D:\Logos\CG_logoReflect.tif

**CMT Tech Team**

**6/25/2019**

Unified Automation Framework– User Document V3.0

Table of Contents

[1. Purpose 2](#_Toc12379435)

[2. Pre-requisite 2](#_Toc12379436)

[3. UAF 2.0 – An Introduction 2](#_Toc12379437)

[4. UAF 2.0 – Architecture 3](#_Toc12379438)

[4.1 Tools: 4](#_Toc12379439)

[4.2 Applications: 4](#_Toc12379440)

[4.3 Test Framework 4](#_Toc12379441)

[4.3.1 Framework Settings: 4](#_Toc12379442)

[4.3.2 Test data: 6](#_Toc12379443)

[4.3.3 Test Scripts: 7](#_Toc12379444)

[4.4 Test Execution 7](#_Toc12379445)

[4.4.1 Maven (POM.xml): 8](#_Toc12379447)

[4.4.2 TestNG (testng.API): 8](#_Toc12379454)

[4.4.3 Continuous Integration: 9](#_Toc12379461)

[4.5 Extent Test Reports 10](#_Toc12379463)

[5. Salesforce DotNeXT Architecture 11](#_Toc12379464)

[5.1 Prerequisite System Requirement: Operating System and supported Browsers 12](#_Toc12379467)

[5.1.1 Operating System 12](#_Toc12379468)

[5.1.2 Supported Browsers 12](#_Toc12379469)

[5.2 JDK and Eclipse IDE Installation 12](#_Toc12379470)

[5.2.1 JDK Installation 12](#_Toc12379472)

[5.2.2 Eclipse IDE Installation 18](#_Toc12379474)

[5.2.3 Selenium Server and other servers 20](#_Toc12379475)

[5.2.4 Get the Framework zip file 20](#_Toc12379476)

[5.3 Import the project 20](#_Toc12379477)

[5.4 SFDC.Next Framework Configuration 25](#_Toc12379478)

[5.5 Creating Object Repository 27](#_Toc12379479)

[5.6 Creating Automation Scripts 33](#_Toc12379480)

[5.7 Creating Business Components 33](#_Toc12379482)

[5.8 List of Framework Reusable library 35](#_Toc12379483)

[6. UAF 2.0 – Execution Flow 41](#_Toc12379484)

[7. UAF 2.0 – Folder Structure 43](#_Toc12379530)

[8. Grid Execution 45](#_Toc12379531)

[7.1 In Selenium: 46](#_Toc12379532)

[7.2 In Appium (Android &iOS): 48](#_Toc12379533)

[8. Parallel Execution 50](#_Toc12379534)

[9. Video Automation Solution 53](#_Toc12379535)

[9.1 Video Testing- Overview 53](#_Toc12379536)

[9.2 Video Testing – Web/Mobile 53](#_Toc12379537)

[9.3 Video Testing – Connected devices 54](#_Toc12379538)

[9.3.1 Set Data.properties 54](#_Toc12379539)

[9.3.2 Run CloneSetup.bat file 55](#_Toc12379540)

[9.3.3 Create New Test Cases 55](#_Toc12379541)

[9.3.4 Set TestRunner.xls 56](#_Toc12379542)

[9.3.5 Python Reusable Methods 56](#_Toc12379543)

[10. Contacts 57](#_Toc12379544)

[11. Change Log 57](#_Toc12379545)

# Purpose

The purpose of this document is to provide a detailed overview of Cognizant Reusable Automation Framework for Desktop & Mobile Application Testing.

# Pre-requisite

|  |  |
| --- | --- |
| S.No | Prerequisites |
| 1 | Windows Version XP/Win 7 (or) MAC Version Lion/Mountain Lion |
| 2 | Android SDK Version R20 and above |
| 3 | Python 2.7 + |
| 4 | JDK Version 1.8.X |
| 4 | Eclipse IDE 4.4 and above |
| 5 | ADT plugin 20 and above |
| 6 | .apk file or signed .ipa file |
| 7 | Appium 1.3.4 and above |
| 8 | Device - Android 4.4 and above or iOS 8.3 and above |

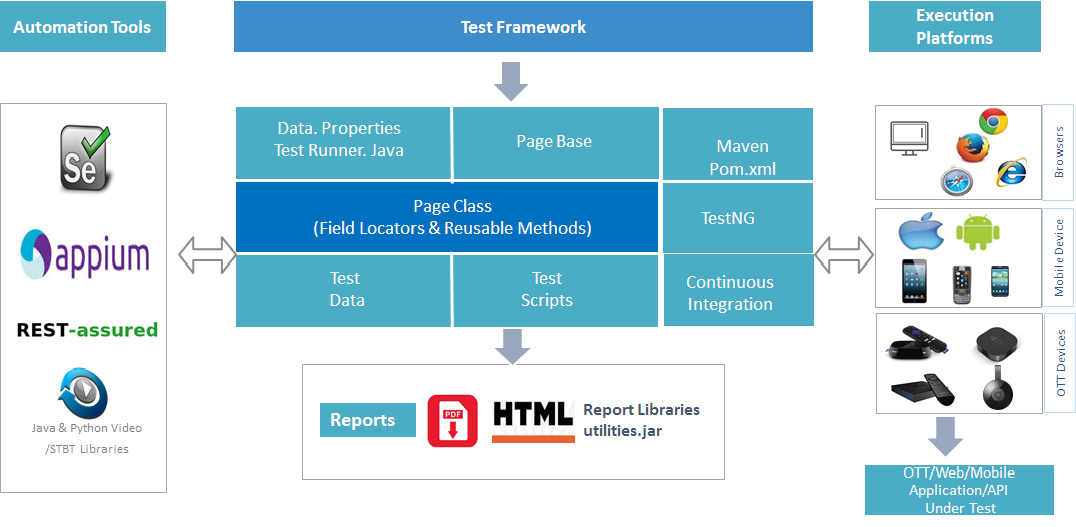
# UAF 2.0 – An Introduction

* + UAF 1.2 - Cognizant Reusable Automation Framework for Testing - is Cognizant’s in-house automation framework based on Page Object Model, proposed to ease the script development and maintenance effort in test automation.
  + Built on Maven structure and follows TestNG Framework.
  + UAF 1.2 follows data driven and modular approach
  + UAF 1.2 is tool agnostic, which means that the UAF 1.2 design principles Supports major test automation tools such as Selenium and Appium.

# UAF 2.0 – Architecture

The following diagram presents a detailed conceptual model of UAF 1.2, and portrays clearly the various modules within the UAF 1.2 architecture. Each of these modules is subsequently explained in detail, in terms of the basic usage of the module (What), the benefits of using the module (Why).

***Schematic representation of UAF 1.2***



## Tools:

UAF 1.2 framework supports tools like Selenium and Appium.

## Applications:

UAF 1.2 framework supports Native Application, Hybrid Application and Web Applications (Mobile and Desktop).

**Native Applications:**

Apps developed exclusively for a specific mobile platform that can leverage all device capabilities. Appium supports Native applications on Android & iOS platform respectively.

**Hybrid Applications:**

Hybrid apps, like native apps, run on the device, and are written with web technologies (HTML5, CSS and JavaScript). Appium supports Native applications on Android & iOS platform respectively.

**Web Applications:**

Web applications on Mobile and Desktop are a [client-server](https://en.wikipedia.org/wiki/Client%E2%80%93server_model) [software application](https://en.wikipedia.org/wiki/Software_application) in which the client (or user interface) runs in a [web browser](https://en.wikipedia.org/wiki/Web_browser). Appium supports Native applications on Android & iOS platform respectively.

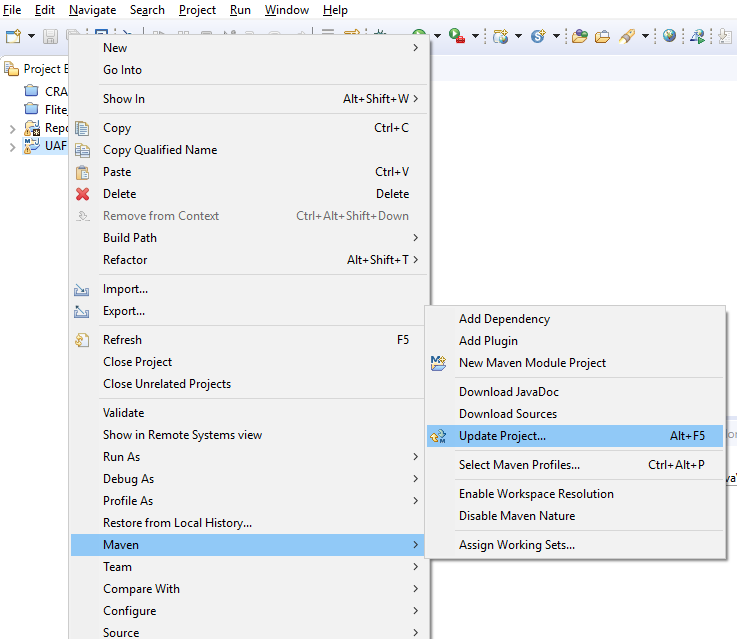
## Test Framework

UAF 1.2 framework simplifies the test design process and test script management.

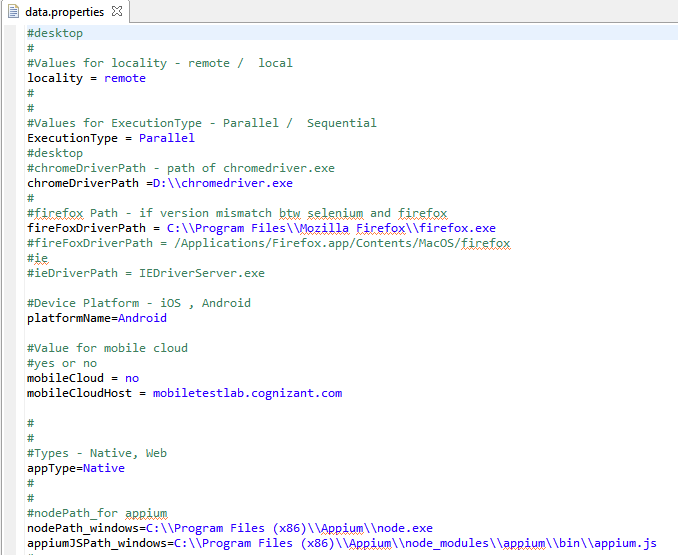
### Framework Settings:

As the name suggests, this is a place where user specifies the settings for the test cases. User can edit tool selection, mobile platform, general settings, and tool capabilities.

Once you import the framework in eclipse make sure to update the project with maven which will download the dependencies from the POM.xml.

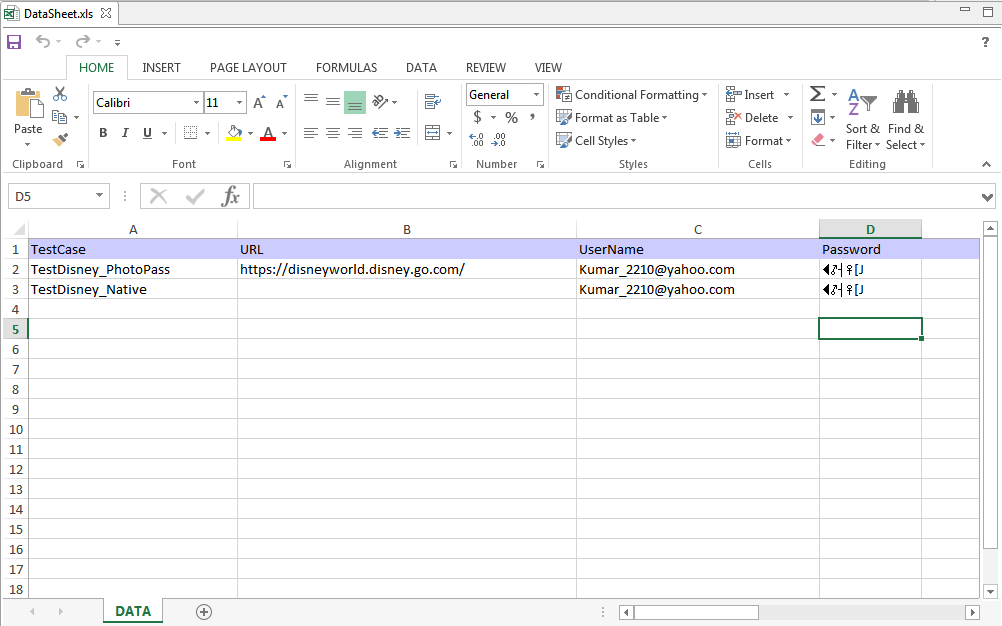


***Framework Settings -UAF 1.2***



### Test data:

The test data is DataSheet.xls file which have the data inputs required by the test script. This helps to implement sound principles of automation script design by separating the data from the scripts.



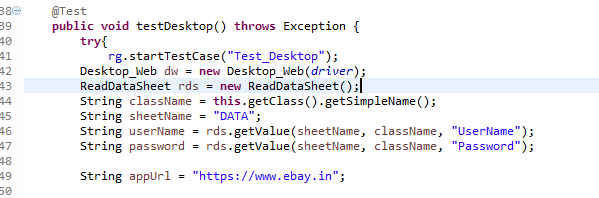
***Sample test data sheet of UAF 1.2***

Above fig. shows a sample test data sheet. Using the sheet name, row and column name, the data inside the test data can be used in the scripts.

**Data** is the sheet name where you can feed the inputs for test scripts. User can also feed the test data in the other two sheets or else they can create the new sheet where they can feed the test data for the script execution.

User has to mention sheet name, test case name and column name in particular test scriptfile for test data during execution.

***How to access Data Sheet***



### Test Scripts:

In this phase, user can design their test scripts and also define the functions (BeforeClass, Test, and AfterClass).

**@** **BeforeClass**

User can add the functional library, set the driver etc.

**@Test**

User can design test scripts for their application

**@** **AfterClass**

It will close the driver session

### 

## Test Execution

In this phase, the execution starts from TestRunner.java class which invokes testng.xml, which in turn calls TestSuite and followed by your test scripts. TestSuite starts the tool session which communicates with device and will launch the browser/install the application and execute test cases etc.



### Maven (POM.xml):

### Maven is a project management and comprehension tool. Maven provides developers a complete build lifecycle framework.

### Maven uses Convention over Configuration which means developers are not required to create build process themselves. Developers do not have to mention each and every configuration details.

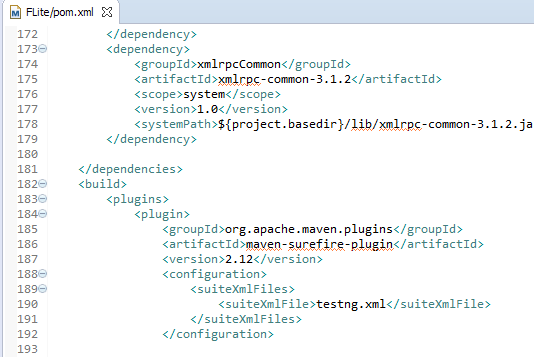
### POM stands for Project Object Model.

### It is fundamental Unit of Work in Maven. It is an XML file.

### It always resides in the base directory of the project as pom.xml.

### It contains information about the project and various configuration details used by Maven to build the project(s).

***Sample pom.xml of UAF 1.2***



### TestNG (testng.API):

## TestNG is an open source automated testing framework, where NG means Next Generation.

## In the older framework we used TestNG.xml configuration files to configure and define TestNG suites.

## The problem with the use of testing.xml is that they are static files and cannot be changed at runtime. Sometimes we may need to create a test suite at runtime, which is based on the Excel sheet or database data. For such problems TestNG provides a feature to define and run TestNG tests at runtime through code by using TestNG API all the configurations that are allowed through xml can be achieved by using the TestNG API.

## Eliminating most of the limitations of the older framework, TestNG API gives the ability to write more flexible and powerful tests. The TestRunner.java class in the framework uses the TestNG API to dynamically create a testing suite file based on the data from the Testrunner.xls sheet.

## Sample code to read testdata from TestRunner.xls.

## 

***Sample TestRunner.Java of UAF 1.2***

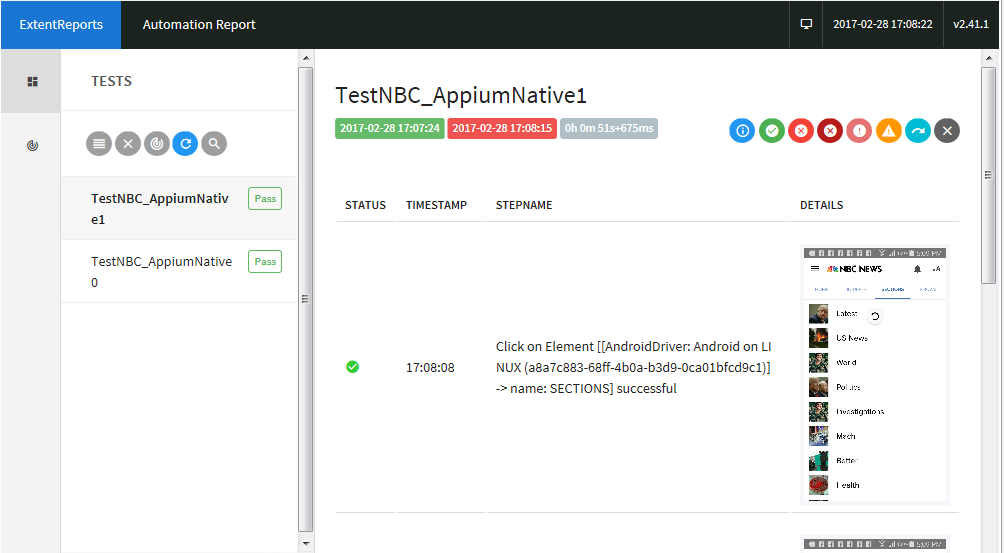
### Continuous Integration:

## *Continuous integration* is a process in which all development work is integrated at a predefined time or event and the resulting work is automatically tested and built. The idea is that development errors are identified very early in the process. UAF supports Continuous Integration through Jenkins.

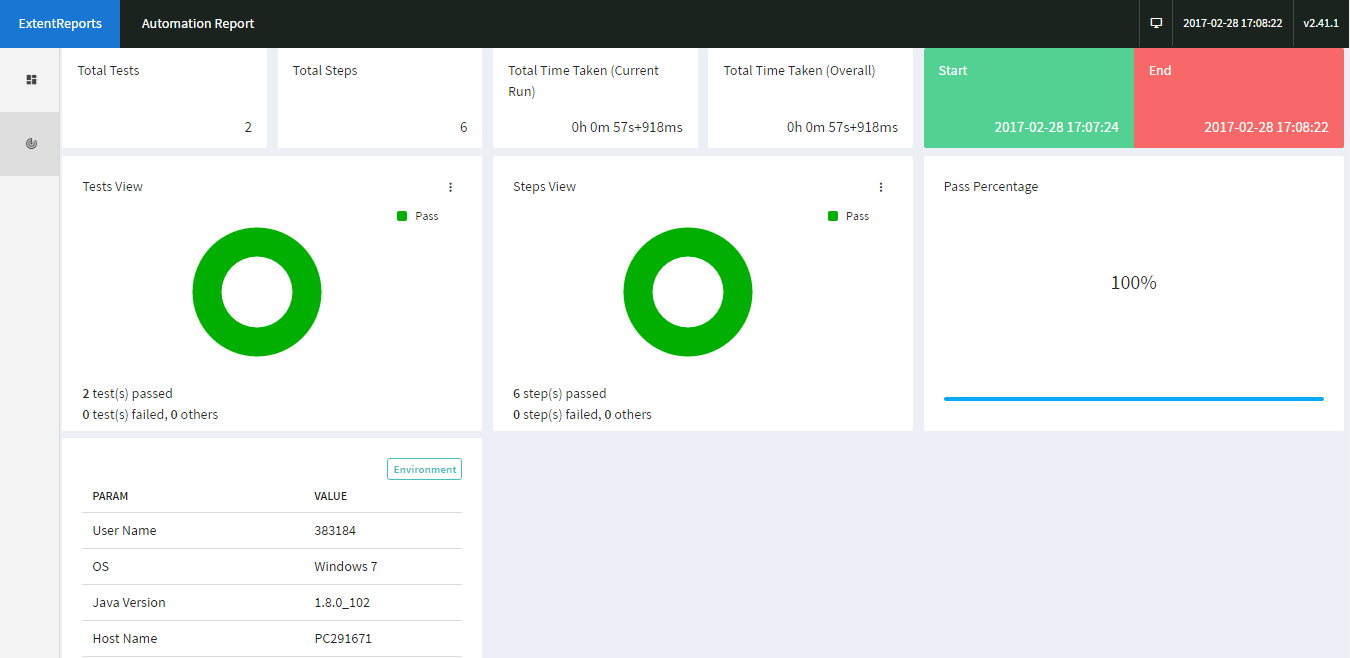
## Extent Test Reports

The Test Reports are generated in HTML format with ExtentReport. It shows the detailed step by step report with screen shots and more analytical data.

* Easy to understand, business facing results.
* It provides detailed and graphical analysis of the project, report, category, author and test to enable you to find the strong and weak areas in your application.
* The results are automatically stored with a timestamp indicating the date and time of execution, for easy reference.
* ExtentReports.html file will be created under your project folder.



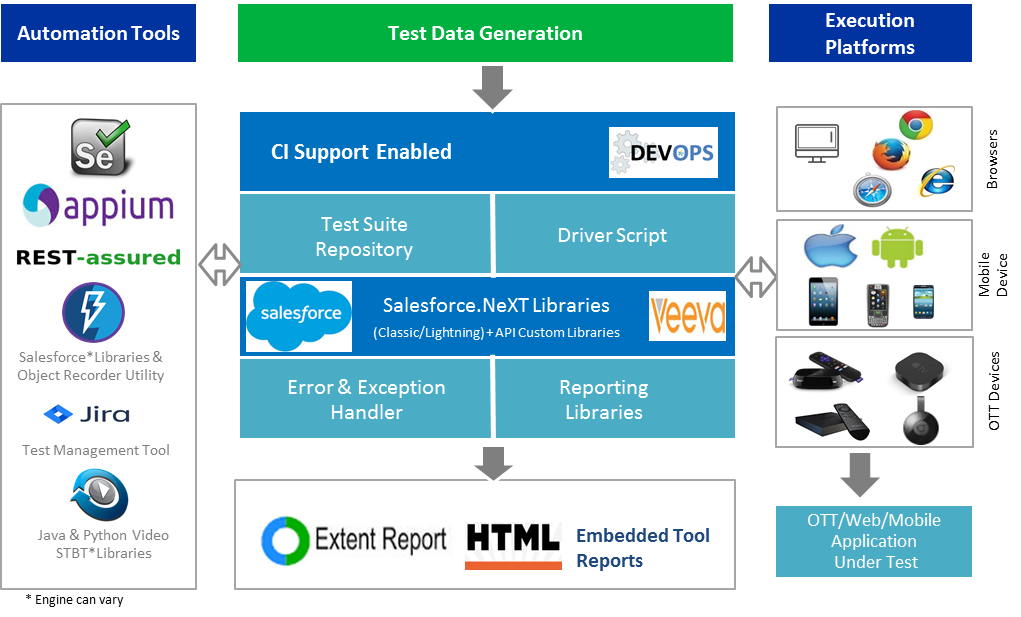
***Sample detailed test case results of UAF 1.2***



***Sample detailed test case report of Test Case***

# Salesforce DotNeXT Architecture

Salesforce DotNeXT Automation Framework is a Tool Agnostic Automation Solution. The following diagram presents a detailed conceptual model of Salesforce DotNeXT Framework and portrays clearly the various modules within the Salesforce DotNeXT Automation Framework.





## Prerequisite System Requirement: Operating System and supported Browsers

## Operating System

List of SFDC.Next framework supported Operating system:

* All Windows 32 bit/64bit version.



Please click on the link for in detail system configurations

## Supported Browsers

List of SFDC.Next framework supported Browsers:

* Google Chrome.
* Internet Explorer 7, 8, 9, 10, and 11 on appropriate combinations of Vista, Windows 7, Windows 8, Windows 8.1, Windows 10
* Firefox
* Safari.



## JDK and Eclipse IDE Installation

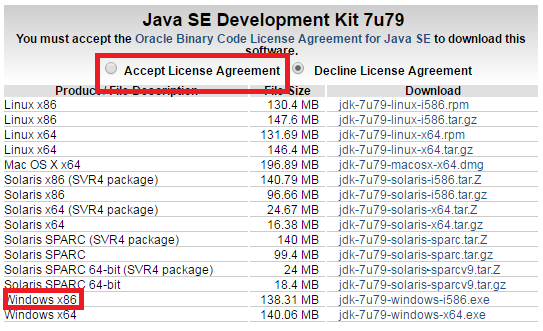


## JDK Installation

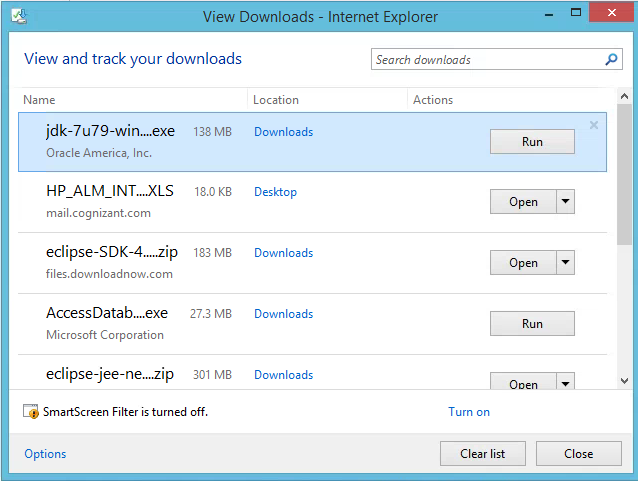
* Download 32 bit (Recommended version JDK 8 version from oracle website link provided below and install on the system.)

<http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html>

**Note: If your system is 64bit download 32 bit version of JDK only.**

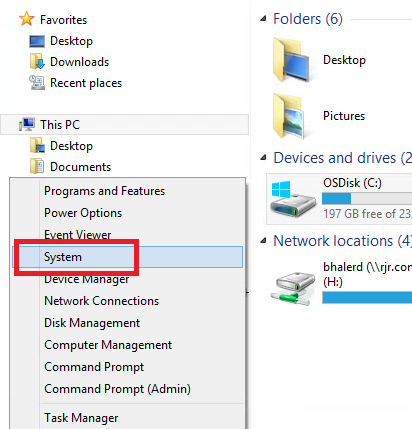


* Click on run and install the JDK through the installation wizard.

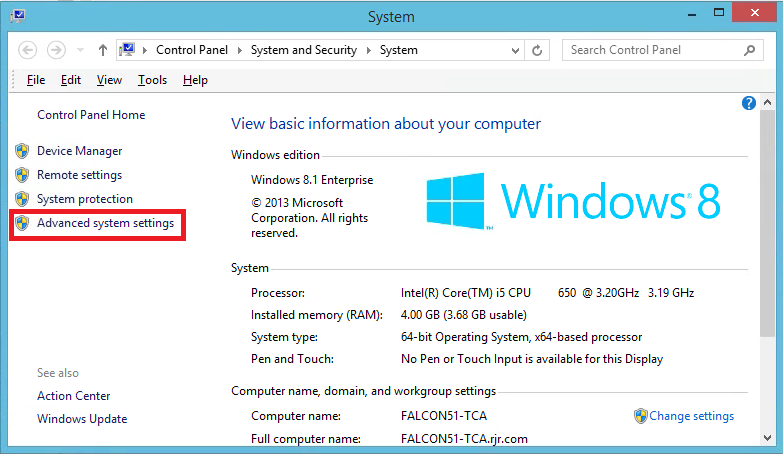


**Set the Environmental Variables**

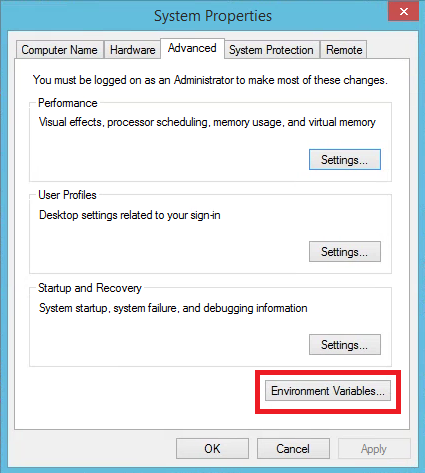
* Right click on the windows and click on System (in windows 8). OR Go to My Computer and click on properties (in windows 7) properties



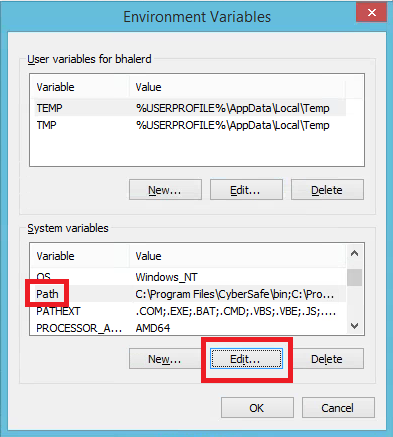
* click on advanced tab

****

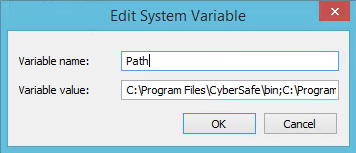
* click on environment variables



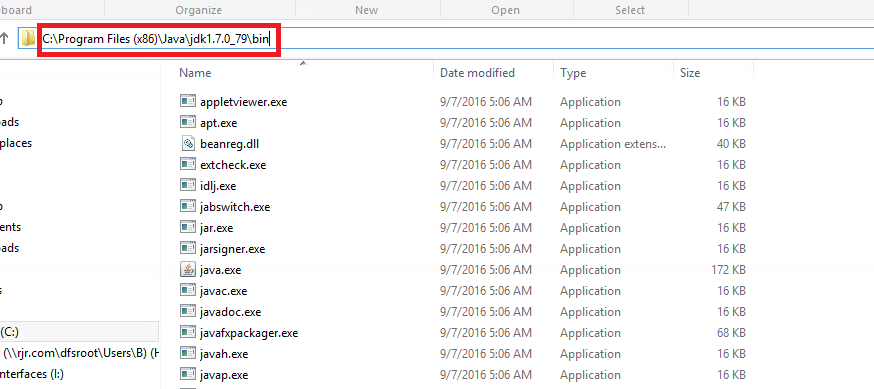
* Search Path under System variables and Click on Edit Button



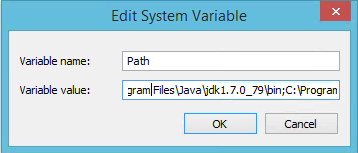
* Edit System variable will be opened.



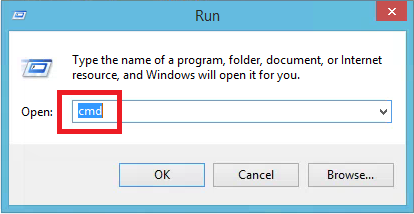
* Copy the path of bin folder of JDK and JRE



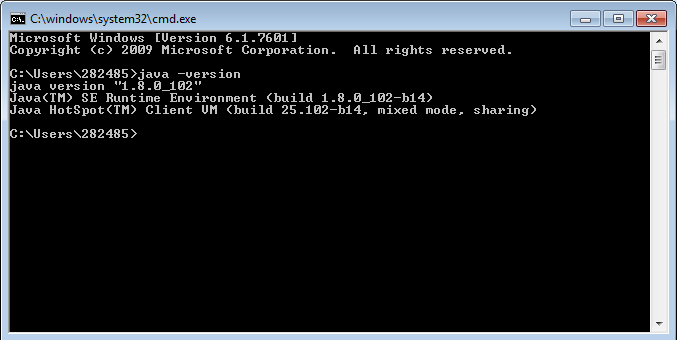
* paste path of bin folder in variable value
* Paste path of bin folder in variable value as Semicolon separated.
* For Example (C:\Program Files\Java\jdk1.8.0\_79\bin;C:\Program> Files\Java\jdk1.8.0\_79\jre\bin)



* Click on ok button
* Now your path is set. You can now execute any program of java from any drive.
* Give cmd in run. And click on ok button.



* Command Prompt will opened Type java –version and click on enter



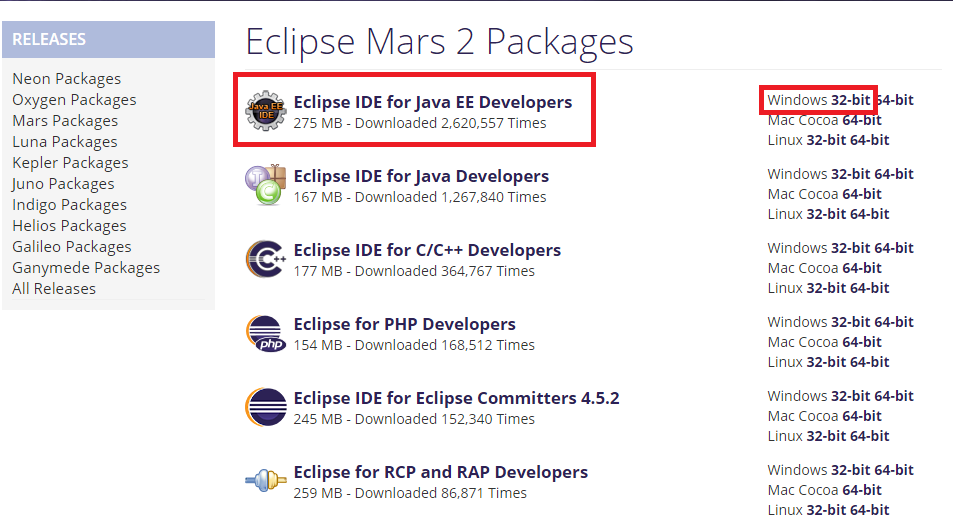
It should show the current installed version of java.

## Eclipse IDE Installation

* + - Go to Eclipse official website and navigate to its download page –<http://www.eclipse.org/downloads/>?

Download Eclipse IDE for Java EE developers. All the recent releases are available on the page. You can use any package.

**Note: If your system is 64-bit download 32 bit Eclipse IDE.**



* Click on the download link, the user is re-directed to the fresh page securing information about the current download. Click on the download icon and you are done.

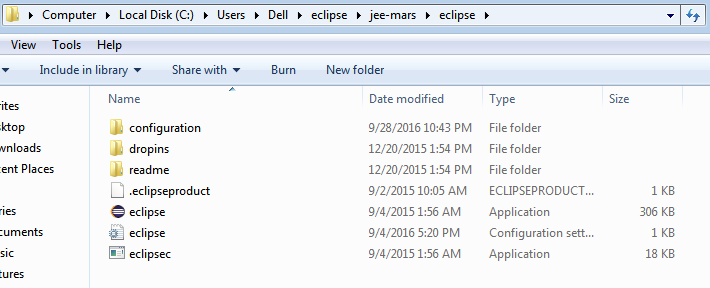


It may take a few minutes before you can download the complete zip folder.

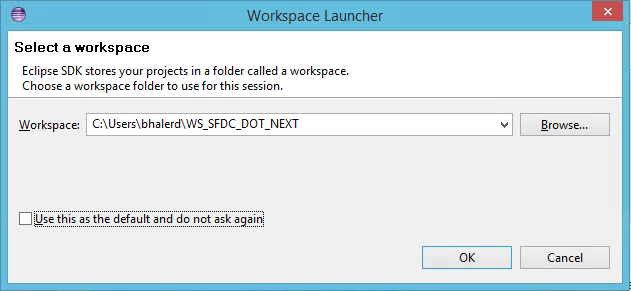
* Once downloaded, copy the folder and place it in the desired location on your file system.



* Extract the zipped folder, a folder named as eclipse can be seen. The folder embodies all the required application and source files.



* Launch the eclipse IDE using “eclipse.exe” residing inside the eclipse folder. Refer the above illustration for the same.
* The application will prompt you to specify the workspace location. Workspace is that location where all your eclipse projects will be residing. Enter/Browse the desired location or the user can simply opt for the default location and click on the OK button. Enter a name and create a workspace say WS\_SFDC\_DOT\_NEXT



## Selenium Server and other servers

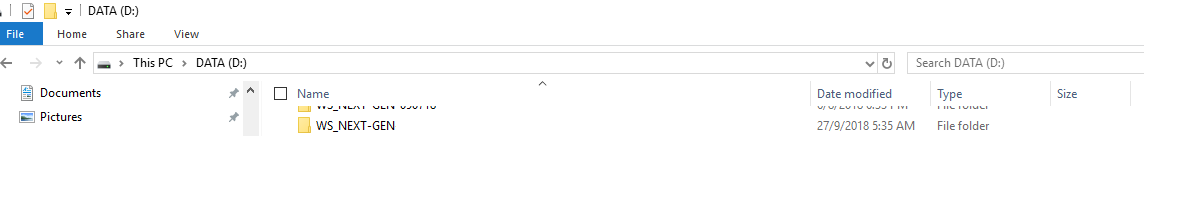
All selenium related jar files and other files i.e. geckodriver, chrome driver, ie driver, selenium server are already present inside Framework.

## Get the Framework zip file

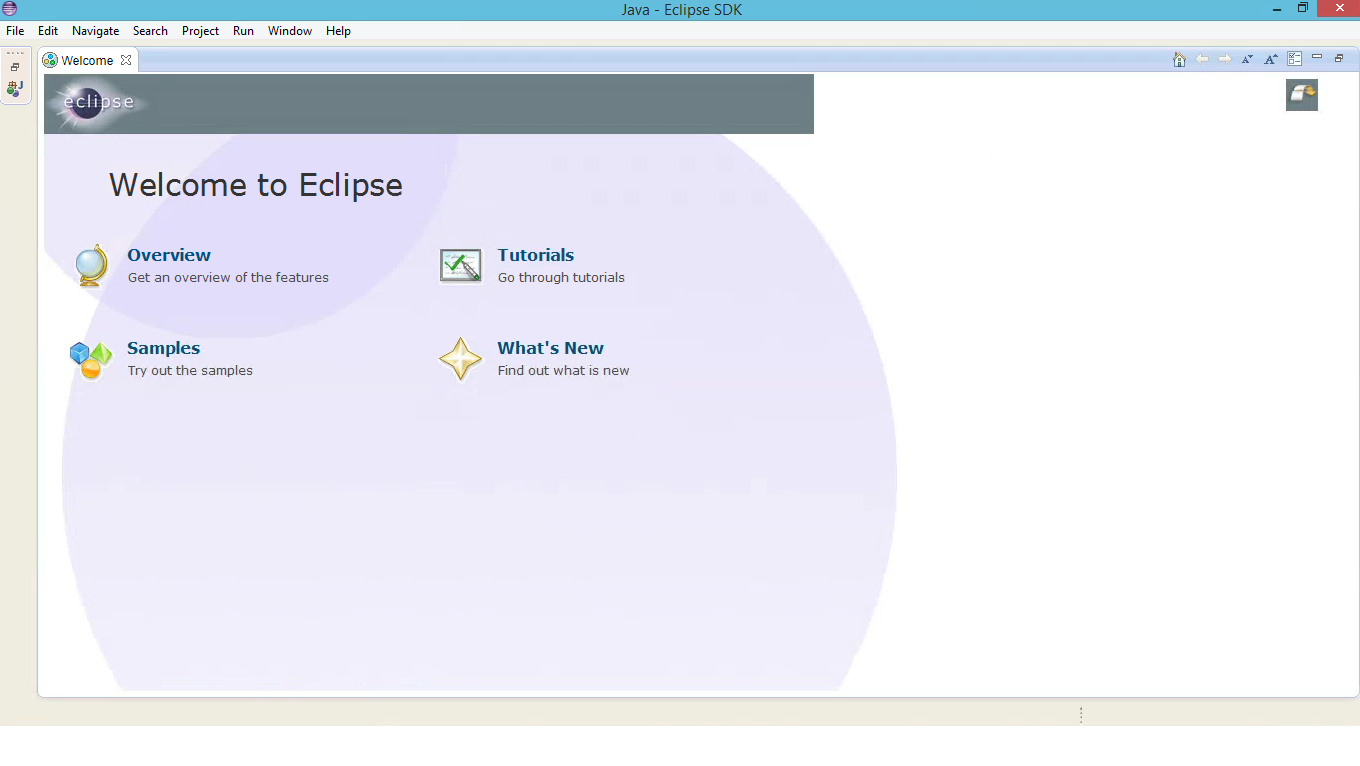
Get the NEXT-GEN framework zip file and place the folder in any drive.

## Import the project

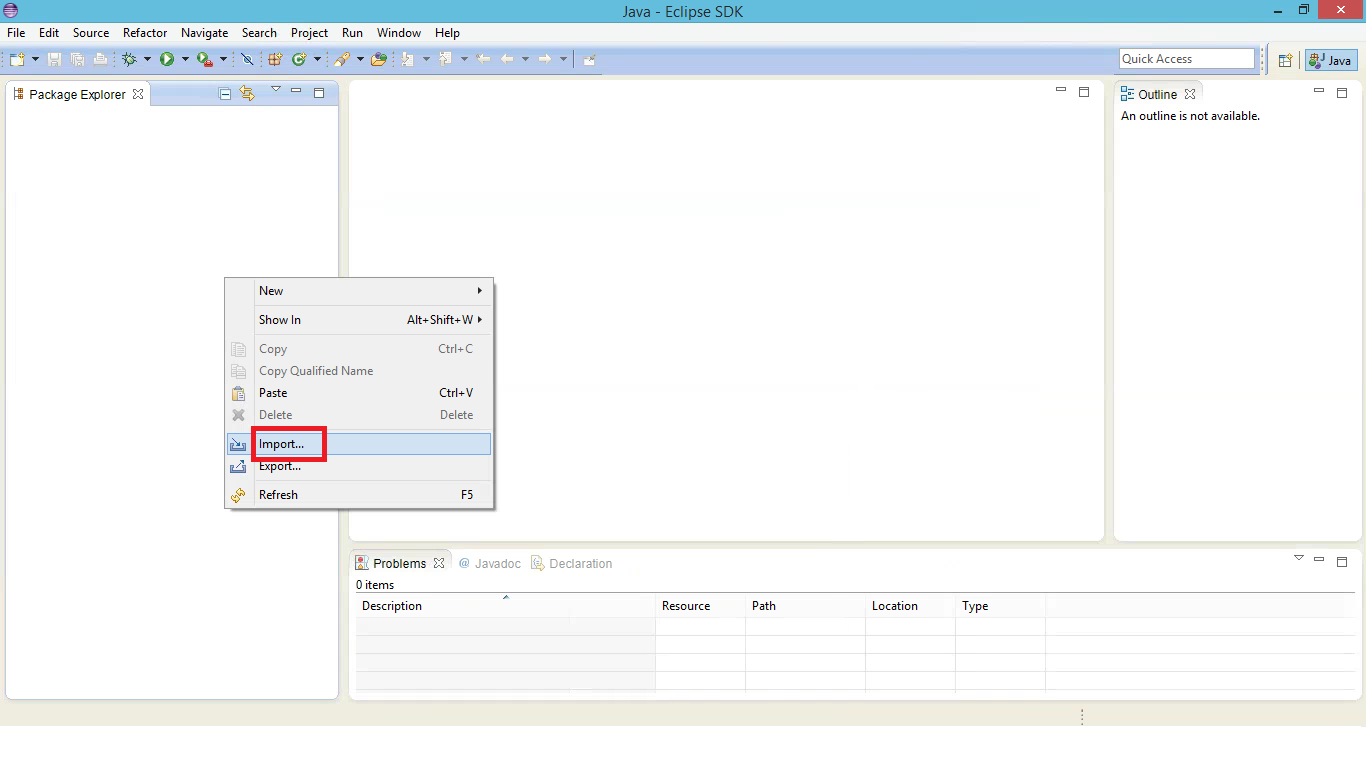
* Create workspace WS\_NEXT-GEN or Any folder into any drive.



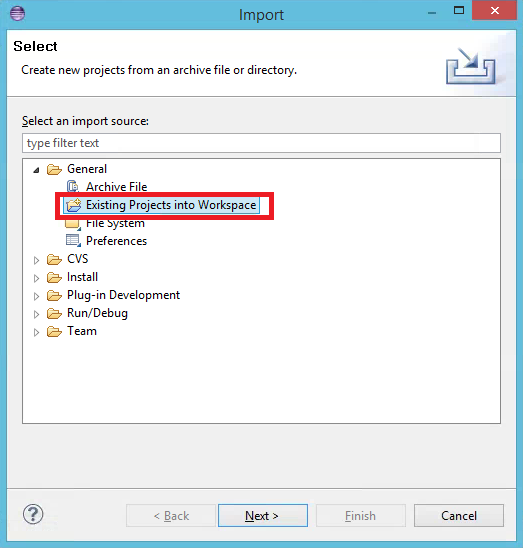
* Open Eclipse Workspace, Welcome to Eclipse page will be displayed. Click on Package Explorer on the left side



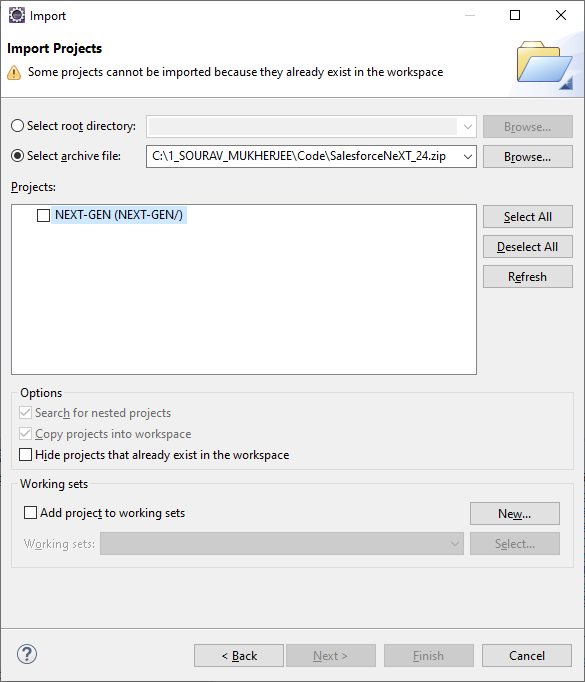
* Blank Package Explorer will be displayed on the left side.
* In Eclipse Right click in the Package Explorer and click on Import.



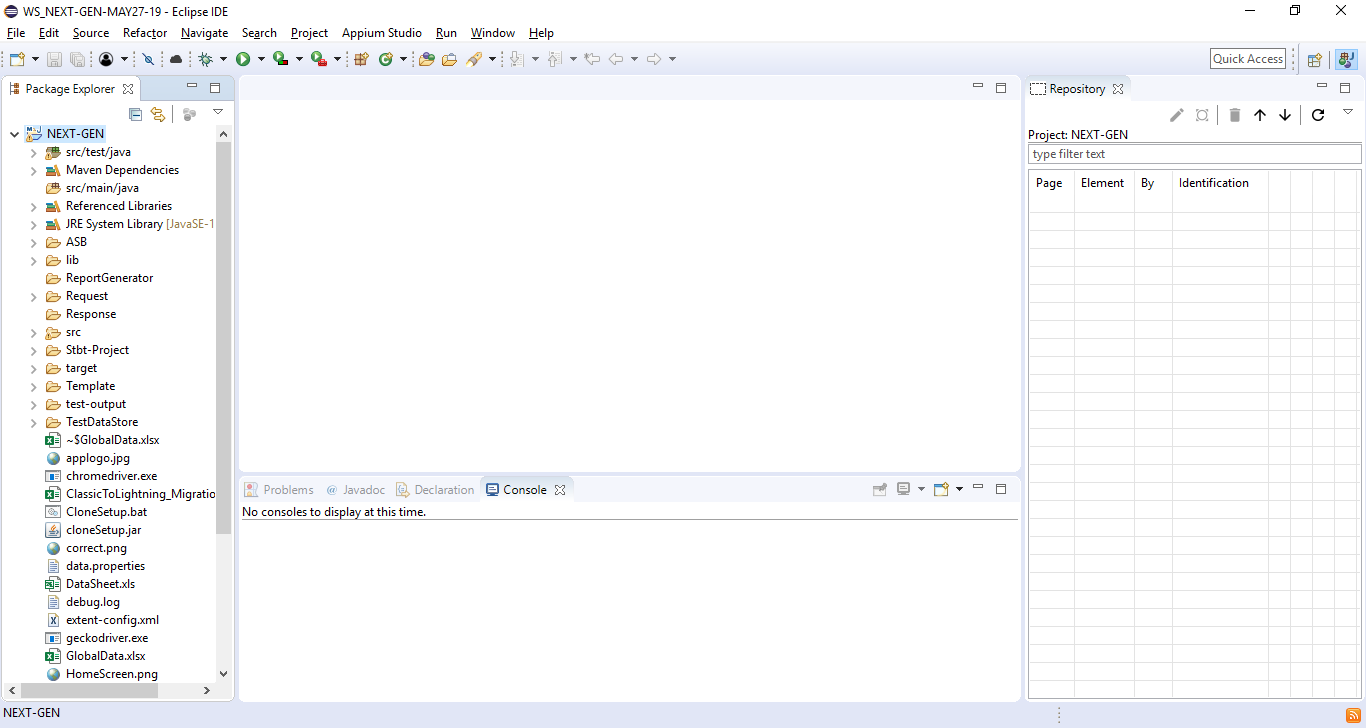
* Select an import Source as Existing Project into Workspace under General and click on Next



* + Click on Select archive file and browse select the project zip file and check the box and click Finish.

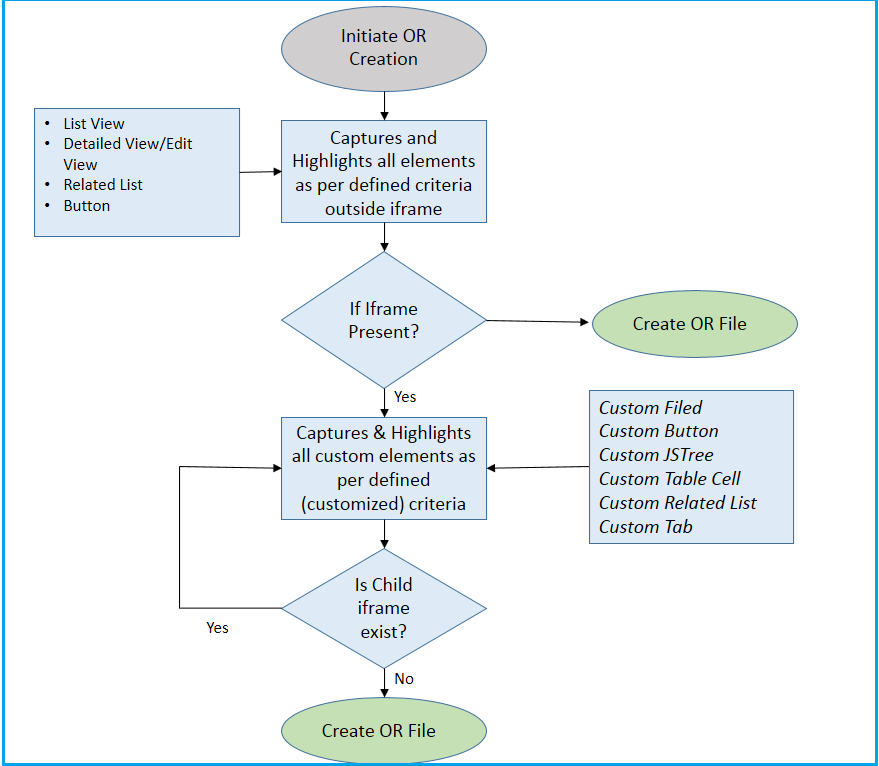


* NEXT-GEN Project will be imported without any errors.

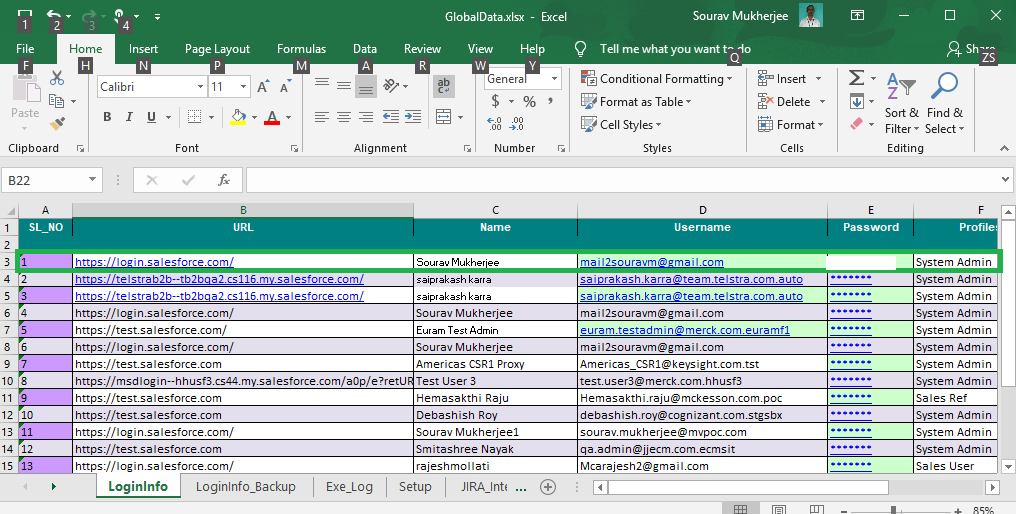


## Creating Object Repository

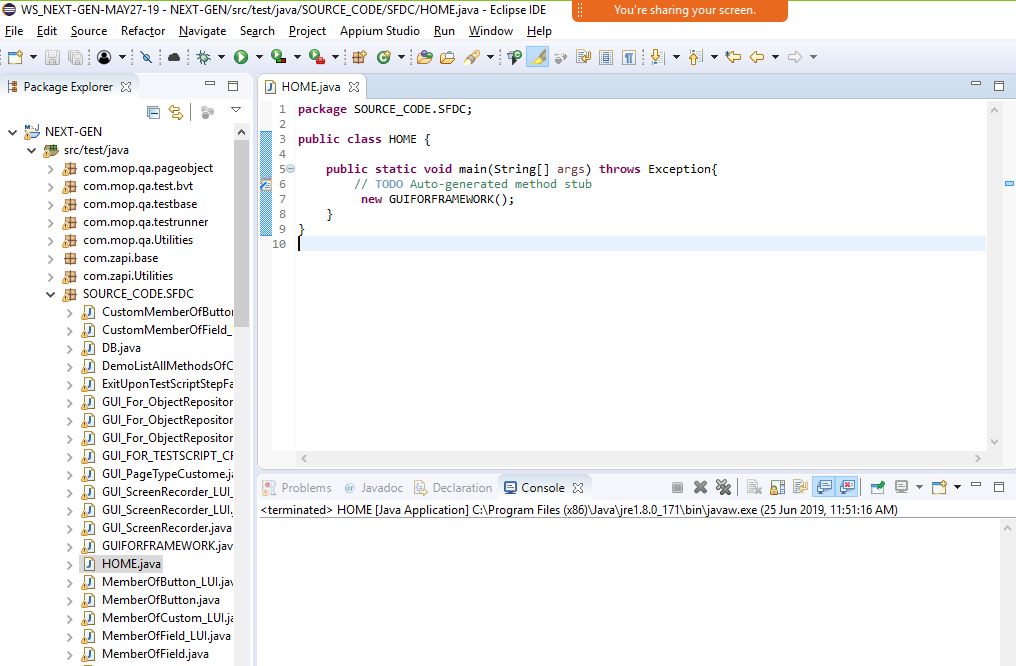
This Framework has been enhanced to support element identification within iframe and outside iframe based on certain criteria. Below flow diagram explains how this utility functions:



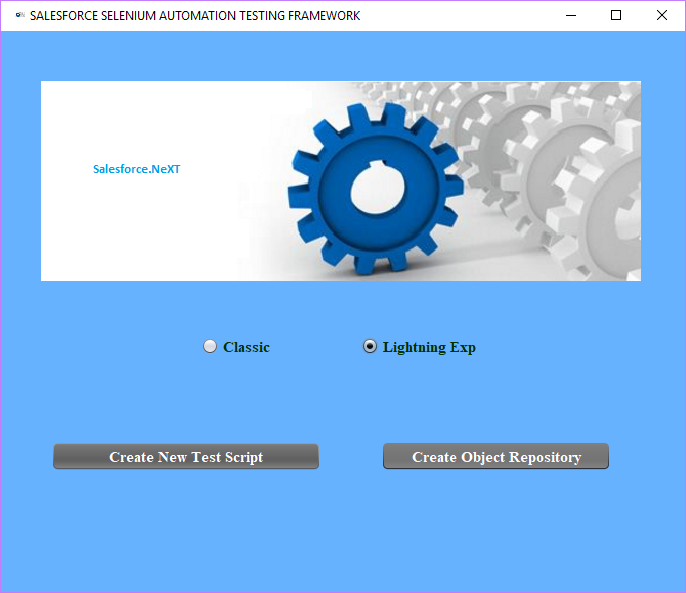
* + To create Object Repository, first login credentials are added in GlobalData.xlsx



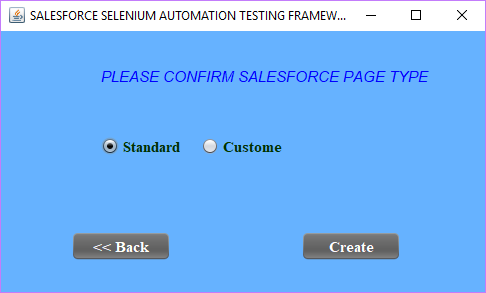
* + Run HOME.java file that is present inside SOURCE\_CODE.SFDC package.



* + A new window opened. The new window contains options to select the version of the Salesforce (Classic or Lightning Exp.)

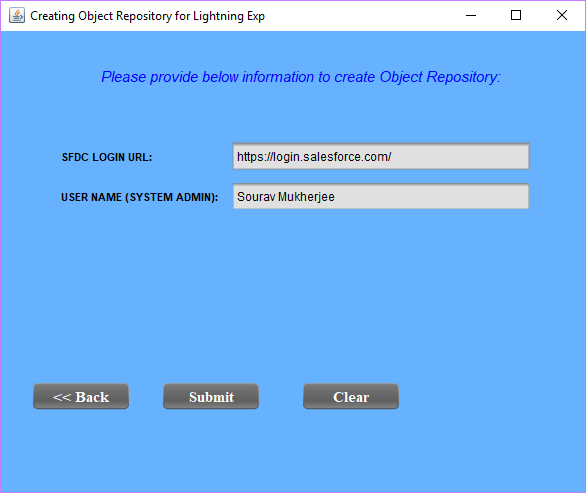


* + Select the version of the Salesforce Application and Click on ‘Create Object Repository’ button.



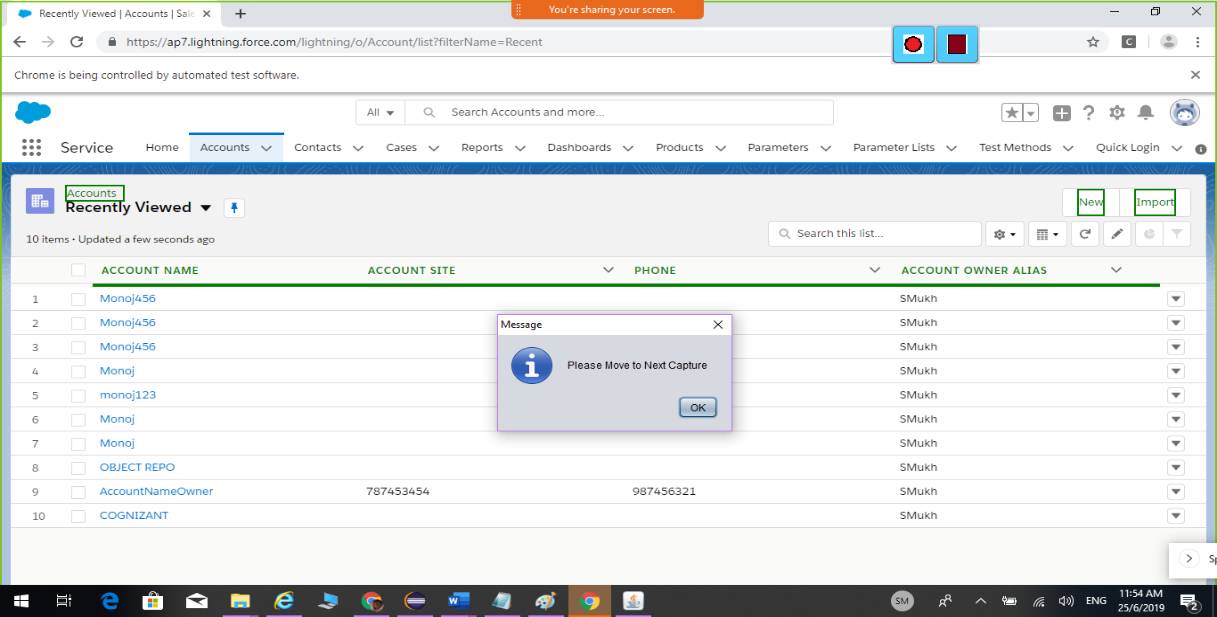
* + Another window displayed to select the page type for which Object Repository will be created.

Select the page type and click on ‘Create’.



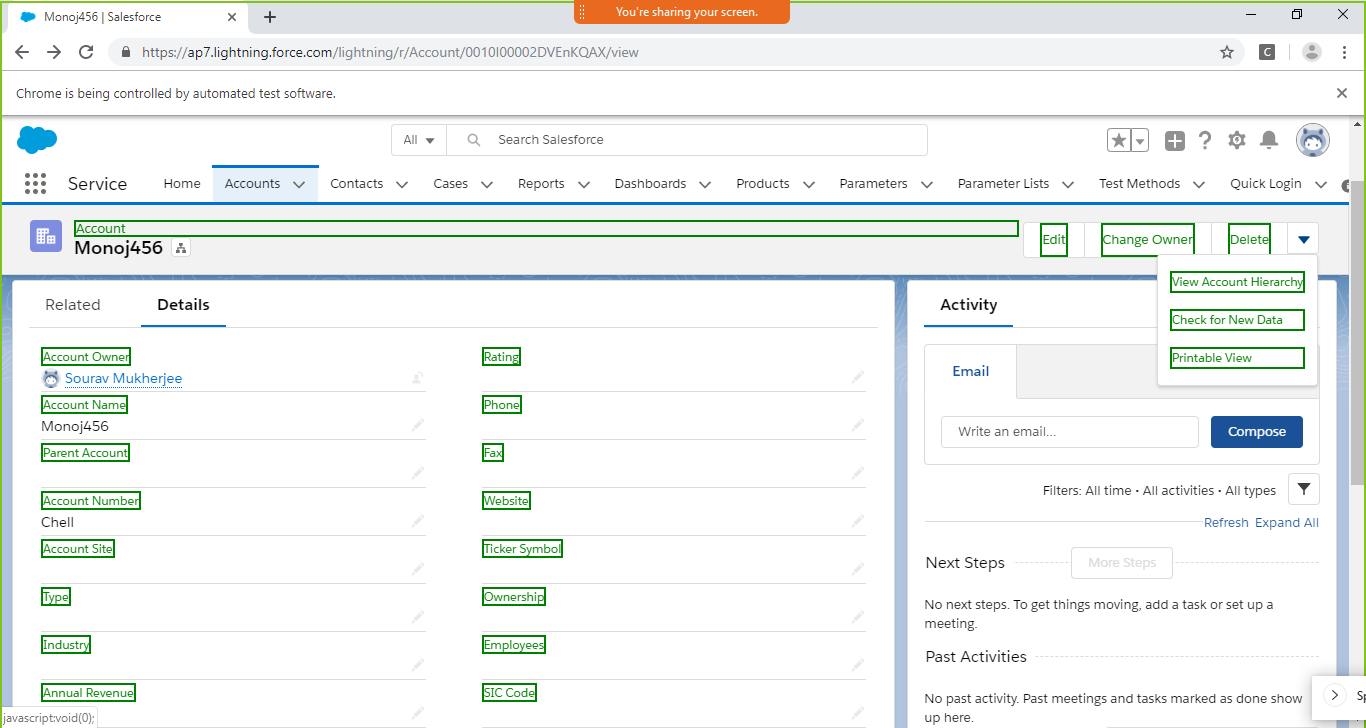
* + Verify once the login credential that is displayed in the next window then click on ‘Submit’ if the credential is valid.
  + User logged into the Salesforce application. ‘Start Recording’ and ‘Stop Recording’ buttons are displayed.
  + Navigate to the page (ex-Accounts page) for which Object Repository needs to be created and click on ‘Start Recording’ button.

Please find below example of Capturing List View Page Elements:



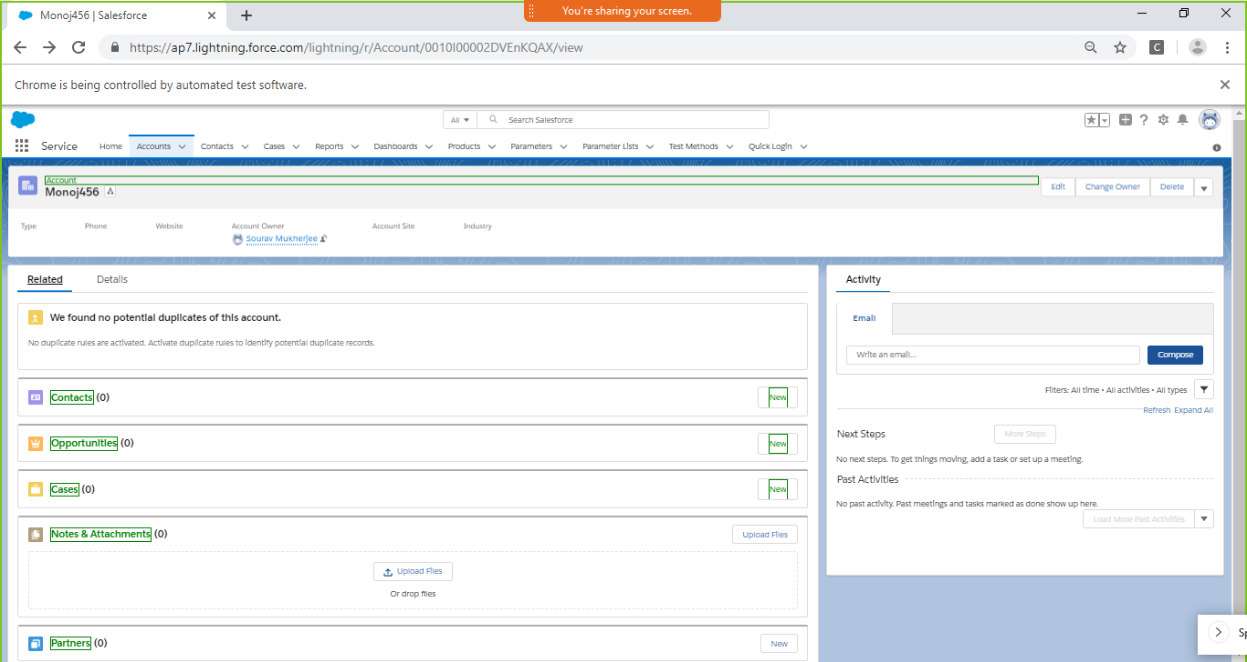
* + Click on Specific record (ex-Account Name) to open Detail view page.

Please find below example of Capturing Detail View Page Elements as highlighted below:



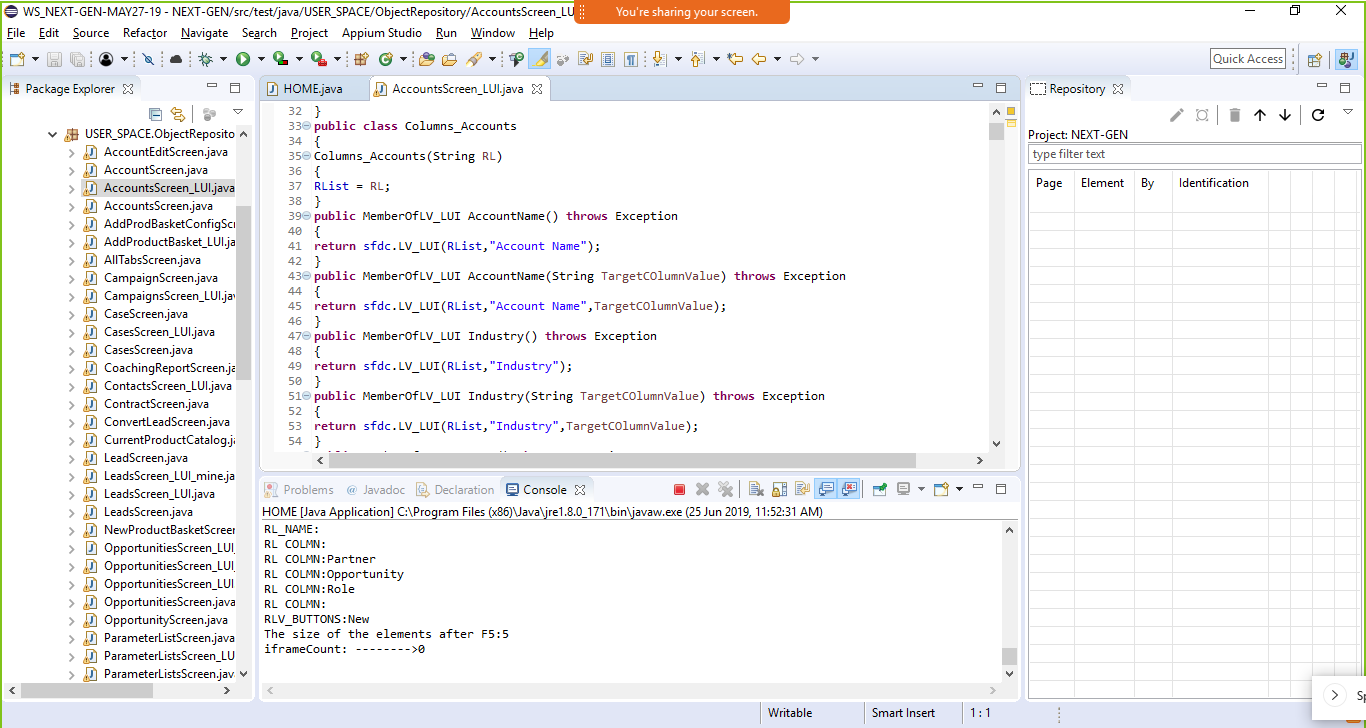
* + Salesforce DotNext Framework automatically captures all Related lists and navigates inside each Related List and captures Columns present in each Related List.

Below in an example of Capturing Related List Element:



* + Once recoding is done Click on ‘Stop Recording’ button. Refresh the project once and check for the Object Repository file in USER\_SPACE.ObjectRepository package.

Below in an example of Object Repository file.



## Creating Automation Scripts

Once object repository is created for a specific screen, it can be used from test script preparation file to implement specific user action. Primarily each step will have three/four layers of automation selection of object and function name as applicable. This has been explained in below figure.

Comment is provided above each step.

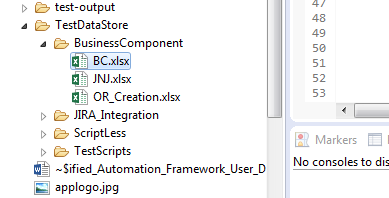
## 

## Creating Business Components

This Framework provides support to add Application specific Business Components and use them in test scripts.

* + We can keep the data related to Business Components in a separate excel file.

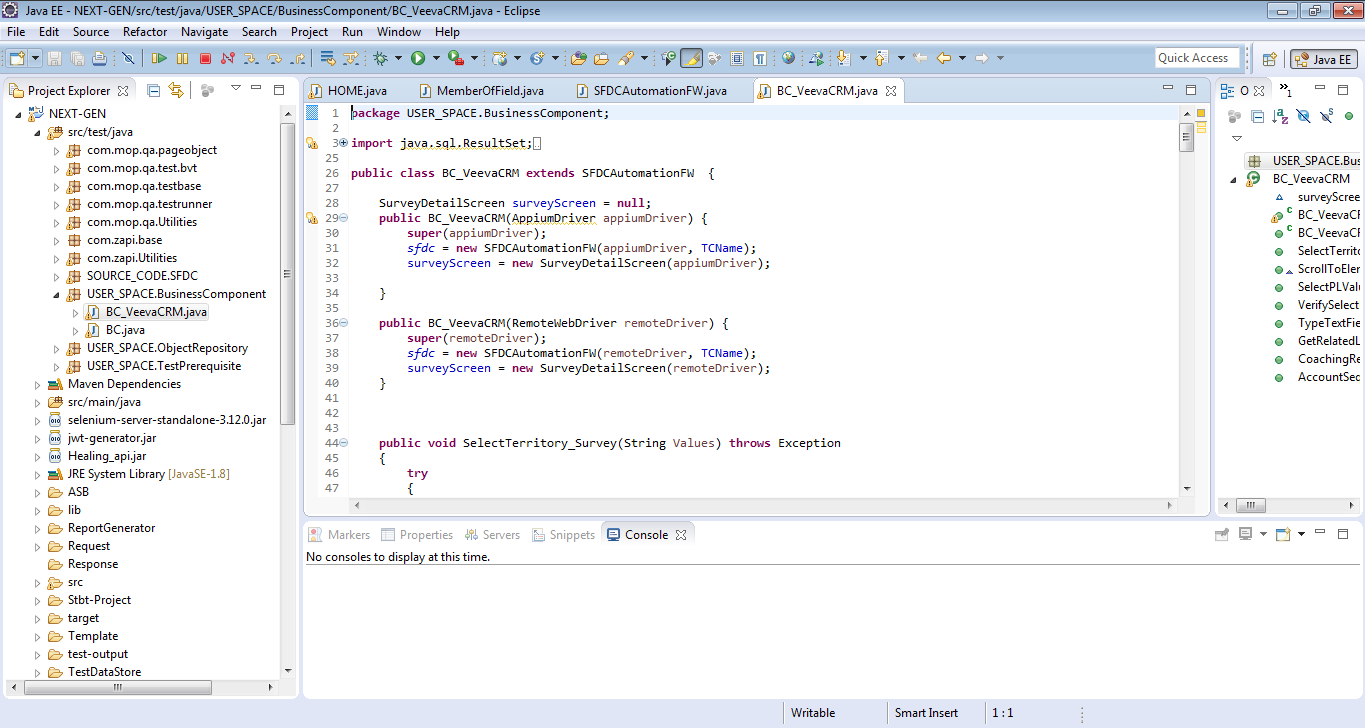
Below screenshot contains the path where the data file is present.



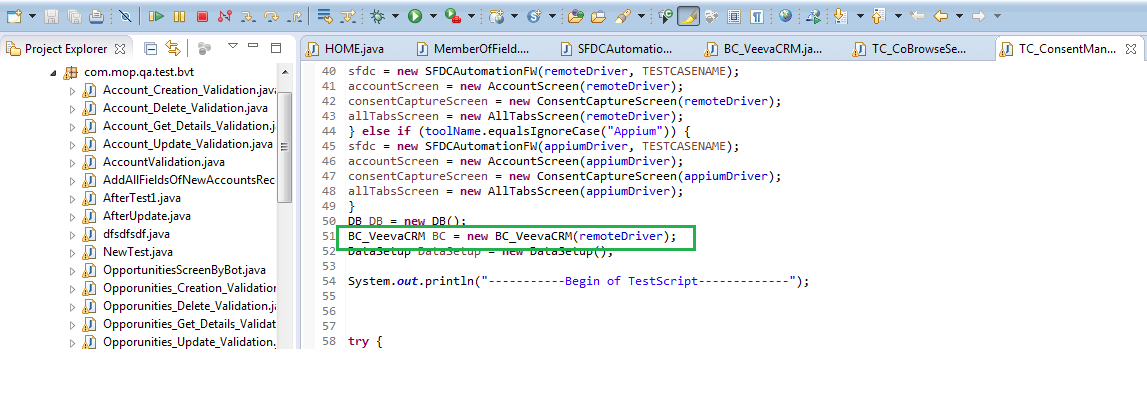
* + Place all Business component java files inside USER\_SPACE.BusinessComponent

package.

Below is an example of a Business Component file.



* + To use the Business Component in the test script, create an instance of the Business Component java file in the test script. Below image shows the way-



## List of Framework Reusable library

|  |  |  |
| --- | --- | --- |
| ***SFDC Element Category*** | ***Reusable Functions*** | ***What it does in SFDC application?*** |
|
| ***Fields / Sections*** | ***ClickONViewOnlyLinkValue*** | *Clicks on the hyperlink present beside a field in View Details page* |
| ***Fields / Sections*** | ***Click*** | *Clicks on a particular field value in the Edit page* |
| ***Fields / Sections*** | ***Type*** | *Type any value in textbox, textarea and also in textbox having lookup field* |
| ***Fields / Sections*** | ***VerifyFieldErrorMsgOnEditPage*** | *Verifies the field level error message in the edit page* |
| ***Fields / Sections*** | ***SelectPL*** | *Selects pick list field value by visible text* |
| ***Fields / Sections*** | ***SelectPLValueByIndex*** | *Selects the picklist value by index where index starts from zero (0)* |
| ***Fields / Sections*** | ***MultiSelectAdd*** | *Selects multiple values in the multi-select picklist field and adds to the Chosen list.* |
| ***Fields / Sections*** | ***MultiSelectRemove*** | *Removes multiple values from chosen list to Available list of multi-select pick list field.* |
| ***Fields / Sections*** | ***MultiSelectAddAll*** | *Selects and Adds all the available values of multi-select pick list field from Available List to Chosen List* |
| ***Fields / Sections*** | ***MultiSelectRemoveAll*** | *Selects and Removes all the available values of multi-select pick list field from Chosen List to Available List.* |
| ***Fields / Sections*** | ***VerifyPLDefaultValue*** | *Verify selected/default value displayed in the pick list field and sends the message to the Log* |
| ***Fields / Sections*** | ***GetPLDefaultValue*** | *Read the displayed value in the picklist field when the page is opened in Edit mode and returns the value.* |
| ***Fields / Sections*** | ***VerifyPLValue*** | *Verify pick list field values from Edit page and sends the message to the Log* |
| ***Fields / Sections*** | ***VerifyMPLAvailable*** | *Verify the values displayed in Available list of a Multi-Select pick list field in Edit page* |
| ***Fields / Sections*** | ***VerifyMPLChosen*** | *Verify the values displayed in Chosen list of a Multi-Select pick list field in Edit page* |
| ***Fields / Sections*** | ***VerifyViewOnlyValueEquals*** | *Verify the value displayed in a field in view only page and send the message to logs* |
| ***Fields / Sections*** | ***GetViewOnlyValue*** | *Read the value of a field from view details page* |
| ***Fields / Sections*** | ***VerifyViewOnlyValueContains*** | *Verify if the field value contains supplied substring* |
| ***Fields / Sections*** | ***VerifyViewOnlyValueDoesNotContain*** | *Verifies that the field value does not contain the supplied string* |
| ***Fields / Sections*** | ***VerifyViewOnlyValueStartsWith*** | *Verify if the field value starts with supplied substring* |
| ***Fields / Sections*** | ***VerifyViewOnlyValueEndsWith*** | *Verify if the field value ends with supplied substring* |
| ***Fields / Sections*** | ***VerifyEditFieldValue*** | *Verify if the field(text, textarea) value is equal to the supplied string in the displayed edit page* |
| ***Fields / Sections*** | ***GetEditFieldValue*** | *Reads the field (text,textarea) value displayed in the edit page and returns the value* |
| ***Fields / Sections*** | ***VerifyChkBoxValue*** | *Verify if the checkbox field is checked or not checked in Edit as well as View only page* |
| ***Fields / Sections*** | ***CheckBoxSelect*** | *Checks the checkbox field in Edit page* |
| ***Fields / Sections*** | ***PrecedingCheckBoxSelect*** | *Checks the checkbox field in Edit page in case checkbox element displays before the field label.* |
| ***Fields / Sections*** | ***PrecedingCheckBoxDeSelect*** | *UnChecks the checkbox field in Edit page in case checkbox element displays before the field label.* |
| ***Fields / Sections*** | ***CheckBoxDeSelect*** | *UnChecks the checkbox field in Edit page* |
| ***Fields / Sections*** | ***SelectFromLookup*** | *Selects the value from SFDC OOB lookup field. If the text box field is editable then it types the value in that field then clicks on the lookup icon and clicks on the hyperlink displayed in search lookup. In case the text box field is read only then directly clicks on the lookup icon and searches the expected value in the search lookup window and clicks on hyperlink. It also sets the focus to the parent window* |
| ***Fields / Sections*** | ***WaitForElement*** | *Waits for specified time for the field to be displayed in the UI* |
| ***Fields / Sections*** | ***LinkClick*** | *Clicks on partial link displayed anywhere in the UI* |
| ***Fields / Sections*** | ***IsDisplayed*** | *Verify if a field label is present in the View Details page* |
| ***Related List*** | ***Click*** | *Clicks on the displayed hyperlink value in the related list present under a "TargetColumn" where "KeyColumn" = "KeyColumnValue"* |
| ***Related List*** | ***VerifyValue*** | *Verifies if the Related List Value under the "TargetColumn" is equal to "ExpectedValueInTargetColumn" where "KeyColumn" = "KeyColumnValue".* |
| ***Related List*** | ***VerifyValueContains*** | *Verifies if the Related List Value under the "TargetColumn" contains "ExpectedValueInTargetColumn" where "KeyColumn" = "KeyColumnValue".* |
| ***Related List*** | ***VerifyCheckBoxValue*** | *Verifies if the check box value in the related list under the "TargetColumn" is "CheckedORNotChecked" where "KeyColumn" = "KeyColumnValue".* |
| ***Related List*** | ***GetValue*** | *Reads the value present in a related list undet the column "TargetColumn" where "KeyColumn" = "KeyColumnValue" and returns the same value* |
| ***Related List*** | ***GetValueByRowIndex*** | *Reads the value under a specified column "ColumnName" in specified "RowIndex" in a related list and returns the value on Success, returns blank value on failure* |
| ***Related List*** | ***VerifyValueByRowIndex*** | *Verifies if the value under a column "ColumnName" of a related list is equal to "ExpectedValue" where related list item index is equal to "RowIndex"* |
| ***Related List*** | ***VerifyValueContainsByRowIndex*** | *Verifies if the value under a column "ColumnName" is equal to "ExpectedValue" for the Line Item index "RowIndex"* |
| ***Related List*** | ***IsDisplayed*** | *Verifies if the supplied related list is displayed in the UI or not as per parameter passed in YesORNo* |
| ***Related List*** | ***CalculateCountofRelatedList*** | *Calculates the total number of related list item* |
| ***Link*** | ***Click*** | *Clicks on the supplied hyperlink with Exact match* |
| ***Link*** | ***IsDisplayed*** | *Verifies if the hyperlink is present in the UI with Exact match* |
| ***Link*** | ***Click\_Partial*** | *Clicks on the supplied hyperlink with partial match* |
| ***Link*** | ***ClickByIndex\_Partial*** | *Clicks by index on the hyperlink displyed in the UI with  partial match. Use this when there are more than one hyperlink with the same name`* |
| ***Link*** | ***ClickByIndex*** | *Clicks by index on the hyperlink displyed in the UI with  exact match. Use this when there are more than one hyperlink  with the same name* |
| ***ListView*** | ***Click*** | *Clicks on the hyperlink displayed under a column of SFDC OOB list Views. For example, All Accounts, Todays Leads so on.* |
| ***ListView*** | ***VerifyValue*** | *This function verifies the value having hyperlink under a particular column from the list view. If the vlaue is not available in the first list it navigates till the end of the list.* |
| ***Tab*** | ***Click*** | *Clicks on the Tab Name displayed on SFDC menu* |
| ***Tab*** | ***IsDisplayed*** | *Verified if a TabName is displayed* |
| ***Button*** | ***Click*** | *Clicks on Salesforce OOB button* |
| ***Button*** | ***IsDisplayed*** | *Checks if the SFDC OOB button is displayed in the UI* |
| ***Button*** | ***VerifyIfDisplayed*** | *Verifies the existence of button in the UI and sends the message in the log* |
| ***General / Framework Level*** | ***OpenURL*** | *Launches the URL of the AUT in specified browser.* |
| ***General / Framework Level*** | ***LoginToSFDC*** | *Logs in to SFDC as a specified nickname of the user. It reads the user name and password from the "Logininfo" sheet of GlobalData.xls file* |
| ***General / Framework Level*** | ***LoginAsUser*** | *login to Salesforce with single sign on feature.* |
| ***General / Framework Level*** | ***GlobalSearch*** | *Searches any sfdc record from SFDC GlobalSearch option* |
| ***General / Framework Level*** | ***SideBarSearch*** | *Searches any sfdc record from SFDC Sidebar Search option* |
| ***General / Framework Level*** | ***ClickONDetailsTab*** | *Clicking on Details tab against the record.* |
| ***General / Framework Level*** | ***AddToXLLogs*** | *Adds the message to excel logs along with Pass/Fail message* |
| ***General / Framework Level*** | ***GetWebDriver*** | *Returns the WebDriver instance* |
| ***General / Framework Level*** | ***GetScreenShot*** | *Captures screen shot of the WebDriver window* |
| ***General / Framework Level*** | ***GetPageLevelErrorMessage*** | *Reads the entire text of the error message displayed on the edit page and returns the same on success and returns blank value on failure* |
| ***General / Framework Level*** | ***VerifyPageLevelErrorMessage*** | *Verifies if the text message/error message displayed on the page level error message contains the partial text specified in the parameter* |
| ***General / Framework Level*** | ***SelectParentWindow*** | *Select the parent when there is only one window opened.This is very useful when you opened multiple window like popup and upon selection of the value in the lookup the popup gets closed. And now you want to switch back to the parent window.* |
| ***General / Framework Level*** | ***SelectPopupWindow*** | *Selects popup window. This is best in use when there are just two open window including main window. If you have more than two window then this function will not work. Please use SelectWindow function in such scenario* |
| ***General / Framework Level*** | ***SelectWindow*** | *Selects the window opened by Selenium Webdriver* |
| ***General / Framework Level*** | ***CloseWindow*** | *Whenever you call this function CloseWindow(), You must call SelectWindow() function after this function call to continue working on the next window, otherwise the focus to the next workable window will be lost.* |
| ***General / Framework Level*** | ***SelectFrame*** | *Selects the Frame by partial text of the title in a window opened by WebDriver* |
| ***General / Framework Level*** | ***SelectiFrame*** | *Selects the iFrame by partial text of the title in a window opened by WebDriver* |
| ***General / Framework Level*** | ***GetQueueMember*** | *Copies the user name present inside the Queue as per requested row index* |
| ***General / Framework Level*** | ***AlertClickYes*** | *Clicks on Yes button on the Alert opened by WebDriver* |
| ***General / Framework Level*** | ***AlertClickNo*** | *Clicks on No button on the Alert opened by WebDriver* |
| ***General / Framework Level*** | ***VerifyAlertMessage*** | *Verifies the text message displayed in the standard alert message* |
| ***General / Framework Level*** | ***ClickOnGoToListLink*** | *Clicks on the Go to list hyperlink which is present under a particular related list.* |
| ***General / Framework Level*** | ***CreateAUTRepository*** | *Creates the repository file for an object. This function identifies all the field labels, section, fields under each sections, related lists and its columns, SFDC buttons. The view details page must be displayed to the respective SFDC record.* |
| ***General / Framework Level*** | ***GetCurrentURL*** | *Copies the Current URL of the WebDriver browser window, on Failure it returns blank value* |
| ***General / Framework Level*** | ***OpenSFDCRecordAsperURL*** | *Opens page in already opened browser as per supplied URL* |
| ***General / Framework Level*** | ***GetCurrentDateTimeStamp*** | *Returns ehe date value in String type as per format ddMMyyHHmmss* |
| ***General / Framework Level*** | ***SelectFromDateLookup*** | *select any specific date by year,month and date from salesforce date lookup* |
| ***General / Framework Level*** | ***SendEmailAttachingTestLog*** | *Sends an email to certain set of users with the attachment of test log excel file* |

# UAF 2.0 – Execution Flow

## Referring to the architecture diagram of UAF, the execution flow is explained as follows

## Add the Test Classes in test.bvt package and add corresponding page class file in page Object package.

## 

## Update the TestRunner.xls file with the following details:

## a) Enter a user specified name for the Test case in the testname column

## b) Choose Test class name in the Test class column

## c) Select YES for the test cases that has to be executed in the Execution column

## d) Choose any one execution type (Parallel / Sequential) in the Execution Type column

## e) Select any one Locality (Hub / Grid) from the Locality column

## f) Choose toolname (Appium / Selenium) from the ToolName column

## 

## If the Execution Type is Parallel

## a) Choose the number of instances to run in parallel in the No\_of\_Instances column

## 

## 

## 4. If the Execution Type is Sequential, then by default one is selected for the No\_of\_Instances column

## 

## If the Toolname is selenium

## a) Choose any browser (chrome / firefox / Internet Explorer) from the Browser column

## b) The Browser column’s drop down options are based on the number of instances

## c) If the Locality is Hub, Specify the path of ChromeDriver or FirefoxDriver or IEDriver accordingly in the data.properties file.

## 

## 

## If the Toolname is Appium,

## a) Choose Application type (web / native) from the AppType column

## b) Select the device platform (Android / ios) from the PlatformName column

## c) Select UDID of the device from the UDID column

## d) The UDID column’s drop down options are based on the number of instances

## 

## If the tool is Appium & Windows

## a) Specify the path of node.exe

## b) Specify the path of Appium.js

## c) Specify the Application properties in data.properties file

## Application Path

## Application Package Name (for Android)

## Application Activity Name (for Android)

## 

## 

## 8. If the tool is Appium & iOS

## a) Specify the path of node.exe

## b) Specify the path of Appium.js

## c) Specify the Application properties in data.properties file

## Application Path (Bundle ID)

## 

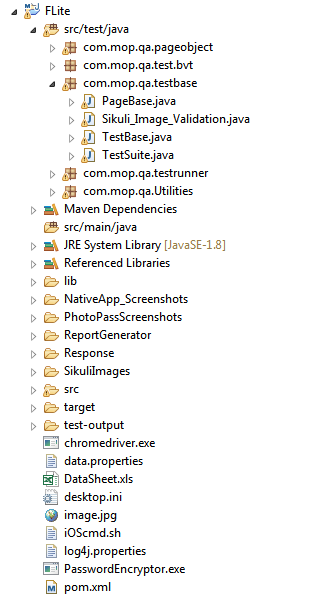
## 9. If the Locality is Grid, choose the remote url of the node machine

## 

## 10. Now run the project from TestRunner.java class. On execution completion, a consolidated report will be generated in Report generator folder.

# UAF 2.0 – Folder Structure

A sample folder structure for the framework is depicted below

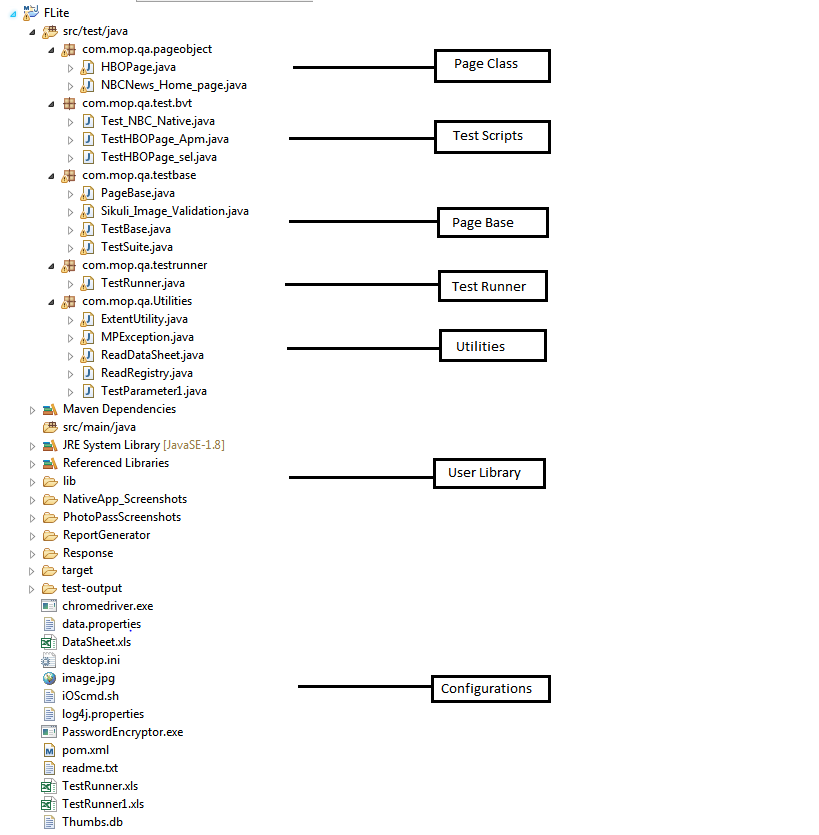


***UAF 1.2 Folder Structure***

The contents of the different folders are explained below:

|  |  |
| --- | --- |
| **Folder** | **Contents** |
| Src | Contains test scripts, page class and page base structured in page object model. |
| Lib | Contains the support libraries |
| ReportGenerator | Contains styles, image and css files for output report. |
| data.properties\* | Where you can do all the application settings. |
| DataSheet.xls\* | Where you can provide inputs to your test cases. |
| TestRunner.java\* | First file to trigger the framework |

**Note\***: User modification is required in these files.



***Folder Structure***

The contents of the different folders are explained below:

|  |  |
| --- | --- |
| **Folder** | **Contents** |
| PageBase | Where all the common methods for Selenium and Appium are coded which can be reused from any pageclass. |

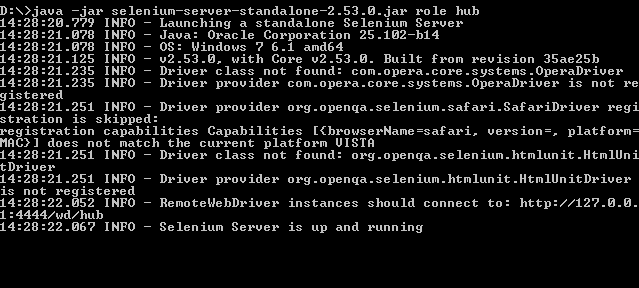
**Note\***: User modification is required in these files.

# Grid Execution

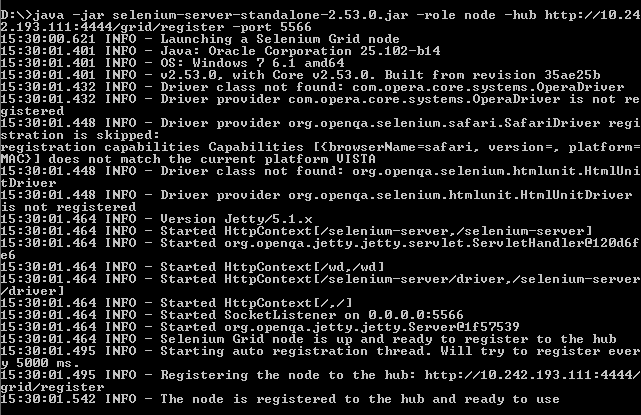
Grid execution allows you to run your tests on different machines against different browsers in parallel/sequential. That is, running multiple tests at the same time against different machines running different browsers and operating systems. Essentially, F-lite Grid support distributed test execution. It allows for running your tests in a distributed test execution environment.

## 7.1 In Selenium:

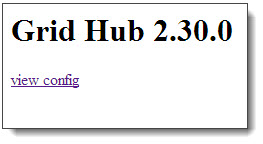
* A grid consists of a single hub, and one or more nodes. Both are started using the selenium-server.
* The hub receives a test to be executed along with information on which browser and ‘platform’ (i.e. WINDOWS) where the test should be run. It ‘knows’ the configuration of each node that has been ‘registered’ to the hub. Using this information it selects an available node that has the requested browser-platform combination. Once a node has been selected, Selenium commands initiated by the test are sent to the hub, which passes them to the node assigned to that test. The node runs the browser, and executes the Selenium commands within that browser against the application under test.
* Selenium Grid Installation Steps:
  + - Download the Selenium Server
    - You can place the Selenium Server .jar file anywhere in your Hard Drive. But for ease use place it on the D drive of both Hub and Node Machine. After doing this, you are now done installing Selenium Grid. The following steps will launch the hub and the node.
    - **Launching Hub:** Go to Hub Machine. Using the command prompt, navigate to the root of Hub Machine's - D drive, because that is the directory where we placed the Selenium Server.
    - On the command prompt, type **java -jar selenium-server-standalone-2.53.0.jar -role** hub
    - The hub should successfully be launched. Your command prompt should look similar to the image below.



* + - **Launching Node:** Now that the hub is already set up, we are going to launch a node. Go to Node Machine and launch a command prompt there.
    - Navigate to the root of Drive D and type the command **java –jar selenium-server-standalone-2.53.0.jar –role node –hub** [**http://10.242193.108:4444/grid/register**](http://10.242193.108:4444/grid/register) **-port 5566**. We used the IP address 10.242.193.108 of the hub machine. We also used port 5566 though you may choose any free port number you desire.
    - The command prompt should be similar to the image below.



* + - Grid is configured and ready to run a test remotely in node Machine.
* A way to verify whether the hub is running by using a browser. Selenium Grid, by default, uses Hub Machine port 4444 for its web interface. Simply open up a browser and go to<http://localhost:4444/grid/console>

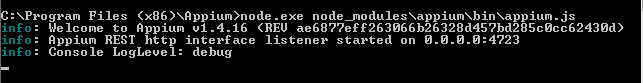
[](http://cdn.guru99.com/images/view_hub_from_browser.jpg)

* **Also, you can check if Node Machine can access the hub's web interface by** navigating to <http://10.242.193.108:4444/grid/console> where **hub machine IP address is 10.242.193.108**

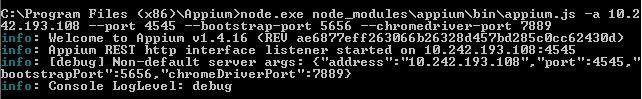


## In Appium (Android &iOS):

* Appium hub is created directly by running the TestRunner.java class. We can also manually create the hub by typing the following command in the command prompt. Before creating the appium server or node you must know location of the node.exe and appium.js
* For 64bit windows:
* Node path: C:\Program Files (x86)\Appium\node.exe
* Appium.js path: C:\Program Files (x86)\Appium\node\_modules\appium\bin\appium.js
* For 32bit windows:
* Node path: C:\Program Files\Appium\node.exe
* Appium.js path: C:\Program Files\Appium\node\_modules\appium\bin\appium.js
* **For iOS**:
* Node path:
* Appium.js path:
* We can also manually create the hub by typing the following command in the command prompt. Navigate to: C:\Program Files (x86)\Appium directory and type the following command.
* Node.exe node\_modules\appium\bin\appium.js
* Your Command Prompt will look like this



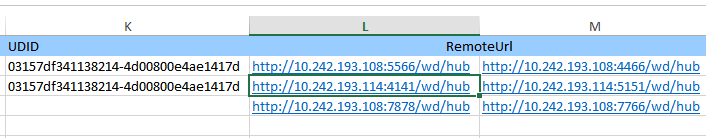
* **Launching Appium Node:** Open command prompt in the node machine where the mobile device is connected. Navigate to: C:\Program Files (x86)\Appium directory and type the following command.
  + - * Node.exe node\_modules\appium\bin\appium.js –a 10.242.193.108 --port 4554 –bootstrap-port 7878 –chromedriver-port 5666
* Where the IP address 10.242.193.108 refers to the node machine where the mobile device is connected and all the ports value are user-defined .After executing the above command Your Command Prompt will look like this:



**Note**: Both in selenium and appium the hub machine refers to the machine where the F-lite framework resides or it refers to the machine the TestRunner.java class is triggered.

Finally after setting up the Grid environment, the user should add the URL of the selenium and appium node into the RemoteUrl column of the TestRunner.xls sheet for Selenium and Appium the Remote Url would be like <http://$ipaddressofthenode$:port/wd/hub>

E.g.: <http://10.242.193.108:5566/wd/hub>



# 8. Parallel Execution

Running tests in parallel is the secret Sauce for accelerating our development process and creating a continuous integration/continuous delivery pipeline. Our F-Lite framework contains an extensive collection of Appium and Selenium scripts for website and mobile application testing that allows running tests in parallel.

Parallel execution in F-lite allows a particular testcase to run in different mobile devices (in case of appium) and in different browsers (in case of selenium) in parallel, based upon the number of instances given by the user in the TestRunner.xls.

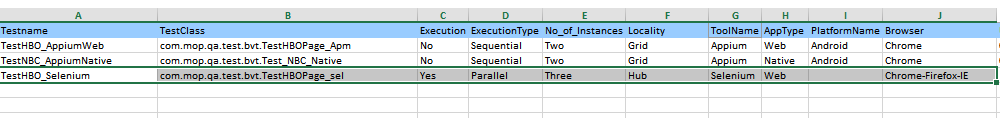
**Execution Flow**:

1. Modify the TestRunner.xls with the following details for the parallel execution.

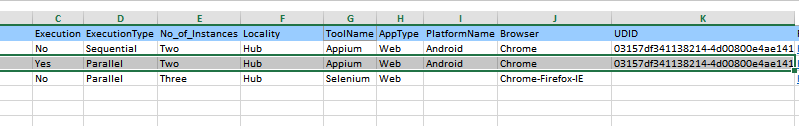
* Select the Execution column as **YES** for the testcase to be executed.
* Make the Execution Type as **PARALLEL** and specify the number of test to run in parallel in Number\_of\_Instances column.

1. If the Locality is HUB then :

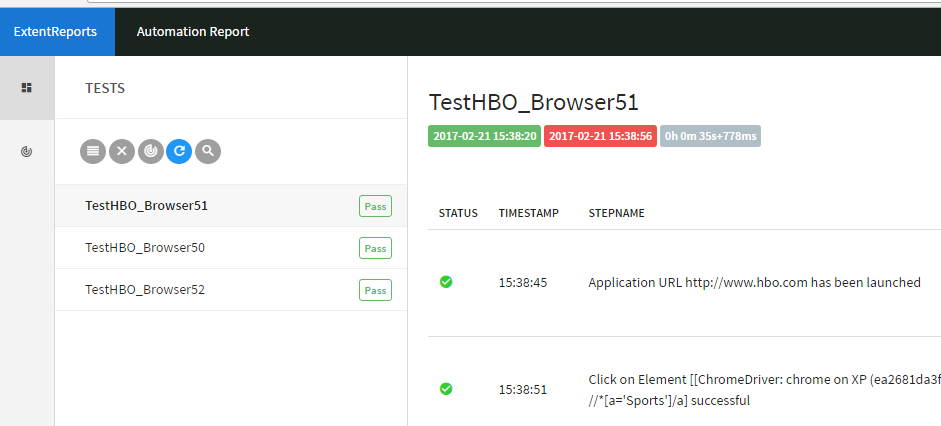
* If the tool name is Selenium select the corresponding browsers



* If the tool name is Appium select the corresponding apptype, PlatformName, browser (if apptype is web) and the UDID of the respective devices.



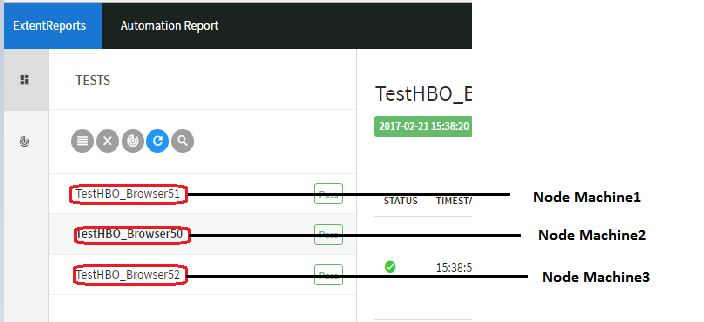
1. Now run the TestRunner.java class and the Corresponding Testcase will be executed.
2. F-LITE framework generates report accordingly without any overlap.



1. If the Locality is GRID, the only change from the Hub execution is to select the Remote URL of the node machines in TestRunner.xls



1. Now the selected test case will run parallel in the specified node machines and the report will be generated in the host or server machine.



# Video Automation Solution

UAF has the capabilities for validating all the Video/Image use cases on Web, Mobile and Connected devices.

* 1. Video Testing- Overview

Video Testing is meant for testing the video content. Main areas we focus on video testing are:

* Advertisement (Pre-roll / Mid-roll/C3 Ads/Non C3  ads)
* Video Player Controls (Play/Pause/Resume/Replay/Volume)
* CC (Closed Captioning)
* Video Testing in different OS/Device combinations
* Live Stream Videos which works with Silverlight/Flash
  1. Video Testing – Web/Mobile

We are achieving Video validations in Web and Mobile applications by leveraging the OpenCV API’s. There are Video validation reusable methods in PageBase.java specifically for Video Automation. Using these methods, the user can validate most of the video testing use cases like, Motion detection, Closed Caption verifications, Advertisement validation, matching an image in a video, extracting texts from video and validating other player controls.

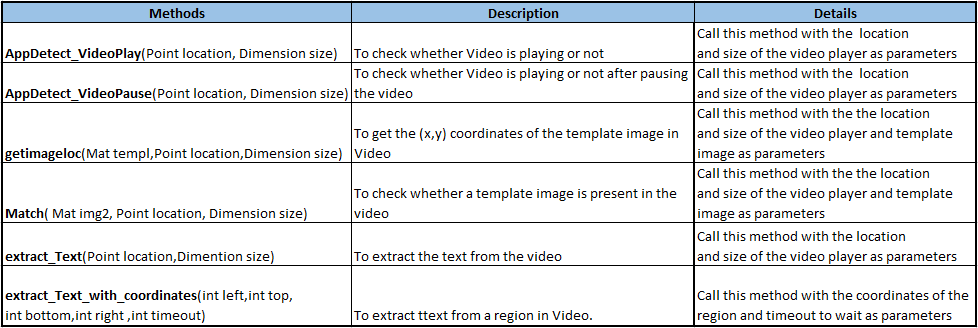
**Steps to set up OpenCV**

* Download and set up OpenCV with eclipse by following the steps from below link:

<http://docs.opencv.org/2.4/doc/tutorials/introduction/java_eclipse/java_eclipse.html>

* After importing the UAF framework, delete the native library “Opencv” in the framework and add the native library which is created in the above step.

The common methods that can be used for video validations are as follows:



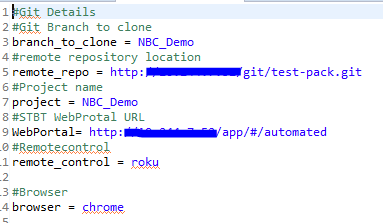
* 1. Video Testing – Connected devices

OTT (Over-The-Top) testing is another area in which the video is Geo restricted. So to test the OTT devices, the actual video content should be available in the remote location. We are using STB Tester ONE as an engine to test the OTT devices such as Set-Top-Boxes, Smart TV, Apple TV etc. We can test the device by connecting it directly to STB-Tester ONE.

The STBT- Tester One Hardware exposes a web Interface using which the user can control the connected device from anywhere in this world.

* + 1. Set Data.properties

Data.properties is the global settings file where you have to specify the new project name, Git server repository path, the branch you want to check out from Git server and driver path.



**branch\_to\_clone** - This property defines which branch of STB-T project you want to clone.

**remote\_repo** - Defines the STB-T remote repository URL.

**Project** – The project name you want to create. If the branch already exists with specified name, that branch will be checkout or else new branch will be created with specified project name.

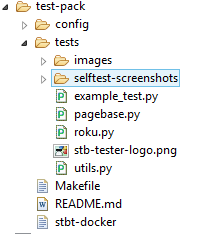
* + 1. Run CloneSetup.bat file

This is a One Time set up. Before starting a new project you have to run ‘ cloneSetup.bat ‘ file.

It will clone the specific branch of the project folder from the Git server location. After that it will checkout to the branch which is mentioned in Data.properties file as ‘project’. If the branch name what we have mentioned as ‘project’ in Data.properties already exists, it will simply clone that branch or else it will check out to the new branch with the name specified. So after running this jar file, a new folder with project files will be created inside the framework with name “Stbt-Project “.

* + 1. Create New Test Cases

After running the ‘cloneSetup.bat‘ file, the STBT Python project folder will be cloned to your workspace. You can create new test cases inside the project folder.

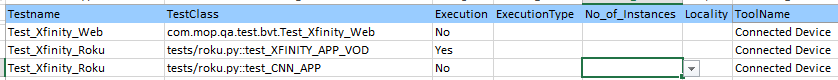


Test cases are Python functions stored in Git repository under tests/\*.py. The function name must begin with test\_ (for example ‘test\_that\_pressing\_EPG\_opens\_the\_guide’). The Framework provides a Python API for supporting the Scripting. You can make use of the **Functions** provided by the Automation Framework, to create new test cases.

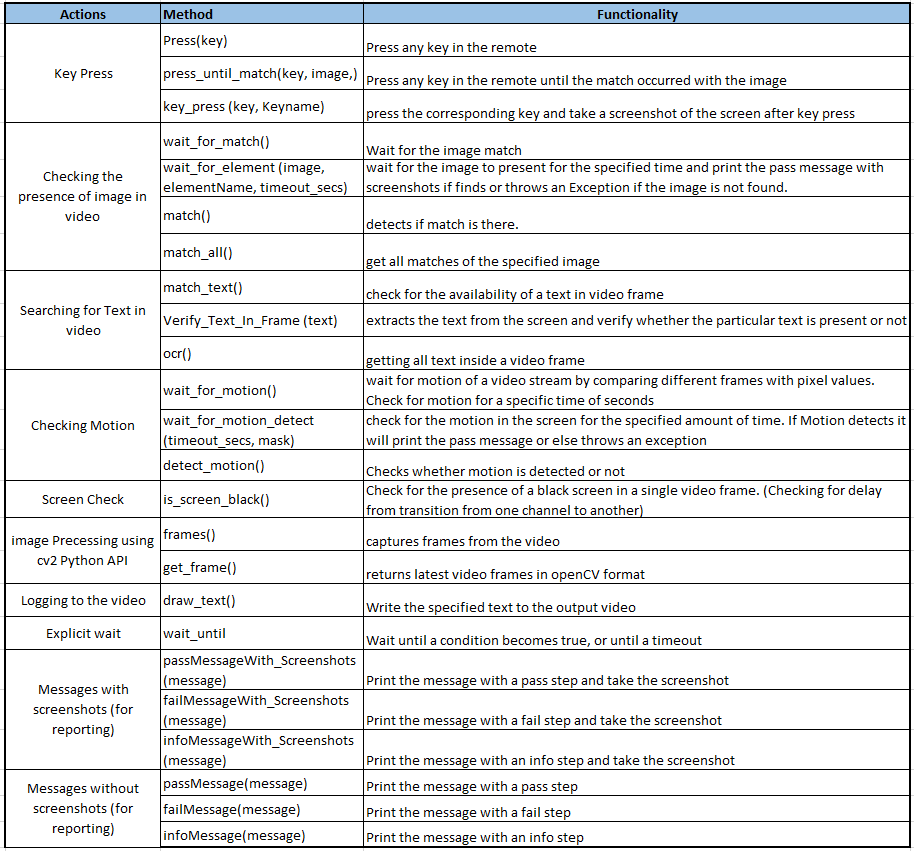
The Framework make use of images in the screen to communicate to device and for comparison purposes. So you have to provide each supporting images in the ‘images’ folder in project folder.

To Take Screenshots:

* Press the *Screenshot* button underneath the live video view in the STB-Tester ONE user interface.
* Save the screenshot into your [test-pack](https://stb-tester.com/manual/getting-started#test-pack) Git repository under /tests/images/.
* Open the image in any image editor and crop the image so that it only contains the part that you want to match in your test case.
  + 1. Set TestRunner.xls

TestRunner.xls is actually the Test Manger in which we give the Test case names with Python file name and the execution status as ‘Yes’ or ’No’. Also we have to select the ToolName as ‘Connected devices’. Only the Test case names with execution status as ‘Yes’ will be executed.

* + 1. Python Reusable Methods

Below is the list of python reusable methods available in the framework**:**

# Contacts

CMT Tech Team ([CMTSATCore@cognizant.com](mailto:CMTSATCore@cognizant.com))

# Change Log

|  |  |  |
| --- | --- | --- |
| Version Number | Changes made | Prepared by |
| V 1.1 | Initial version | CMT Tech Team |
| V 1.2 | Extent report  Parallel and Grid Execution | CMT Tech Team |
| V2.0 | Video automation solution | CMT Tech Team |