Description: Description: Description: Description: Description: Description: Description: Description: C:\Users\pmike\Desktop\OptumGraphic.gif

**SELKEY ASK FRAMEWORK**

**An excel base framework to help transition from manual tester to Automation**

**Kumar, Naveen**

**INDEX**

**Pre-Requisites……………………………………………………………2**

**PRE-REQUISITES**

**Jboss: (**[**http://appstore.uhc.com/AppInfo/AppVersionId/11755**](http://appstore.uhc.com/AppInfo/AppVersionId/11755)**) *(For Java Dev Only)***

**Java *(Compulsory for all)***

**Check you have Java installed:**

Go to: C:\Program Files\Java

Check you have “jdk1.8.0\_51” folder.

***Install Java(If you don’t have):*** *Raise request for Java at* [*http://avs.uhc.com/Profile.aspx?AppKeyID=16747*](http://avs.uhc.com/Profile.aspx?AppKeyID=16747)*, fill up the attached sheet and send BSL for approval:*



If you have any of the folder, go to command prompt and type “java –version”

If installation is correct you will get the version else some error.

If you get error raise a request for correction.

**Maven**

**Check Maven Installed in System: *(Compulsory for all)***

Go to command prompt and type “mvn –version”

If version is not shown raise it from app store: <http://appstore.uhc.com/AppInfo/AppVersionId/10244?BackToList=/AppList/AppList>

**Check you have Maven plugin in Jboss: *(For Java Dev Only)***

Go to File-> New -> Others

Scroll and verify you can find “Maven” in the ‘Select a wizard’ window.

***If you don’t have maven plugin in Jboss, install it:***

Go to url: <https://marketplace.eclipse.org/content/maven-integration-eclipse-luna-and-newer>

Drag and drop the install button to your eclipse to install maven plugin

**GIT**

**Check GIT Installed in System: *(Compulsory for all)***

Right click anywhere in any folder, should see “Git Bash” and “Git GUI” in context menu

If not shown raise it from app store: <http://appstore.uhc.com/AppInfo/AppVersionId/12090?BackToList=/AppList/AppList>

**Check you have Git plugin in Jboss: *(For Java Dev Only)***

Go to File-> New -> Others

Scroll and verify you can find “Maven” in the ‘Select a wizard’ window.

***If you don’t have Git plugin in Jboss, install it***

Go to url: <https://marketplace.eclipse.org/content/egit-git-team-provider>

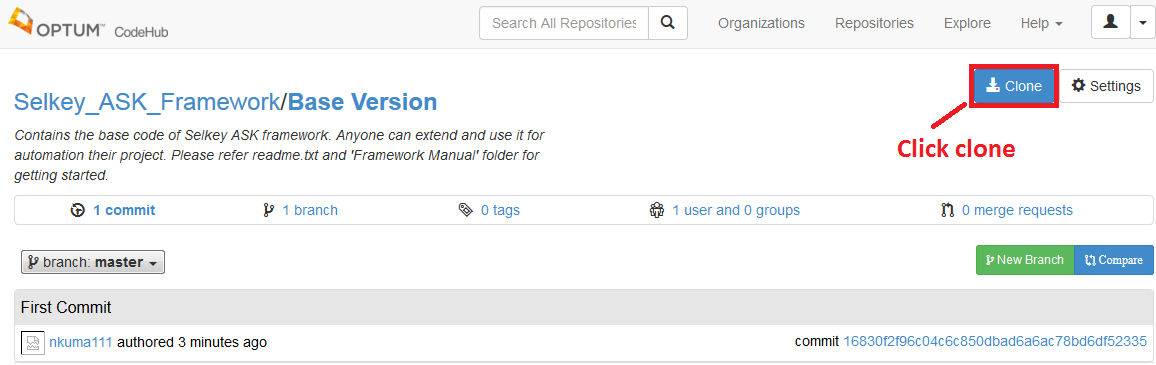
Drag and drop the install button to your eclipse to install maven plugin

**Check System Variables: *(Compulsory for all)***

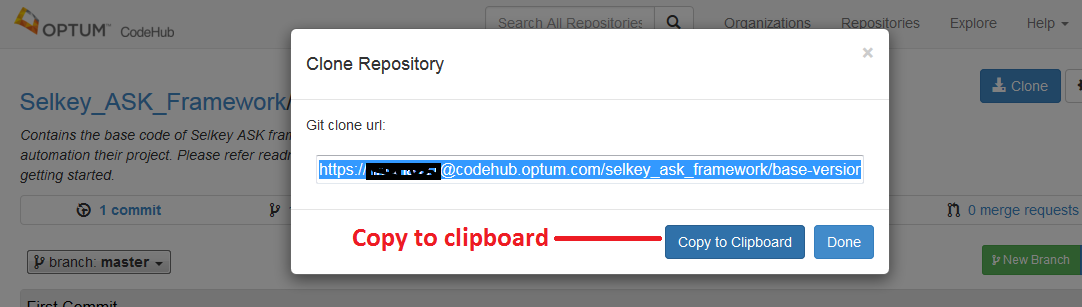
1. Go to Control Panel -> System
2. Click Advanced System Settings link
3. Click Environment Variables button
4. Under the System variables section make sure you have “JAVA\_HOME” variable set to “C:\Program Files\Java\jdk1.8.0\_51” (if not please set)
5. Under the System variable section make sure you have “Path” variable containing “%MAVEN\_3\_2\_3%\bin” (if not please append with a “;”).

**Get your ASK copy from Optum Codehub**

1. Now go to <https://codehub.optum.com/selkey_ask_framework/base-version/tree/master>
2. Click ‘Clone’ button

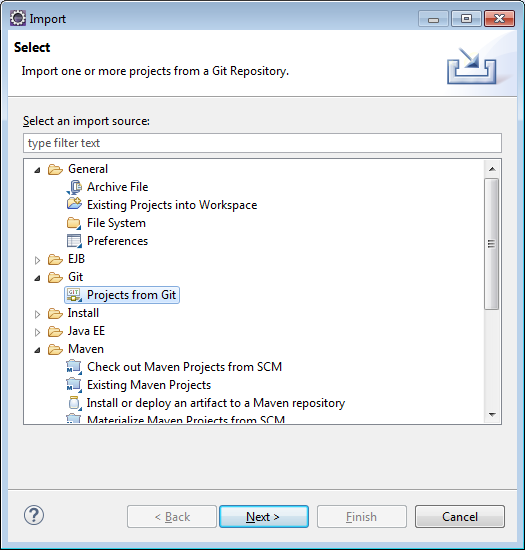


1. Copy git path to clipboard:

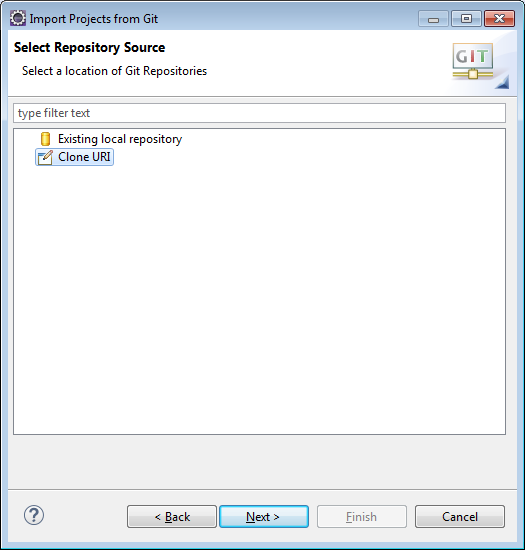


Let us say this copied link as **Link1**

1. In Eclipse go to File -> import

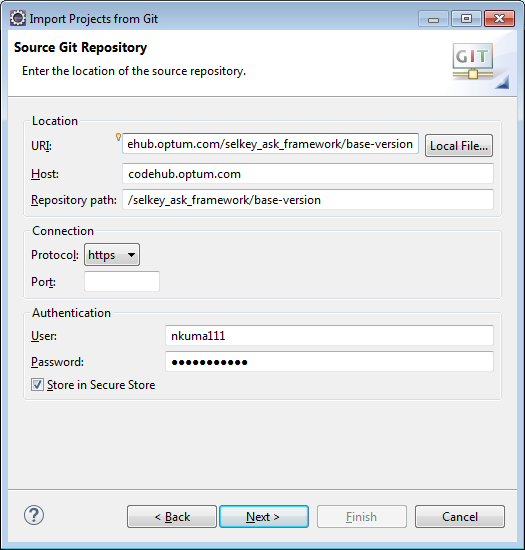


Select “Projects from Git” hit next



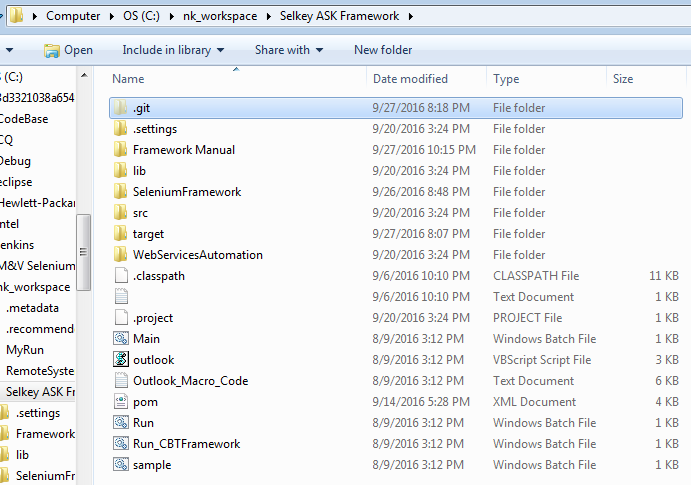
Select ‘Clone URI’ and hit next





Paste the url you copied from the codehub in the URI section and delete the postfix “.git” from it. Enter your username password for codehub and hit next button.

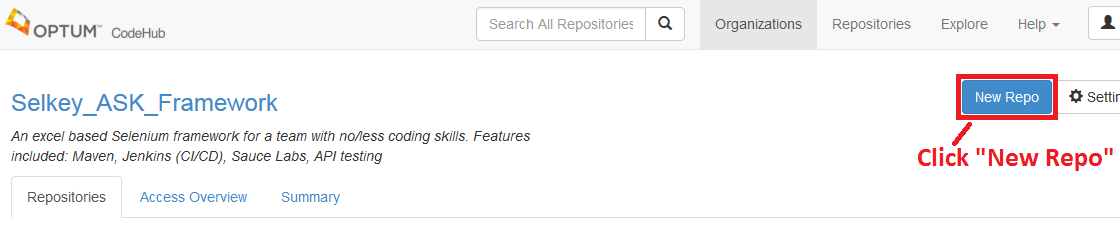
1. Choose the master branch and hit next and finish to download the project.
2. Now go to the actual location in windows folder where the project is installed.
3. Select the hidden folder ‘.git’ and delete the folder.

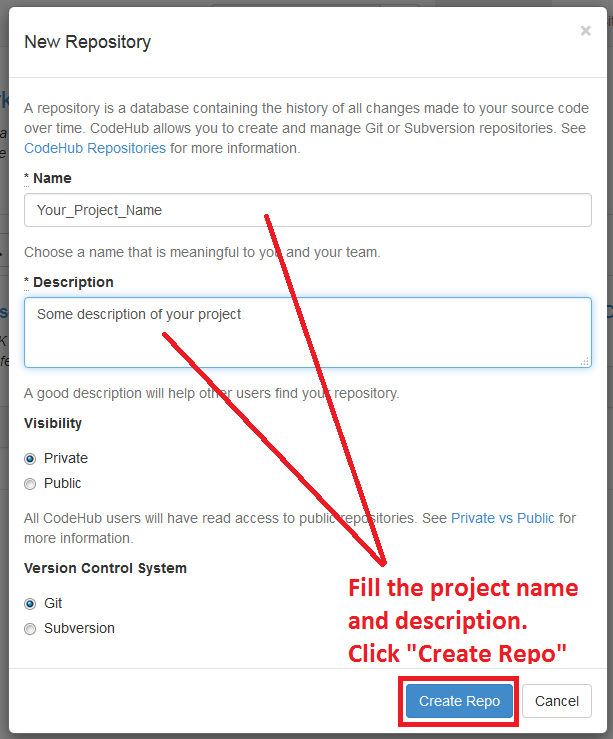


**Create your ASK Project on Codehub**

**Create a Repo on code hub first:**

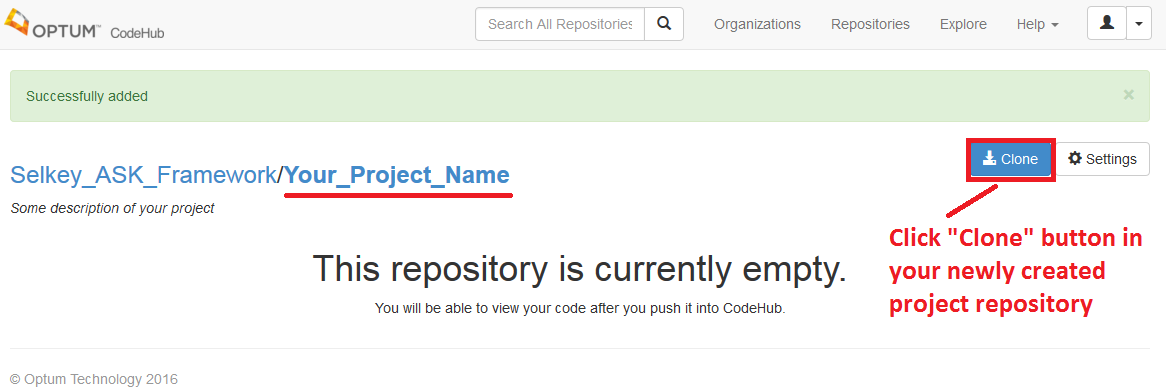
1. Login to Optum code hub and go to <https://codehub.optum.com/selkey_ask_framework>



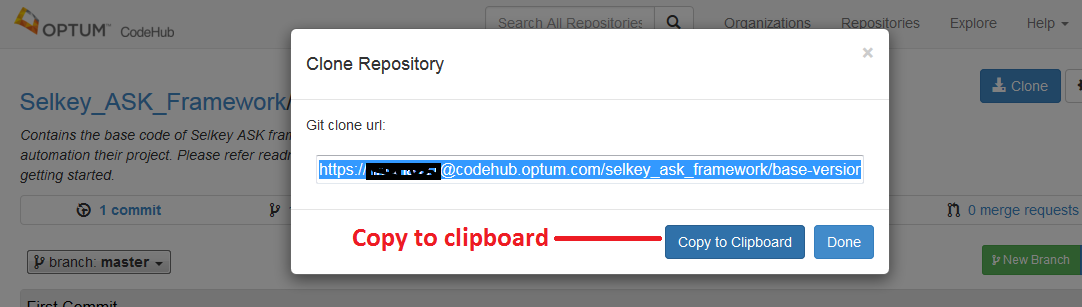


An overlay like above image will open. Fill in you project name and description, chose visibility as per your choice and click “Create Repo” button.

1. When done. Click “Clone” button in your newly created project repository.

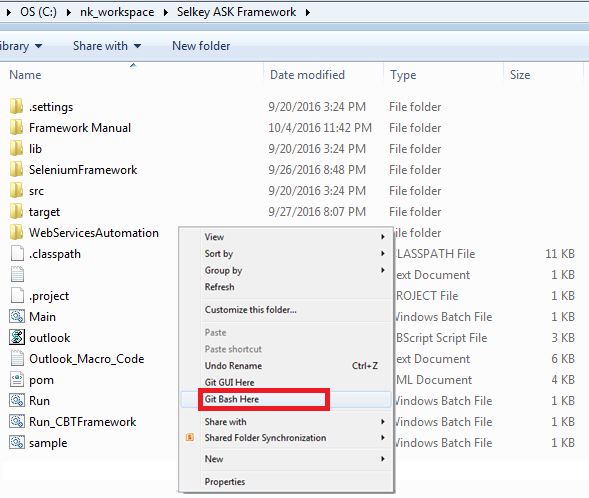


1. Copy git path to clipboard:

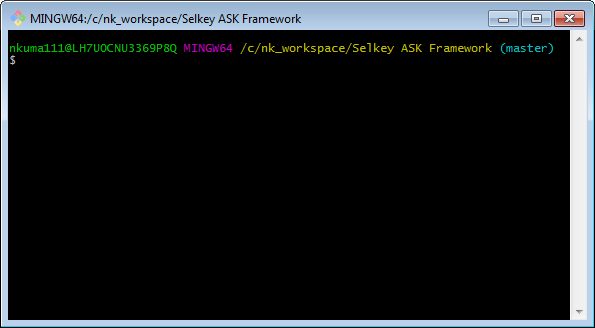


Let us say this copied link as **Link2**

**Upload your code to the repo just created:**

1. Go the folder location where you have the project code 

Right click and click “Git Bash Here” from the context menu.



A window like above will open to take your git commands.

1. Give following commands to upload your code on the code hub repository of your project.

Have **Link2** handy to use in the “git remote add origin” command:

git init

git add .

git add –A

git commit –m ‘First Commit’

git remote add origin [https://<YourNTID>@codehub.optum.com/selkey\_ask\_framework/<your\_project\_name>.git](https://%3cYourNTID%3e@codehub.optum.com/selkey_ask_framework/%3cyour_project_name%3e.git)

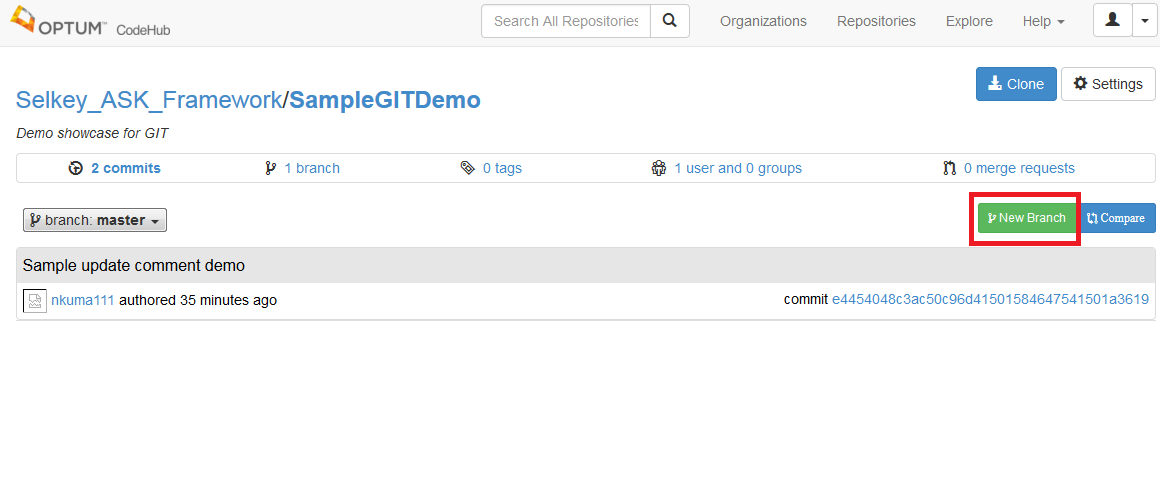
git push –u origin master

1. Now when you go to your project repository on code hub. You should see your project code uploaded.

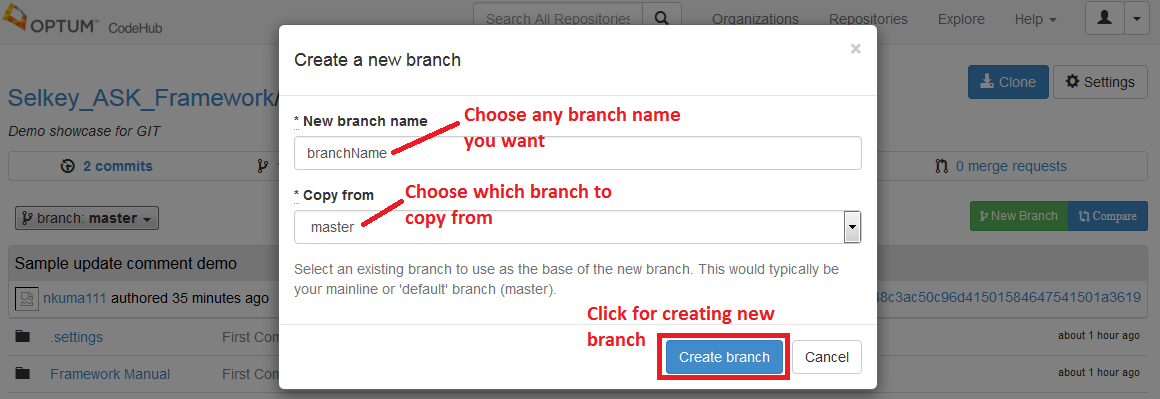
**Creating a branch for the repo created:**

By default a master branch is created. However just to save the originally of what we have in start we create other branches for the teams to work on and update at.

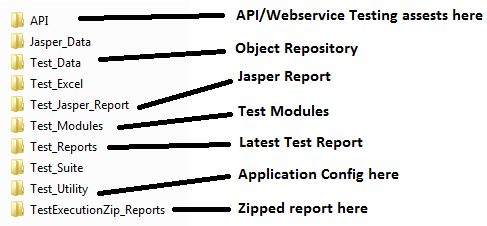
1. Click New Branch button in the repo you created:



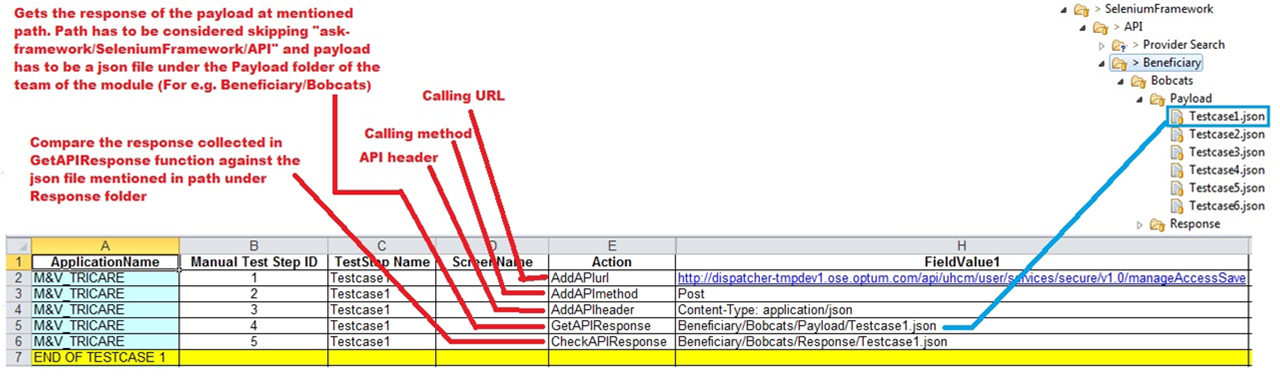
1. Enter branch name and copy from branch name and click Create branch button:



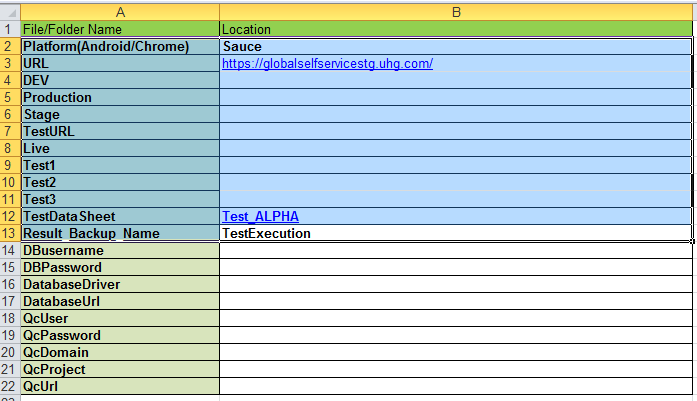
**Framework Folder Structure**:



**Working with Web service**:



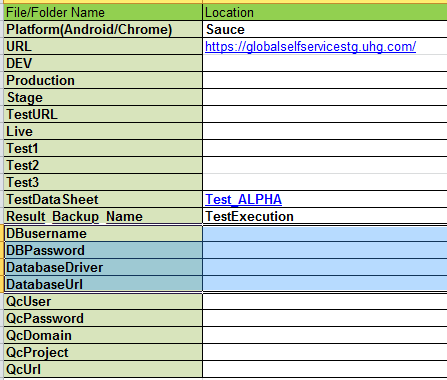
**Configure Execution in Application\_Config**

****

|  |  |
| --- | --- |
| **Field** | **Values** |
| Platform | Sauce/Firefor/InternetExplorer/Chrome  Choose any browers of your choice.  Choose it to Sauce if want a cloud run with Jenkins. |
| URL | Common url to launch |
| DEV | Environment specific url |
| Production | Environment specific url |
| TestDataSheet | TestData sheet location in Modules. By default set to "Test\_Alpha |

**Configure Database connection**

Step 1: Please provide information to the below fields in Application\_Config.xls file.



* DBusername - Database username for the specific database
* DBpassword - Database password for the specific database
* DatabaseDriver - Defaults to oracle.jdbc.driver.OracleDriver. Optional to change.
* DatabaseUrl - Database url to be used.

**For eg.: Jdbc: mysql://localhost:3306/sonoo** where

“jdbc” is the API,

“Mysql” is the database,

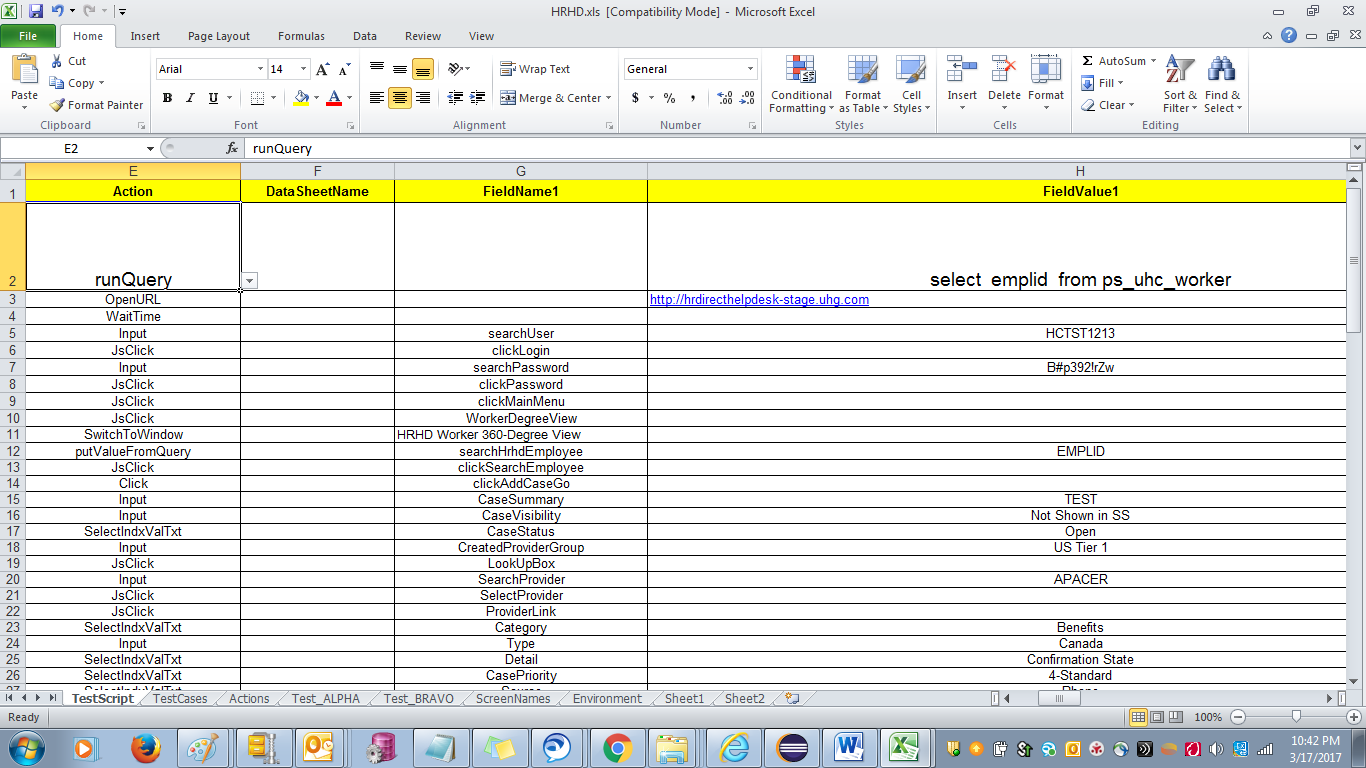
“Localhost” is the server name on which mysql is running

“3306” is the port number

“sonoo” is the database name

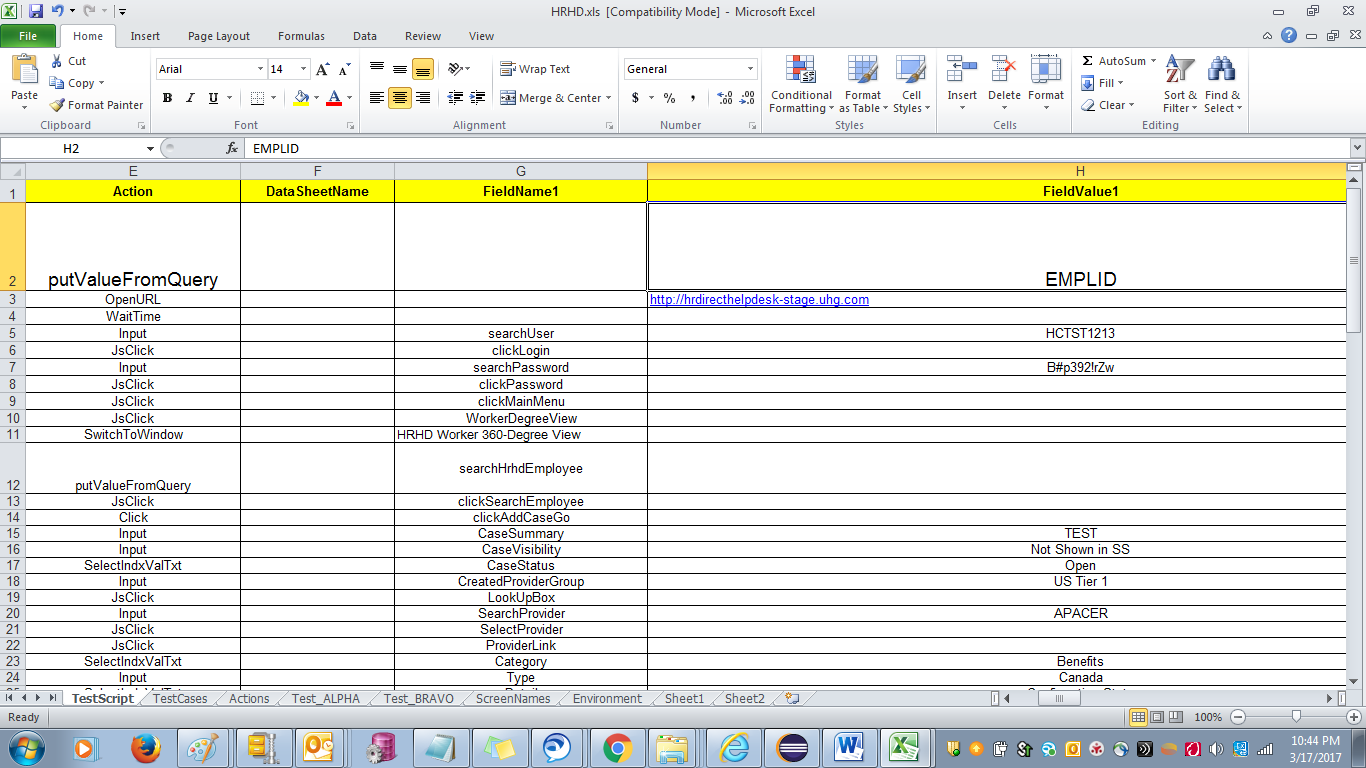
**Usage of Database keywords**

* Action Name: ***runQuery*** Field value: give **Sql query** to execute



Action “runQuery” functionality

* Opens the connection, runs the given query in the Field value and closes the connection.
* Stores the query results for only 1st row of the query.
* Action Name: **putValueFromQuery** Field Value: give the column name of the result record you got when you ran the latest query using “runQuery” action



Action “putValueFromQuery” functionality

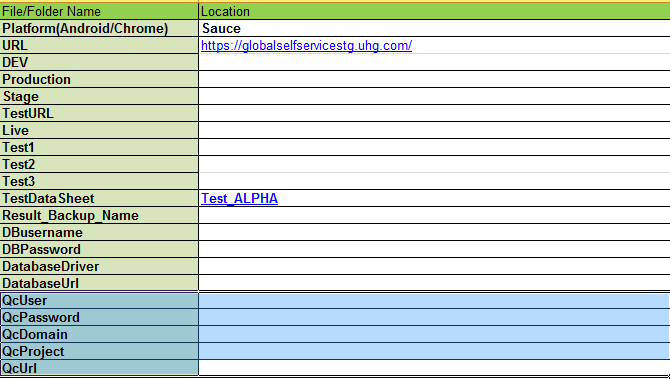
* This Action can be used only with input text box and text area as it is dependent on “runQuery” action
* It is used to enter one of the column value which you got as result of the query ran using “runQuery” action

**Configure ALM connection**:

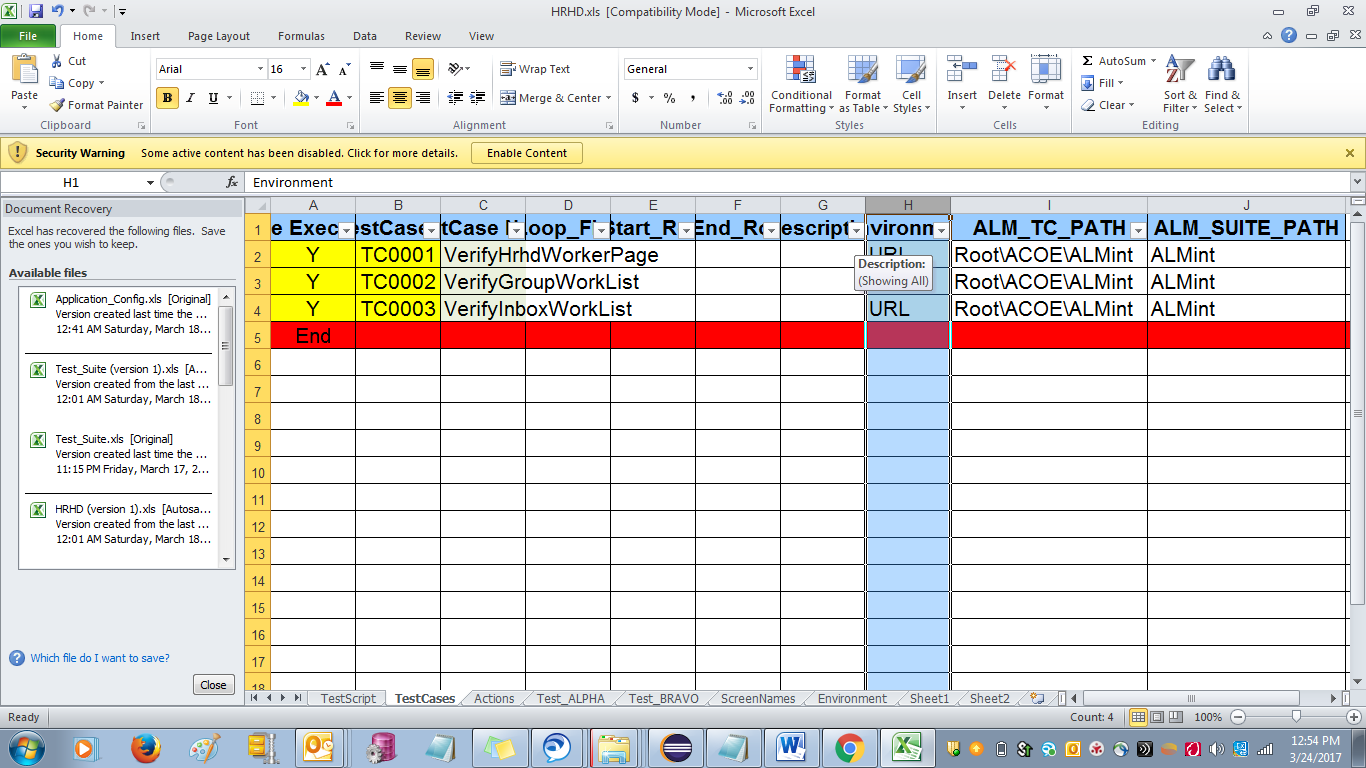
1. Update SeleniumFramework\Test\_Utility\Application\_Config.xls:

* Please fill information for fields
  + QcUser
  + QcPassword
  + QcDomain
  + QcProject
  + QcUrl

fields to connect to your ALM project.



1. Be sure to mention the path of ALM testcase and AML suite in the “Testcases” worksheet of test module workbook against the test cases expected to update result on ALM.



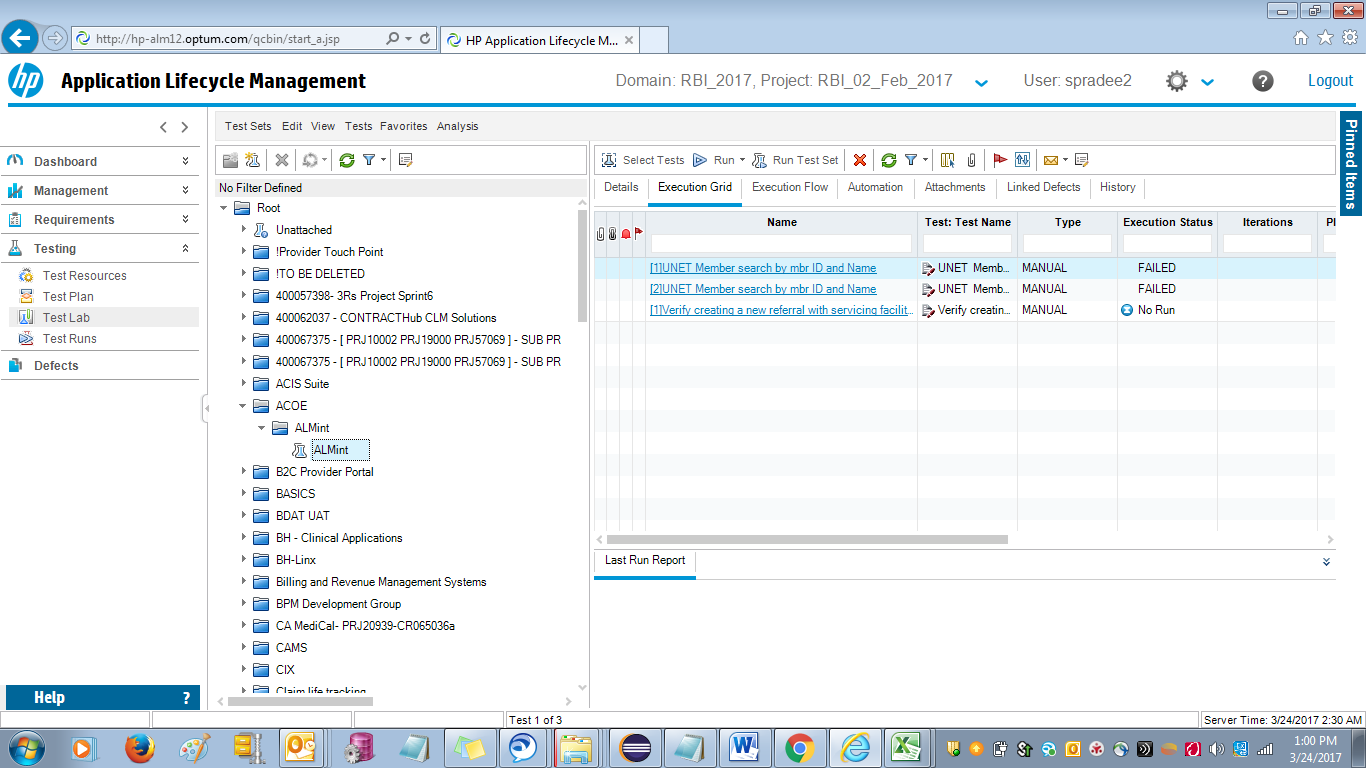
ALM\_SUITE\_PATH value

ALM\_TC\_PATH value

Test Case sheet

|  |  |
| --- | --- |
| **ALM\_TC\_PATH** | **ALM\_SUITE\_PATH** |
| Root\ACOE\ALMint | ALMint |

Also, the test case name has to be same as ALM to update the result of execution to ALM.



ALM\_SUITE\_PATH value

ALM\_TC\_PATH value

1. Run Dbconnection.bat for updating the results after running your test from run.bat file.

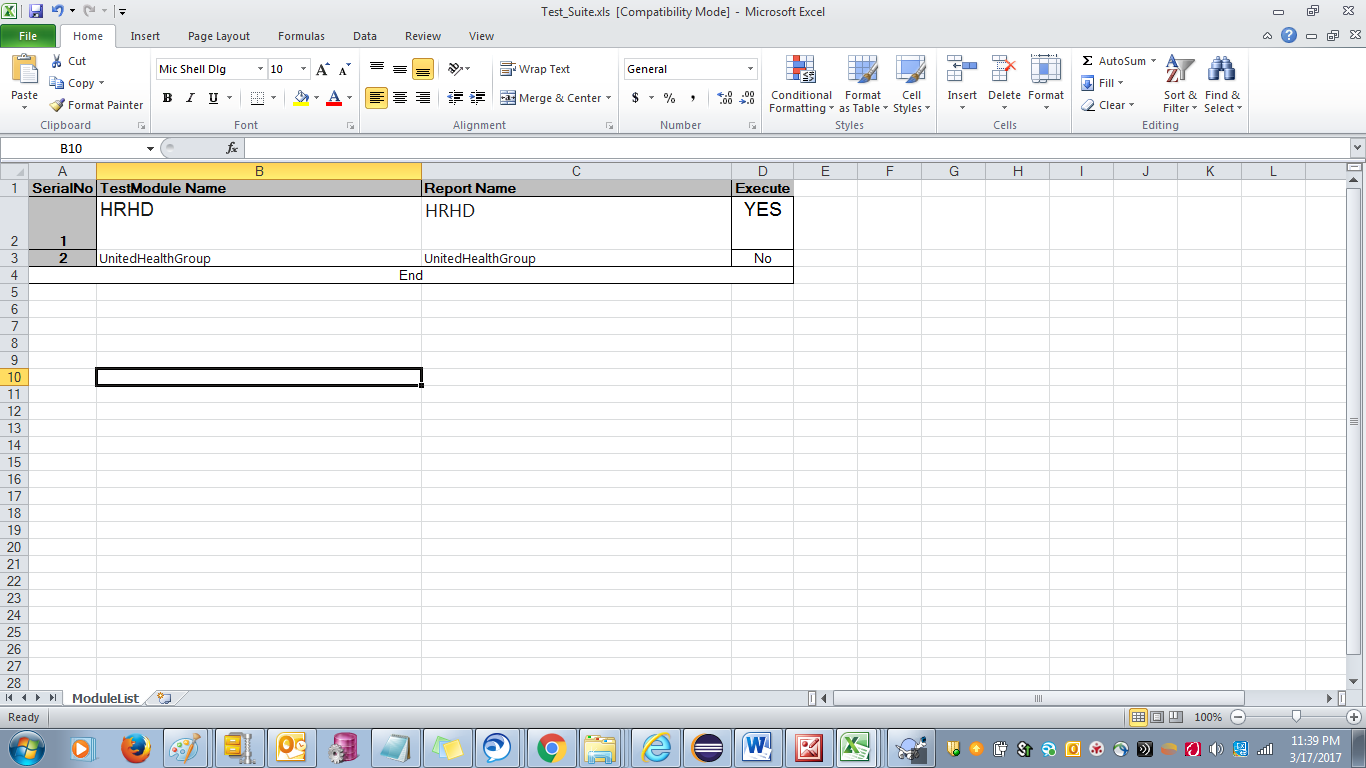
**Steps to start with automation**

The below 4 excel workbooks under “SeleniumFramework” folder are important to configure an automation script

* TestSuite
* Object\_Repository
* TestModules

**TestSuite**

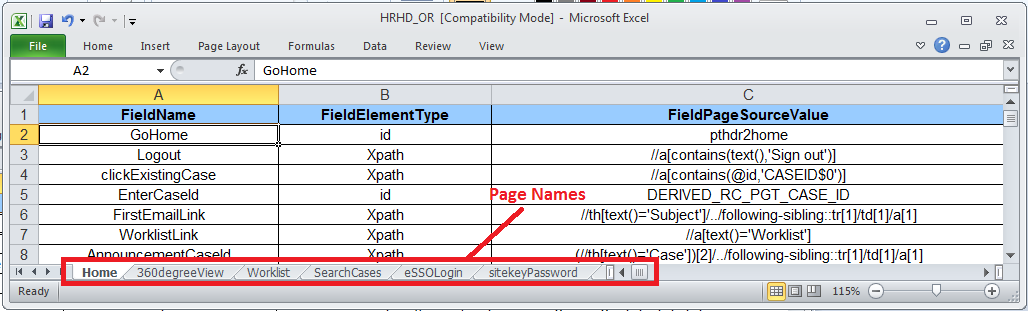
In order to run any application you have to provide the Test module name in SeleniumFramework\Test\_Suite\Test\_Suite.xlsx sheet. Set the modules to execute by mentioning “Yes” or “No”



**Object\_Repository**

In order to run any application you have to provide the locator with whom you would like to interaction.

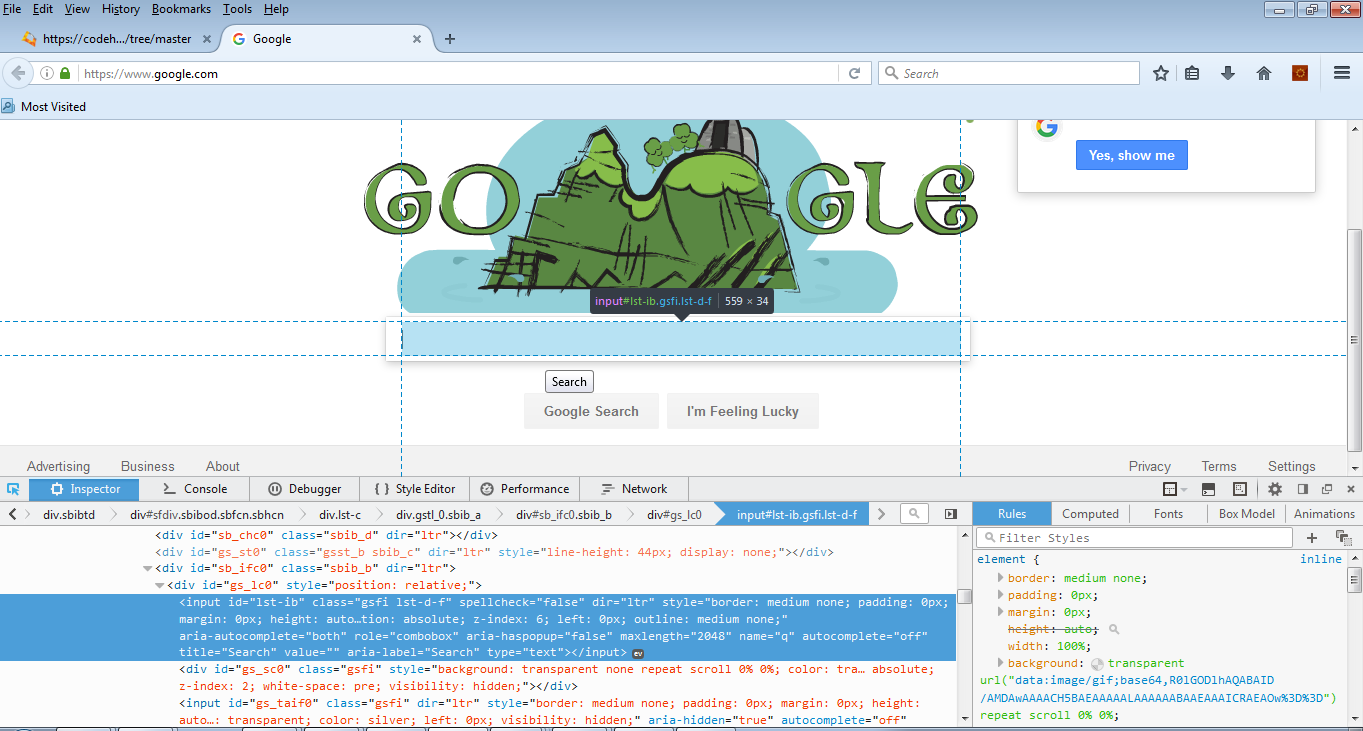
So an object repository is maintained in “SeleniumFramework\Object\_Repository” file. The file name will be “YourApplicationName\_OR.xls” where “YourApplicationName” is what you give in “Application” column of your test script.



|  |  |
| --- | --- |
| **Field** | **Values** |
| PageName | Worksheet name, any name you want to give a page. The reference will be taken in “ScreenName” field in the TestScript sheet of Test Module. |
| FieldName | Any name you want to give to that element on that page |
| FieldElement Type | Id/Name/Xpath/DynamicXpath |
| FieldPageSourceValue | Value of "FieldElementType" you mentioned for that webelement |

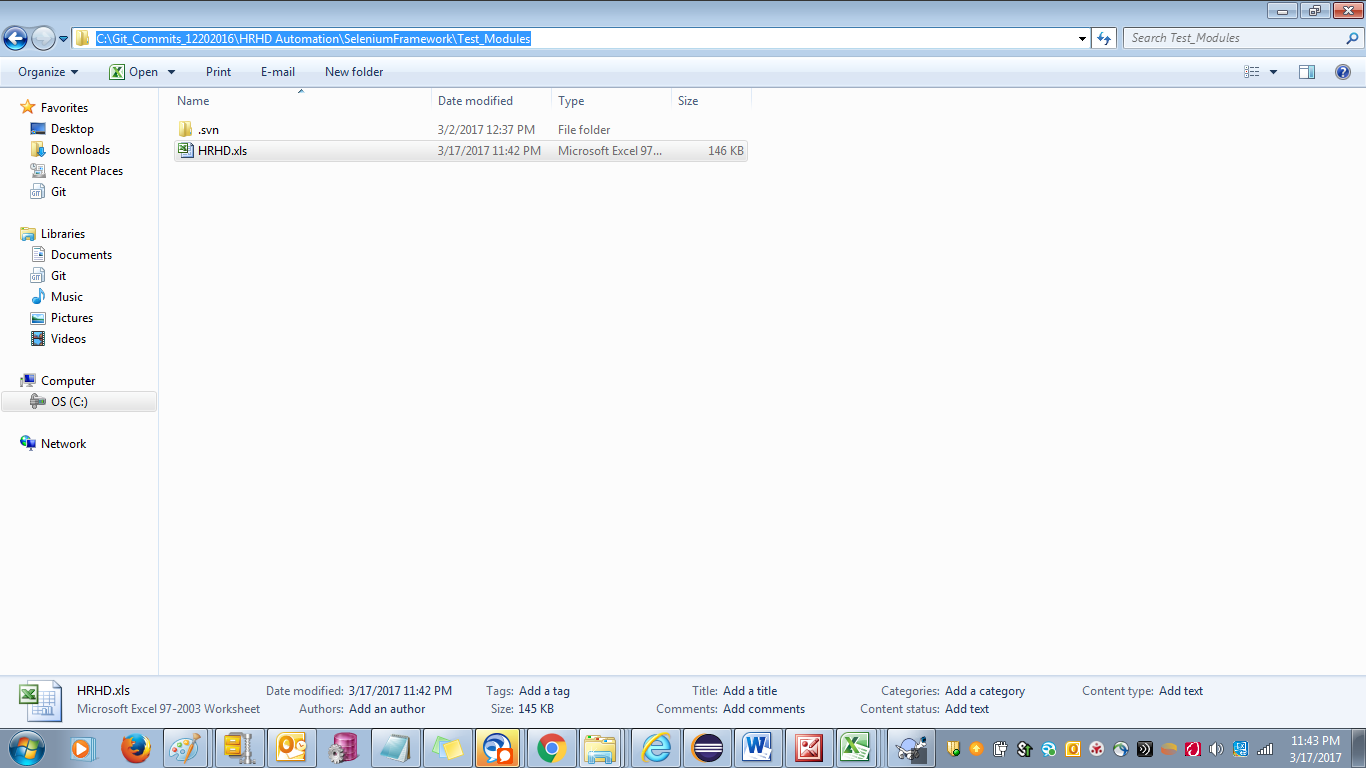
**How to take out locator from a webpage on firefox browser:**

Use F12 key and inspect functionality on any brower to get FieldElementType and FieldPageSourceValue

****

**TestModules:**

After giving the Test Module name, create an Excel file at SeleniumFramework\Test\_Modules with the same name mentioned in Test Suite. Use any excel sheet already available in module folder when you downloaded as template.



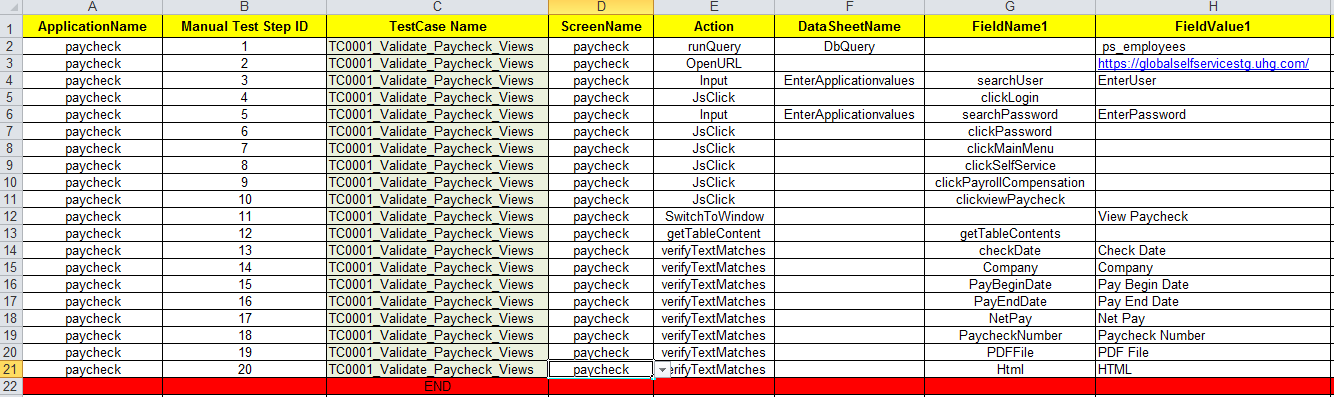
**TestCases worksheet fields:**

****

|  |  |
| --- | --- |
| **Fields** | **Value Suggestions** |
| TestCase Execution Flag | Yes/No |
| TestCase\_ID | Any |
| TestCase Name | Any/Same as Testcase name in ALM |
| Loop\_Flag | Optional. Set it to "Loop" if want to iterate more than 1 time. Iteration count will depend on the records in Datasheet you are using |
| Start\_Row | Starting row number of testcase in TestScript sheet |
| End\_Row | End row number of the testcase in TestScript sheet |
| Description | Any |
| Environment | Choose from the Application sheet |
| ALM\_TC\_PATH | Test case path in ALM |
| ALM\_SUITE\_PATH | Test suite path in ALM |

**TestScript Worksheet fields:**

Test Script sheet is used to write the test steps for your test cases provided in the Test cases sheet



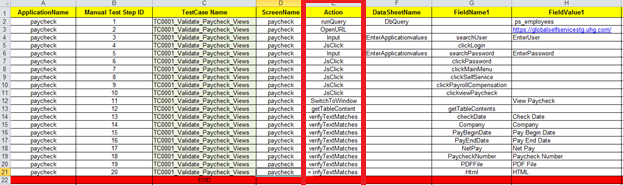
|  |  |
| --- | --- |
| **Field** | **Value suggestion** |
| ApplicationName | Any. Keep same in a testcase |
| Manual Test Step ID | Step numbering. Start from 1 for step 1 of a test case |
| TestCase Name | Test case name, same as mentioned in TestCases worksheet |
| ScreenName | Page name, Refer the SeleniumFramework\Test\_Data\Element\_Collection.xls sheet to get correct screenname |
| Action | Interactions like Click/Input/Verify. Choose from the dropdown |
| DataSheetName | Empty or Some name from Column A of the Test\_Alpha sheet if want to use DataSheetName |
| FieldName1 | Page element, Refer the SeleniumFramework\Test\_Data\Element\_Collection.xls sheet. |
| FieldValue1 | Field of the datasheet mentioned in DataSheetName. Or mention a direct field if DataSheetName field is emply |

***Always end a testcase step with the word “END” as last row in Column A.***

***Note:***

Screen Name + Field Name are always to be taken from “ElementCollection.xlsx” for interaction

**ACTIONS:**



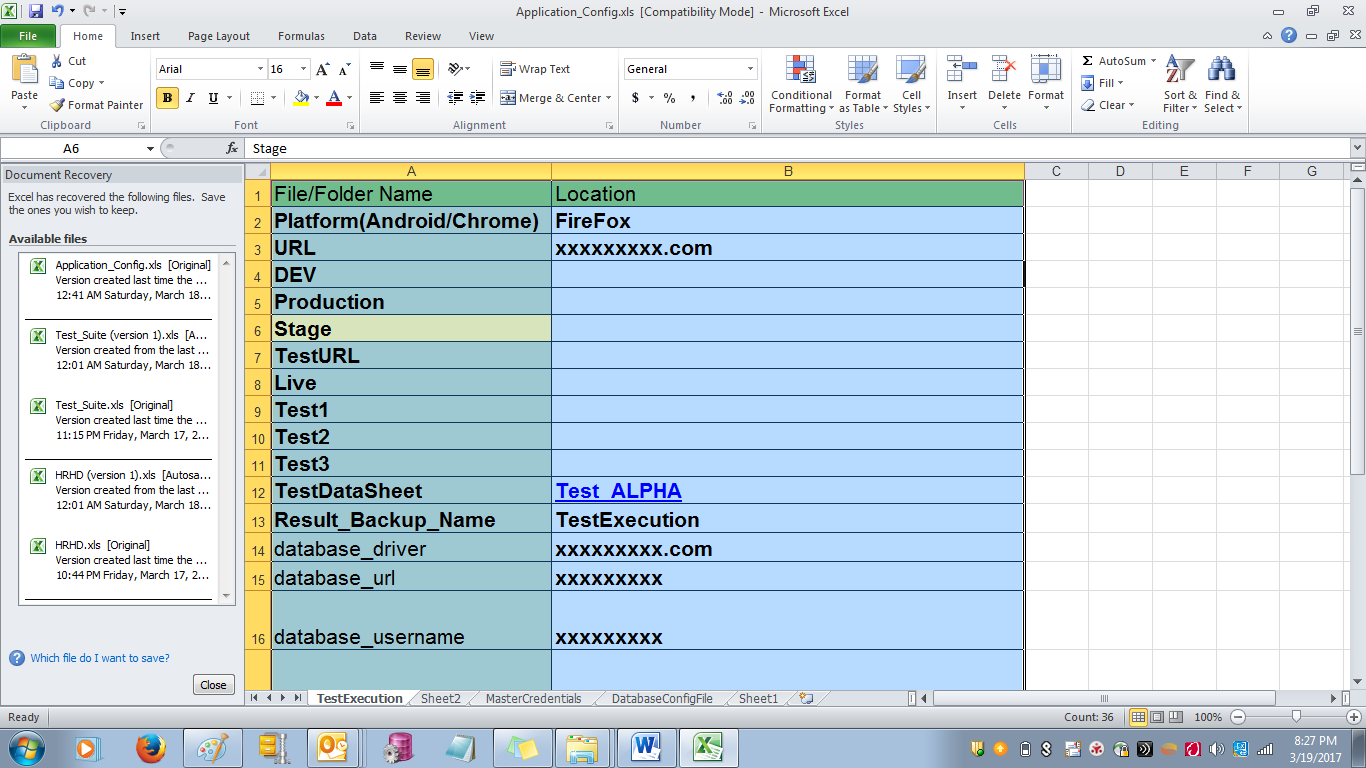
Below is the description of actions available to use in Test Script Sheet:

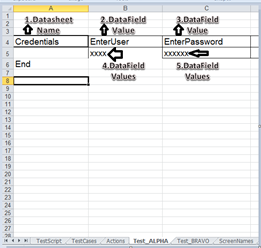
|  |  |
| --- | --- |
| **OpenURL** | Open the given url |
| **Input** | Type the given value in textbox |
| **Click** | Click the located webelement |
| **CheckByIndex** | Use this when you have multiple elements located with same locator and you want some specific element to be click and you can give positon number. Pass the index/position number of the webelement in the FieldValue1 starting from 0 for the 1st webelement |
| **SetCheckBox** | Set a check box to checked/uncheck state as per what you pass in FieldValue1. |
| For e.g.: Pass "check" if you want the checkbox checked else pass "uncheck" |
| **ClearAndType** | Cleans the textbox and then type the pass value from FieldValue1 |
| **Clear** | Cleans the textbox |
| **WaitTime** | specify the no of milliseconds to wait in FieldValue1 For Ex- 1000ms = 1 sec |
| **VerifyElementExists** | Check if the webelement exists |
| **VerifyDynamicElementExists** | Check if the webelement exists replacing the word "VARIABLE" in xpath with the given value at runtime |
| **VerifyElementByValue** | Verify the webelement's attribute "value" with the value passed in FieldValue1 field of TestScript |
| **VerifyElementProperty** | Verifies the attribute of a given element to a Value/storedVariable  Example for FieldValue1: attribute#directCompareValue or attribute#storeVariableName  **Note:** If using storedVariable, you may need to use HoldElementText or getEleProperty functions first. |
| **VerifyURL** | Verify the url of the page with the value passed in FieldValue1 field of TestScript |
| **VerifyTextPresent** | Verify is text is present anywhere on page, no webelement need to be referred |
| **VerifyLink** | Verify the link/webelement is displayed |
| **VerifyMultiLinks** | Verify the count of the links matched with the value passed in FieldValue1 field of TestScript |
| **VerifyFalseEleExist** | Verify webelement don't exists on page |
| **VerifyAlertText** | Verify the alert text matches with the value passed in FieldValue1 field of TestScript |
| **SelectIndxValTxt** | Selects a dropdown to the given text passed in the value passed in FieldValue1 field of TestScript To select by index pass index#[indexNumber] in the value passed in FieldValue1 field of TestScript To select by value attribute pass value#[Value] in the value passed in FieldValue1 field of TestScript |
| **HoldelementText** | Stores the text of the given webelement including textboxes or dropdown to the  a runtime variable named in FieldValue1 |
| **SendelementValue** | Types value of the variable stored in "HoldelementValue" action in textbox webelement. The variable name need to be given in the value passed in FieldValue1 field of TestScript |
| **Geteleproperty** | |  | | --- | | Geteleproperty Same as HoldelementValue. Here you are not restricted to store only webelement text like HoldelementText and can store any attribute's value to the variable named mentioned in fieldValue1 Tips:  1. For storing text of webelement pass "innertext". 2. Pass the attributeName#variableNameToStore in fieldValue1 to store the element property For Ex- id#main where id is attribute and main is variable which stores it's value. 3. Miscellenous attributes supported bg-color - Gives back ground color of webelement width- Gives the width of webelement height-Gives the width of webelement | |
| **Asserteleproperty** | Compares the text value of a textbox or any webelement to the valued stored using Getelepropery. NOTE: Pass attribute "text" as value in FieldValue1 field of TestScript |
| **ClosewindowByTitle** | Closed the window have the title mentioned as value in FieldValue1 field of TestScript |
| **CloseBrowser** | Closes the latest current browser window |
| **SwitchToWindow** | Switch to the window whose title is passed in FieldValue1 |
| **SwitchToframe** | Switch to the frame whose id is passed in FieldValue1 |
| **VerifywindowTitle** | Verify the current window title to the one passed in FieldValue1 |
| **Actionclick** | Click action alternative |
| **MouseHoverclick** | Click action alternative |
| **MouseHoverJs** | Hover mouse to the webelement |
| **TypeRandomNbr** | Types random numbers to the textbox. Range need to passed as [lowerLimit, upperLimit] in the FieldValue1. E.g. 0, 99: will generate random number between 0 - 99 |
| **HandleAlert** | Accept an alert |
| **KeyEvent** | Alias the keyboard typing. Takes argument from the FieldValue1. Support keyevents: CTRL+END, CTRL+HOME, ENTER, TAB, CTRLDELETE, ARROWDOWN |
| **GetPreviousDate** | Enters 18 months prior date in the asked webelement |
| **NavigateBack** | Performs navigating back in browser |
| **JscriptExecutor** | Execute the passed JavaScript in FieldValue1 |
| **VerifyTableRowCount** | Verify the table webelement's row count to the value passed in FieldValue1 |
| **MouseHover** | Hover mouse to the webelement |
| **JsClick** | Click action alternative |
| **UncheckAllSelectbyIndex** | Will click all webelements you are able to locate. Can be used to check/uncheck list of checkboxes located by same xpath |
| **OptionalClick** | Clicks a webelement if it is present. Throws no error if the webelement don't exists |
| **ActionInputEnter** | Types the value passed from FieldValue1 and hit enter to the same textbox |
| **runQuery** | Runs the query passed from FieldValue1. Stores 1st record for later use. |
| **putValueFromQuery** | Pass the Fieldvalue1 as Query Name#Column Name which Runs the query and stores the Value from the given Column Name further used to  send the value to InputTextBox of Web Element given in FieldName1 |
| **switchToDefaultContent** | It comes out of the previous frame and switches to default content pass the previous step field name as input |
| verifyTextMatches | Verify the value matches “FieldValue1” with the text of Webelement in “Field Name1”. |
| checkElementCountEquals | Verify the row count of the Web table given in “Field Name1”. If count is >0 results as pass else it is vice versa |
| **AddAPIurl** | Use this for webservice testing. Store the url you want to hit for webservice. Pass using FieldValue1 |
|  |  |
| **AddAPImethod** | Use this for webservice testing. Store the method for the webservice. Pass values as either get or post in FieldValue1 |
| **AddAPIheader** | Use this for webservice testing. Stores the header for the webservice. Use multiple times if have more headers. Pass values as either get or post in FieldValue1. For e.g.: Content-Type: application/json |
| **GetAPIResponse** | Use this for webservice testing. Store the webservice. Pass the payload in json format and give location in FieldValue1. Store the json file in API location and mention the path ahead of API folder location |
| **CheckAPIResponse** | Use this for webservice testing. Compare the response got from the action GetAPIResponse. Pass the expected in json format and give location in FieldValue1. Store the json file in API location and mention the path ahead of API folder location |
| ***verifyDBtextMatches*** | Pass the Fieldvalue1 as Query Name#Column Name which Runs the query and stores the Value from the given Column Name further used to verify the Query output with the Text of Web Element |
| ***verifyMatchesText*** | Verify the value/storedVariable given in “FieldValue1” matches the text in webelement  mentioned in “Field Name”. Note: If using storedVariable, you may need to use HoldElementText or  getEleProperty functions first. |
| ***UploadUsingRobot*** | Mention the file name in FieldValue1, and the Webelement value in FieldName1 to be clicked  Note: It's Mandatory to Place the file to be uploaded in SeleniumFramework\Upload folder |
| ***verifyDBtextSmallerThan*** | Pass the FieldValue1 as Query Name#Column Name which Runs the query and stores the Value for the given Column Name further used to verify the DB text is Smaller than the Text of Web Element |
| ***verifyDBtextGreaterThan*** | Pass the FieldValue1 as Query Name#Column Name which Runs the query and stores the Value for the given Column Name further used to verify the DB text is Greater than the Text of Web Element |
| ***verifytextSmallerThan*** | Verify the Given FieldValue1 is Smaller than the Text of Web Element |
| **NavigateForward** | Performs navigating Forward in browser |
| **NavigateTo** | Performs navigation to the given url in the FieldValue1 |
| ***verifytextGreaterThan*** | Verify the Given FieldValue1 is Greater than the Text of Web Element |

***Test Data sheet***

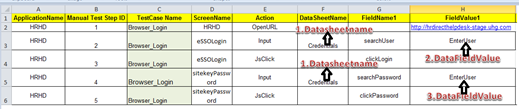
In order to keep data in an excel sheet for achieving data driven approach, we can use another Test data sheet. The Test data sheet name should be provided in the 12th row, Column B in Application\_Config.xlsx.

For Example: Test\_ALPHA is the default worksheet to be used in a Test Module to mention data.



**Screenshot of the Test\_Alpha worksheet in a module workbook:**

**Note:** The TestDatasheet name (E.g.-Test\_Alpha) in Application\_Config.xlsx should be created with the same name (sheet name) in your Application.xlsx file in Test\_Modules folder.



**How to link the data in “TestScript” worksheet to “Test\_ALPHA” worksheet of a test module**

1. For a specific “Action” which is used to provide our required test data, there should be provided with a convenient “DataSheetName”. For Ex- “EnterApplicationValues” is our DataSheetName.
2. The same Name should be provided in the Test\_ALPHA sheet “A” column as a first value.

Note: Please refer with Arrows “1”.

1. In “FieldValue1” column for specific Action, we will provide the specific name which conveys which “data” is being used. For Example, Application is sending the “username” to input box, then it can be “EnterUser”, If It is password which is being sent to the Input box, it could be “EnterPassword”.
2. The ““FieldValue1” column values like “EnterUser” and “EnterPassword” should be started from the “B” Column in the sequence. Please refer the Arrows “2” and “3”.
3. Now the Real Test data which are being used using should be provided below the “EnterUser” and “EnterPassword”. Please refer the Arrows “4” and “5”.

**Running the automation**

Step 1: Run the “run.bat” at root location of framework.