8/29/2016

Certification Project

Testing With Selenium WebDriver

By Pankaj Dhande

TABLE OF CONTENTS

TESTING WITH SELENIUM WEBDRIVER

INTRODUCTION

This document enlists the architecture and functioning of Hybrid automation framework that is established to perform automation testing using Selenium WebDriver. This document also enlists the features of the framework, details about the automated scenarios, and limitations/risks. The framework is a hybrid one, i.e. it combines the features of both, keyword as well as data driven framework. Framework accepts inputs from the excel sheets, and after execution, the results are updated also in an excel sheet. The framework uses Page Factory and Page Object Model, both of the design pattern for ease of coding and maintenance.

FRAMEWORK ARCHITECTURE

The framework comprises of below packages and classes:

- **Package appModules:** This package consits of the application module, i.e. the code releated to the AUT. It contains classes like setup, mercuryRegistration and mercuryFlightBooking. Here is the information about the classes in this package in breif:
    - Class **setup:** Consists of code related to login and other validations.
    - Class **mercuryRegistration**: Consists of code for new user registration functionalty.
    - Class **mercuryFlightBooking**: Consists of coder for flight booking functionality.

- **Package commonLibs:** As the name suggests, this package consists of all the methods that are common across the framework. This package addresses the problem of code duplication by maintaining all the frequently needed methods at single point. Here is the information about the classes in this package in brief
    - Class **dataProvider**: This class contains methods that acts as data supplier. These methods fetch data from input excel file, and pass it on to the calling method. The method usually can return data as a single value, arrays, and objects.
    - Class **excelDriver**: This class contains all the methods that are needed to interact with excel files. Excel file opening, reading, writing and saving can be performed using the methods of this class.
    - Class **keywordUtility**: This is the class where framework keywords are designed. These keywords are called from the main class of the framework. The methods are specified as test case steps in an excel sheet. This class also initialized WebDriver, that could be passed across different page object classes to perform that specific page related operations.
    - Class **Extent**: This class contains methods to write the framework reporting and logs. A utility named "ExtentReports" is used here to fulfil the purpose.
    - Class **utils**: This class contains various methods which are needed across the timespan of automation testing. Those methods are accumulated in this single class, in order to avoid code duplication and easy maintenance.

- Package **frameworkDriver**: This package contains class **automationEngine,** which holds the main method for the framework. Here is the information about the classes in this package in brief:
    - Class **automationEngine**: This class contains the main method. It reads the test suite from excel file and fires the test execution.

FRAMEWORK FEATURES

Here are the features offered by this framework:

- Keyword driven: The test cases and test steps are nothing but the keywords. You may use them in any logical order and they should work
- Data driver: All of the data needed for the execution is fetched from Excel sheets, which makes it easier to modify the data.
- Easy to maintain: POM and PageFactory makes this framework very easy to manage and maintain.
- Detailed reports: Extent reports used in framework provided beautiful reports in HTML format, which takes out need of any other reporting.
- Automated emails: Reports are automatically emailed to the addresses specified in the input excel sheet once the execution is over. Hence execution status can be checked without logging into the automation machine

Note: To send automated email, you need

1. Log into your gmail account which you are going to use as a "FROM" account.
2. Navigate to account security settings using URL: https://myaccount.google.com/security
3. Set Allow less secure apps: ON

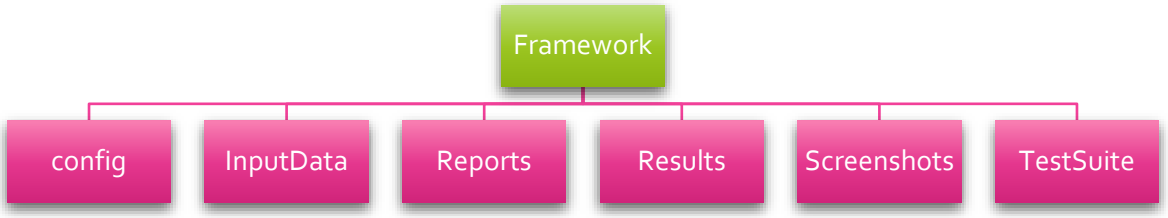
FRAMEWORK CONTROL FLOW

Here is the flow of the framework for a sample test case which shows how the framework works.

1. Excel sheets are prepared, with input data from test suite. The sample excels are attached with the code.
2. When the framework is invoked, first the framework reads the properties files to get the folder paths for input test data file and test suite file.
3. Framework reads the test suite excel file using the methods from **commonLibs.excelDriver()** and reads the test suite to fetch the list of test cases. Test suite is read by **testSuiteDriver** method. The test cases then one by one are passed to method **testCaseDriver.**
4. We have kept the test case names and the test case excel worksheet names same, which makes it easy for the framework to navigate within the excel sheet.
5. Thus the method **testCaseDriver** navigates to the respective test case sheet and gets a list of test steps within it.
6. The test steps are nothing but a list of keywords that framework understand. These test steps/keywords are passed to **keywordUtility** one by one.
7. **keywordUtility.runMethod** identifies the keywords and calls the respective method from the respective page class.
8. Page class methods return the execution results to **keywordUtility**, which is then passed onto **testCaseDriver** method.
9. **testCaseDriver** method updates the result right in front of the respective test step.
10. Once all the test steps are executed, the test case result is then passed onto **testSuiteDriver** method.
11. **testSuiteDriver** method updates the test case result in the test suite, and then picks up the next test case from the list.
12. Once all the test cases are executed, the final results are then saved to a new excel file in the Results folder.
13. HTML reports are saved in the "Reports" folder.
14. Captured screenshots are saved under **screenshots** directory.

DIRECTORY STRUCTURE

The framework uses different directory in order to interact with excel files, and to save results and screenshots. Below is the directory structure:



Config: This directory contains the properties file (config.properties).

InputData: This directory contains the excel file which holds all the input test data (TestData.xlsx).

Reports: This directory contains HTML reports generated by framework

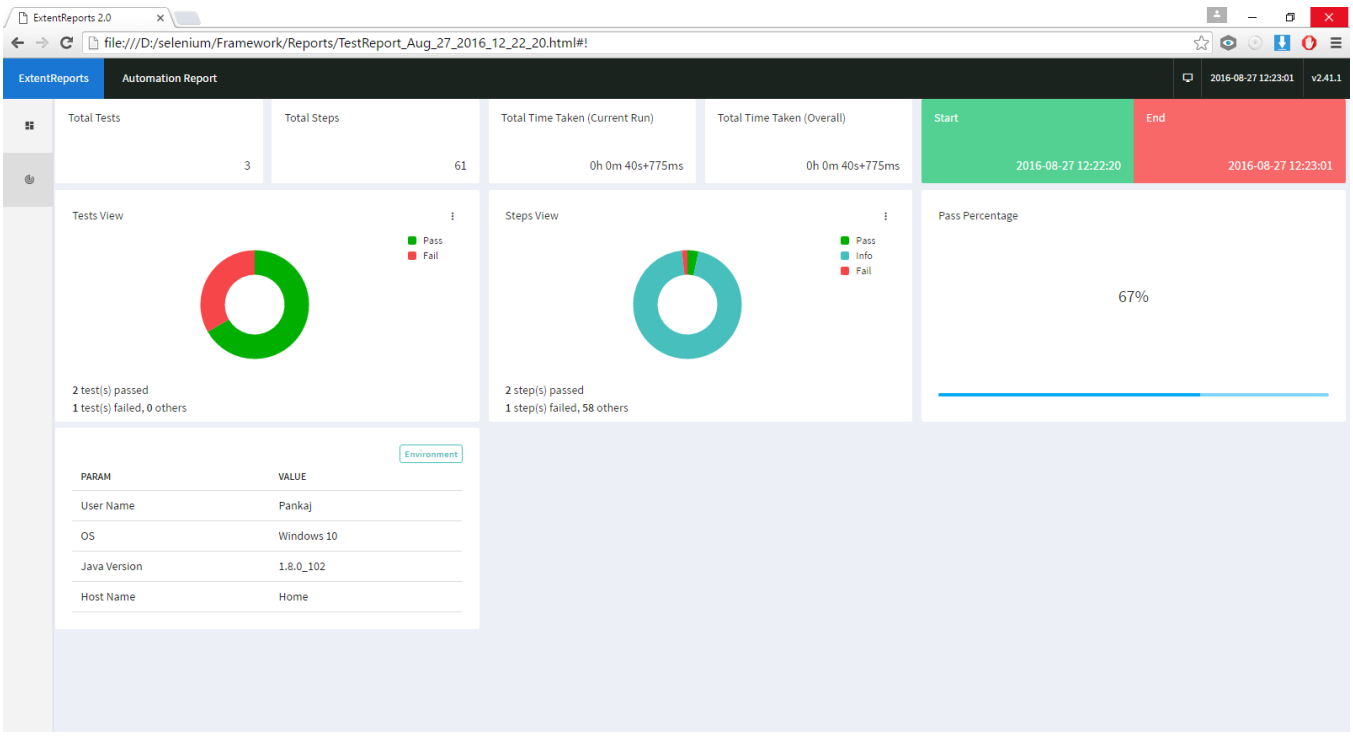
Results: The results generated by the framework are kept in this directory.

Screenshots: Screenshots captured during execution are kept in this directory.

TestSuite: Test suite excel is kept in this directory (AutomationTestSuite.xlsx)

REPORTING AND LOGGING

This framework uses and external plugin called "ExtentReports" for reporting and logging purpose. Extent report provides .html reports in graphical format along with test case and test step wise status. The best part is that it also handles logging part, hence we get everything at the same place. Here is a sample report



HOW TO RUN THE TESTS

I have written a set of test cases with this framework for demonstration purpose. These test cases demonstrate the features of this framework. Below is how you can get the test care running.

1. Open the zip file and extract the contents. Load the project in eclipse.
2. I have configured default build path as D:\selenium\lib. You may either put all of your jar files in this path, or you may configure your own build path.
3. Copy the directory Framework to D:\selenium
4. From the eclipse, right click the file automaionEngine.java and select run as->Java Application.
5. The framework will invoke browser and perform the automated runs.
6. The results are saved at Results folder (under path D:\selenium\Framework\Results)
7. Screenshots are saved at screenshots folder (D:\selenium\Framework\Screenshots)
8. Logs are saved in the directory where code resides.

COVERAGE

There are a set of functionalities/actions that have been asked in the certification project PDF file. Below is the brief picture of what is covered as a part of this project.

Action	Status	Comment
page synchronization	Covered	
verification of page title and URL	Covered	
validation of read-only and disabled property	Not Covered	No disabled/read-only UI control found in AUT
existence of web element and validation of checkbox and radio button for check/selected	Covered	
take screenshots with time stamp	Covered	
find object using id	Covered	
find object using css	Covered	
find object using classname	Not Covered	No UI control available with classname in AUT
find object using tag	Covered	
find object using xpath	Covered	
Find objects using absolute and relative xpath	Covered	
Project should be designed using TestNG	Not Covered	Used hybrid framework for more scalability and customizations

APPLICATION UNDER TEST

I tried several applications mentioned in the certification project PDF file, but most of them didn't work. They were down all the time. I could find only one application that was running all the time, i.e. "Mercury Tours". All the automation was performed on Mercury Tours application.

Here is the URL for the AUT: http://newtours.demoaut.com

FUTURE IMPROVEMENTS

The current implementation of the framework is very basic as the project submission is time bound. This framework will further be developed to incorporate below features

1. Database connectivity for input data and test suites configuration.
2. Report export to PDF format
3. Saving report parameters to database and generating report any time for any execution run.
4. Selenium Grid integration
5. Email integration for options other than Gmail

Further development on this framework could be checked at Github:
https://github.com/pankdhnd/CertificationProject-HybirdFramework

REFERENCES

1. www.edureka.co
2. www.guru99.com/
3. http://learn-automation.com/
4. http://toolsqa.com/