switchTo().window("window");
You can also switch from frame to frame:
Eg: driver.switchTo().frame("frame");

Pop Up Dialogs

A web application can even include pop up dialog boxes which is also known as alert. Usually we need to click Yes or No. Sometimes we might even have to enter value into a text box on the alert dialog box. In order to do any of these, it is first required to move the control to the alert dialog box.

Eg: Alert alertBox = driver.switchTo().alert();

Navigation

There exists another method along with “get” to open a new web page on a browser. It is called as “navigate”. It is very similar to the “get” method. Additionally, it also provides an option to move forward or backward on the browser’s history.

Example:

driver.navigate().to("http://www.vtu.ac.in");
driver.navigate().forward(); driver.navigate().back();

Enter Values to a text box

To enter values into a text box, the text box must be located first using any of the above-mentioned location strategies. Further a method called “sendKeys” must be executed on the text box.

Eg: WebElement element = driver.findElement(By.id("txtBox"));
Element.sendKeys(“HelloWorld”);

Selecting an option from a drop down

To select an option from a drop down, it is required to locate the drop-down element first and then search for the option by its text using a method called “selectByVisibleText”.

Example:

Select select =
new Select(driver.findElement(By.id("selectPlaces")));
select.selectByVisibleText("Bangalore");

Clicking a button

In forms after filling information, a submit button has to be clicked to submit the information to another page. To perform a submit, it is required to locate the submit button first and then perform a “click” operation on it.

Eg: driver.findElement(By.id("submit")).click();

IV. PROPOSED METHODOLOGY

Creation of an Automation Test Framework is a process of creating an infrastructure for software testing, utilizing various test automation tools. The major requirements of this automation infrastructure are to support entire software testing processes of all applications under a team, help in improving the quality of software testing process in all the applications, have a real or conceptual structure, be inclusive of set of processes, standards and interactions between the components where scripts are designed and executed.

The entire architecture of the framework is discussed in Fig4.1. Certain modules of the architecture have been created and utilized. The rest of the modules are yet to be developed or utilized.

The underlying code of the framework is selenium web driver. A wrapper code of selenium is used to create Automation Engine. The framework also has a reporting layer. Test reports can be sent as an e-mail, screenshot, log files or just as an html report. This can involve the code to be executed on a server such as Jenkins according to a schedule.

Using the Automation Engine, Page Objects are created. Page Objects include web elements as attributes and methods to access the web elements. Page validators are also used to validate the loading of a page. This helps for the control to wait for the complete loading of the page before accessing the web elements. A page load time out is associated with the page validation.

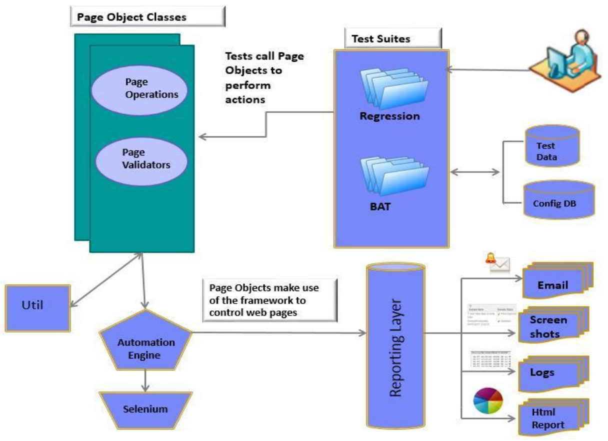

Fig 4.1. Detailed Architecture

The dependency between classes has been removed with the help of Dependency Injection by Google Guice. It provides annotations like @Inject. Dependency between classes are provided through constructors.

Multiple web browsers are supported by the framework such as Mozilla Firefox, Google Chrome and Internet Explorer.

Tests are created with the help of TestNG framework. It is available as a jar file or even as an eclipse Plug in. Tests can be run as a test suite, involving multiple tests in a single test class or independent tests. Even tests that are dependent on each other can be performed. Tests can be Regression tests or Build Acceptance Tests. Build Acceptance tests check if the particular data is accepted by the test or not and Regression tests are used to check if existing features in previous versions are still valid in current versions.

THE PAGE OBJECT MODEL

This is a design pattern followed to maintain details about a page separately from a test scenario. This involves creating a class called Page class for every page of a web application. The page class contains as attributes every web element and as methods operations performed on these methods. Methods are written using the WebElement. Separate classes called Test classes contain the test code. This is useful while making changes to a functionality, because changes made at one place need not be made in all the different tests. During tests, we can move control from one page to another by creating instancesof these page classes which are called as Page Objects.

Example of Page Object Pattern: -

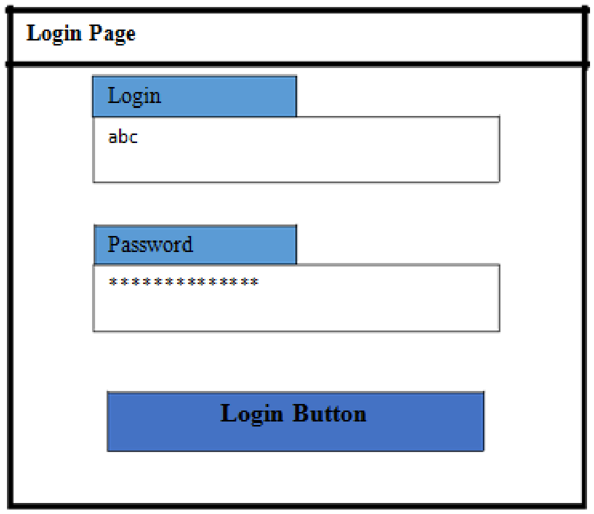

Fig 4.2. Sample Login Page

The above page is a Login Page. It contains the following elements:

- login field
- password field
- Login Button

And the following functionality:

- To enter username.
- To enter password.
- To click on the login button
- To log in to the application;

Hence a class 'LoginPage' should be created describing the above functionalities as well as the elements.

V. RESULTS

The requirement of this project was to reduce as much as time possible that is spent on manual tests by automating the tests. Example there are six retailers and each retailer has a minimum of 1500 tests. A minimum of ten tests on each retailer are performed every day which amounts to a minimum of 60 manual tests per day. Some of the regular manual tests have been automated. This reduces wastage of time which can be used for other productive work.

Earlier methodologies had a traditional pattern of coding. The locating strategies and test code were written together in a single class. Here isolation of code is very difficult in case of changes. A change made at one location will have to be made at all the locations. The new design pattern of Page Objects has made remarkable re use of code and efficient usability.

VI. CONCLUSION

Automating manual tests have been quite successful when compared to the existing methodologies used for the same. Even though initially it was difficult, to create page objects and automate tests, over a course of time, working code could be re-used and tests could be automated at a faster rate. The difficulty of automating tests increases with the complexity of the technology used to develop the web application. The only accessibility to the application for an automation tester is with UI code like HTML. Hence things can get difficult when unique ids or names are not available or when certain web elements are un reachable. Selenium Web Driver has handled almost all the available scenarios required for automation testing, and mostly the failures or difficulties in automation testing are due to lack of practice in certain areas. Experience is the only way we can tackle situations, and these situations vary from applications to applications. Being able to tackle situations of one application does not make a tester an expert in handling all web applications. Automation framework creation not only requires the knowledge of Selenium Web Driver, but certain areas requires reasonable amount of knowledge of Java to create the framework and re use it for creating new automated tests.

VII. FUTURE SCOPES

Timing issues can be a problem while automating tests. Certain web elements might take time to load and hence the control should wait until the element is available to continue the workflow of the test. Page validation is required to handle the loading of web elements. This is a scenario which still needs to be understood better in this project. Identifying the web element that loads the last has to be identified for this purpose and it has been difficult to identify the same. Another scenario, is logging into the application more than once. Currently the framework provides a single login system and hence multiple logins have to be considered and implemented. The new flexibility to run tests from command line and also from eclipse has been provided. Scheduling these tests to run on the server un monitored by testers is a next step to be achieved, along with increasing the number of automation tests of the application.

REFERENCES

[1] Chandraprabha, Ajeet Kumar, Sajal Saxena - "SYSTEMATIC STUDY OF A WEB TESTING TOOL:SELENIUM". IJARSE -ISSN-2319-8354(E), Vol. No.2, Issue No.11,Pages-113-120 , November 2013

[2] Ms. Rigzin Angmo, Mrs. Monika Sharma-"Selenium Tool:AWebbasedAutomationTesting

Framework".IJETCAS-ISSN (Online): 2279-0055. 8(4),March-May, 2014

[3] Deepti Gaur, Dr. Rajender Singh Chhillar- "Implementation of Selenium with JUNIT and Test-Ng".IJCSMS International Journal of Computer Science and Management Studies, Vol. 12, Issue 03,ISSN(Online): 2231-5268. Sept 2012

[4] Fei Wang, Wencai Du-"A Test Automation Framework Based on WEB" 11th International Conference onComputer and Information Science (ICIS), ISBN:978-1-4673-1536-4 IEEE/ACIS May 30 2012-June 12012

[5] de Castro, A.M.F.V. Macedo, G.A. ; Collins, E.F. ; Dias-Neto, A.C.- "Extension of Selenium RC tool to perform automated testing with databases in web applications" 8th International Workshop on Automation of Software Test (AST), -INSPEC Accession Number: 13752093 - IEEE 18-19 May 2013

[6] Dianxiang Xu, Weifeng Xu; Bavikati, B.K. ; Wong,W.E.- "Mining Executable Specifications of WebApplications from Selenium IDE Tests"- IEEE Sixth International Conference on Software Security andReliability (SERE), ISBN: 978-1-4673-2067-2 -IEEE 20-22 June 2012