Overview:

Mercury project deals with the happy path journey of a registered user logging on to the travel site, search & select proper flight options, furnish the passengers/payment details, reserve the flight and finally makes a secure purchase successfully and logs out of the website.

I used ‘Test Automation’ process with Java/Selenium to accomplish the project accordingly, alongside of several functional testing tasks incl. exploratory manual/automation tests on hyperlinks, text links social contact links.

Furnished below is bit more information in relation to how I went about completing the task.

Prerequisites:

1. Download and install ***Java***
2. Download and install ***Eclipse***
3. Created project ***Mercury\_XML*** to work on the project using parameters
4. Download ***Reference libraries*** (JRE system libraries) via build path
5. Download all Selenium ***Jars*** (common, parameter, poi, selenium server standalone etc.)
6. Download and attach Selenium ***TestNG*** to the project
7. Download and configure build path for ***Chrome driver*** in the dedicated folder, e.g. ***driver*** in the project
8. Created folder ***images*** for the screenshots
9. Created folder ***documentation*** for the project docs to reside for instance-“Mercury project”

Project tasks:

Created four ***packages*** under ***src*** folder with a unique naming convention that states the project name(Mercury) after com. Followed by the specific name of the package, namely as below followed by step 3 of the prerequisites.

1. com.mercury.**basedriver**:
2. Created base driver class and initialized the chrome browser (open & close), while using several predefined methods such as delete cookies, time outs, sleeps etc.
3. Base driver class is extended to all other classes and majority of the methods to reuse the code and the relevant functionality especially we are dealing with one static website hence the url is also embedded here in the base driver.
4. @before suite, @after suite, @before test, @after test, , annotations were also used here to make the tests robust.
5. com.mercury.**pages**:
6. Created few classes/pages (features)s like login, flight, passengers, credit card, billing and delivery addresses with the extension to the base driver.
7. Similar naming conventions used for class names.
8. Each class is made up of several methods that are built with various web driver pre-defined methods/connections and with the usage of variables, objects etc.
9. Required data for the methods is derived from the xl document */Mercury\_Excel/driver/Mercury\_ExcelData.xlsx*
10. The above excel sheet consists of data scattered across sheets and location indexes that is exported to project via excel predefined method.
11. Web elements were located by using the xpath mostly.
12. Screenshots, validation methods were also written under page classes.
13. System.out.println and reporter.log pre-defined methods were used to see the confirmations/user messages in console and TestNG reports.
14. com.mercury.**flows**:
15. Classes under flows package were created and linked to the page classes with objects with the extension to the base driver thus minimizing the code visibility.
16. Similar naming conventions used for class names.
17. com.mercury.**testcases**:
18. Created number of test case classes corresponding to the flows, which in turn condensed the code visibility to nearly one liner.
19. Combined all of the above pages and flows to create the test cases that will be used in the testing.xml to run the proper tests thru TestNG suite.
20. Test cases are named as TC followed with an increment of the numeric values to maintain consistency.
21. **@Test** annotations and **parameters** were specified on top of the methods.

@Test

@Parameters ({"cardtype", "number", "expmonth", "expyear", "firstname", "middlename", "lastname"})

Test case execution:

After the completion of the test case creation at the test cases package, it is imperative that we got to run these tests in order to accomplish the task. The actual test execution is done by following the below steps.

1. Selected the Mercury project and right clicked to choose the option **TestNG** and then selected **convert to TestNG,** which opens a window that creates the testng.xml at the project location.
2. Folder **test.output** and **testing.xml** are now visible associated with the mercury project.
3. Open the testing.xml and add the parameters under annotation-suite 🡪 @before suite, @after suite
4. Specify the name and supply their values for the **parameters** used throughout the project within the parameter structure in the xml file for e.g.

<parameter name=*"url"* value=*"http://newtours.demoaut.com/"*/>

1. Class names that are under test along with the base driver class name are specified under the classes and within the class parameterization.

<classes>

<class name=*"com.mercury.basedriver.SetupBrowser"*/>

Run TestNG Suite:

1. Now right clicked on the xml file and selected “Run As” then opted for ***TestNG Suite,***
2. Test cases then ran in the order that they are specified.
3. Chrome initialized, website opened and did all of the activities that were coded in the pages package.
4. Finally, the browser closes as per the suite requirement.

Deliverables:

1. Console:
2. Test report is shown under suite, total test cases ran and pass/fail numbers - as expected
3. System.out.println statement messages are printed – as expected
4. ResultsOfRunningSuite:
5. Test report is shown under suite, total test cases ran and pass/fail numbers - as expected
6. Summary and failure exception messages shown – as expected
7. Test-output:
8. Information regarding the test report as to total test cases/passed/failed etc is stored in the test-output folder. - as expected
9. Test results are stored here in different formats, so we can use them in our convenient format. - as expected
10. Index.html and emailable reports are the prominent one’s that can be used to send reports to stake holders. - as expected
11. Images:
12. Screenshots acquired from the tests are stored in the images folder - as expected
13. Screenshots are stored in png format
14. Comments:
15. Additional information/description of activities is noted on top of methods/classes.
16. Forward slashes (//) is used to denote those comments as below

*driver*.get(url); //travel site opened

Conclusion:

It is hereby decided that we follow the same procedure for the rest of the modules of this project.

*Author: Gowri*

*Job title: Test analyst*

*Created date: 17th June 2020*

*Modified date: 19th June 2020*

*Document source: C:\Users\gowri\OneDrive\Desktop\SeleniumAutomation\SeleniumWorkSpace\Mercury\_XML\documentation\Mercury Project*