Automation Testing Using Data Driven Approach

Ayush Gondhali
Computer Department
RMD Sinhgad School Of Engineering
Pune-58, INDIA
yush.gondhali@gmail.com

Rishikesh Chandra
Computer Department
RMD Sinhgad School Of Engineering
Pune-58, INDIA
rishirebel@gmail.com

Ashish Shinde
Computer Department
RMD Sinhgad School Of Engineering
Pune-58, INDIA
ashish.shinde1010@gmail.com

Sanket Pimple
Computer Department
RMD Sinhgad School Of Engineering
Pune-58, INDIA
sanketpimple@gmail.com

Abstract— As we all know that Software testing is time –consuming and expensive process, especially in safety-secured applications. Testing Automation would allow us to reduce development cost and will help us in increasing the quality of Software. Automation Framework follows Data Driven Approach which passes data stored in Spreadsheet as input to the number of test scripts written for executing test cases.
We have developed this Framework by using Selenium IDE and languages such as Java and HTML. Framework will automate the testing of software in different phases using a Driver Script and each phase will test the web application and generate the final reports as screenshots, logger report and HTML report. These reports are generated automatically with the help of the driver script and selenium webdriver and the complete final report is sent to the client as an electronic mail.

Keywords*- Selenium Web driver, Automation, Data Driven, Driver Script, Test Case.*

*****

I. INTRODUCTION

We all know that the need of technology is growing day by day and so is the speed of manufacturing such equipment's and software's capable of satisfying the same is also increasing with the same pace. But at the same time we also need the quality assurance for all those products and software's so that they can meet the needs and the standards of the clients and the customers which will result in having a long term beneficial and happy relationship with the customer/clients or end user. And to do the same we need to monitor and test the tool on regular basis either by providing the product lifetime support or by regular testing

The abrupt increase in the growth of Software Industry has brought numerous web applications in the market and these applications are being used by almost everyone hence the correctness of these applications is very much necessary as a small error in the application can lead in customers disappointment and can bring a patch on the name of the Developing Company.

Hence testing of all these application is mandatory before launching them in the market between so many users. But at the same time testing all these application will require a lot of time and human effort and can also have some probability of human errors. Therefore Automation in testing can bring a revolutionary change and can help in saving a lot of time and resources.

Our Automated Testing Framework automatically takes the test cases and accordingly select the testing functions/scripts and side by side generates a logger report, HTML report and screenshots of each instance of testing and finally generating a complete report on tested application/software, after comparison of the actual result with the expected result.

II. FRAMEWORK ARCHITECTURE AND FLOE DIAGRAM

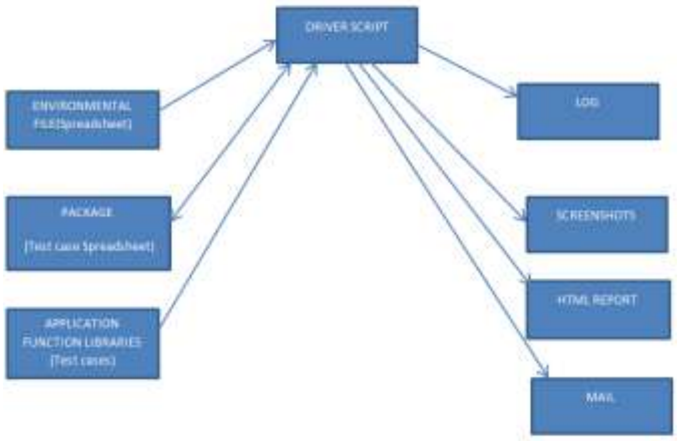

DRIVER SCRIPT
ENVIRONMENTAL FILE(Spreadsheet)
PACKAGE
(Test case Spreadsheet)
APPLICATION FUNCTION LIBRARIES (Test cases)
LOG
SCREENSHOTS
HTML REPORT
MAIL

Figure 1. Work Flow Diagram

The above diagram shows the working and organization of different components involved in the framework.

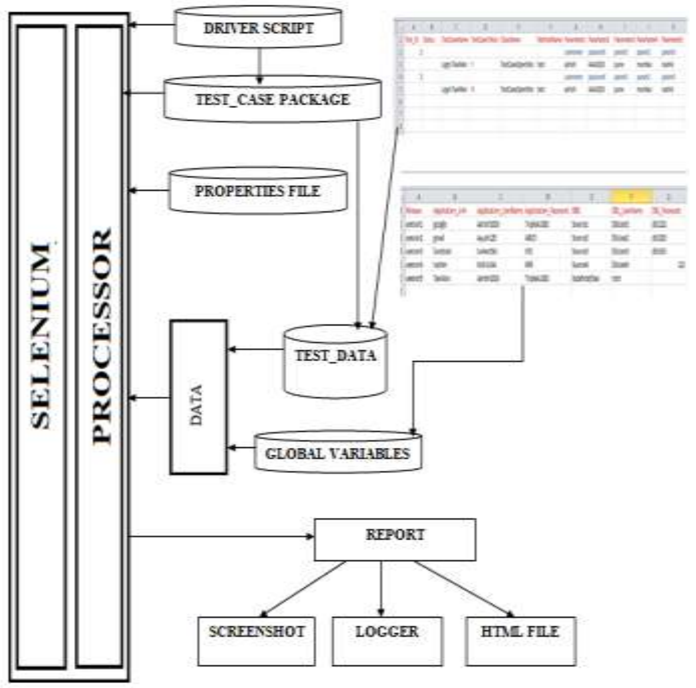


Figure 2. Framework Architecture

The architecture as shown above in Figure 2 shows the structure of the framework involving the Selenium Web driver as the main block for testing since it automates the browser and its components as needed to test the WebApp.

III. TOOLS USED

Selenium Web-driver is an open source testing tool for web applications. It directly runs on any of the web browsers such as Internet Explorer, Mozilla Firefox, and Google Chrome etc. As we have already used and did the execution job on all the browsers, we have observed that it works fast on Google Chrome.

Also we have used various .jar files such as Apache POI API for reading the data from excel files, Apache log4j for logging services, various .jar files for Selenium Web-driver and jdk -jre with Eclipse for building the whole framework. Also we have developed our own log file generator as there are some limitations with Apache log4j.

IV. ABOUT THE FRAMEWORK

Selenium Web-driver is an open source testing tool for web applications. It directly runs on any of the web browsers such as Internet Explorer, Mozilla Firefox, and Google Chrome etc. As we have already used and did the execution job on all the browsers, we have observed that it works fast on Google Chrome.

Also we have used various .jar files such as Apache POI API for reading the data from excel files, Apache log4j for logging services, various .jar files for Selenium Web-driver and jdk -jre with Eclipse for building the whole framework**.**

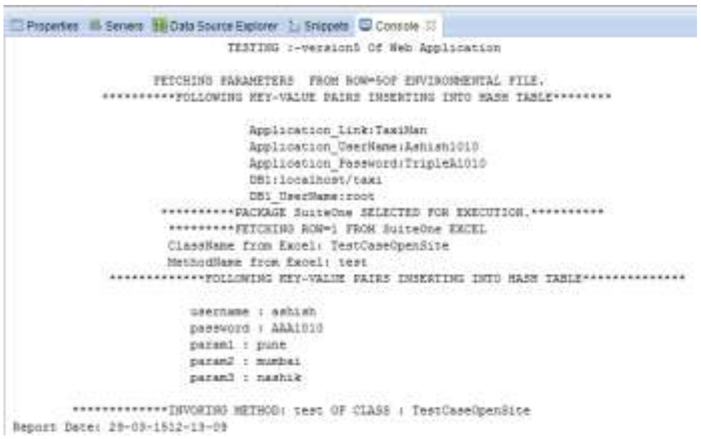
But the main framework is made in Eclipse so from there also we can run the framework. We can stop or pause the testing anytime. The testing process will stop automatically after finishing the test which also includes generation of success/failure screenshots, HTML reports, XSLT reports, Logger report and writing Passed/Failed/Skipped status in the reports and the excel files.

An integral part of the test execution is the verification of the test results. Our framework is made in such a way that it compares the actual outcome with the expected outcome and makes a decision. The expected outcome result is already provided in the excel sheet and the framework fetches this data from there.

V. WORKING OF FRAMEWORK

The basic data which will be needed while the execution of the test is provided in the excel sheets. These fields contain the columns such as Test Id, Status, Class Name, Test Case Name, Method Name, Test Case To Run, Parameter 1-5, Release, Application Link, Application Username, Application Password, DB1, DB1_Username, DB1_Password etc. These fields contain the necessary data for the execution of the Framework**.** Some of the data is fetched and loaded while some of the data is written in the columns after the execution in the excel sheets. Test Id contains the unique identification no for identifying the different tests. Status is the column which is left null at the start which is then occupied by the status such as Pass/Fail/Skip, returned by the execution of the test conducted. Class Name contains the name of the test case. Method Name contains the method which is to be used. Parameter 1-5 contains the different parameters which could be required while the execution of the Framework for example it could be Login name, Password or any name or number etc. Test Case To Run is the column which is specially designed to intake values as "y" or "Y" and "n" or "N" so that the tester has the option to either run the test or to skip it. Application Link contains the link of the application which is to be tested and further it contains Application Username, Application Password and other records which hold the necessary data to be used while testing.

Once the framework is started as shown in Fig.3, it loads the necessary data from the excel sheets and works according to the driver scripts.



```
                    TESTING :-version5 Of Web Application

              FETCHING PARAMETERS  FROM ROW=5OF ENVIRONMENTAL FILE.
        **********FOLLOWING KEY-VALUE PAIRS INSERTING INTO HASH TABLE********

                        Application_Link:TaxiMan
                        Application_UserName:Ashish1010
                        Application_Password:TripleA1010
                        DB1:localhost/taxi
                        DB1_UserName:root
               **********PACKAGE SuiteOne SELECTED FOR EXECUTION.**********
                *********FETCHING ROW=1 FROM SuiteOne EXCEL
                ClassName from Excel: TestCaseOpenSite
                MethodName from Excel: test
         *************FOLLOWING KEY-VALUE PAIRS INSERTING INTO HASH TABLE*************

                   username : ashish
                   password : AAA1010
                   param1 : pune
                   param2 : mumbai
                   param3 : nashik

      *************INVOKING METHOD: test OF CLASS : TestCaseOpenSite
Report Date: 29-03-1512-13-09
```
Figure 3. Showing the successful initialization of the Framework

Here the driver script are the java files which contain the commands and codes for the whole working of the framework. As this script is actually carrying out the working of the Framework that's why we call it the Driver Script.

While execution it opens the webpage as shown in Fig.4 to be tested which is already provided in the file called

Parameter.properties in which is used by the driver script by simply calling a function for it**.**

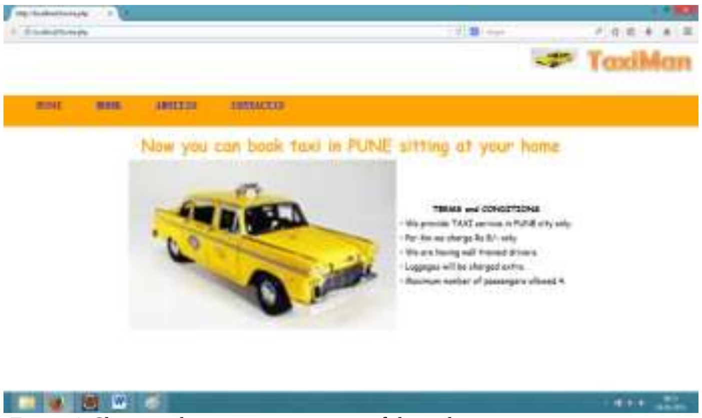
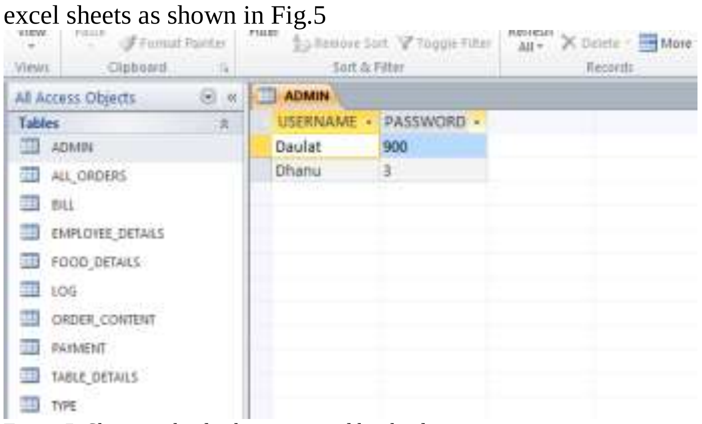


Figure 4. Showing the automatic opening of the webpage

This file also contains the browser which is to be used for opening the webpage. Hence due to this a webpage is opened and once the webpage is opened completely then the necessary input data is automatically inserted into that webpage. This automatic insertion of data. While this execution is carried out by the driver script, at the same time the driver script is also fetching the data from the web application database and other excel sheets as shown in Fig.5



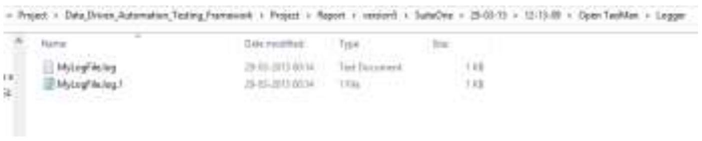
ADMIN

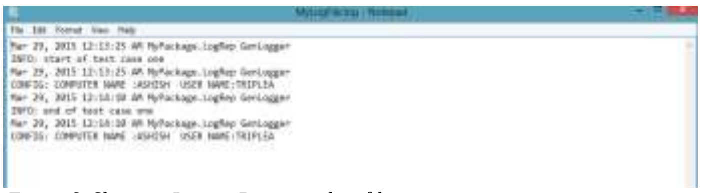
USERNAME	PASSWORD
Daulat	900
Dhanu	3

Figure 5. Showing the database accessed by the driver script

Also after this execution some of the required data is written on the excel sheet for example in the Status column of the excel sheet.

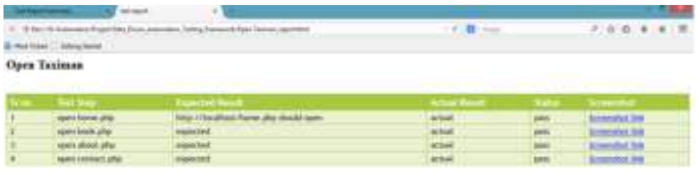
Once the process is over, the report files are generated by the framework automatically generates the Logger report as shown in Fig.4, the Screenshots as shown in Fig.5 and the HTML report as shown in Fig.6


Project › Data_Driven_Automation_Testing_Framework › Project › Report › version5 › SuiteOne › 29-03-15 › 12-13-09 › Open TaxiMan › Logger
Name
Date modified
Type
Size
MyLogFile.log
MyLogFile.log.1
Text Document
1 KB
1 KB



```
Mar 29, 2015 12:13:25 AM MyPackage.LogRep GenLogger
INFO: start of test case one
Mar 29, 2015 12:13:25 AM MyPackage.LogRep GenLogger
CONFIG: COMPUTER NAME :ASHISH  USER NAME:TRIPLEA
Mar 29, 2015 12:14:10 AM MyPackage.LogRep GenLogger
INFO: end of test case one
Mar 29, 2015 12:14:10 AM MyPackage.LogRep GenLogger
CONFIG: COMPUTER NAME :ASHISH  USER NAME:TRIPLEA
```

Figure 6. Showing Logger Report and its file



Open Taximan

Sr.no	Test Step	Expected Result	Actual Result	Status	Screenshot
1	open home.php	http://localhost/home.php should open	actual	pass	Screenshot link
2	open book.php	expected	actual	pass	Screenshot link
3	open about.php	expected	actual	pass	Screenshot link
4	open contact.php	expected	actual	pass	Screenshot link

Figure 7. Showing the HTML report

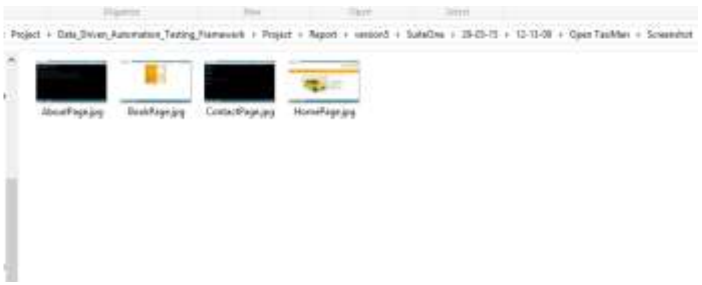

Project › Data_Driven_Automation_Testing_Framework › Project › Report › version5 › SuiteOne › 29-03-15 › 12-13-09 › Open TaxiMan › Screenshot
AboutPage.jpg
BookPage.jpg
ContactPage.jpg
HomePage.jpg

Figure 8. Showing the Screenshots generated

Also there is one more report which is automatically generated, which demonstrated the results in pie chart for i.e the XSLT report.

VI. Directory Hirarchy Of Framework

The Framework works in following steps:

1. Driver script fetches rows of Environmental file and compares its release version with that given in properties file.
2. Driver script generates Environment hash table for that row.
3. Now it reads spreadsheet whose name is given in properties file.
4. Now it generates Test Data hash table for a row and invokes the given test case.
   (The Tester may include the Framework's methods to generate HTML report, create Logs and store screenshots according to his will.)
5. The Logs, Screenshots and HTML reports are generated in respective directory.

   (Steps 4 and 5 repeat for every test case given in the spreadsheet.)

The files created and used by the framework are organized in the manner as shown in Fig.7. The well-organized hierarchy allows the tester to manage a plethora of reports generated as a result of testing large projects. The report folders are generated dynamically as needed and the logs, screenshots and the HTML reports are stored in the respective folders.

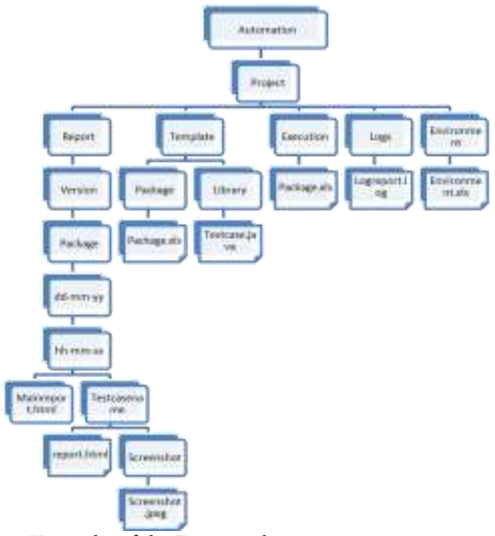


Figure 9. Directory Hierarchy of the Framework

VII. ACKNOWLEDGEMENT

We express our heartfelt gratitude to our respected Project guide Prof. Vina M. Lomte for her kind and inspiring advice which helped us to understand the subject and its semantic significance. She enriched us with valuable suggestions regarding our topic and presentation issues.

VIII. CONCLUSION

A test automation framework can be built around model based testing that begins with automating the test design exercise and the development of manual test procedures. It further uses automation frameworks to attain modularity in automated test script design, and integrates with test execution tools for automated test execution.

Such a framework will help test organizations to:

Greatly scale when test designers and test automation developers are in short supply by training regular testers in model-based testing and test automation

Deliver test automation quickly so that regression cycles can be automated right from the first cycle, and save effort and cost that would otherwise have been expended on manual testing.

REFERENCES

[1] Wikipedia: Test Automation http://en.wikipedia.org/wiki/Test_automation
[2] The History of Software Testing by Joris Meerts http://www.testingreferences.com/testinghistory.php
[3] BUSY DEVELOPERS' GUIDE TO HSSF AND XSSF FEATURES HTTPS://POI.APACHE.ORG/SPREADSHEET/QUICK-GUIDE.HTML
[4] JAVA LOGGING TUTORIAL:HTTP://WWW.VOGELLA.COM/TUTORIALS/LOGGING/ARTICLE.HTML
[5] Java I/O Manual https://docs.oracle.com/javase/tutorial/essential/io/
[6] Robot for Screenshot http://stackoverflow.com/questions/4490454/how-to-take-a-screenshot-in-java
[7] Java Robot Class http://docs.oracle.com/javase/7/docs/api/java/awt/Robot.html
[8] Data Driven and Keyword Driven Test Automation Frameworks by Pekka Laukkanen http://eliga.fi/Thesis-Pekka-Laukkanen.pdf