International Journal of Engineering Technology and Applied Science
(ISSN: 2395 3853), Vol. 2 Issue 8 August 2016

Advance Automation Script in Selenium using Page Object Model

Mr. Sumit Sabu[1], Mrs. Priyanka Dhasal[2]

M. Tech, Information Technology Branch, Rajiv Gandhi Proudyogiki Vishwavidyalaya, Patel College of Science & Technology (PCST), Ralamandal Indore, Madhya Pradesh 452020

Assistant Professor, Information Technology Branch, Rajiv Gandhi Proudyogiki Vishwavidyalaya, Patel College of Science & Technology (PCST), Ralamandal Indore, Madhya Pradesh 452020
sabu.sumit@gmail.com[1], priyanka.dhasal@patelcollege.com[2]

ABSTRACT

Every automation script uses the locators to find the elements and perform action on those elements. Sometimes same element is used for different actions or more than one action are performed on single element, this leads to use of that particular element multiple time. This eventually leads to the duplicity of functionality.

As a result of that automation code of that element also got duplicated. If some locator has changed, then person needs to go through whole test code in order to change the locator whenever it is necessary.
By using Page Object Model we can reduce the duplicity of the code. By using this technique the readability and consistency of the code is much more improved.[1] An implementation of the page object model can be achieved by separating the abstraction of the test object and the test scripts.

Current automation scripts are designed to locate HTML elements by using Page Object Model.This script will help the user to perform repetitive regression testing task with much more less effort and minimum amount of time.

KEYWORDS

Automation script, Java code in selenium, UI Element Automation, POM automation.

INTRODUCTION

Test automation has been quite hot topic for multiple years. There are many problems in that field. Some people are seeing it as "total no no no - just waste of money and time", some people think it is the silver bullet, which solves all problems. My view is somewhere between those two extremes.

While using the Page Object Model, test objects are separated with test scripts. With this type of coding architecture, we create a code which is more usable, more easy to maintain as instead of the multiple usage of same element, whenever it is necessary; we need to maintain the locator identity at one place only. Any architecture should contain various important modules like Planning, Scripting, Coding and Reporting. Planning, Scripting and Coding is a part where most of the technical persons understand the importance of the standard practices.

But for a non-technical user Reporting is the main concern. They want the report to be as user friendly as possible. By using the reporting architecture of TestNG, we will be able to create a report in canonical form. In the current implementation of our algorithm, we have taken the screenshot for various cases (like page-navigation, register form various validations etc.) We have programmatically arrange those screenshots date wise in an output folder of our own choosing. In this way we can keep track of the daily recorded screenshots.

ARCHITECTURE AND TECHNIQUE

The architecture of current automation script is POM (Page Object Model).[2][3] A Page Object Model is a design pattern that can be implemented using selenium web driver. It essentially models the pages/screen of the application as objects called Page

Objects, all the functions that can be performed in the specific page are encapsulated in the page object of that screen. In this way any change made in the UI will only affect that screens page object class thus abstracting the changes from the test classes.

The reporting module of the system is our own designed custom folder which contains the screenshots of the navigated pages and also the screenshot of the same page under different conditions. All these screenshots are managed in such a way that any user (technical or non-technical) can easily understand the result. The TestNG reporting module creates the report on method basis, as Pass, Fail or Skip.

Furthermore we are creating data driven program, i.e. we will fetch the test data from the excel sheet. We are collaborating the test classes with XML files. From these XML files we will invoke the test class's methods. For any non-technical user, all these stuff is quite cumbersome. So we came up with a solution of .BAT file. For running the XML file, we are creating a .BAT file. User can simply double click on .BAT file, and our test will run like any other softwares usually install in the system.

The main advantage of Page Object Model is that if the UI changes for any page, it don't require us to change any tests, [4] we just need to change only the code within the page objects (Only at one place).

EXPECTED RESULT

Main aim of creating this algorithm is to provide such a reporting mechanism which is easily understandable and user friendly. User can customize the output folder according to its need.

In the current implementation the algorithm produces the resulting screenshots folder inside the main project folder. The screenshots folder contains another folder named after the method name. So that end user could easily identify which test just runned and screenshot belongs to that particular test. Programmatically setting the folder name points towards the method name will provide a lot of ease for the end user. Then for the periodic run of the test we have again goes one stop dipper and create another folder with Current DateTime name. This folder contains the screenshot with name of the html page. We can use date only but if user runs the test for multiple times during same day then too the mechanism creates another folder.

Below figure shows the folder structure of the current implementation.

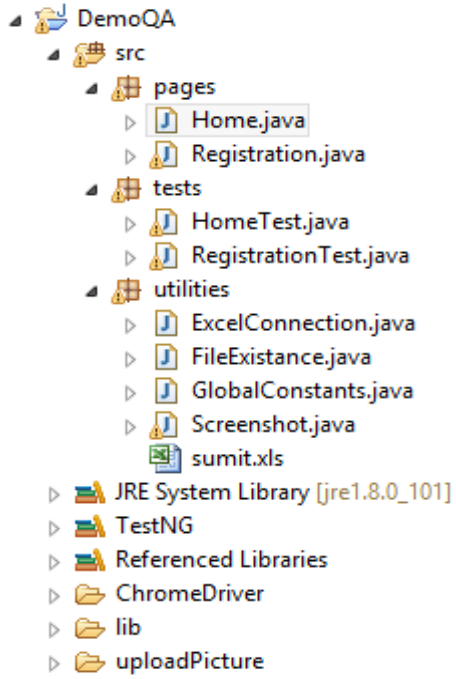

Fig 1 : Folder Structure

The figure 1 shows the folder structure of the current implementation. We have separated the Test classes with corresponding Page Object classes. The "lib" folder contains the necessary libraries for selenium to run the program.

Figure 2 shows a sample for the Page Objects for a particular page. Basically we are converting the various html elements of the web page to the corresponding page objects. We have used the selenium methods for extracting the web elements. Current figure shows the usage of " XPath", we can also use the "ID","NAME", or "CSS" to locate that web element.

*Home.java

```
package pages;
import org.openqa.selenium.By;

public class Home {

    WebDriver driver;
    Screenshot screenshotObj;

    public Home(WebDriver driver) {
        this.driver = driver;
    }

    By img_logo = By.xpath(".//*[@id='site_navigation'
    By home_tab = By.xpath(".//*[@id='menu-item-38']/a
    By aboutUs_tab = By.xpath(".//*[@id='menu-item-158
    By services_tab = By.xpath(".//*[@id='menu-item-15
    By demo_tab = By.xpath(".//*[@id='menu-item-66']/a
    By demoDraggable_tab = By.xpath(".//*[@id='menu-it
    By demoTabs_tab = By.xpath(".//*[@id='menu-item-15
    By blog_tab = By.xpath(".//*[@id='menu-item-65']/a
    By contact_tab = By.xpath(".//*[@id='menu-item-64'

    public void pageNavigate() {
        screenshotObj = new Screenshot(driver);

        if ((driver.getTitle()).equals("Demoqa | Just
            screenshotObj.CaptureAndSaveScreenshot(Glo
```

Fig 2 : Pa

APPLIED ALGORITHM

1. Create page objects for all available pages in our application.
2. In those pages we will try to get the handle for the elements present on the corresponding pages using predefined By class.
3. We will need to apply different approaches to extract the element. (For example : ID,NAME,CSS or XPath).[6]
4. Create test classes for each individual page present in the application. These classes contain the methods, which are responsible for invoking the action methods of the corresponding pages.
5. Screenshots of the desired screen and condition (either Fail or Pass) with suitable name and location is taken and saved to the directory.
6. Store the screenshots in the folder of our choosing.

RESULT

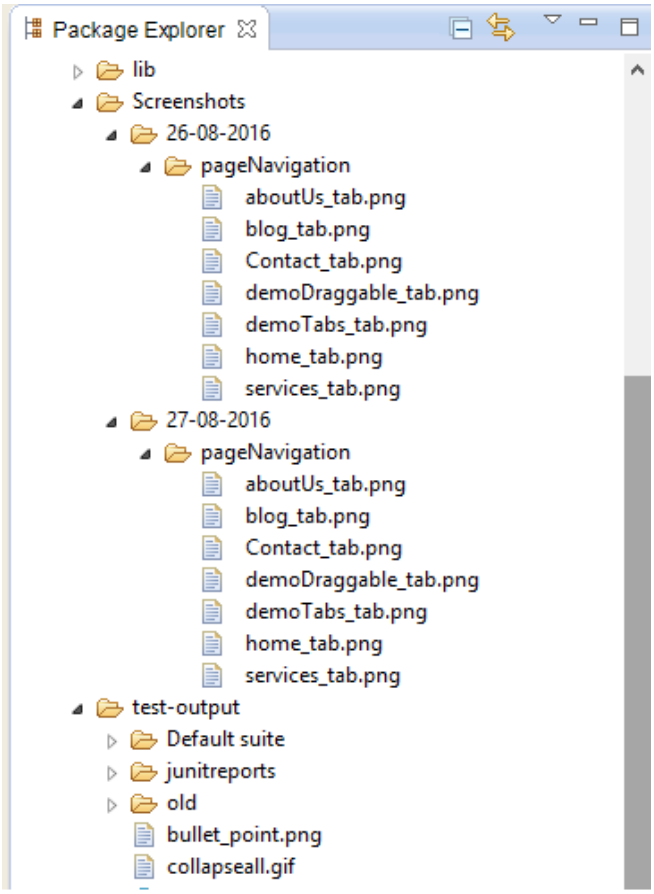

Fig 3: Resulting screenshots

After the implementation of this script of selenium, users will be able to see the screenshot (at desired conditions) of the desired pages at any particular time (depends on the requirements/need of that). Previously explained methodologies provides the reporting format of TestNG architecture.
Current automation script provides the customized folder structure for the resulting screenshots. User can use those screenshots for further analysis. Date and time wise arranged screenshots are named after the page we navigate or method we are executing.

Figure 3 shows the structure of the Resulting Screenshots folder. This folder contains the screenshots properly arranged according to the date.

This figure shows 2 new folders:

1. **Test-output Folder** ; This is a automatically generated folder which is generated by TestNG architecture. There is an index.html file inside this folder. That

file holds the results of the currently run scripts methods.

The methods are shown in the canonical form, with Pass/Fail/Skip tagged attached to them.

2. **Screenshots Folder** : This is a custom folder created by this script. This folder holds the screenshots that were taken during the execution of the code.

CONCLUSION AND FUTURE WORK

Automation is the technique which reduces human efforts. Automation enables user to perform the test scripts quickly and repeatedly. Selenium being a open tool provides support for many languages.[5] We have created a custom output folder which provides the ease and comfort for the user to easily analyze and review the screen of the site.

As different browsers behave in different manner.. Sometimes button's corner are rounded in Google chrome and the same button's corner are edged in the Firefox. IE(Internet Explorer) also behave in different manner. In order to be able to 100% sure, we need to take the screenshot for different browsers and also now a days websites are not only adaptive but they are also responsive in nature. So in future we need to extend this script to be able to run on different browsers and also on the mobile devices with different platforms like Android, iOS, Windows etc.

REFERENCES

[1]. http://www.ijarcce.com/upload/2016/may-16/IJARCCE%20105.pdf
[2].https://www.seleniumeasy.com/selenium-tutorials/test-automation-scripts-structure
[3].http://www.sciencedirect.com/science/article/pii/S1877050915005396
[4].http://www.ijcsit.com/docs/Volume%205/vol5issue01/ijcsit20140501196.pdf
[5]. http://eliga.fi/Thesis-Pekka-Laukkanen.pdf
[6]. http://www.guru99.com/page-object-model-pom-page-factory-in-selenium-ultimate-guide.html