

**Java Selenium**

**Automation Framework**

For

All Test Automation Projects

Version 0.1

Last Updated **2/14/2017**

**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](#_Toc474750363)

[1.0 Application/Project Setup 5](#_Toc474750364)

[1.1 GitHub 5](#_Toc474750365)

[**1.1.1** **Push Code into GitHub Repository for the First Time** 5](#_Toc474750366)

[**1.1.2** **Configure Eclipse to Clone Repository from Git First Time** 5](#_Toc474750367)

[1.2 Directory/Folder Structure 6](#_Toc474750368)

[1.2.1 logs 6](#_Toc474750369)

[1.2.2 src 6](#_Toc474750370)

[1.2.3 maps 6](#_Toc474750371)

[1.2.4 variables 6](#_Toc474750372)

[1.2.5 webservices 6](#_Toc474750373)

[1.2.6 testdata 7](#_Toc474750374)

[1.2.7 bin 7](#_Toc474750375)

[1.2.8 driver 7](#_Toc474750376)

[1.3 Configuration Map 7](#_Toc474750377)

[1.3.1 **ALM** 7](#_Toc474750378)

[1.3.2 **BrowserType** 7](#_Toc474750379)

[1.3.3 **Environment** 8](#_Toc474750380)

[1.3.4 **ExecParms** 8](#_Toc474750381)

[1.3.5 **EventNames** 8](#_Toc474750382)

[1.3.6 **Object** 8](#_Toc474750383)

[1.3.7 **TC{ALMTestID}** 9](#_Toc474750384)

[2.0 Event to Function Mapping 12](#_Toc474750385)

[3.0 Java Development Standards 20](#_Toc474750386)

Document Glossary of Terms

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

# Introduction

**This document outlines the detailed setup for automating all applications using the Java Selenium Automation framework established for the TMO Automation Team at Charter.**



# Application/Project Setup

The application/project setup will include GitHub, directory/folder structure, configuration map spreadsheet and Main function setup.

## GitHub

One spreadsheet needs to be created for managing test case data input and test configuration and object

### **Push Code into GitHub Repository for the First Time**

1. On workstation 6659....Go into GitHub and create new project called nextgen

2. Log into the box as svc\_automation (as an example)

3. create directory C:\Users\svc\_automation\git\nextgen (nextgen is an example)

4. Launch DOS Prompt

5. cd to that directory just created

6. git init

7. git add --all

8. git commit -m "20161107 - Gateway Baseline"

9. git remote add origin https://svc\_automation@git.corp.chartercom.com:8443/scm/taa/nextgen.git

NOTE: IF YOU GET ERROR that origin already exists, then run this command first...

git remote rm origin

10. git push -u origin master

NOTE: if you get a Fatal Authentication because credentials to Git are preset and they need to be edited, then go to Control Panel\All Control Panel Items\Credential Manager and edit the Git entry.

### **Configure Eclipse to Clone Repository from Git First Time**

1. Launch Eclipse

2. In left hand project explorer, select Import

3. Select Git\Projects from Git

4. Select Next

5. Select CloneURI

6. Fill out information

URI: ssh://melking@git.corp.chartercom.com:7999/taa/nextgen.git

Host git.corp.chartercom.com

Repository Path /taa/nextgen.git

Protocol ssh

Port 7999

User melking

Password ..........

click Next button

7. Hit Next

8. Click Finish

## Directory/Folder Structure

The directory folder structure will need to be created for the application/project. There are required folders.

### logs

The Java Selenium execution output log files will be placed in this directory.

### src

All of the Java code will be located in this directory including .java class files and all functions.

### maps

The Configuration Map file will be located in this directory. There will be one .xlsm configuration map file per project/application.

### variables

Many functions create temporary variable files when called and store those temporary variable files in this directory.

### webservices

All webservice files reside in this directory. There are 2 mandatory subdirectories, 1) requests and 2) responses. Also, there are 2 manadatory subdirectories under requests, namely, a) templates and b) runtime. The templates directory contains all of the XML requests which are parameterized with variables to be substituted. All variables should be prefixed with “XML\_”. An example is “XML\_PHONE\_NUMBER”. During runtime, the XML runtime request is generated and stored in webservices\requests\runtime directory. Also, the response of the xml request is placed into the webservices\responses directory.

### testdata

Contains all of the

### bin

Contains all of the binary files.

### driver

Contains all of the

## Configuration Map

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.

### **ALM**

Contains all of the ALM configuration parameters for the given application

### **BrowserType**

Contains a list of the browser types supported for the given application

### **Environment**

Contains list of all environments available for automation of that application

For webservices, 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.

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

CREDS example chtr\svc\_tst\_automation:H2i1fL9!

### **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.

### **EventNames**

List of TC events available. This is a fixed list and corresponds to a function call in the Framework fw\_event function.

### **Object**

Must have 3 columns named,

* + - 1. **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.

* + - 1. **COLUMN TagName,Attribute,Value**

the value should be tagname + “,” + attribute + “,” + value. Examples include:

Button,NA,Search

NA,id,addressLine1

input,xpath,//\*[@id=”container”]/button

* + - 1. **COLUMN ExtraInfo**

Any other information that you need to have associated per field can be used in the ExtraInfo column (if needed)

### **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.

* + - 1. **COLUMN TCObjectName**

which is a dropdown data validation list linked to ObjectName named range on Object worksheet.

* + - 1. **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}

* + - 1. **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.

For Webservices, do the following…..

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.

* + - 1. **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.

* + - 1. **COLUMN TCEventName**

which is a dropdown data validation list linked to EventNames worksheet. See Section 2.0 – Event to Function Mapping for a list of the available Event Names.

# Event to Function Mapping

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

| **ID** | **Event Name (or Calling Function)** | **TCTestData** | **Function/Description** |
| --- | --- | --- | --- |
| 1 | Manual execution via Eclipse or via Jenkins | NA | Main function inside of {Application}Tests.java. See main function example in GatewayTests.java |
| 2 | Main function | NA | fw\_get\_list\_of\_test\_cases\_to\_execute  Called from the Main function. Pulls all test IDs from the ExecParms worksheet which have a corresponding value of “Y”. |
| 3 | Main function | NA | fw\_create\_output\_log\_file – called to create text output log file. |
| 4 | Main function | NA | 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 | LaunchBrowser | IE or CHROME | fw\_launch\_browser  This will open the browser and sets current driver to IE or CHROME. |
| 6 | NavigateToURL | make sure the 2nd part of the object\_name corresponds to the column in Environments worksheet. For example, if you want to navigate to a CDC URL, then the object definition should be NAVIGATE\_CDCURL and the corresponding column header in the Environments worksheet should be CDCURL. If you want to navigate to a Retail Sales URL, then the object definition should be NAVIGATE\_RETAILSALESURL and the corresponding column header in the Environments worksheet should be RETAILSALESURL. | fw\_navigate\_to\_url  Navigates to url and maximizes window. |
| 7 | fw\_event | Value not needed here | fw\_check\_for\_loading\_page |
| 8 | TerminateWindowProcesses | List of window processes to terminate comma delimited  For example  “excel.exe,chrome.exe” | fw\_terminate\_window\_processes |
| 9 | EnterDataTextbox | Value to enter into textbox  For example “1234”.  If you want to pull value from variable file, then indicate “FILE\_{name of variable file}” to pull from.  If you want to send TAB, then put “KEYTAB,9”…..9 represents the number of times to tab.  If you want to send ENTER, then put “KEYENTER”. | fw\_enter\_data\_into\_text\_field |
| 10 | SelectCheckbox | Value not needed here | fw\_select\_checkbox |
| 11 | ClickButton | Value not needed here | fw\_click\_button |
| 12 | SelectListValueByValue | Value to be selected from the list. | fw\_select\_from\_a\_list\_by\_value |
| 13 | SelectListValueByVisibleText | Value to be selected from the list. | fw\_select\_from\_a\_list\_by\_visible\_text |
| 14 | GetText | Value not needed here | fw\_get\_text |
| 15 | Currently called from fw\_closedown\_test (which is in Logging.java) | NA | fw\_quit\_driver  Quits the active driver. |
| 16 | SwitchToDriver | IE or CHROME | fw\_switch\_to\_driver  This function switches current driver to IE or CHROME. |
| 17 | No Event | Using locator and locatorvalue, the webelement is determined and then……  *driver*.switchTo().frame(webelement); | fw\_switch\_frame |
| 18 |  | Gets current handle, switches to new window, closes new window, then returns back to original window. | fw\_close\_window |
| 19 | GetWindowHandle | Gets current window handle. | fw\_get\_window\_handle |
| 20 | SwitchToNewWindow | Switches to new window | fw\_switch\_to\_new\_window |
| 21 | SwitchToWindow | window\_handle\_value  which gets passed into the following method  *driver*.switchTo().window(window\_handle\_value); | fw\_switch\_to\_window |
| 22 | AcceptAlert | Value not needed here | fw\_accept\_alert |
| 23 | ClickJAVASCRIPT | Value not needed here | fw\_click\_element\_using\_javascript |
| 24 |  |  | fw\_get\_webelements\_object  Returns a list of webelementss |
| 25 |  |  | fw\_get\_element\_object  Returns by value for selecting object. |
| 26 | ValidateText | {Expected Value},{Actual Value}  Example is “Jingle,JingleBells”. | fw\_validate\_text  The validate text function checks to see if the expected value is CONTAINED inside the actual value. It’s not an EQUALS comparison. It’s a contains comparison. |
| 27 | XMLExecute | 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.  Enter values to substitute in TCTestData column in following format 1) XML\_LOCATION\_ID,12345 (hard coded example), or 2) 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. | fw\_execute\_xml  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. |
| 28 | 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). | fw\_get\_value\_from\_xml\_based\_on\_tagname |
| 29 | SetVariable | Put name of the variable (case-sensitive) followed by a comma follow by value you want for this variable. Below example:  Sysprin,82451200 NOTE: A file named “Sysprin” will be created in your “variables” folder with the variable value “82451200”. You must have a variables folder defined in your workspace. NOTE: the TCObjectName should be “NA”. | fw\_set\_variable |
| 30 | CheckForElementExistence | 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. | fw\_check\_element\_existence |
| 31 | GetAttribute | Value is attribute name you want to get value of like (img or data-reactid, etc). | fw\_get\_attribute\_value |
| 32 | GetCurrentDate | Value is “{variable name},{format of date}”. Here is an example: “CurrentDate,yyyy-MM-dd”  NOTE: TCObjectName value is NA. | fw\_generate\_datetime\_current |
| 33 | GetFutureDate | Value is “{variable name},{format of date},{number of days in future}”. Here is an example: “FutureDate,yyyy-MM-dd,10”.  NOTE: TCObjectName value is NA. | fw\_generate\_datetime\_future |
| 34 | WriteLogHeader | the header block name that you want to appear in your output log file. | fw\_writeLogEntry |
| 35 | 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. | fw\_validate\_text\_in\_xml\_response |
| 36 | XMLGetValueByMultipleTagnames | First delimiter is &&, value to right is the output file that the value will be stored in.  All values to the left are the search criteria delimited by “--“.  First search string is always SEARCHFORWARD.  2nd search string can be forward or backward and you must indicate how many tags to traverse looking for whatever string value you want to look for.  3rd search string in below example searches forward and then once all search is done, then the value to the right between > and < will be placed into output file.  Example is  SEARCHFORWARD--E911 INSERT</osmc:ApsIntent>, SEARCHBACKWARD(5)--FILE\_Number##</osmc:TelNumber>, SEARCHFORWARD(3)--<osmc:ClientOrderNumber&&ClientOrderNumber1 | fw\_get\_value\_from\_xml\_based\_on\_multiple\_tagnames |
| 37 | IncrementValueByOne | {Variable File Name}  Example: AptNumber | fw\_increment\_value\_by\_one  This event will open the file, pull the integer value out of the file and increment it by one and store that incremented value back into the same file. |
| 38 | NA | NA | fw\_get\_value\_from\_file  The file name full path is passed into this function. |
| 39 | NA | NA | fw\_get\_workspace  This function gets the active workspace. |
| 40 | Component  Logic to execute Component events in Main.java | The value should be the prefix of all of the ObjectNames. So for example, if you have 3 objects named….  SEARCH\_Address1  SEARCH\_ZipCode  SEARCH\_Submit  Then you can have a Component event with a TCTestData value of SEARCH and it will navigate to the Component worksheet and execute all test case events for that given test case which have a prefix of SEARCH. | 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. |

# 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)

# Execution

The application/project setup will include GitHub, directory/folder structure, configuration map spreadsheet and Main function setup.

## Eclipse

One spreadsheet needs to be created for managing test case data input and test configuration and object

## Jenkins

The directory folder structure will need to be created for the application/project. There are required folders.