

**Java Selenium**

**Automation Framework**

For

All Test Automation Projects

Version 0.1

Last Updated **1/30/2017**

**Prepared by Mark Elking**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author(s)** | **Description of Change** | **Approved By** |
| 1.0 | 12/29/2016 | Mark Elking | Initial Draft |  |
| 2.0 |  |  |  |  |
| 3.0 |  |  |  |  |

Revision History

Table of Contents

[Introduction 5](#_Toc473557878)

[1.0 Input Spreadsheets 5](#_Toc473557879)

[2.0 Java Functions 11](#_Toc473557880)

[3.0 Java Development Standards 13](#_Toc473557881)

Document Glossary of Terms

|  |  |
| --- | --- |
| **Term** | **Description** |
| Example | Brief description or definition |
|  |  |
|  |  |

# Introduction

**This document outlines the detailed setup for automating a new GUI application using the Java Selenium Automation framework established for the TMO Automation Team at Charter.**



# Input Spreadsheets

One spreadsheet needs to be created for managing test case data input and test configuration and object definition parameters. The file format must be .XLSM and must be located under directory…..{Application source directory}\maps\{Application}ConfigurationMap.xlsm. Below is the list of mandatory worksheets in this spreadsheet.

* **WORKSHEET ALM** – contains all of the ALM configuration parameters for the given application
* **WORKSHEET BrowserType** – contains a list of the browser types supported for the given application
* **WORKSHEET Environment** – contains list of all environments available for automation of that application
* **WORKSHEET ExecParms** – this worksheet has a button that when clicked will read all of the worksheets in the spreadsheet and update the list of SheetName column values with all worksheets in the spreadsheet. Any comments associated with a given TC will be preserved and re-associated. BrowserType dropdown values pull from BrowserType worksheet. When a BrowserType is selected it will automatically update all TCTestData values for all corresponding LaunchBrowser events across all TC prefixed worksheets. Environment dropdown values pull from Environment worksheet. When an Environment is selected it will automatically update all TCTestData values for all corresponding NavigateToURL events across all TC prefixed worksheets. Each TC sheetname will have a corresponding Value column dropdown value of Y or N. If Y is selected, then that test will be executed when Java main function is executed. If N is selected, that test will not be executed. You can have multiple test cases with the value Y and the Java main function will run those tests in the order listed from top to bottom on the ExecParms worksheet.
* **WORKSHEET EventNames** – list of TC events available. This is a fixed list and corresponds to a function call in the Framework fw\_event function.
* **WORKSHEET Object** - must have 3 columns named,
  + **COLUMN ObjectName** – the format of the value should be PAGENAME\_FIELDNAME, example is SEARCH\_Address1 where SEARCH is the page name and Address1 is the field label.
  + **COLUMN TagName,Attribute,Value** – the value should be tagname + “,” + attribute + “,” + value. Examples include:
    - Button,NA,Search
    - NA,id,addressLine1
    - input,xpath,//\*[@id=”container”]/button
  + **COLUMN ExtraInfo** – any other information that you need to have associated per field can be used in the ExtraInfo column (if needed)
* **WORKSHEET TC{ALMTestID}** – There must be N worksheets named “TC{ALMTestID}”, this is case-sensitive. If you are automating 20 test cases, then you will need 20 corresponding worksheets. Examples include: “TC27473”, “TC27474”. TC prefixed worksheets must have 5 columns named, All of the column names are case-sensitive.
* **COLUMN TCObjectName** – which is a dropdown data validation list linked to ObjectName named range on Object worksheet.
* **COLUMN TCTestData** – free form text field. If EnterDataTextbox event, then TCTestData should have some text value. NOTE: all text is automatically Cleared before entering the data into the text field. If you do NOT want to CLEAR the textbox value before entering a value into the textbox, then you would put in “NOCLEAR,3”.
  + NOTE: if the value of TCEventName is Component, then TCObjectName can be NA and TCTestData value must be the {PageName}
* **COLUMN ObjectToLookForAfterObjectEvent** – if you want to control the rate at which the script is executing by interrogating the page document AFTER you click a button (for example), then you can search for a given text on the page document until it’s found or until the loop counter/time you configure is met. Format of this value is “NA” or the following……..
  + tagname + “,” + text to search for + “,” + number of loops to keep looking + “,” + milliseconds to wait per loop

Examples include:

* + - h4,Search,15,1000
    - span,Customize Offers,30,1000
    - Default value is NA

An example is “label,Order Reasons(s),15,1000”. So if I click a button and this column value is NOT NA and it has value of “label,Order Reasons(s),15,1000” then the code will look for the Order Reasons(s) text in the label tagname on the page for a total of 15 seconds, checking every 1 second 15 times. If the text is found at 3 seconds, then it will not continue to look for the additional 12 seconds. Inside control is passed on to the next event in the test case.

* + **COLUMN MillisecondsToWaitAfterObjectEvent** – default value is 0. If you want the script to wait for some hard coded time interval after the event. NOTE: regardless of whether or not there is an ObjectToLookForAfterObjectEvent specified, this MillisecondsToWaitAfterObjectEvent will occur. The ObjectToLookForAfterObjectEvent (if not NA) will occur first, then the MillisecondsToWaitAfterObjectEvent will occur next.
  + **COLUMN TCEventName** – which is a dropdown data validation list linked to EventNames worksheet.
    - LaunchBrowser
    - NavigateToURL
    - CheckForElementExistence – in TCTestData make sure to provide value something like (10,1000) where 10 is number of loops to check for element and 1000 is the number of milliseconds to wait per loop.
    - ClickButton
    - ClickJAVASCRIPT
    - EnterDataTextbox
    - SelectListValueByValue
    - SelectListValueByVisibleText
    - SelectCheckbox
    - GetText
    - ValidateText – this event will call the fw\_validate\_text function. To implement you could do the following in Main inside of Component as an example…..see highlighted in Yellow.
    - TerminateWindowProcesses
    - Component
    - WriteLogHeader – in TCTestData indicate the header block name that you want to appear in your output log file.
    - GetAttribute – in TCTestData indicate the attribute name you want to get value of like (img or data-reactid, etc).
    - SwitchToDriver – in TCTestData indicate the browser type of IE or CHROME which calls fw\_switch\_to\_driver function.
    - AcceptAlert – used to accept an alert.
    - StopExecution
    - NA
    - SetVariable – in TCTestData put name of the variable (case-sensitive) followed by a comma follow by value you want for this variable. Here are some examples:

Sysprin,82451200

Address1,502 Jingle Bells Lane

NOTE: A file named “Address1” will be created in your “variables” folder with the variable value “502 Jingle Bells Lane”. You must have a variables folder defined in your workspace.

NOTE: the TCObjectName should be “NA”.

* + - GetCurrentDate – TCObjectName value is NA. TCTestData value is “{variable name},{format of date}”. Here is an example: “CurrentDate,yyyy-MM-dd”
    - GetFutureDate - TCObjectName value is NA. TCTestData value is “{variable name},{format of date},{number of days in future}”. Here is an example: “FutureDate,yyyy-MM-dd,10”.
    - XMLExecute
      * Create an Object on the Object worksheet entitled “WEBSERVICE\_{webservicename}. Example is “WEBSERVICE\_addHouse”. Make sure TCObject definition is “NA,NA,NA”
      * Create a template xml file located under webservices\templates directory. Substitute any hard coded values in that template request xml file with variables prefixed with XML. For example, “XML\_ACCOUNT\_NUMBER”. Should be all caps.
      * On the Environment worksheet, add 2 new columns for each new webservice. The names of the columns should be {webservicename}ENDPOINT and {webservicename}CREDS.

Here are examples….

addOrderENDPOINT

addOrderCREDS

For the corresponding environment for which you plan to execute this webservice, provide the ENDPOINT and CREDS information.

Examples of each of those are as follows….

ENDPOINT example

<https://ebs-uat.corp.charterom.com/csg_cter/2.06/OrderDetailService.asmx>

CREDS example

chtr\svc\_tst\_automation:H2i1fL9!

* + - * Values can be passed into an XML template in 2 ways, 1-hard coded in spreadsheet OR 2-by referencing a file which has a value in there for substitution.
      * On TC prefixed worksheet…..
        + Select TCEventName “XMLExecute”
        + Select TCObjectName value from list created in previous step
        + Enter values to substitute in TCTestData column in following format

XML\_LOCATION\_ID,12345 (hard coded example)

XML\_LOCATION\_ID,FILE\_LocationID (dynamic example where LocationID is the name of the file in the variables directory which contains value of LocationID. To reference the file, make sure the word file is capitalized and has an underscore.

* + - * + If you want the webservice to keep executing until a certain value in the response is found, then update the ObjectToLookForAfterObjectEvent column with the following value

{text to look for in the XML response}--{number of loops to check}--{milliseconds to wait per loop}

Here is an example….

ReciveFromActivation--3--5000

If you have dynamic data to pass, then do something like the following……

ReciveFromActivation<Account>,FILE\_AccountId,</Account>--3--5000

…..where the text highlighted in Yellow is literal text. NOTE: the commas are NOT literal text to be validated but they separate the literal text from the dynamic text. Dynamic text comes from the variable file referenced. So in this example, if account ID value 12345 was in the file “AccountId”, then the text that will be validated in the XML response file will be the following…..

ReciveFromActivation<Account>12345</Account>

And the check will occur every 5 milliseconds for 3 loops. So total of 15 seconds.

* + - XMLGetValueByTagName

This event gets a value for a specified tagname from the webservice response xml file. After the XMLExecute event is called, a webservice response file is created. This XMLGetValueByTagName event will retrieve the corresponding value out of the response file for the tagname that you specify in TCTestData. NOTE: do NOT include the “<” and the “>” values in the TCTestData. If a tagname is <address> in the response xml file. Then in TCTestData, you would indicate “address” (no < or > needs to be specified).

* + - XMLValidateTextinXMLResponse

This event will validate text in an XML Response. The text to validate should be put into TCTestData column.

If you have dynamic data to pass, then do something like the following……

ReciveFromActivation<Account>,FILE\_AccountId,</Account>

…..where the text highlighted in Yellow is literal text. NOTE: the commas are NOT literal text to be validated but they separate the literal text from the dynamic text. Dynamic text comes from the variable file referenced. So in this example, if account ID value 12345 was in the file “AccountId”, then the text that will be validated in the XML response file will be the following…..

ReciveFromActivation<Account>12345</Account>

NOTE: this validation check will occur only 1 time.

* Component – there is only 1 Component worksheet, and it’s case-sensitive. The format of this worksheet is identical to the TC prefixed worksheet. The intent of this worksheet is to house all of the common code used across many test cases to avoid redundancy in the TC worksheets. For example if you have a set of 10 events on the Customer page which have absolutely identical values for all 5 columns across all test cases, then you can put all of those 10 events into the Component worksheet and then inside each of the TC prefixed remove those 10 rows and replace them with 1 row referencing the Component.

**else** **if** (tc\_eventname.equals("Component"))

{

**for** (**int** w=2;w<RowComponent+1;w++)

{

tc\_eventname\_comp = xls.getCellData("Component", "TCEventName", w);

tc\_objectname\_comp = xls.getCellData("Component", "TCObjectName", w);

tc\_testdata\_comp = xls.getCellData("Component", "TCTestData", w);

objecttolookforafterobjectevent\_comp = xls.getCellData("Component", "ObjectToLookForAfterObjectEvent", w);

millisecondstowaitafterobjectevent\_comp = xls.getCellData("Component", "MillisendsToWaitAfterObjectEvent", w);

**if** (tc\_objectname\_comp.contains(tc\_testdata + "\_"))

{

**if** (tc\_objectname\_comp.equals("TAKEAPAYMENT\_ConfirmationMessage"))

{

*fwgui*.fw\_event(input\_filename, "Component", "GetText", tc\_objectname\_comp, tc\_testdata\_comp, objecttolookforafterobjectevent\_comp, millisecondstowaitafterobjectevent\_comp);

String actual\_text = *fwgui*.*return\_get\_text*;

*fwgui*.fw\_event(input\_filename, "Component", "ValidateText", tc\_objectname\_comp, tc\_testdata\_comp + "," + actual\_text, objecttolookforafterobjectevent\_comp, millisecondstowaitafterobjectevent\_comp);

}

**else**

{

*fwgui*.fw\_event(input\_filename, "Component", tc\_eventname\_comp, tc\_objectname\_comp, tc\_testdata\_comp, objecttolookforafterobjectevent\_comp, millisecondstowaitafterobjectevent\_comp);

}

}

}

}

# Java Functions

The Java Functions that support this automation solution include the following.

|  |  |
| --- | --- |
| **ID** | **Function Description** |
| **1** | Main function inside of {Application}Tests.java. See main function example in GatewayTests.java |
| **2** | fw\_get\_list\_of\_test\_cases\_to\_execute |
| **3** | fw\_create\_output\_log\_file – called to create text output log file. |
| **4** | fw\_event – this function has 7 arguments   1. configuration\_map\_fullpath – full path of input file. 2. tab\_name – worksheet that is to be used i.e. TC24734 or Component 3. tc\_event\_name – TCEventName value (ClickButton, EnterDataTextbox, SelectListValue, SelectCheckbox,GetText,StopExecution,NA). 4. tc\_object\_name – TCObjectName value, example is “OFFER\_Next”. However, if you want to execute all of the objects on a given page, then just indicate the value “OFFER” and it will execute all of the objects on that OFFER page. 5. tc\_test\_data – TCTestData value. Freeform value to enter data into text box or select from listbox or key prefixed value for Component worksheet. 6. object\_to\_look\_for\_after\_object\_event – value from the TC prefixed worksheet ObjectToLookForAfterObjectEvent 7. milliseconds\_to\_wait\_after\_object\_event – value from the TC prefixed worksheet MillisecondsToWaitAfterObjectEvent |
| **5** | fw\_launch\_browser |
| **6** | fw\_nagivate\_to\_url |
| **7** | fw\_check\_for\_loading\_page |
| **8** | fw\_terminate\_window\_processes |
| **9** | fw\_enter\_data\_into\_text\_field |
| **10** | fw\_select\_checkbox |
| **11** | fw\_click\_button |
| **13** | fw\_select\_from\_a\_list\_by\_value |
| **14** | fw\_select\_from\_a\_list\_by\_visible\_text |
| **15** | fw\_get\_text |
| **16** | fw\_quit\_driver |
| **17** | fw\_switch\_to\_driver |
| **18** | fw\_switch\_frame |
| **19** | fw\_close\_window |
| **20** | fw\_get\_window\_handle |
| **21** | fw\_switch\_to\_new\_window |
| **22** | fw\_switch\_to\_window |
| **23** | fw\_accept\_alert |
| **24** | fw\_click\_element\_using\_javascript |
| **25** | fw\_get\_webelements\_object |
| **26** | fw\_get\_element\_object |
| **27** | fw\_validate\_text |
| **28** | fw\_execute\_xml |
| **29** | fw\_get\_value\_from\_xml\_based\_on\_tagname |
| **30** | fw\_set\_variable |
| **31** | fw\_check\_element\_existence |
| **32** | fw\_get\_attribute\_value |
| **33** | fw\_generate\_datetime\_current |
| **34** | fw\_generate\_datetime\_future |
| **35** | Any custom functions that you need to create for your application can be put into the {Application}Functions.java class. Example is GatewayFunctions.java |

# Java Development Standards

The Java development standards are as follows.

1. One GitHub project repository will equate to one corresponding Maven project in Eclipse.
2. All source code will live under the structure {Java Project Name}\src\main\java

Example is Gateway\src\main\java

1. The package name will be named as follows: com.chtr.tmoauto.{Java Project Name}
2. All MainEngine classes should be named as follows: {Application}Tests.java. The class name should be {Application}Tests.java. An example is TPSITests.java. NOTE: there will only be one function in this Class, namely, main.
3. If any application requires any custom functions to be built, then those custom functions will live in the following class {Application}Functions.java. An example is TPSIFunctions.java
4. One input spreadsheet will be located at: {Java Project Name}\maps\{Application}ConfigurationMap.xlsm. The file must be xlsm.
5. The main function will call the fw\_get\_list\_of\_test\_cases\_to\_execute function to identify the list of test cases to execute, pulling this list from the configuration map.xlsm.
6. The main function will loop through that list of tests for execution.
7. The fw\_create\_output\_log\_file will be called inside this loop. Thus, one log file will be generated per test case.
8. The log files will be located at: {Java Project Name}\logs directory. Ensure this maps directory is created.
9. Dynamic variables will be placed in the {Java Project Name}\variables directory. Ensure this variables directory is created.
10. Webservices files will be located at:

{Java Project Name}\webservices\templates

{Java Project Name}\webservices\runtime\requests

{Java Project Name}\webservices\runtime\responses

1. The TC prefixed worksheet will be read determining number of test case steps to execute. A loop will be established inside the test case loop and the fw\_event function will be called/executed per test case step.
2. Last function call for each test case is fw\_closedown\_test.
3. All Framework functions are prefixed with “fw\_”. An example is “fw\_enter\_data\_into\_textbox”.
4. All application related custom functions must be prefixed with application name. Examples are “Gateway\_Login” and “Gateway\_Search”. Make sure you do not have some functions prefixed with Gateway and some functions prefixed with GW. Make consistent across all functions.
5. All functions should have standard comment block headers. Example is…..

/\*\*

\* This function is will log you into the ALM application using URL, userID and passID inputs.

\* @param: URL

\* @param: userID

\* @param: passID

\* @since: 11/16/2016

\* @author: Mark Elking

\*/

public void ALM\_Login (URL, userID, passID)

1. Standard Commenting inside function/code

It is encouraged to comment some of the code in order for others maintaining to have some insight into what the code is doing without having to read all the lines of code in order to interpret what the function code is doing.

Comment blocks start with /\* and end with \*/

Single line of comment start with //

1. All business functions should call the fw\_writeLogEntry (log\_message, return\_code)