Prolifics
Global Technology Solutions Provider

ARSIN FRAMEWORK SELENESE COMMANDS

Malleshwar Goud Cheruku,
Test Automation Project Manager
IV&V Email: mcheruku@arsin.com
(o): +91-40-39991999 Extn. 1207
Mobile: +91-9642775888

Contents

Q) Why we are Using Arsin Framework?

Ans:-

1) Report Generation

2) Simple commands which will be used instead of the complex webdriver commands.
3) Commands for reading data from Excel are included.

Q) Is Arsin Framework is sufficient for Automation?

Ans: It will be useful for almost 80% of any automation scripts. But, there would be special requirements based on the projects in that case Selenium webdriver classes and methods are must.

 # The following lines of code used to set up the test environment and the oneTimeSetUp method is common for all the examples given in this document:

```
class Example
{
ArsinSeleniumAPI  oASelFW = null;
//ArsinFrameWorkAPI is a class and oASelFW is a instance.

public void oneTimeSetUp(String prjName, String testEnvironment,String
instanceName, String sauceUser, String moduleName,String testSetName)
throws InterruptedException
 {
                String[] environment = new
ArsinSeleniumAPI().getEnvironment(testEnvironment,
this.getClass().getName());
                String os = environment[0];
                String browser = environment[1];
                String testCasename = this.getClass().getSimpleName();
                System.out.println(os+"---"+browser);
                oASelFW = new ArsinSeleniumAPI(prjName, testCasename,
browser, os,instanceName, sauceUser, moduleName, testSetName);
```

```
            String Url = http://www.ksrtc.in/;
            //you can give any url that you want to test
            oASelFW.startSelenium(url);
            //opening the ksrtc page to automate in this example
        }

}
```

The arguments for oneTimeSetUp method can be passed from xml file or can pass from main method.

#Note:

In this document we only included the part of the code as examples for understanding. So, please place the example lines inside of a method.

#Syntax:

```
class  <ClassName>
{
        ArsinSeleniumAPI oASelFW = null;
        @Parameters({ "prjName", "testEnvironment", "instanceName",
        "sauceUser","moduleName", "testSetName" })
                @BeforeClass
        //copy the above oneTimeSetUpMethod code
        public void oneTimeSetUpMethod(arg1, arg2, ....,arg6)
        {
                //code

        }
        @Test
        public void <exampleMethod()>
        {

                //example code given under the commands Details
        }
}
```

// we are using TestNG and parameters send using xml file. So, no main.

1. Command Summary

eturn Type	Command
String	assertAlert Asserts (validates) JavaScript alert message with given expected message string or fail if there were no alerts.
String	assertConfirmation Asserts (validates) JavaScript confirmation dialog with given expected message string and also accepts confirmation dialog (i.e., clicks 'ok' by default) or fail if there were no alerts.
String	assertSubText Asserts whether the given string appeared (fully or partially) in the text of WebElement.
String	assertText Gets the text of an element. Verifies that the specified Text pattern appears somewhere on the rendered page shown to the user.
String	assertTitle Asserts the current page title with given expected title.
String	assertValue Asserts (validates) value of a targeted WebElement (input field or option tag) with given expected value.
String	captureEntirePageScreenshot Saves the entire contents of the current window canvas to a PNG file.

String	click Clicks on a link, button, checkbox or radio button.
String	clickSpecifiedLink
String	clickSpecifiedLinkNWait Clicks on a specified link and waits for a new page to appear.
String	clickAndWait Waits for element to be clickable and then click and again waits for new page to load.
String	clickAndWaitForElementPresent Click on link or button and wait for the next element to be visible.
String	closeBrowser Closes the current browser window.
String	deleteAllVisibleCookies Delete all cookies i.e., (driver.deleteAllCookies – webdriver).
String	eClickAndWait Clicks on a link, button, checkbox or radio button and waits for a new page to load (or) for a default time of 60 seconds. (The WebElement is specified in the Excel input data sheet.)

String[]	eOpen Open the URL in test browser (The URL is specified in Excel input data sheet). (driver.get(Url) - Webdriver)
String	getXpathCount Returns the number of nodes those matches with the specified Xpath.
String	getSelectedLabel
String[]	getAllWindowNames Returns the titles of all windows that the browser knows about. (driver.getWindowHandles() - WebDriver)
String	getCookieByName Returns the value of the cookie with the specified name. or Throws an error if the cookie is not present. (Only the cookie names of the current website)

String	getSelectOptions Gets all option labels in the specified select drop-down.
String	getText Gets the text of an element (i.e. text of , , <a> etc.)
String	getTitle Gets the title of the current page.
String	getURL Gets the URL of the current page.
String	getValue Navigating to “previous” page. (clicking ‘←’ back on the browser)
String	goBack Navigating to “previous” page. (clicking ‘←’ back on the browser)
String	isChecked Determine whether a toggle-button (checkbox/radio) is selected or not.

String	isElementInVisible Determine whether the specified element is displayed or not on current page. (<webelement>.isDisplayed( ) – WebDriver)
String	isElementPresent Determine whether the specified element is present somewhere on the page. (<webelement>.isElementPresent( ) – WebDriver)
String	isSpecifiedOptionChecked Determine whether the specified toggle-button (checkbox/radio) is checked or not.
String	openURL Opens an URL in the test frame.
String	openWindow Opens a popup window (if a window with that URL isn't already open). (driver.navigate().to(URL) - Webdriver)
void	sendReport This method adds step details to array lists used for Reporting.
void	sendReportWithOutExit This method adds step details to array lists used for Reporting and it continues further even though the condition is Fails.

String	select Select an option from a drop-down using an option label.
String	selectWindow Selects a popup window; once a popup window has been selected, all commands go to that window.
String	selectFrame Selects a frame within the current window using index or locator of frame. (You may invoke this command multiple times to select nested frames.)
String	type Sets the value of an input field, as though you typed it in.
String	typeEmpty Sets the value of an input field to empty string.
String	typePassword Type the password in the text box and characters will not be displayed instead the '*' will be displayed in the text box.
String	verifySubText Verifies whether expected sub text appears in the actual text of the specified web element. (The subtext to be verified is specified in the Excel input data sheet.)

String	verifyText Verifies that the specified text pattern appears in the specified web element.
String	verifyTextPresent Verifies that the specified text pattern appears in the specified web element. (The text to be verified is specified in the Excel input data sheet.)
String	verifyValue Verifies the (whitespace-trimmed) value of an input field (or anything else with a value parameter). (The value to be verified is specified in the Excel input data sheet.)
String	verifyTitle Verifies the title of the current page. (The title to be verified is specified in the Excel input data sheet.)
String	verifyConfirmation Verifies the message of a JavaScript confirmation dialog generated during the previous action. And accepts the confirmation dialog.
String	verifyAlert Verifies the message of a JavaScript alert generated during the previous action, or fail if there were no alerts.

String	waitForAlertPresent Waits for the specified time in seconds for an alert message to appear on the screen.
String	waitForElementPresent Waits until specified element is present on the page. (If element is not present, waits for given time and exits).
String	waitForPageToLoad Waits for a new page to load and maximum wait time is 60 seconds.
String	waitForValue Waits for the specified time in seconds for a WebElement value (i.e., value of <input> <option> etc.)
String	windowMaximize To maximize browser window
String	waitForFrameAndSelect Waits for specified time and select Frame by index.
String	waitUntilElementVisible Waits until the specified element is visible on the page.

2. Command Details

assertAlert

Description:

Asserts (Validates) JavaScript alert message with the given "**expected Alert Message"** or fail if there were no alerts.

If the **"expected Alert Message"** is equals to **"actual Alert Message"** then
{
returns the **"actual Alert Message"**.
//PASS
}
Else
{
reports Error and **exits** .
//FAIL
}

Usage:

Two-arguments:

oASelFW.Effecta("**assertAlert**","ExpectedAlertMsg");

Returns:

JavaScript generated Alert Message Text. (If AssertAlert Pass)

Example:

assertAlert PASS case:

String expAlertMsg = "'From' place is required";
String alertMsg = oASelFW.effecta("**assertAlert**", expAlertMsg);

//Here the Expected alert message is → 'From' place required
System.out.println(alertMsg);

//**Output:** 'From' place required

assertAlert FAIL case:

String expAlertMsg = "'From' place"; // wrong expected message passing
String alertMsg = oASelFW.effecta("**assertAlert**", expAlertMsg);

//Here the Expected alert message is → 'From' place
System.out.println(alertMsg);

// **output** : Report step details method path:Displayed alert message: 'From' place is
required.Screenshot:D:\WORKSPACE\Results\Screenshots\Alerts_29-01-15 16:22:51_2.png

assertConfirmation

Description:

Asserts (Validates) JavaScript confirmation dialog box message with the given "**expected Confirmation Dialog box Message"** or fail if there were no alerts. And also by default accepts the Confirmation Box (i.e., same as clicking 'ok' manually)

Confirmation Box**. Accept**;
If the **"expected Confirmation Dialog box Message"** is equals to **"actual Confirmation Dialog box Message"** then
{
returns the **"actual Confirm Box Message"**.
//PASS

}

Else
{
reports Error and **Exits.**
//FAIL
}

Usage:

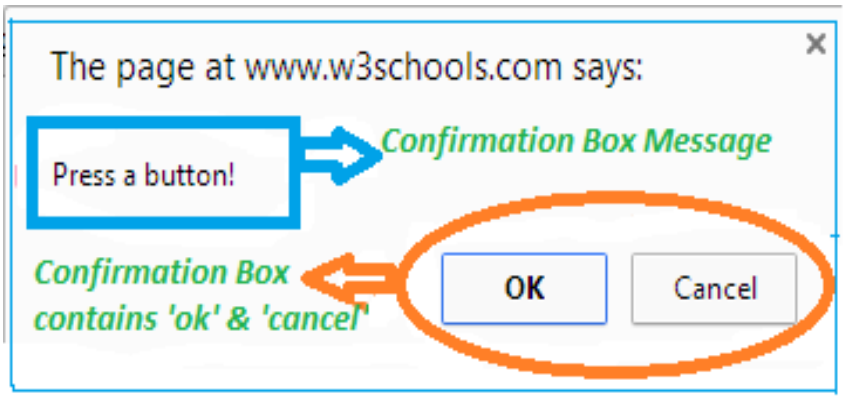
Two-Arguments:

oASelFW.Effecta("**assertConfirmation**","ExpectedConfirmationDiaglog BoxMessage");

Returns:

JavaScript generated ConfirmationMessage Text. (If AssertConfirmation Pass)

Example



assertConfirmation PASS case:

```
String expConfirmMsg = “Press a button!”;
String confirmMsg = oASelFW.effecta("assertConfirmation ",
expConfirmMsg);
// Here the Expected Confirmation box message is → 'Press a button!
System.out.println(confirmMsg);


//Output:   Press a button!
```

assertConfirmation FAIL case:

```
String expConfirmMsg = “Press a  Box”; // wrong expected message is passing
String confirmMsg = oASelFW.effecta("assertConfirmation ",
expConfirmMsg);
// Here the Expected Confirmation box message is → 'Press a Box!
System.out.println(confirmMsg);

// Output : Report step details method path: Displayed confirmation
message:  Press a
button!Screenshot:D:\WORKSPACE\Results\Screenshots\Alerts_29-01-
15 16:01:45_3.png
```

assertSubText

Description:

Asserts (validates) whether the given string appeared (fully or partially) in the text of targeted WebElement.

If the **“actual Text of targeted WebElement”** contains **“given sub Text”** then

```
{
        returns the “actual Text of Targeted WebElement”.
        //Pass
}
```

Else
{
reports Error and **Exits.**
//FAIL
}

Usage:

Three-Arguments:

oASelFW.Effecta("**assertSubText**", "target", "expectedSubText");

Parameters:
target → Locator of the targeted WebElement (any locating strategies i.e., id, name, xpath, css, link text etc.)
expectedSubText → A string that matches with WebElement text partially or fully.

Returns:
The Actual Text of targeted WebElement (if assertSubText Pass)

Example:
String target = "//div[1]/span/a";
String expectedSubText = "tutorial";
System.out.println(oASelFW.effecta("**assertSubText**", target, expectedSubText));
// Returns target element actual text

//**Output :** Selenium tutorials

Four-Arguments:

oASelFW.Effecta("**assertSubText**", "target", "expectedSubText", "ElementTile ToBeDisplayedInReport");

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id,

name, xpath, css, link text etc.)
expectedSubText → A string that matches with WebElement text partially or fully.
ElementTitleToBeDisplayedInReport → Title to be displayed for the element in the report.

Returns:
The Actual Text of targeted WebElement (if assertSubText Pass)

Example:
String target = "//div[1]/span/a";
String expectedSubText = "tutorial";
String elementTitleDisplayInReport = " first Tutorial Link";
System.out.println(oASelFW.effecta("**assertSubText**", target, expectedSubText,elementTitleDisplayInReport));
// Returns target element actual text

//**Output :** Selenium tutorials

assertText

Description:

Asserts (validates) whether the given string matches with the text of targeted WebElement.
If the **"givenText"** equals **"actual Text of targeted WebElement"**
then
{
returns the **"actual Text of Targeted WebElement"**.
//PASS
}
Else
{
reports Error and **Exits.**
//FAIL
}

Usage:

Three-Arguments:

oASelFW.Effecta("**assertText**", "target", "expectedText");

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)
expectedText → A string that exactly matches with the WebElement Text.

Returns:
The Actual Text of targeted WebElement (if assertText Pass)

Example:
String target = "//div[1]/span/a";
String expectedText = "Selenium tutorials";

System.out.println(oASelFW.effecta("**assertText**", target, expectedText));
// Returns target element actual text

//**Output :** Selenium tutorials

Four-Arguments:

oASelFW.Effecta("**assertText**", "target", "expectedText", "ElementTile ToBeDisplayedInReport");

Parameters:

target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)

expectedText → A string that exactly matches with the WebElement Text.

ElementTitleToBeDisplayedInReport → Title to be displayed for the element in the report.

Returns:
The Actual Text of targeted WebElement (if assertText Pass)

Example:
String target = "//div[1]/span/a";
String expectedText = "Selenium tutorials";
String elementTitleDisplayInReport = " first Tutorial Link";

System.out.println(oASelFW.effecta("**assertText**", target, expectedSubText,elementTitleDisplayInReport));
// Returns target element actual text

//**Output :** Selenium tutorials

assertTitle

Description:

Asserts the current page title with given expected title.

If the **"expected title"** equals **"actual Title of the current page"**
then
{
returns the **"actual Title of the current page"**.
//PASS

}
Else
{
reports Error and **Exits.**
//FAIL
}

Usage:

Two-Arguments:

oASelFW.Effecta("**assertTitle**", "expectedTitle");

Parameters:

expectedTitle → The expected title of the page.

Returns:
The Actual Title of current page(if assertTitle Pass)

Example:
String expectedTitle = "Selenium - Web Browser Automation";
System.out.println(oASelFW.effecta("**assertTitle**", expectedTitle));
// Returns Actual Title of the current Page

//**Output :** Selenium - Web Browser Automation

assertValue

Description:

Asserts (validates) value of a targeted WebElement (input field or option tag) with given expected value.

If the "**expected value"** equals **"actual value of targeted WebElement"** then
{
returns the **"actual Value of Targeted WebElement"**.
//PASS
}
Else
{
reports Error and **Exits.**
//FAIL
}

Usage:

Three-Arguments:

oASelFW.Effecta("**assertValue**", "target", "expectedValue");

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)
expectedValue→ A string that exactly matches with the WebElement Value.

Returns:
The Actual Value of targeted WebElement (input or option) (if assertValue Pass)

Example:
String target = "//select[1]/option[2]";
String expectedValue = "Hyderabad";
System.out.println(oASelFW.effecta("**assertValue**", target, expectedValue));
// Returns target element actual value

//**Output :** Hyderabad

Four-Arguments:

oASelFW.Effecta("**assertValue**", "target", "expectedValue", "ElementTile ToBeDisplayedInReport");

Parameters:

target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)
expectedValue→ A string that exactly matches with the WebElement Value.
ElementTitleToBeDisplayedInReport → Title to be displayed for the element in the report.

Returns:
The Actual Value of targeted WebElement (input or option) (if assertValue Pass)

Example:
String target = "//select[1]/option[2]";
String expectedValue = "Hyderabad";
elementTitleDisplayInReport = " CitySelected";
System.out.println(oASelFW.effecta("**assertValue**", target,
expectedSubText,elementTitleDisplayInReport));
// Returns target element actual value

//**Output :** Hyderabad

captureEntireScreenShot

Description:
Saves the entire content of the current window canvas to a PNG file.

Usage:
Three-arguments:

oASelFW.Effecta ("**captureEntirePageScreenshot**", "verification", "description");

Parameters:
verification → verification/validation text to be written to the report.
description → description to be appeared in the report

Returns:
The result string "PASS", if command executed successfully.

Example:
String verification = "Hyderabad";
String description = "Selecting city as Hyderabad";
System.out.println(oASelFW.Effecta ("**captureEntirePageScreenshot**", verification, description);

//Output: PASS

click

Description:

Clicks on a link, button, checkbox or radio button.

If "**target element found"** then
{
1. Click on element
2. Returns string **"PASS"**

}
Else
{
Reports **ElementNotFoundException** and **Exits**
}

Usage:

Two-arguments:

oASelFW.Effecta ("**click**","target");

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)

Returns:
The result string "PASS", if command executed successfully.

Example:
String target= "//button";
System.out.println(oASelFW.Effecta ("**click**", target));

//Output: PASS

Three-arguments:

oASelFW.Effecta(**"click"**,"target"," ElementTile ToBeDisplayedInReport");

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)
ElementTile ToBeDisplayedInReport → Title to be displayed for the element in the report.

Returns:
The result string "PASS", if command executed successfully.

Example:
String target= "//button";
String ElementTile ToBeDisplayedInReport = "signupbutton"
System.out.println(oASelFW.Effecta ("**click**", target, ElementTile ToBeDisplayedInReport));

//Output: PASS

clickSpecifiedLinkNWait

Description:

Clicks on a specified targeted link and waits for a new page to appear.

If "**target element found"** then
{
1. Click on link & Wait for new Page to appear (max 60 seconds timeout)
2. Returns string **"PASS"**

}

Else
{
Reports **ElementNotFoundException** and **Exits**
//FAIL
}

Usage:

Three-arguments:

oASelFW.Effecta(**"clickSpecifiedLinkNWait"**,"target",
" ElementTileToBeDisplayedInReport");

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)
ElementTitleToBeDisplayedInReport → Title to be displayed for the element in the report.

Returns:
The result string "PASS", if command executed successfully.

Example:
String target= "//button";
String ElementTileToBeDisplayedInReport = "signupbutton"
System.out.println(oASelFW.Effecta ("**clickSpecifiedLinkNWait**", target, ElementTileToBeDisplayedInReport));

//Output: PASS

clickAndWait

Description:

Waits for element to be clickable and then click and again waits for new page to load.
If "**target element found"** then
{

1. WebDriver waits until element to be clickable (max. wait 250 seconds)
2. Click the element
3. Wait for new page to load (max. wait 250 seconds)
4. Returns string **"PASS"**

}

Else
{
Reports **ElementNotFoundException** and **Exits**
}

Usage:

Two-arguments:

oASelFW.Effecta ("**clickAndWait**","target");

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)

Returns:
The result string "PASS", if command executed successfully.

Example:
// Examples tested on www.vijethagrocer.com
String target= "//*[@id="nav"]/li[3]/a/span";
System.out.println(oASelFW.Effecta ("**clickAndWait**", target));

//Output: PASS

Three-arguments:

oASelFW.Effecta(**"clickAndWait"**,"target","
ElementTitleToBeDisplayedInReport");

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)
ElementTitleToBeDisplayedInReport → Title to be displayed for the element in the report.

Returns:
The result string "PASS", if command executed successfully.

Example:
// Examples tested on www.vijethagrocer.com
String target= "//*[@id="nav"]/li[3]/a/span";
String ElementTitleToBeDisplayedInReport = "Bevarages"
System.out.println(oASelFW.Effecta ("**clickAndWait**", target, ElementTitleToBeDisplayedInReport));

//Output: PASS

clickAndWaitForElementPresent

Description:

Clicks the given targeted WebElement and Waits until visibility of resulted expected WebElement.
If "**target element found"** then
{
1. Click the targeted element
2. WebDriver waits until resulted Expected element to be visible (max. wait 250 seconds)
3. Returns string **"PASS"**
}
Else
{
Reports **ElementNotFoundException** and **Exits**
}

Usage:

Four-arguments:

oASelFW.Effecta(**"clickAndWaitForElementPresent"**,"target",
" ElementTitleToBeDisplayedInReport","resultedExpectedWebElement");

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)

ElementTitleToBeDisplayedInReport → Title to be displayed for the element in the report.
resultedExpectedWebElement → Locator of the resulted web element (any locating strategies i.e., id,name, xpath, css, link text etc.)

Returns:
The result string "PASS", if command executed successfully.

Example:
// Examples tested on www.vijethagrocer.com
String target= "//*[@id="nav"]/li[10]/a/span";
String ElementTitleToBeDisplayedInReport = "Frozen Meat";
resultedExpectedWebElement = "//a[@title=' VENKY BREST 2PC 430G']";
System.out.println(oASelFW.Effecta ("**clickAndWaitForElementPresent**", target, ElementTitleToBeDisplayedInReport, resultedExpectedWebElement));

//Output: PASS

Explanation: - First the targeted WebElement will be clicked and the webdriver waits until visibility of resultedExpectedWebElement and if that is visible then webdriver returns "PASS".

closeBrowser

Description:
Closes the current browser window.

Usage:
One-argument:

oASelFW.Effecta("**closeBrowser**");

Returns:
Returns the result string "PASS", if command executed successfully.

deleteAllVisibleCookies

Description:

Deletes all the cookies of the domain.

Usage:

One-argument:

oASelFW.Effecta("**deleteAllVisibleCookies**");

Returns:

Returns the result string "PASS", if command executed successfully.

Example:

oASelFW.Effecta("**deleteAllVisibleCookies**");

// if the current web page is let us say: www.flipcart.com then it deletes All cookies related to flipcart.

//Output: Pass

eClickAndWait

Description:

Clicks on a link, button, checkbox or radio button and waits for a new page to load (or) for a default time of 60 seconds. (The WebElement is specified in the Excel input data sheet.

If "**target element found"** then

{

1. Click the element
2. Wait for new page to load (max. wait 60 seconds)
3. Returns string **"PASS"**

}

Else
{
Reports **ElementNotFoundException** and **Exits**
}

Usage:

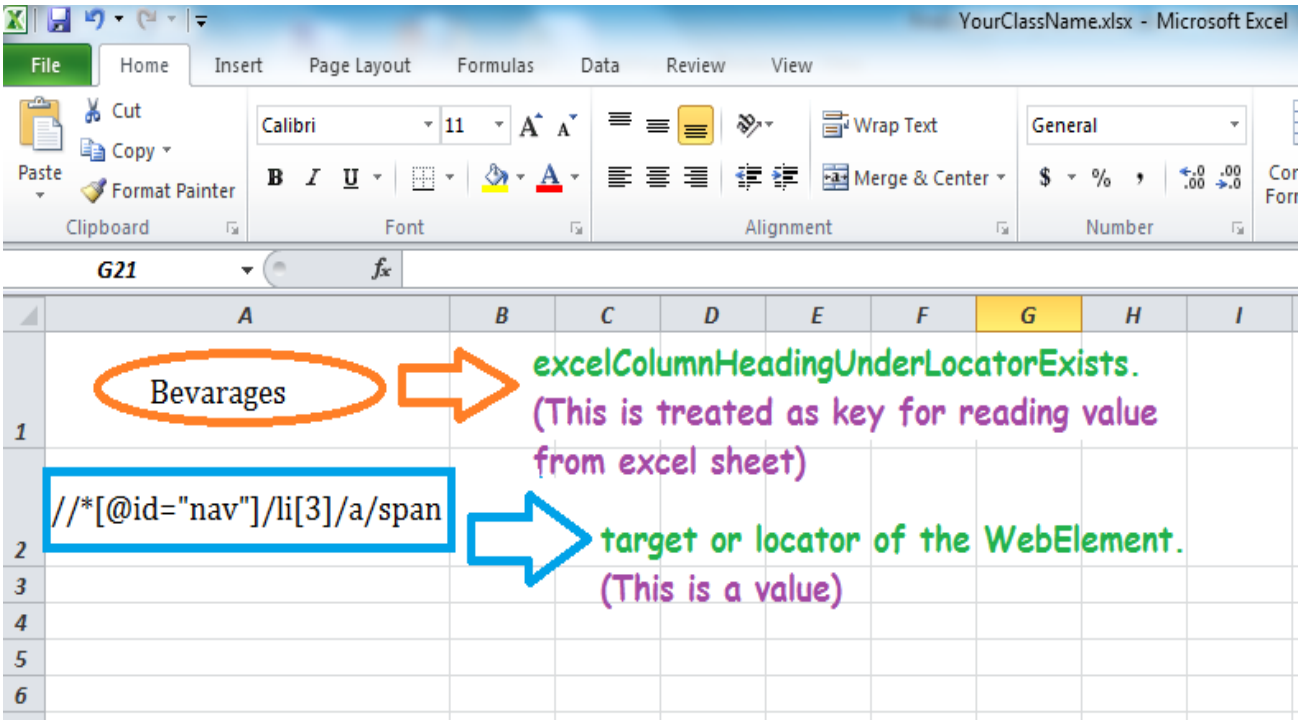
Two-arguments:

oASelFW.Effecta ("**eClickAndWait**","
excelColumnHeadingUnderLocatorExists");

Parameters:
excelColumnHeadingUnderLocatorExists → The header is treated as key and value is Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)

Returns:
The result string "PASS", if command executed successfully.

Example:



// Examples tested on www.vijethagrocer.com
String excelColumnHeadingUnderLocatorExists = “Bevarages”;
System.out.println(oASelFW.Effecta (“**eClickAndWait**”, excelColumnHeadingUnderLocatorExists));

//Output: PASS

Explanation:-
We will pass only key i.e., “Beverages” and internally the command **“eClickAandWait”** will take care of reading value and does its functionality.

eOpen

Description:

Opens the URL in test browser (The URL is specified in Excel input data sheet).

Usage:

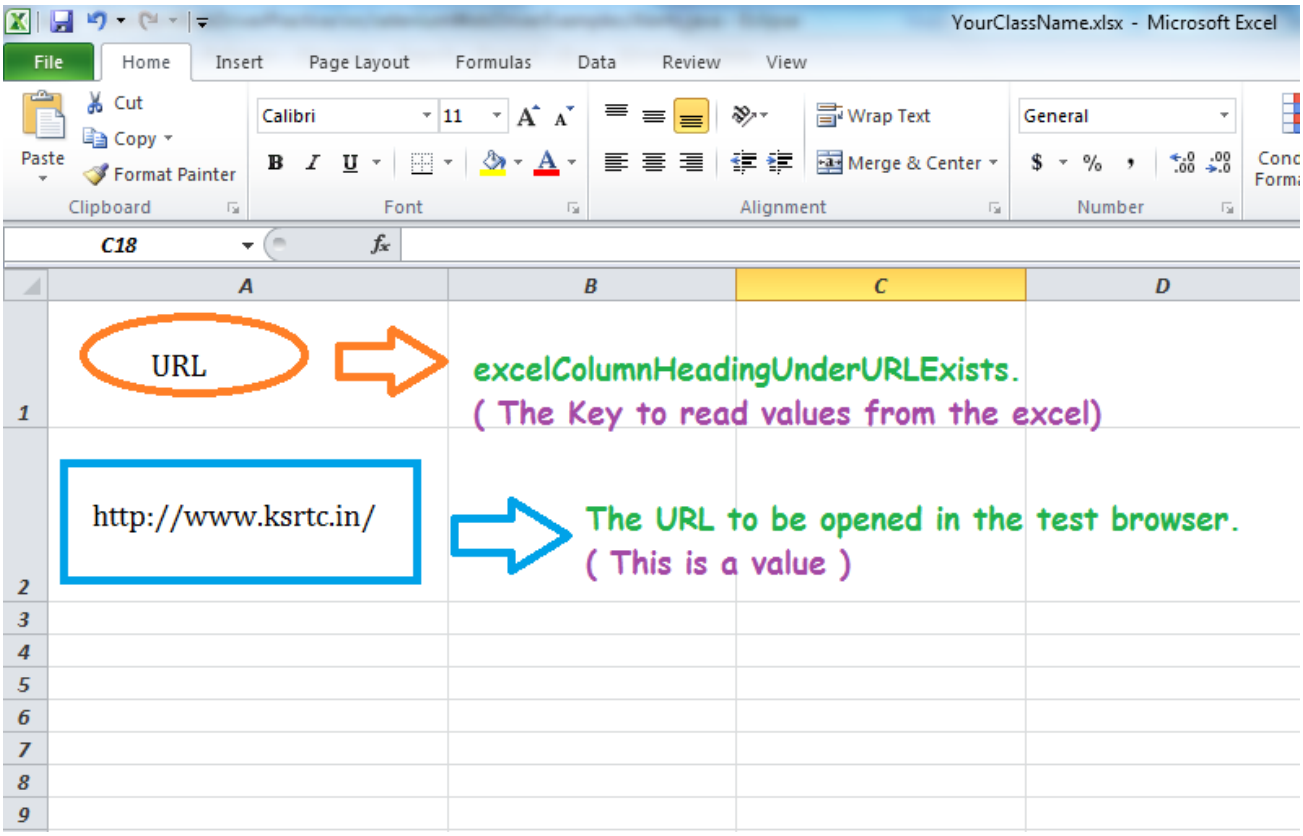
Two-arguments:

oASelFW.Effecta (“**eOpen**”,” excelColumnHeadingUnderURLExists”);

Parameters:
excelColumnHeadingUnderURLExists → The header is treated as key and value is Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)

Returns:
The result string “PASS”, if command executed successfully.

Example:



String excelColumnHeadingUnderURLExists = "URL";
System.out.println(oASelFW.Effecta ("**eOpen**", excelColumnHeadingUnderURLExists));

//Output: PASS

Explanation:-
We will pass only key i.e., "URL" and internally the command **"eOpen"** will take care of reading value and does its functionality.

getXpathCount

Description:

Returns the number of nodes those matches with the specified Xpath.

If **"specified Xpath matches"** then
{
Returns total number of WebElements (nodes) matched.
}
Else
{
Reports **ElementNotFoundException** and **Exits**
}

Usage:

Two-arguments:

oASelFW.Effecta ("**getXpathCount**","xpath");

Parameters:
xpath → Locator of the targeted web element specified using xpath.

Returns:
The total number of WebElements (nodes) matched.

Example:
String xpath= "//button";
System.out.println(oASelFW.Effecta ("**getXpathCount**", xpath));

//Output: 4 (The page contains 4 buttons, so it returns 4).

Three-arguments:

oASelFW.Effecta(**"click"**,"xpath"," ElementTile ToBeDisplayedInReport");

Parameters:
xpath → Locator of the targeted web element specified using xpath.

ElementTile ToBeDisplayedInReport → Title to be displayed for the element in the report.

Returns:
The total number of WebElements (nodes) matched.

Example:

String xpath= "//button";
String ElementTile ToBeDisplayedInReport = "The total no.of buttons";
System.out.println(oASelFW.Effecta ("**getXpathCount**", xpath, ElementTile ToBeDisplayedInReport));

//Output: 4 (The page contains 4 buttons, so it returns 4).

getAllWindowNames

Description:

Returns the titles of all windows that the browser knows about.

Usage:

One-argument:

oASelFW.Effecta ("**getAllWindowNames"**);

Returns:
The string array with titles of all windows.

Example:
String[] windowNames =oASelFW.Effecta ("**getAllWindowNames**");

```
for( int i=0; i<windowNames.length; i++)
{
        System.out.println(windowName[i]);
}
```

//Output:
Book Bus Ticket Online – KSRTC
W3Schools Online Web Tutorials
Selenium - Web Browser Automation

Explanation:- The above output is of titles of 3 websites that are opened currently.

getCookieByName

Description:

Returns the value of the cookie with the specified name or Throws an error if the cookie is not present. (Only the cookie names of the current website)

Usage:

Two-arguments:

oASelFW.Effecta ("**getCookieByName**","cookieName");

Parameters:
cookieName → cookie name under the current website.

Returns:
The value of Cookie name matched.

Example:

```
String  cookiName ="__utmt";
System.out.println(oASelFW.Effecta ("getCookieByName","cookieName"));
```

//Output: __utmt=1; expires=Tue, 03 Feb 2015 05:16:55 IST; path=/; domain=.ksrtc.in

getSelectOptions

Description:

Gets all option labels in the specified select drop-down.

If **"Specified target of select drop-down found"** then
{

1. Get all options of Select
2. Return string array which holds all options

}
Else
{
Reports **ElementNotFoundException** and **Exits**
}

Usage:

Two-Arguments:

oASelFW.Effecta("**getSelectOptions**", "selectLocator");

Parameters:
selectLocator → Locator of the select (any locating strategies i.e., id, name, xpath, css, link text etc.)

Returns:
Return string array which holds all options

```
String selectLocator= "//select[@id='month']";
System.out.println(oASelFW.Effecta ("getSelectOptions", selectLoactor));

//Output:
 January
 February
……
December (prints all 12 months)
```

getText

Description:

Gets the text of an element. (i.e., text of , , <a> etc.)
Same as "**getText**" in Selenium Webdriver.

If "**target element found"** then
{
Returns text of WebElement.
}
Else
{
Reports **ElementNotFoundException** and **Exits**
}

Usage:

Two-arguments:

oASelFW.Effecta ("**getText**","target");

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)

Returns:
The text of a WebElement.

Example:

```
// Examples tested on www.vijethagrocer.com
String  target= "//*[@id="nav"]/li[3]/a/span";
System.out.println(oASelFW.Effecta ("getText", target));
```

//Output: Bevarages

Three-arguments:

oASelFW.Effecta ("**getText**","target"," ElementTitleToBeDisplayedInReport");

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)
ElementTitleToBeDisplayedInReport → Title to be displayed for the element in the report.

Returns:
The text of a WebElement.

Example:
// Examples tested on www.vijethagrocer.com
String target= "//*[@id="nav"]/li[3]/a/span";
String ElementTitleToBeDisplayedInReport = "Main-menu -2 text";
System.out.println(oASelFW.Effecta ("**getText**", target, ElementTitleToBeDisplayedInReport));

//Output: Bevarages

getTitle

Description:
Gets the title of the current web page.

Usage:
One-argument:

oASelFW.Effecta ("**getTitle**");

Returns:
The title of the current web page.

Example:
// Examples tested on www.vijethagrocer.com
System.out.println(oASelFW.Effecta ("**getTitle**"));

//Output: Online Grocery in Hyderabad | Online Grocery Shopping | Vijetha

getURL

Description:
Gets the URL of the current page.

Usage:
One-argument:

oASelFW.Effecta ("**getURL**");

Returns:
The URL of the current web page.

Example:
// Examples tested on www.ksrtc.in
System.out.println(oASelFW.Effecta ("**getURL**"));

//Output: http://www.ksrtc.in/site/bus-services

getValue

Description:
Gets the (whitespace-trimmed) value of an input field (or any element with a value parameter).

If "**target element found"** then
{
Returns value of a WebElement.
}
Else
{
Reports **ElementNotFoundException** and **Exits**
}

Usage:

Two-arguments:

oASelFW.Effecta ("**getValue**","target");

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)

Returns:
The value of a WebElement.

Example:

// Examples tested on
https://hr.prolifics.com/employees/profile/personal/general

String target="/html/body/div[1]/div[12]/div/div[1]/div[1]/div[20]/input";
System.out.println(oASelFW.Effecta ("**getValue**", target));

//Output: IVV // (Department)

Three-arguments:

oASelFW.Effecta ("**getValue**","target","
ElementTitleToBeDisplayedInReport");

Parameters:

target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)
ElementTitleToBeDisplayedInReport → Title to be displayed for the element in the report.

Returns:
The value of a WebElement.

Example:
// Examples tested on
https://hr.prolifics.com/employees/profile/personal/general

String target="/html/body/div[1]/div[12]/div/div[1]/div[1]/div[20]/input";
String ElementTitleToBeDisplayedInReport = "Department";
System.out.println(oASelFW.Effecta ("**getValue**", target));

//Output: IVV // (Department)

goBack

Description:
Navigating to "previous" page. (clicking '←' back on the browser)

Usage:
One-argument:

oASelFW.Effecta ("**getBack**");

Returns:
The String "Pass".

isChecked

Description:

Determine whether a toggle-button (checkbox/radio) is selected or not

If "**target element found"** then
{
1. Checks whether target element selected or not.
2. Returns "true" , if selected or "false"

}
Else
{
Reports **ElementNotFoundException** and **Exits**
}

Usage:

Two-arguments:

oASelFW.Effecta ("**isChecked**","target");

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)

Returns:
String "true", if selected or "false".

Example:

// Examples tested on http://www.ksrtc.in/

String target="//input[@name='chkLadiesSeatOnly']";
System.out.println(oASelFW.Effecta ("**isChecked**", target));

//Output: false

Three-arguments:

oASelFW.Effecta ("**isChecked**","target","
ElementTitleToBeDisplayedInReport");

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)
ElementTitleToBeDisplayedInReport → Title to be displayed for the element in the report.

Returns:
String "true", if selected or "false".

Example:
// Examples tested on // Examples tested on http://www.ksrtc.in/

String target="//input[@name='chkLadiesSeatOnly']";
String ElementTitleToBeDisplayedInReport = "Department";
System.out.println(oASelFW.Effecta ("**isChecked**", target,
ElementTitleToBeDisplayedInReport));

//Output: false

isElementInVisible

Description:

Determine whether the specified element is displayed or not on current page.

If "**target element found"** then
{
    1. Waits for 3 seconds
    If **"target element displayable"**
    {
        Return "Element is Visible"
    }

```
        Else
        {
                Return "Element is not Visible"
        }
}
Else
{
         Reports ElementNotFoundException and Exits
}
```

Usage:

Three-arguments:

oASelFW.Effecta ("**isElementInVisible**","target","
ElementTitleToBeDisplayedInReport");

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)
ElementTitleToBeDisplayedInReport → Title to be displayed for the element in the report.

Returns:
The String "Element is Visible" if element is displayed or "Element is not visible"

Example:

```
// Examples tested on // Examples tested on  http://www.ksrtc.in/

String target="//select[@id=' country-code']";
String ElementTitleToBeDisplayedInReport = "country Code";
System.out.println(oASelFW.Effecta ("isElementInVisible", target,
ElementTitleToBeDisplayedInReport));


//Output:  false
```

isElementPresent

Description:

Determine whether the specified element is present somewhere on the page.

If "**target element Present"** then
{
Returns "true"
}
Else
{
Returns "false"
}

Usage:

Two-arguments:

oASelFW.Effecta ("**isElementPresent**","target")

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)

Returns:
The String "true" if element present, or "false" if element not present.

Example:

```
// Examples tested on // Examples tested on  http://www.ksrtc.in/

String target="//input[@name='chkLadiesSeatOnly']";
String ElementTitleToBeDisplayedInReport = "chexk box";
System.out.println(oASelFW.Effecta ("isElementPresent", target));

//Output:  true
```

openURL

Description:

Opens an URL in the test frame.

Usage:

Two-arguments:

oASelFW.Effecta ("**openURL**","URL")

Parameters:
URL → URL to be opened.

Returns:
The String "Pass".

Example:
// Examples tested on // Examples tested on http://www.ksrtc.in/

String URL=" http://www.ksrtc.in/";
System.out.println(oASelFW.Effecta ("**openURL**", URL));

//Output: Pass

openWindow

Description:

Opens a popup window (if a window with that URL isn't already open).

Usage:

Three-arguments:

oASelFW.Effecta ("**openWindow**","URL",
"ElementTitleToBeDisplayedInReport")

Parameters:
URL → URL to be opened.
ElementTitleToBeDisplayedInReport → Title to be displayed for the element in the report.

Returns:
The String "Pass".

Example:
// Examples tested on // Examples tested on http://www.ksrtc.in/

String URL=" http://www.ksrtc.in/site/bus-services";
ElementTitleToBeDisplayedInReport ="bus services";
System.out.println(oASelFW.Effecta ("**openWindow**", URL, ElementTitleToBeDisplayedInReport));

//Output: Pass

sendReport

Description:
This method adds step details to array lists used for reporting.

Usage:
Four-arguments:

oASelFW.effecta("**sendReport**","report heading 1","report heading 2","PASS/FAIL");

Parameters:
report heading 1 → The Heading that need to be displayed in the first column of the report . (Normally the expected string will be passed)
report heading 2 → The Heading that need to be displayed in the Second column of the report .(Normally the actual string will be passed)
pass/fail → Specify "PASS" or "FAIL" to be specified in the report.

Example:
// Examples tested on // Examples tested on http://www.ksrtc.in/

//assert Text without effecta methods and calling sendReport for report generation.

```
String expected = "Search Bus";
String actual = oASelFW.driver.findElement(By.xpath("//a[@title='Search For Services']")).getText();
System.out.println(actual);
if (actual.equalsIgnoreCase(expected))
{
	oASelFW.effecta("sendReport",expected,actual,"PASS");
}
else
{
	oASelFW.effecta("sendReport",expected,actual,"FAIL");
}
```

// Output: (fail details will be seen in the report)

sendReportWithOutExit

Description:

This method adds step details to array lists used for reporting and it continues further even though the condition is 'Fail'.

Usage:

Four-arguments:

oASelFW.effecta("**sendReportWithOutExit**","report heading 1","report heading 2","PASS/FAIL");

Parameters:
report heading 1 → The Heading that need to be displayed in the first column of the report . (Normally the expected string will be passed)
report heading 2 → The Heading that need to be displayed in the Second column of the report .(Normally the actual string will be passed)
pass/fail → Specify "PASS" or "FAIL" to be specified in the report.

Example:
// Examples tested on // Examples tested on http://www.ksrtc.in/

//assert Text without effecta methods and calling sendReport for report generation.

```
String expected = "Search Bus";
String actual = oASelFW.driver.findElement(By.xpath("//a[@title='Search For Services']")).getText();
System.out.println(actual);

// Determine expected text is equal to actual text of web element.
if (actual.equalsIgnoreCase(expected))
{
        oASelFW.effecta("sendReportWithOutExit ",expected,actual,"PASS");
}
else
{
        oASelFW.effecta("sendReportWithOutExit ",expected,actual,"FAIL");
}
```

Output: (fail details will be seen in the report)

//In the detailed report this step details will be specified.

select

Description:

Select an option from a drop-down using a label or text of option.

If **"select locator found"** then
{
1. Gets all the options under the Select WebElement
2. Compares the optionToBeSelected with the actual options

If **"any actual Option equals with optionToBeSelected"**
{
Click option. (Select the option)
Return Option text.
}
Else
{
Return "Fail" and Exit
}
}
Else
{
Reports **ElementNotFoundException** and **Exits**
}

Usage:

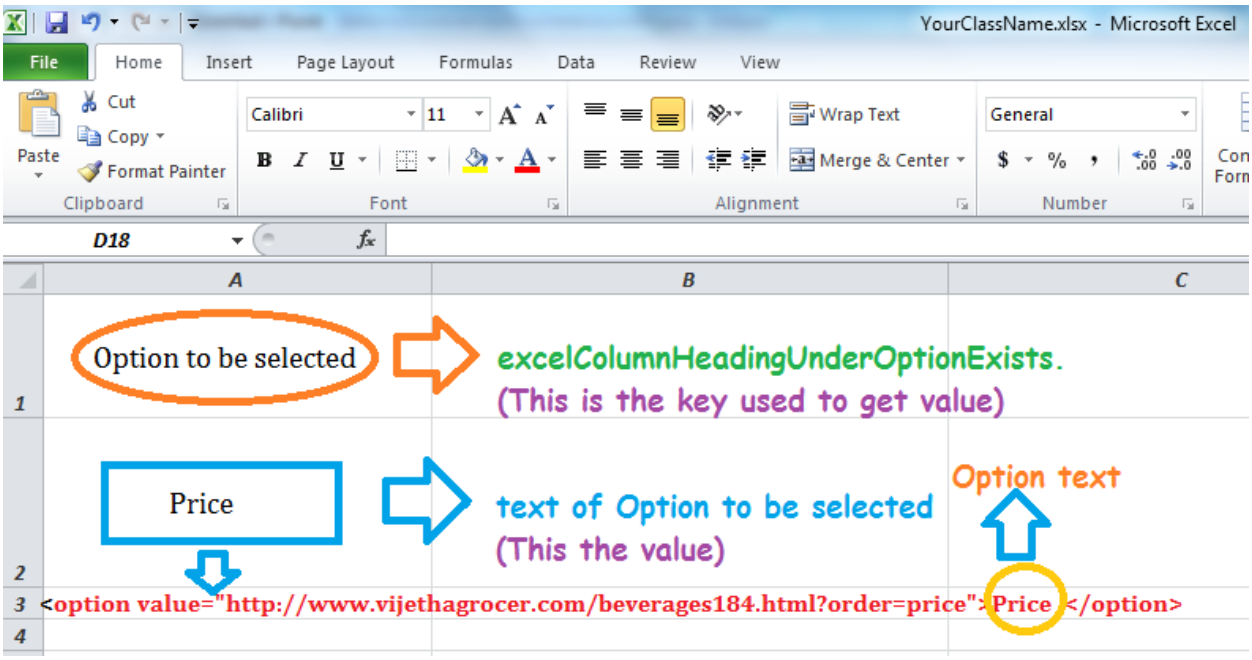
Three-arguments:

oASelFW.effecta("**select**","selectLocator","
excelColumnHeadingUnderOptionLabelExists ");

Parameters:
selectLocator → Locator of the targeted select (WebElement) (any locating strategies i.e., id,name, xpath, css, link text etc.)
excelColumnHeadingUnderOptionExists→ The header is treated as key . (Value is text of option to be selected.)

Returns:
Selected option text.

Example:



// Examples tested on http://www.vijethagrocer.com/beverages-184.html

String excelColumnHeadingUnderOptionExists = "Option to be selected";
String selectLocator = “id=RESULT_RadioButton-5”;
System.out.println(oASelFW.effecta("**select**",selectLocator,excelColumnHeadingUnderOptionExists));

Output: Price

Four-arguments:

oASelFW.effecta("**select**","selectLocator", "optionToBeSelected", “ElementTileToBeDisplayedInReport”);

Parameters:
selectLocator → Locator of the targeted select (WebElement) (any locating strategies i.e., id,name, xpath, css, link text etc.)

optionToBeSelected → The text of an option to be selected.
ElementTitleToBeDisplayedInReport → Title to be displayed for the element in the report.

Example:
// Examples tested on http://www.vijethagrocer.com/beverages-184.html

String selectLocator = "id=RESULT_RadioButton-5";
String OptionToBeSelected = "Price";
String ElementTitleToBeDisplayedInReport = "selecting price option";
System.out.println(oASelFW.effecta("**select**",selectLocator,optionToBeSelected, ElementTitleToBeDisplayedInReport));

Output: Price

selectWindow

Description:
Selects a popup window; once a popup window has been selected, all commands go to that window.

Usage:
Two-arguments:
oASelFW.Effecta("**selectWindow**","windowID");

Parameters:
windowID → the JavaScript window ID of the window to select.

Returns:
Returns the result string "Pass", if command executed successfully.

Example:
// Examples tested on http://www.ksrtc.in/
String windowID = "{40019950-255a-44e1-8d41-0382b5679ff7}";
System.out.println(oASelFW.effecta("**selectWindow** ", windowID));
Output: Pass

selectFrame

Description:

Selects a frame within the current window using index or locator of frame.
(You may invoke this command multiple times to select nested frames.)

Usage:

Two-arguments:

oASelFW.Effecta("**selectFrame**","frameIndex/frameLocator");

Parameters:
Index → The index number is used for specifying which frame to be selected.
(The frame index starts from '0')

Returns:
Returns the result string "Pass", if command executed successfully.

Example:
// Examples tested on https://www.formsite.com/examples/order-form-examples.html

a)

```
String frameIndex = "0";
System.out.println(oASelFW.effecta("selectFrame",  frameIndex));
```

Output: Pass

b)

```
WebElement  frameLocator  =
oASelFW.driver.findElement(By.xpath("/html/body/div[2]/div/div[1]/
div[1]/iframe"));
System.out.println(oASelFW.effecta("selectFrame",  frameLocator));
```

Output: Pass

type

Description:

Sets the value of an input field (target element), as though you typed it in.

If **"target element found"** then
{
1. Sets the target element (input field) value as given string (inputValueToBeSet).
2. Returns value of inputValueToBeSet String.

}
Else
{
Reports **ElementNotFoundException** and **Exits**
}

Usage:

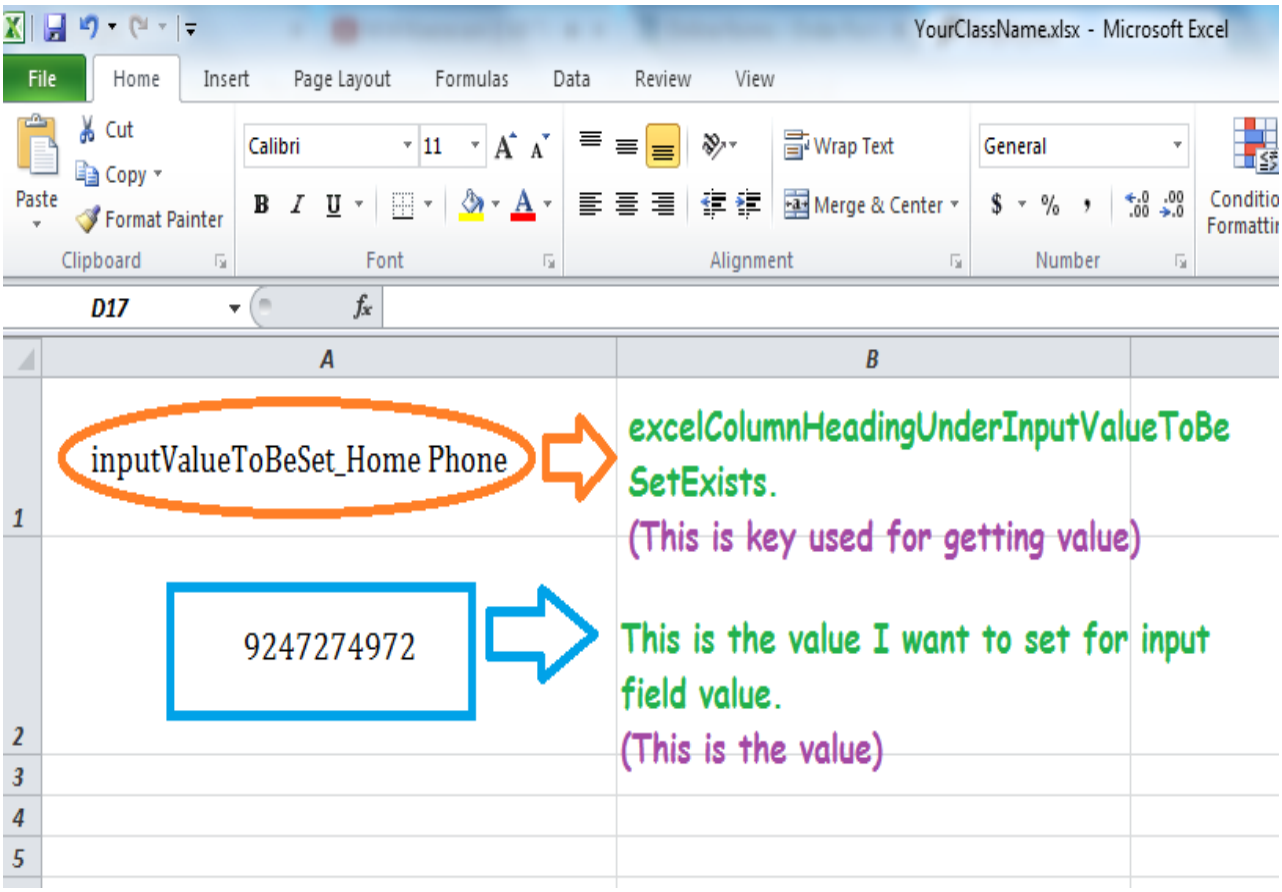
Three-arguments:

oASelFW.effecta("**type**","inputFieldLocator","
excelColumnHeadingUnderOptionLabelExists ");

Parameters:
inputFieldLocator → Locator of the targeted input field (WebElement) (any locating strategies i.e., id,name, xpath, css, link text etc.)
excelColumnHeadingUnderInputValueToBeSetExists→ The header is treated as key .
(Value is input value to be passed string.)

Returns:
The value of inputValueToBeSet String

Example:



// Examples tested on
https://hr.prolifics.com/employees/profile/personal/general

String inputFieldLocator = "//input[@id='empHPhone']";
String excelColumnHeadingUnderInputValueToBeSetExists = "inputValueToBeSet_Home Phone";
System.out.println(oASelFW.effecta("**type**",inputFieldLocator,excelColumnHeadingUnderInputValueToBeSetExists));

Output : 9247274972

Four-arguments:

oASelFW.effecta("**type**","inputFieldLocator", "inputValueToBeSet", "ElementTile ToBeDisplayedInReport");

Parameters:
inputFieldLocator → Locator of the targeted input filed(WebElement) (any locating strategies i.e., id,name, xpath, css, link text etc.)
inputValueToBeSet → The input field value to be set.
ElementTitleToBeDisplayedInReport → Title to be displayed for the element in the report.

Example:
// Examples tested on http://www.vijethagrocer.com/beverages-184.html

String inputFieldLocator = "//input[@id='empHPhone']";
String inputValueToBeSet = "92472724972";
String ElementTitleToBeDisplayedInReport = "setting Home Phone value";
System.out.println(oASelFW.effecta("**type**",inputFieldLocator, inputValueToBeSet, ElementTitleToBeDisplayedInReport));

Output: 9247274972

typeEmpty

Description:

Sets the value of an input field to empty string.
If **"target element found"** then
{
1. Sets the target element (input field) value as empty.
2. Returns "Pass"

}
Else
{
Reports **ElementNotFoundException** and **Exits**
}

Usage:

Two-arguments:

oASelFW.effecta("**typeEmpty**","inputFieldLocator");

Parameters:
inputFieldLocator → Locator of the targeted input field (WebElement) (any locating strategies i.e., id,name, xpath, css, link text etc.)

Returns:
The string "Pass"

Example:

// Examples tested on
https://hr.prolifics.com/employees/profile/personal/general

String inputFieldLocator = "//input[@id='empHPhone']";
System.out.println(oASelFW.effecta("**typeEmpty**",inputFieldLocator));

Output : Pass

Three-arguments:

oASelFW.effecta("**typeEmpty**","inputFieldLocator", "ElementTile ToBeDisplayedInReport");

Parameters:
inputFieldLocator → Locator of the targeted input filed(WebElement) (any locating strategies i.e., id,name, xpath, css, link text etc.).

ElementTitleToBeDisplayedInReport → Title to be displayed for the element in the report.

Example:
// Examples tested on http://www.vijethagrocer.com/beverages-184.html

String inputFieldLocator = "//input[@id='empHPhone']";
String ElementTitleToBeDisplayedInReport = "setting Home Phone value to empty";
System.out.println(oASelFW.effecta("**typeEmpty**",inputFieldLocator, ElementTitleToBeDisplayedInReport));

Output: Pass

.

typePassword

Description:

Type the password in the text box and characters will not be displayed instead the '*****' will be displayed in the text field.
(If My password is 8 characters this will display 8 **"*"** in the text field.)

If **"target element (text box) found"** then
{
1. Type the given password in the textField but '*' will be displayed.
2. Returns the given Password.

}
Else
{
Reports **ElementNotFoundException** and **Exits**
}

Usage:

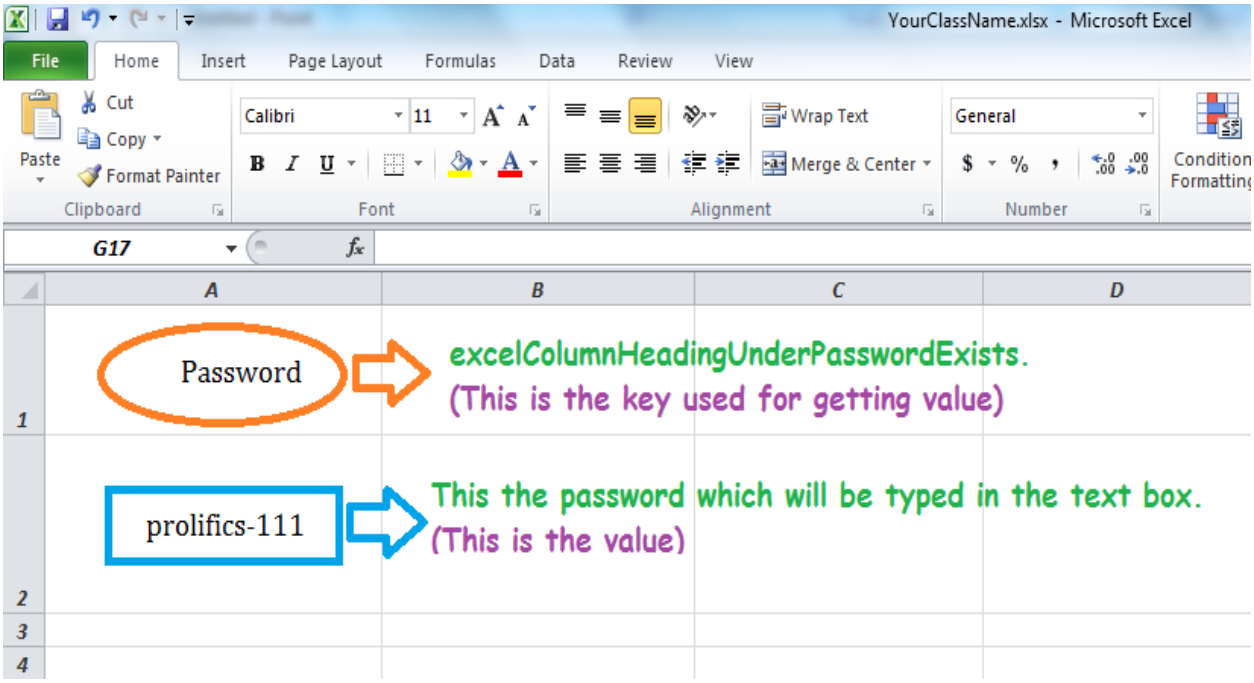
Three-arguments:

oASelFW.effecta("**typePassword**","textFieldLocator","
excelColumnHeadingUnderOptionLabelExists ");

Parameters:
textFieldLocator → Locator of the targeted text field(WebElement) (any locating strategies i.e., id,name, xpath, css, link text etc.)
excelColumnHeadingUnderPasswordExists→ The header is treated as key . (Value is input value to be passed string.)

Returns:
Returns given Password

Example:



// Examples tested on
https://hr.prolifics.com/employees/profile/personal/general

String textFieldLocator = "//input[@id=' EmployeeHashCode ']";
String excelColumnHeadingUnderPasswordExists = "Password";
System.out.println(oASelFW.effecta("**typePassword**",textFieldLocator,excelColumnHeadingUnderInputValueToBeSetExists));

Output: prolifics-111

Four-arguments:

oASelFW.effecta("**typePassword**","textFieldLocator", "password", "ElementTile ToBeDisplayedInReport");

Parameters:
inputFieldLocator → Locator of the targeted text filed(WebElement) (any locating strategies i.e., id,name, xpath, css, link text etc.)
password → The password string which to be typed in the text field.
ElementTitleToBeDisplayedInReport → Title to be displayed for the element in the report.

Example:
// Examples tested on
https://hr.prolifics.com/employees/profile/personal/general

String textFieldLocator = "//input[@id=' EmployeeHashCode ']";
String Password = "prolifics-111";
String ElementTitleToBeDisplayedInReport = "Typing password";
System.out.println(oASelFW.effecta("**typePassword**",textFieldLocator, Password, ElementTitleToBeDisplayedInReport));

Output: prolifics-111

verifySubText

Description:

Verifies whether expected sub text appears in the actual text of the specified web element. (The subtext to be verified is specified in the Excel input data sheet.)
If the **"actual Text of targeted WebElement"** contains **"given sub Text"** then
{
returns the **"actual Text of Targeted WebElement"**.
//Pass
}

```
Else
{
        Reports Details and screenshot and Proceeds to next line.
       //FAIL
}
```

Usage:

Three-Arguments:

oASelFW.Effecta("**verifySubText**", "target", "execlColumnHeadingUnderVerifySubTextExists");

Parameters:

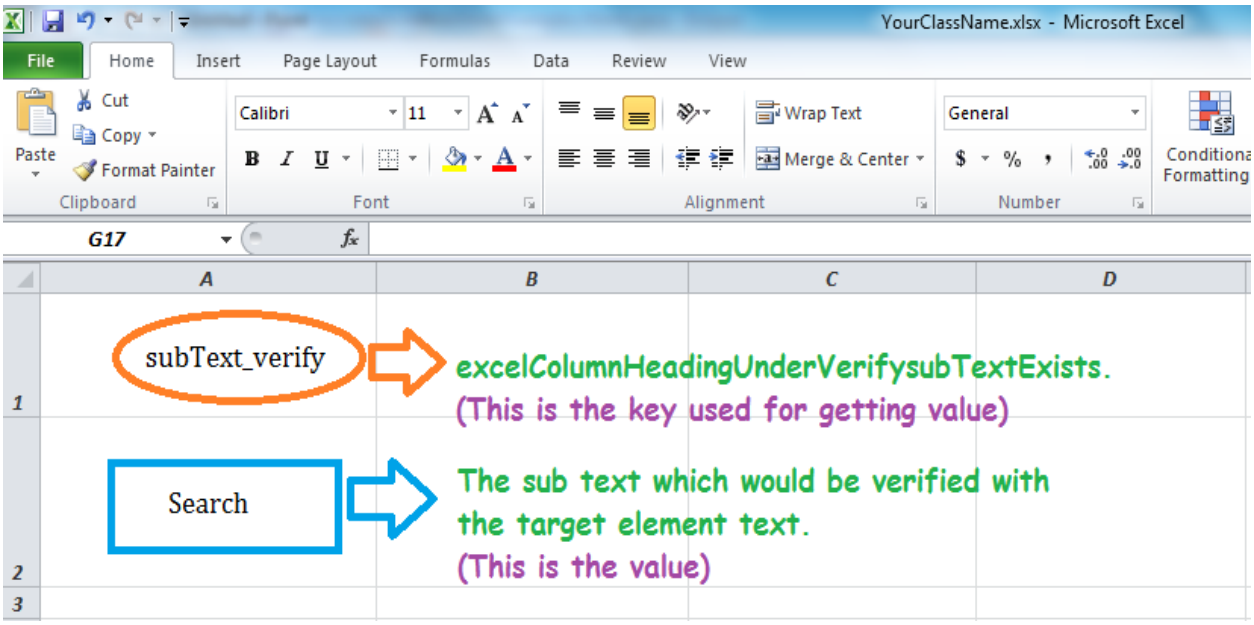
target → Locator of the targeted WebElement (any locating strategies i.e., id, name, xpath, css, link text etc.)

execlColumnHeadingUnderVerifySubTextExists → The header is treated as key .(Value is sub Text to be verified with target element's text .)

Returns:

The Actual Text of targeted WebElement (if verifySubText Pass)

Example:



```
// Examples tested on  http://www.ksrtc.in/
String  target =  "//div[1]/span/a";
String  execlColumnHeadingUnderVerifySubTextExists = "subText_verify";
System.out.println(oASelFW.effecta("verifySubText", target,
execlColumnHeadingUnderVerifySubTextExists));
// Returns target element actual text

//Output : Search Services
```

verifyText

Description:

Verifies that the specified text pattern appears in the specified web element.

If the **"givenText"** equals **"actual Text of targeted WebElement"** then

```
{
        returns the "actual Text of Targeted WebElement".
        //PASS
}
Else
{
        Reports Details and screenshot and Proceeds to next line.
       //FAIL
}
```

Usage:

Three-Arguments:

oASelFW.Effecta("**verifyText**", "target", "expectedText");

Parameters:

target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)

expectedText → A string that exactly matches with the WebElement Text.

Returns:
The Actual Text of targeted WebElement (if verifyText Pass)

Example:
String target = "//div[1]/span/a";
String expectedText = "Selenium tutorials";

System.out.println(oASelFW.effecta("**verifyText**", target, expectedText));
// Returns target element actual text

//**Output :** Selenium tutorials

Four-Arguments:

oASelFW.Effecta("**verifyText**", "target", "expectedText", "ElementTile ToBeDisplayedInReport");

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)
expectedText → A string that exactly matches with the WebElement Text.
ElementTitleToBeDisplayedInReport → Title to be displayed for the element in the report.

Returns:
The Actual Text of targeted WebElement (if verifyText Pass)

Example:
String target = "//div[1]/span/a";
String expectedText = "Selenium tutorials";
String elementTitleDisplayInReport = " first Tutorial Link";
System.out.println(oASelFW.effecta("**verifyText**", target, expectedSubText,elementTitleDisplayInReport));
// Returns target element actual text

//**Output :** Selenium tutorials

verifyTextPresent

Description:

Verifies that the specified text pattern appears in the specified web element.
(The text to be verified is specified in the Excel input data sheet.)

If the **"givenText"** equals **"actual Text of targeted WebElement"**
then
{
returns the **"actual Text of Targeted WebElement"**.
//PASS
}
Else
{
Reports Details and screenshot and Proceeds to next line.
//FAIL
}

Usage:

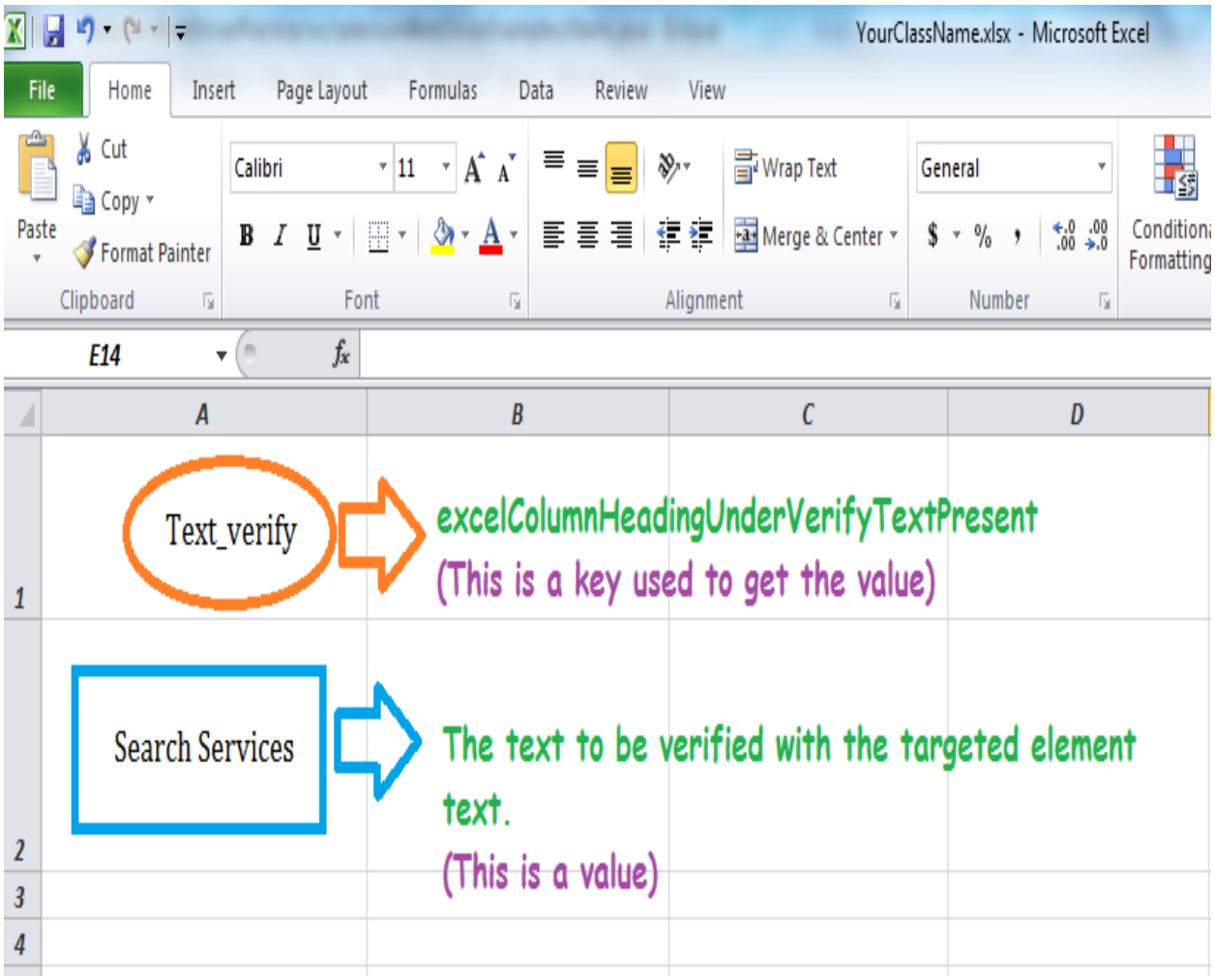
Three-Arguments:

oASelFW.Effecta("**verifyTextPresent**", "target",
"execlColumnHeadingUnderVerifyTextExists");

Parameters:
target → Locator of the targeted WebElement (any locating strategies i.e., id, name, xpath, css, link text etc.)
execlColumnHeadingUnderVerifyTextExists → The header is treated as key .(Value is Text to be verified with target element's text .)

Returns:
The Actual Text of targeted WebElement (if verifyTextPresent Pass)

Example:



String target = "//div[1]/span/a";
String execlColumnHeadingUnderVerifySubTextExists = "Text_verify";
System.out.println(oASelFW.effecta("**verifyTextPresent**", target, execlColumnHeadingUnderVerifySubTextExists));
// Returns target element actual text

//**Output :** Search Services

verifyValue

Description:

Verifies the (whitespace-trimmed) value of an input field (or anything else with a value parameter).
(The value to be verified is specified in the Excel input data sheet.)

If the "**expected value"** equals **"actual value of targeted WebElement"** then
{
returns the **"actual Value of Targeted WebElement"**.
//PASS
}
Else
{
Reports Details and screenshot and Proceeds to next line.
//FAIL
}

Usage:

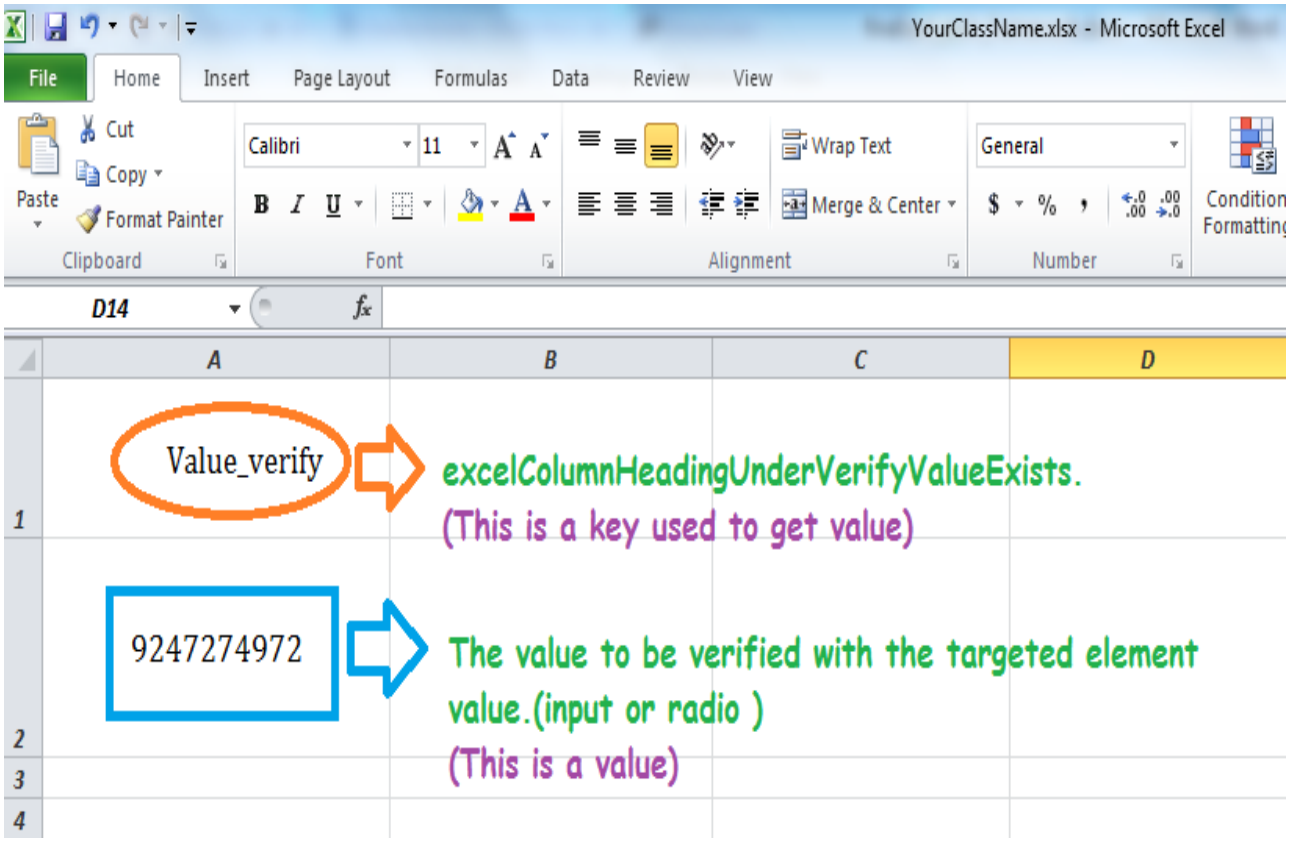
Three-Arguments:

oASelFW.Effecta("**verifyValue**", "target", "execlColumnHeadingUnderVerifyValueExists");

Parameters:
target → Locator of the targeted WebElement (any locating strategies i.e., id, name, xpath, css, link text etc.)
execlColumnHeadingUnderVerifyValueExists → The header is treated as key. (Value is expected input value to be verified with target element's value .)

Returns:
The Actual value of targeted WebElement (if verifyValue Pass)

Example:



// Examples tested on
https://hr.prolifics.com/employees/profile/personal/general

String target= "//input[@id='empHPhone']";
String execlColumnHeadingUnderVerifyValueExists = "Value_verify";
System.out.println(oASelFW.effecta("**verifyValue**", target,
execlColumnHeadingUnderVerifyValueExists));
// Returns target element actual value

//**Output :** 9247274972

verifyTitle

Description:

Verifies the title of the current page. (The title to be verified is specified in the Excel input data sheet.)

If the **"expected title"** equals **"actual Title of the current page"**
then
{
returns the **"actual Title of the current page"**.
//PASS
}
Else
{
Reports Details and screenshot and Proceeds to next line
//FAIL
}

Usage:

Two-Arguments:

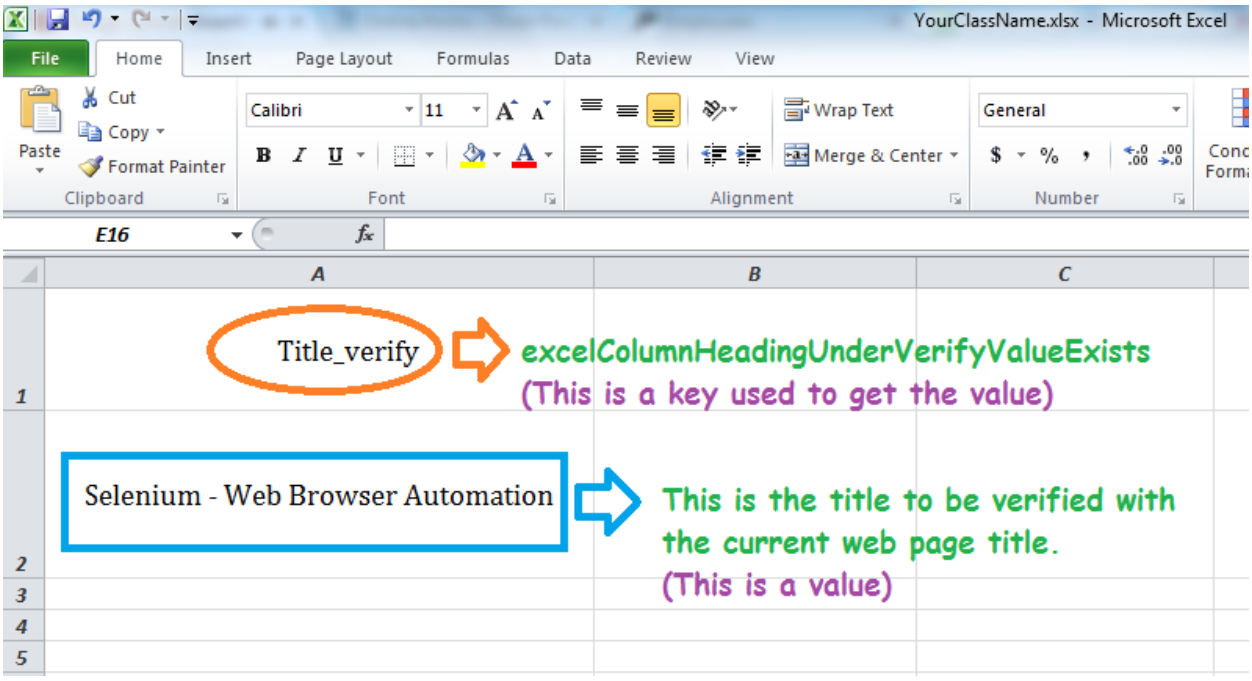
oASelFW.Effecta("**verifyTitle**",
"execlColumnHeadingUnderVerifyTitleExists");

Parameters:

execlColumnHeadingUnderVerifyTitleExists → The header is treated as key to get the value from excel sheet. (The value is expected title of the page.)

Returns:
The Actual Title of current page(if verifyTitle Pass)

Example:



// Examples tested on **http://www.seleniumhq.org/**

String execlColumnHeadingUnderVerifyTitleExists = "Title_verify";
System.out.println(oASelFW.effecta("**verifyTitle**", expectedTitle));
// Returns Actual Title of the current Page

//**Output :** Selenium - Web Browser Automation

verifyConfirmation

Description:

Verifies JavaScript confirmation dialog box message with the given "**expected Confirmation Dialog box Message"** or fail if there were no alerts. And also by default accepts the Confirmation Box (i.e., same as clicking 'ok' manually)
(The confirmation box message to be verified is specified in the Excel input data sheet.)

Confirmation Box**. Accept**;
If the **"expected Confirmation Dialog box Message"** is equals to **"actual Confirmation Dialog box Message"** then
{
returns the **"actual Confirm Box Message"**.
//PASS

}

Else
{
Reports Details and screenshot and Proceeds to next line
//FAIL
}

Usage:

Two-Arguments:

oASelFW.Effecta("**verifyConfirmation**","execlColumnHeadingUnderConfirmationBoxMessageExists");

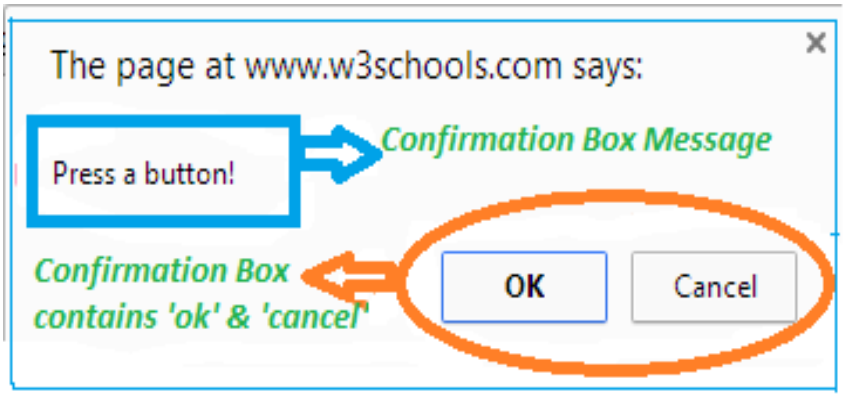
Parameters:

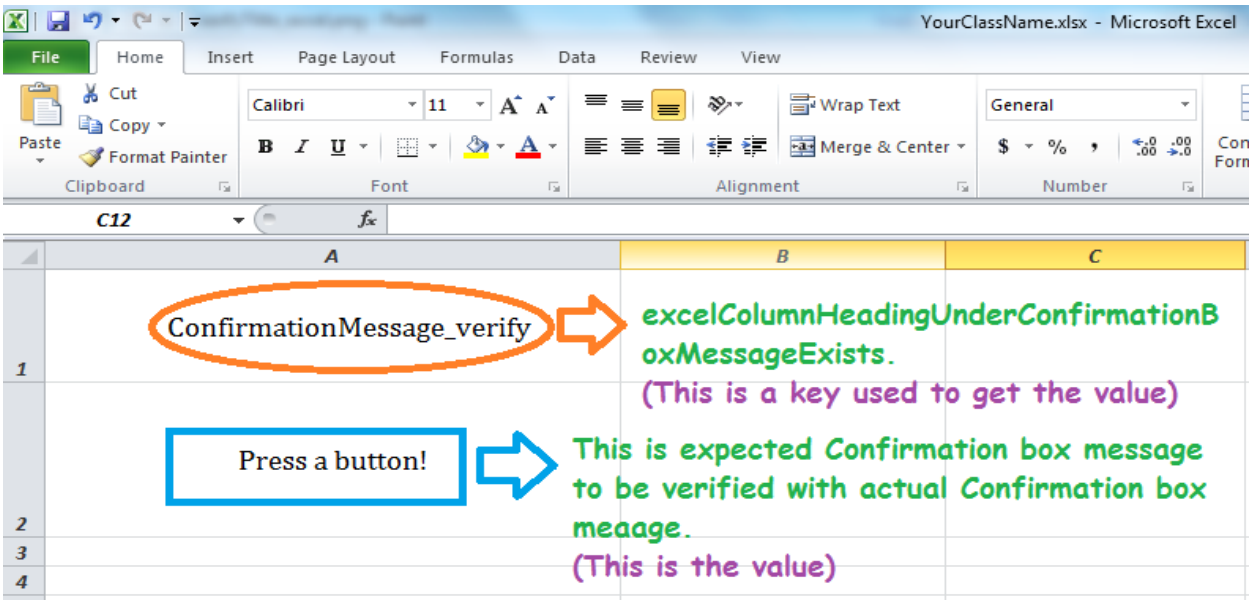
execlColumnHeadingUnderConfirmationBoxMessageExists → The header is treated as key to get the value from excel sheet. (The value is expected message of confirmation box).

Returns:

JavaScript generated ConfirmationMessage Text. (If verifyConfirmation Pass)

Example





// Examples tested on
http://www.w3schools.com/js/tryit.asp?filename=tryjs_confirm

String execlColumnHeadingUnderConfirmationBoxMessageExists = "ConfirmationMessage_verify";
System.out.println(oASelFW.effecta("**verifyConfirmation** ", execlColumnHeadingUnderConfirmationBoxMessageExists));

Output: Press a button!

verifyAlert

Description:

Verifies the message of a JavaScript alert generated during the previous action, or fail if there were no alerts.
(The Alert message to be verified is specified in the Excel input data sheet.)

If the **"expected Alert Message"** is equals to **"actual Alert Message"** then
{
    returns the **"actual Alert Message"**.
    //PASS
}
Else
{
    Reports Details and screenshot and Proceeds to next line.
    //FAIL
}

Usage:

Two-arguments:

oASelFW.Effecta("**verifyAlert**","ExpectedAlertMsg");

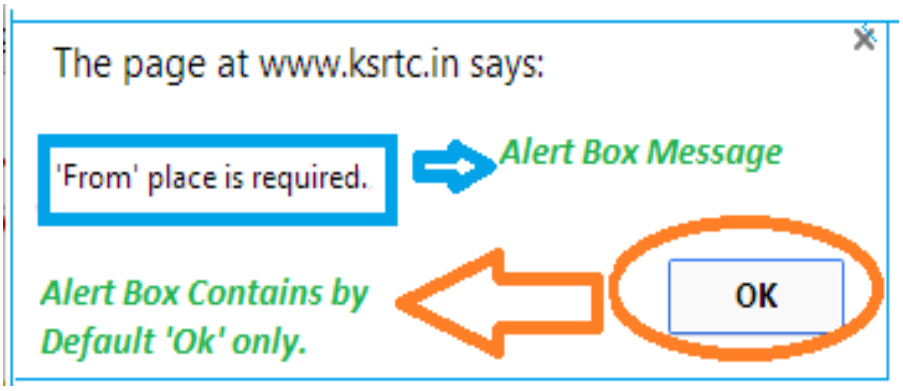
Parameters:

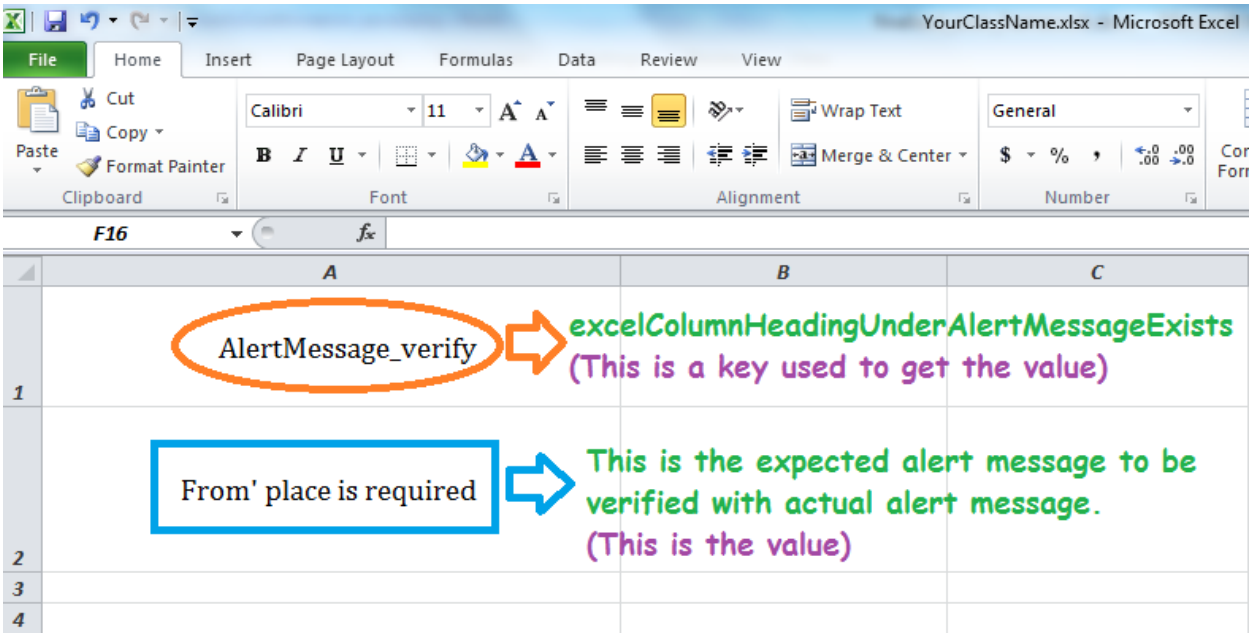
execlColumnHeadingUnderAlertMessageExists → The header is treated as key to get the value from excel sheet. (The value is expected message of Alert box).

Returns:

JavaScript generated Alert Message Text. (If verifyAlert Pass)

Example:





// Examples tested on **http://www.ksrtc.in/**

String execlColumnHeadingUnderAlertMessageExists = "AlertMessage_verify";
System.out.println(oASelFW.effecta("**verifyAlert** ", execlColumnHeadingUnderAlertMessageExists));

Output: 'From' place is required

waitForAlertPresent

Description:

Waits for the specified time in seconds for an alert message to appear on the screen.

If the **"Alert "** appeared
{
    returns the string **"true"**
}
Else
{
    returns the string **"false"**
}

Usage:

Two-arguments:

oASelFW.Effecta("**waitForAlertPresent**","secondsToWait");

Parameters:
secondsToWait → The total number of seconds to wait specified as string.

Returns:

The string "true" (if waitForAlertPresent pass)

Example:
// Examples tested on **http://www.ksrtc.in/**

String secondToWait = "10";

//wait 10 seconds for alert to be present
System.out.println(oASelFW.Effecta("**waitForAlertPresent**",secondsToWait)
);

```
//verify alert once alert present
String  execlColumnHeadingUnderAlertMessageExists =
"AlertMessage_verify";
System.out.println(oASelFW.effecta("verifyAlert ",
execlColumnHeadingUnderAlertMessageExists));
```

Output:

```
 true    //waitForAlertPresent
'From' place is required    //verifyAlert
```

waitForElementPresent

Description:

Waits until specified element is present on the page. (If element is not present waits for given time and exits).
If the **"Element found on the page "** then
{
    returns the string **"true"**
}
Else
{
    reports Error and **exits** .
}

Usage:

Two-arguments:

oASelFW.Effecta("**waitForElementPresent**","target");

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)

Returns:

The string “true” (if waitForElementPresent pass)

Example:

// Examples tested on www.vijethagrocer.com

String target= “//*[@id="nav"]/li[10]/a/span”;
System.out.println(oASelFW.Effecta(“**waitForElementPresent**”,target));

Output: true

Three-arguments:

oASelFW.Effecta(“**waitForElementPresent**”,“target”, “secondsToWait”);

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)
secondsToWait → The total number of seconds to wait specified as string.

Returns:

The string “true” (if waitForElementPresent pass)

Example:

// Examples tested on www.vijethagrocer.com
String target= “//*[@id="nav"]/li[10]/a/span”;
String secondsToWait = “20”;
System.out.println(oASelFW.Effecta(“**waitForElementPresent**”,target,seconds ToWait));

Output: true

Four-arguments:

oASelFW.Effecta("**waitForElementPresent**","target", "secondsToWait", "ElementTitleToBeDisplayedInReport");

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)
secondsToWait → The total number of seconds to wait specified as string
ElementTitleToBeDisplayedInReport → Title to be displayed for the element in the report.

Returns:

The string "true" (if waitForElementPresent pass)

Example:

```
// Examples tested on www.vijethagrocer.com
String target= "//*[@id="nav"]/li[10]/a/span";
String secondsToWait = "20";
String ElementTitleToBeDisplayedInReport = "waitforElementPresent_20sec";
System.out.println(oASelFW.Effecta("waitForElementPresent",target,secondsToWait, ElementTitleToBeDisplayedInReport));
```

Output: true

waitForPageToLoad

Description:

Waits for a new page to load. The waiting time is specified or the default maximum waiting time is 60 sec.

If the **"Page load time out reaches "** then
{
returns the string **"Pass"**
}

Usage:

One-argument:

oASelFW.Effecta("**waitForPageToLoad**");

Returns:

The string "true" (if waitForPageLoadPresent pass)

Example:

// Examples tested on: **http://www.ksrtc.in/**

//wait for default 60 seconds for page to load
System.out.println(oASelFW.Effecta("**waitForPageToLoad**"));
oASelFW.driver.navigate( ).To("**http://www.ksrtc.in/");**

Output: Pass

Two-arguments:

oASelFW.Effecta("**waitForPageToLoad**","secondsToWait");

Parameters:
secondsToWait → The total number of seconds to wait specified as string.

Returns:

The string "true" (if waitForPageLoad pass)

Example:
// Examples tested on: **http://www.ksrtc.in/**

String secondToWait = "30";
//wait 30 seconds for page to load
System.out.println(oASelFW.Effecta("**waitForPageToLoad**",
secondsToWait));
oASelFW.driver.navigate( ).To("**http://www.ksrtc.in/**");

Output: Pass

waitForValue

Description:

Waits for the specified time in seconds for a WebElement value (i.e., value of <input> <option> etc.)
If the **" Targeted Element found on the page"** then
{
1. Wait for specified time in seconds
2. Get the target element value
3. Return the value of targeted element

}
Else
{
Reports Error and **exits.**
}

Usage:

Two-arguments:

oASelFW.Effecta("**waitForValue**","target");

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)

Returns:

The target element value (if waitForValue pass)

Example:

// Examples tested on
https://hr.prolifics.com/employees/profile/personal/general

//By default waits for 10 seconds before get the element value
String target="/html/body/div[1]/div[12]/div/div[1]/div[1]/div[11]/input";
System.out.println(oASelFW.Effecta("**waitForValue**",target));

Output: 9247274972

Three-arguments:

oASelFW.Effecta("**waitForValue**","target", "secondsToWait");

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)
secondsToWait → The total number of seconds to wait specified as string.

Returns:

The target element value (if waitForValue pass)

Example:

// Examples tested on
https://hr.prolifics.com/employees/profile/personal/general

String target="/html/body/div[1]/div[12]/div/div[1]/div[1]/div[11]/input";
String secondsToWait = "20";

// waits for 20 seconds before get the element value
System.out.println(oASelFW.Effecta("**waitForValue**",target,secondsToWait));

Output: 9247274972

windowMaximize

Description:

To maximize browser window

Usage:

One-argument:

oASelFW.Effecta("**windowMaximize**");

Returns:

The string "**Pass**"

Example:

oASelFW.driver.navigate( ).To("**http://www.ksrtc.in/**");

System.out.println(oASelFW.Effecta("**windowMaximize**"));

Output: Pass

waitForFrameAndSelect

Description:

Waits until frame is available and Selects a frame within the current window using index or locator of frame. (You may invoke this command multiple times to select nested frames.)

Usage:

Two-arguments:

oASelFW.Effecta("**waitForFrameAndSelect**","frameIndex/frameLocator");

Parameters:
Index → The index number is used for specifying which frame to be selected. (The frame index starts from '0')

Returns:
Returns the result string "Pass", if command executed successfully.

Example:
// Examples tested on https://www.formsite.com/examples/order-form-examples.html

a)

```
String frameIndex = "0";
System.out.println(oASelFW.effecta("waitForFrameAndSelect",
frameIndex));
```

Output: Pass

b)

```
WebElement frameLocator =
oASelFW.driver.findElement(By.xpath("/html/body/div[2]/div/div[1]/
div[1]/iframe"));
System.out.println(oASelFW.effecta("waitForFrameAndSelect",
frameLocator));
```

Output: Pass

waitUntilElementVisible

Description:

Waits until specified time in seconds and searches for element is visible on the page. (If element is not visible waits for given time and exits).

Waits until specified time in seconds
If the **"Element is vibile on the page "** then
{
returns the string **"true"**
}
Else
{
reports Error and **exits** .
}

Usage:

Three-arguments:

oASelFW.Effecta("**waitForElementVisible**","target", "secondsToWait");

Parameters:
target → Locator of the targeted web element (any locating strategies i.e., id, name, xpath, css, link text etc.)

secondsToWait → The total number of seconds to wait specified as string.

Returns:

The string "true" (if waitForElementVisible pass)

Example:

// Examples tested on www.vijethagrocer.com
String target= “//*[@id="nav"]/li[10]/a/span”;
String secondsToWait = “20”;
System.out.println(oASelFW.Effecta(“**waitForElementVisible**”,target,seconds ToWait));

Output: true