Katalon Keywords Explanation

**KMS Technology**

**123 Cong Hoa Street,  
Ward 12,Tan Binh District,  
Ho Chi Minh City**

**(84 8) 3811 9977**

**(84 8) 3811 9955**

**11/20/2014**

This document explains the usage of keywords built into Katalon.

Table of Content

[1. Alert Class Action 3](#_Toc427840817)

[**1.1** **acceptAlert** 3](#_Toc427840818)

[**1.2** **dismissAlert** 3](#_Toc427840819)

[**1.3** **getAlertText** 3](#_Toc427840820)

[**1.4** **setAlertText** 4](#_Toc427840821)

[**1.5** **verifyAlertNotPresent (not implemented yet)** 4](#_Toc427840822)

[**1.6** **verifyAlertPresent (not implemented yet)** 4](#_Toc427840823)

[**1.7** **waitForAlertNotPresent (not implemented yet)** 5](#_Toc427840824)

[**1.8** **waitForAlertPresent (not implemented yet)** 5](#_Toc427840825)

[2. Browser Action 5](#_Toc427840826)

[**2.1** **authenticate** 5](#_Toc427840827)

[**2.2** **back** 6](#_Toc427840828)

[**2.3** **closeBrowser** 6](#_Toc427840829)

[**2.4** **closeWindowIndex** 6](#_Toc427840830)

[**2.5** **closeWindowTitle** 7](#_Toc427840831)

[**2.6** **closeWindowUrl** 7](#_Toc427840832)

[**2.7** **deleteAllCookies** 8](#_Toc427840833)

[**2.8** **forward** 8](#_Toc427840834)

[**2.9** **getUrl** 8](#_Toc427840835)

[**2.10** **getWindowTitle** 9](#_Toc427840836)

[**2.11** **maximizeWindow** 9](#_Toc427840837)

[**2.12** **navigateToUrl** 9](#_Toc427840838)

[**2.13** **openBrowser** 10](#_Toc427840839)

[**2.14** **refresh** 10](#_Toc427840840)

[**2.15** **switchToDefaultContent** 10](#_Toc427840841)

[**2.16** **switchToWindowIndex** 11](#_Toc427840842)

[**2.17** **switchToWindowTitle** 11](#_Toc427840843)

[**2.18** **switchToWindowUrl** 11](#_Toc427840844)

[**2.19** **verifyTextNotPresent** 12](#_Toc427840845)

[**2.20** **verifyTextPresent** 12](#_Toc427840846)

[**2.21** **waitForPageLoad** 13](#_Toc427840847)

[3. Keyboard Action 13](#_Toc427840848)

[**3.1** **sendKeys** 13](#_Toc427840849)

[4. Mouse Action 13](#_Toc427840850)

[**4.1** **click** 13](#_Toc427840851)

[**4.2** **doubleClick** 14](#_Toc427840852)

[**4.3** **dragAndDropByOffset** 14](#_Toc427840853)

[**4.4** **dragAndDropToObject** 14](#_Toc427840854)

[**4.5** **mouseOver** 14](#_Toc427840855)

[**4.6** **rightClick** 15](#_Toc427840856)

[5. Select Class Action 15](#_Toc427840857)

[**5.1** **deselectAllOption** 15](#_Toc427840858)

[**5.2** **deselectOptionByIndex** 16](#_Toc427840859)

[**5.3** **deselectOptionByLabel** 17](#_Toc427840860)

[**5.4** **deselectOptionByValue** 17](#_Toc427840861)

[**5.5** **getNumberOfSelectedOption** 18](#_Toc427840862)

[**5.6** **getNumberOfTotalOption** 19](#_Toc427840863)

[**5.7** **selectAllOption** 19](#_Toc427840864)

[**5.8** **selectOptionByIndex** 20](#_Toc427840865)

[**5.9** **selectOptionByLabel** 21](#_Toc427840866)

[**5.10** **selectOptionByValue** 22](#_Toc427840867)

[**5.11** **verifyAnyOptionNotSelected (not implemented yet)** 22](#_Toc427840868)

[**5.12** **verifyAnyOptionSelected (not implemented yet)** 23](#_Toc427840869)

[**5.13** **verifyOptionNotPresentByIndex (not implemented yet)** 24](#_Toc427840870)

[**5.14** **verifyOptionNotPresentByLabel** 25](#_Toc427840871)

[**5.15** **verifyOptionNotPresentByValue** 25](#_Toc427840872)

[**5.16** **verifyOptionNotSelectedByIndex** 26](#_Toc427840873)

[**5.17** **verifyOptionNotSelectedByLabel** 27](#_Toc427840874)

[**5.18** **verifyOptionNotSelectedByValue** 28](#_Toc427840875)

[**5.19** **verifyOptionPresentByIndex (not implemented yet)** 29](#_Toc427840876)

[**5.20** **verifyOptionPresentByLabel** 30](#_Toc427840877)

[**5.21** **verifyOptionPresentByValue** 31](#_Toc427840878)

[**5.22** **verifyOptionSelectedByIndex** 32](#_Toc427840879)

[**5.23** **verifyOptionSelectedByLabel** 33](#_Toc427840880)

[**5.24** **verifyOptionSelectedByValue** 34](#_Toc427840881)

[6. Utility Action 35](#_Toc427840882)

[**6.1** **callTestCase** 35](#_Toc427840883)

[**6.2** **concatenate** 35](#_Toc427840884)

[**6.3** **delay** 35](#_Toc427840885)

[**6.4** **modifyObjectProperty** 36](#_Toc427840886)

[**6.5** **removeObjectProperty** 36](#_Toc427840887)

[**6.6** **takeScreenshot** 37](#_Toc427840888)

[**6.7** **verifyEqual** 37](#_Toc427840889)

[**6.8** **verifyGreaterThan** 37](#_Toc427840890)

[**6.9** **verifyGreaterThanOrEqual** 38](#_Toc427840891)

[**6.10** **verifyLessThan** 39](#_Toc427840892)

[**6.11** **verifyLessThanOrEqual** 39](#_Toc427840893)

[**6.12** **verifyMatch** 40](#_Toc427840894)

[**6.13** **verifyNotEqual** 40](#_Toc427840895)

[**6.14** **verifyNotMatch** 41](#_Toc427840896)

[7. Web Element Action 41](#_Toc427840897)

[**7.1** **check** 41](#_Toc427840898)

[**7.2** **focus** 42](#_Toc427840899)

[**7.3** **getAttribute** 42](#_Toc427840900)

[**7.4** **getText** 42](#_Toc427840901)

[**7.5** **setText** 43](#_Toc427840902)

[**7.6** **uncheck** 43](#_Toc427840903)

[**7.7** **verifyElementChecked** 43](#_Toc427840904)

[**7.8** **verifyElementEnabled (not implemented yet)** 44](#_Toc427840905)

[**7.9** **verifyElementNotChecked** 44](#_Toc427840906)

[**7.10** **verifyElementNotEnabled (not implemented yet)** 45](#_Toc427840907)

[**7.11** **verifyElementNotPresent** 45](#_Toc427840908)

[**7.12** **verifyElementNotVisible (not implemented yet)** 45](#_Toc427840909)

[**7.13** **verifyElementPresent** 46](#_Toc427840910)

[**7.14** **waitForElementClickable** 46](#_Toc427840911)

[**7.15** **waitForElementNotPresent** 46](#_Toc427840912)

[**7.16** **waitForElementPresent** 47](#_Toc427840913)

[**7.17** **waitForElementVisible** 47](#_Toc427840914)

1. **Alert Class Action**
   1. **acceptAlert**

|  |  |
| --- | --- |
| Description | Simulate users clicking on “OK” button of an alert popup (alert, confirmation popup, prompt popup) |
| Object | N/a |
| Input | N/a |
| Output | N/a |

**Example** **Scenario** – Users want to click on “OK” button of an alert.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| acceptAlert |  |  |  |

* 1. **dismissAlert**

|  |  |
| --- | --- |
| Description | Simulate users clicking on “Cancel” button of an alert popup (alert, confirmation popup, prompt popup) |
| Object | N/a |
| Input | N/a |
| Output | N/a |

**Example** **Scenario** – Users want to click on “Cancel” button of a confirmation popup.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| dismissAlert |  |  |  |

* 1. **getAlertText**

|  |  |
| --- | --- |
| Description | Get displayed text of an alert popup (alert, confirmation popup, prompt popup) |
| Object | N/a |
| Input | N/a |
| Output | String result – text of the alert |

**Example** **Scenario** – Users want to get displayed text of an alert. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| getAlertText |  |  | var\_result |

* 1. **setAlertText**

|  |  |
| --- | --- |
| Description | Simulate users typing text into a prompt popup |
| Object | N/a |
| Input | 1. String value – text to type into the prompt popup |
| Output | N/a |

**Example** **Scenario** – Users want to type “KMS Technology” into a prompt popup.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| setAlertText |  | “KMS Technology” |  |

* 1. **verifyAlertNotPresent (not implemented yet)**

|  |  |
| --- | --- |
| Description | Verify if there is NOT an alert |
| Object | N/a |
| Input | N/a |
| Output | boolean result – **true** if there is **NOT** an alert; otherwise, **false** |

**Example** **Scenario** – Users want to verify if there is **NOT** an alert. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyAlertNotPresent |  |  | var\_result |

* 1. **verifyAlertPresent (not implemented yet)**

|  |  |
| --- | --- |
| Description | Verify if there is an alert |
| Object | N/a |
| Input | N/a |
| Output | boolean result – **true** if there is an alert; otherwise, **false** |

**Example** **Scenario** – Users want to verify if there is an alert. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyAlertPresent |  |  | var\_result |

* 1. **waitForAlertNotPresent (not implemented yet)**

|  |  |
| --- | --- |
| Description | Wait for an alert to not present (disappear) within the given time in second unit |
| Object | N/a |
| Input | 1. int seconds – the amount of seconds to wait |
| Output | N/a |

**Example** **Scenario** – Users want to wait for an alert to disappear within 10 seconds.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| waitForAlertNotPresent |  | 10 |  |

* 1. **waitForAlertPresent (not implemented yet)**

|  |  |
| --- | --- |
| Description | Wait for an alert to present (appear) within the given time in second unit |
| Object | N/a |
| Input | 1. int seconds – the amount of seconds to wait |
| Output | N/a |

**Example** **Scenario** – Users want to wait for an alert to appear within 10 seconds.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| waitForAlertPresent |  | 10 |  |

1. **Browser Action**
   1. **authenticate**

|  |  |
| --- | --- |
| Description | Navigate to a page that requires authentication. System will enter username and password |
| Object | N/a |
| Input | 1. String url – url of the page to navigate  2. String username  3. String password  4. int waitingTime – time to wait since navigating to the page until entering username |
| Output | N/a |

**Example** **Scenario** – Users want to navigate to page requiring authentication.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| authenticate |  | “https://www.example.com”, “testuser”, “P@ssword”, 2 |  |

* 1. **back**

|  |  |
| --- | --- |
| Description | Simulate users clicking “back” button on their browser |
| Object | N/a |
| Input | N/a |
| Output | N/a |

**Example** **Scenario** – Users want to click on “back” button.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| back |  |  |  |

* 1. **closeBrowser**

|  |  |
| --- | --- |
| Description | Close the browser. This action will close all windows of the browser. |
| Object | N/a |
| Input | N/a |
| Output | N/a |

**Example** **Scenario** – Users want to close browser.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| closeBrowser |  |  |  |

* 1. **closeWindowIndex**

|  |  |
| --- | --- |
| Description | Close window with the given index. Window index is counted by order of window's appearance and starts from 0. After closing a window, all windows appeared after that window will have index decreased by 1. For example: we have 4 windows (index: 0, 1, 2, 3); close window index 1, after closing, the window with old index 2 will become new index 1, the window with old index 3 will become new index 2.  If users close current window, system will switch to the first window. If current window happens to be the first window, system will switch to the new first window. However we strongly recommend users switch to another window before closing current window to prevent any confusion. |
| Object | N/a |
| Input | 1. int index – index of the window to close |
| Output | N/a |

**Example** **Scenario** – Users want to close the second window (index 1).

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| closeWindowIndex |  | 1 |  |

* 1. **closeWindowTitle**

|  |  |
| --- | --- |
| Description | Close window with the given title.  If users close current window, system will switch to the first window. If current window happens to be the first window, system will switch to the new first window. However we strongly recommend users switch to another window before closing current window to prevent any confusion. |
| Object | N/a |
| Input | 1. String title – title of the window to close |
| Output | N/a |

**Example** **Scenario** – Users want to close the window with title “Contact Page”.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| closeWindowTitle |  | “Contact Page” |  |

* 1. **closeWindowUrl**

|  |  |
| --- | --- |
| Description | Close window with the given url.  If users close current window, system will switch to the first window. If current window happens to be the first window, system will switch to the new first window. However we strongly recommend users switch to another window before closing current window to prevent any confusion. |
| Object | N/a |
| Input | 1. String url – url of the window to close |
| Output | N/a |

**Example** **Scenario** – Users want to close the window with url “http://www.kms-technology.com/contact.html”.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| closeWindowUrl |  | “http://www.kms-technology.com/contact.html” |  |

* 1. **deleteAllCookies**

|  |  |
| --- | --- |
| Description | Delete all cookies of all windows. |
| Object | N/a |
| Input | N/a |
| Output | N/a |

**Example** **Scenario** – Users want to delete all cookies.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| deleteAllCookies |  |  |  |

* 1. **forward**

|  |  |
| --- | --- |
| Description | Simulate users clicking “forward” button on their browser |
| Object | N/a |
| Input | N/a |
| Output | N/a |

**Example** **Scenario** – Users want to click “forward” button.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| forward |  |  |  |

* 1. **getUrl**

|  |  |
| --- | --- |
| Description | Get url of the current window |
| Object | N/a |
| Input | N/a |
| Output | 1. String url – url of the current window |

**Example** **Scenario** – Users want to get url of the current window. The output is stored into variable var\_url.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| getUrl |  |  | var\_url |

* 1. **getWindowTitle**

|  |  |
| --- | --- |
| Description | Get title of the current window |
| Object | N/a |
| Input | N/a |
| Output | 1. String title – title of the current window |

**Example** **Scenario** – Users want to get title of the current window. The output is stored into variable var\_title.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| getWindowTitle |  |  | var\_title |

* 1. **maximizeWindow**

|  |  |
| --- | --- |
| Description | Resize current window to take up the entire screen |
| Object | N/a |
| Input | N/a |
| Output | N/a |

**Example** **Scenario** – Users want to maximize the window.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| maximizeWindow |  |  |  |

* 1. **navigateToUrl**

|  |  |
| --- | --- |
| Description | Navigate to the specified web page |
| Object | N/a |
| Input | 1. String url – url of the web page to navigate to |
| Output | N/a |

**Example** **Scenario** – Users want to navigate to the page “http://www.kms-technology.com/”.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| navigateToUrl |  | “http://www.kms-technology.com/” |  |

* 1. **openBrowser**

|  |  |
| --- | --- |
| Description | Open browser and navigate to the specified url; if url is left empty, just open browser |
| Object | N/a |
| Input | 1. String url – url of the web page to be opened, can be left empty |
| Output | N/a |

**Example** **Scenario** – Users want to open browser and navigate to the page “http://www.kms-technology.com/”.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| openBrowser |  | “http://www.kms-technology.com/” |  |

* 1. **refresh**

|  |  |
| --- | --- |
| Description | Simulate users clicking “refresh” button on their browser |
| Object | N/a |
| Input | N/a |
| Output | N/a |

**Example** **Scenario** – Users want to refresh the page.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| refresh |  |  |  |

* 1. **switchToDefaultContent**

|  |  |
| --- | --- |
| Description | Use after an action on child object of an iframe. |
| Object | N/a |
| Input | N/a |
| Output | N/a |

**Example** **Scenario** – Users want to set text into “txtbx\_username” and “txtbx\_password” textboxes. Those elements lie on an iframe.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| setText | txtbx\_username | “test” |  |
| switchToDefaultContent |  |  |  |
| setText | txtbx\_password | “secret” |  |
| switchToDefaultContent |  |  |  |

* 1. **switchToWindowIndex**

|  |  |
| --- | --- |
| Description | Switch to the window with given index. Window index is counted by order of window's appearance and starts from 0. After switching, window index does NOT change, i.e. the first window still has index 0, the second window still has index 1 and so on. |
| Object | N/a |
| Input | 1. int index – index of the window to switch to |
| Output | N/a |

**Example** **Scenario** – Users want to switch to the third window (index 2) then switch back to the first window (index 0).

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| switchToWindowIndex |  | 2 |  |
| switchToWindowIndex |  | 0 |  |

* 1. **switchToWindowTitle**

|  |  |
| --- | --- |
| Description | Switch to the window with the given title |
| Object | N/a |
| Input | 1. String title – title of the window to switch to |
| Output | N/a |

**Example** **Scenario** – Users want to switch to the window with title “Home Page”.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| switchToWindowTitle |  | “Home Page” |  |

* 1. **switchToWindowUrl**

|  |  |
| --- | --- |
| Description | Switch to the window with the given url |
| Object | N/a |
| Input | 1. String url – title of the window to switch to |
| Output | N/a |

**Example** **Scenario** – Users want to switch to the window with url “http://www.kms-technology.com/contact.html”.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| switchToWindowUrl |  | “http://www.kms-technology.com/contact.html” |  |

* 1. **verifyTextNotPresent**

|  |  |
| --- | --- |
| Description | Verify if the given texts do NOT present anywhere in the page source |
| Object | N/a |
| Input | 1. String text – text to be verified if **NOT** presenting anywhere in the page source 2. boolean isRegex – **true** if text is regular expression; otherwise, **false** |
| Output | boolean result – **true** if text does **NOT** present anywhere in the page source; otherwise, **false** |

**Example** **Scenario** – Users want to verify if text “KMS Technology” does **NOT** present anywhere in the page source. Not use regular expression. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyTextNotPresent |  | “KMS Technology”;false | var\_result |

* 1. **verifyTextPresent**

|  |  |
| --- | --- |
| Description | Verify if the given texts present anywhere in the page source |
| Object | N/a |
| Input | 1. String text – text to be verified if presenting anywhere in the page source 2. boolean isRegex – **true** if text is regular expression; otherwise, **false** |
| Output | boolean result – **true** if text presents anywhere in the page source; otherwise, **false** |

**Example** **Scenario 1** – Users want to verify if text “KMS Technology” presents anywhere in the page source. Not use regular expression. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyTextPresent |  | “KMS Technology”; false | var\_result |

**Scenario 2** – Users want to verify if texts “KMS” and “Technology” present anywhere in the page source. In addition, there could be some characters between those two texts. Use regular expression. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyTextPresent |  | “.\*KMS.\*Technology.\*”; true | var\_result |

* 1. **waitForPageLoad**

|  |  |
| --- | --- |
| Description | Wait for the web page to load within the given time in second unit |
| Object | N/a |
| Input | 1. int seconds – the amount of seconds to wait |
| Output | N/a |

**Example** **Scenario** – Users want to wait for the web page to load within 10 seconds.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| waitForPageLoad |  | 10 |  |

1. **Keyboard Action**
   1. **sendKeys**

|  |  |
| --- | --- |
| Description | Simulates keystroke events on the specified element, as though you typed the value key-by-key |
| Object | TestObject object – represent a web element |
| Input | 1. value – the combination of keys to type |
| Output | N/a |

**Example** **Scenario** – Users want to type “KMS” into the text-box “txtbx\_Search”.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| sendKeys | txtbx\_Search | “KMS” |  |

1. **Mouse Action**
   1. **click**

|  |  |
| --- | --- |
| Description | Click on the given web element |
| Object | TestObject object – represent a web element |
| Input | N/a |
| Output | N/a |

**Example** **Scenario** – Users want to click on the object “btn\_Login”.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| click | btn\_Login |  |  |

* 1. **doubleClick**

|  |  |
| --- | --- |
| Description | Double click on the given web element |
| Object | TestObject object – represent a web element |
| Input | N/a |
| Output | N/a |

**Example** **Scenario** – Users want to double click on the object “btn\_Login”.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| doubleClick | btn\_Login |  |  |

* 1. **dragAndDropByOffset**

|  |  |
| --- | --- |
| Description | Drags an object a certain distance and then drops it. |
| Object | TestObject object – represent a web element |
| Input | 1. int xOffset - x offset in pixels from the current location to which the element should be moved, e.g., 70, -30  2. int yOffset - y offset in pixels from the current location to which the element should be moved, e.g., 70, -30 |
| Output | N/a |

**Example** **Scenario** – Users want to drag image “img\_Pic1” and drop it at a location that is 30 pixels on the left and 70 pixels down.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| dragAndDropByOffset | img\_Pic1 | -30;70 |  |

* 1. **dragAndDropToObject**

|  |  |
| --- | --- |
| Description | Drags an object and then drops it to another object. |
| Object | TestObject object – represent a web element |
| Input | 1. TestObject destinationObject |
| Output | N/a |

**Example** **Scenario** – Users want to drag image “img\_Pic1” and drop onto image “img\_Pic2”. Currently, users must switch to Advanced mode to add destination object.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| dragAndDropToObject | img\_Pic1 | img\_Pic2 |  |

* 1. **mouseOver**

|  |  |
| --- | --- |
| Description | Simulate users hovering a mouse over the given element |
| Object | TestObject object – represent a web element |
| Input | N/a |
| Output | N/a |

**Example** **Scenario** – Users want to hover over the object “btn\_Login”.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| mouseOver | btn\_Login |  |  |

* 1. **rightClick**

|  |  |
| --- | --- |
| Description | Right click on the given web element |
| Object | TestObject object – represent a web element |
| Input | N/a |
| Output | N/a |

**Example** **Scenario** – Users want to right click on the object “btn\_Login”.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| rightClick | btn\_Login |  |  |

1. **Select Class Action**
   1. **deselectAllOption**

|  |  |
| --- | --- |
| Description | Deselect all options |
| Object | TestObject object – represent a web element |
| Input | N/a |
| Output | N/a |

**Example** **Scenario** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to deselect all options.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| deselectAllOption | sl\_Commands |  |  |

* 1. **deselectOptionByIndex**

|  |  |
| --- | --- |
| Description | Deselect the options at the given indices. Index starts from 0. |
| Object | TestObject object – represent a web element |
| Input | 1. String index\_range – index range of the options to be deselected   Example: 2 - index 2  2,3 - index 2 and 3  2-5 - index 2 to 5 (2, 3, 4, 5) |
| Output | N/a |

**Example** **Scenario** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to deselect “Navigation Commands” option (index 1).

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| deselectOptionByIndex | sl\_Commands | “1” |  |

Users want to deselect all options from index 0 to 3.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| deselectOptionByIndex | sl\_Commands | “0-3” |  |

Users want to deselect “Browser Commands” option (index 0) and “Wait Commands” option (index 3).

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| deselectOptionByIndex | sl\_Commands | “0,3” |  |

* 1. **deselectOptionByLabel**

|  |  |
| --- | --- |
| Description | Deselect the options with the given label (displayed text) |
| Object | TestObject object – represent a web element |
| Input | 1. String label – displayed text of the options to be deselected 2. boolean isRegex – **true** if label is regular expression, **false** if not |
| Output | N/a |

**Example** **Scenario** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to deselect “Browser Commands” option.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| deselectOptionByLabel | sl\_Commands | “Browser Commands”, false |  |

Users want to deselect “Wait Commands” option and “WebElement Commands” option.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| deselectOptionByLabel | sl\_Commands | “^W(ai|ebElemen)t Commands$”, true |  |

* 1. **deselectOptionByValue**

|  |  |
| --- | --- |
| Description | Deselect the options with the given value |
| Object | TestObject object – represent a web element |
| Input | 1. String value – value of the options to be deselected 2. boolean isRegex – **true** if value is regular expression, **false** if not |
| Output | N/a |

**Example** **Scenario** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to deselect “Browser Commands” option (value “browser\_commands”).

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| deselectOptionByValue | sl\_Commands | “browser\_commands”, false |  |

Users want to deselect “Wait Commands” option (value “wait\_commands”) and “WebElement Commands” option (value “webelement\_commands”).

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| deselectOptionByValue | sl\_Commands | “^w(ai|ebelemen)t\_commands$”, true |  |

* 1. **getNumberOfSelectedOption**

|  |  |
| --- | --- |
| Description | Count the number of options which are being selected |
| Object | TestObject object – represent a web element |
| Input | N/a |
| Output | int number – the number of selected options |

**Example** **Scenario** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to count the number of selected options of the web element. Assume that there are 2 options being selected. The output (2) is stored into variable var\_number.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| getNumberOfSelectedOption | sl\_Commands |  | var\_number |

* 1. **getNumberOfTotalOption**

|  |  |
| --- | --- |
| Description | Count the total number of options the given web element has |
| Object | TestObject object – represent a web element |
| Input | N/a |
| Output | int number – the total number of options |

**Example** **Scenario** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to count the total number of options of the web element. The output (5) is stored into variable var\_number.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| getNumberOfTotalOption | sl\_Commands |  | var\_number |

* 1. **selectAllOption**

|  |  |
| --- | --- |
| Description | Select all options |
| Object | TestObject object – represent a web element |
| Input | N/a |
| Output | N/a |

**Example** **Scenario** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to select all options.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| selectAllOption | sl\_Commands |  |  |

* 1. **selectOptionByIndex**

|  |  |
| --- | --- |
| Description | Select the options at the given indices. Index starts from 0. |
| Object | TestObject object – represent a web element |
| Input | 1. String index\_range – index range of the options to select. Index starts from 0.   Example: 2 - index 2  2,3 - index 2 and 3  2-5 - index 2 to 5 (2, 3, 4, 5) |
| Output | N/a |

**Example** **Scenario** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to select “Navigation Commands” option (index 1).

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| selectOptionByIndex | sl\_Commands | “1” |  |

Users want to select all options from index 0 to index 3.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| selectOptionByIndex | sl\_Commands | “0-3” |  |

Users want to select “Browser Commands” option (index 0) and “Wait Commands” option (index 3).

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| selectOptionByIndex | sl\_Commands | “0,3” |  |

* 1. **selectOptionByLabel**

|  |  |
| --- | --- |
| Description | Select the options with the given label (displayed text) |
| Object | TestObject object – represent a web element |
| Input | 1. String label – displayed text of the options to select 2. boolean isRegex – **true** if label is regular expression, **false** if not |
| Output | N/a |

**Example** **Scenario** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to select “Browser Commands” option and “Wait Commands” option.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| selectOptionByLabel | sl\_Commands | “Browser Commands”, false |  |

Users want to select “Wait Commands” option and “WebElement Commands” option.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| selectOptionByLabel | sl\_Commands | “^W(ai|ebElemen)t Commands$”, true |  |

* 1. **selectOptionByValue**

|  |  |
| --- | --- |
| Description | Select the options with the given value |
| Object | TestObject object – represent a web element |
| Input | 1. String value – value of the options to select 2. boolean isRegex – **true** if value is regular expression, **false** if not |
| Output | N/a |

**Example** **Scenario** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to select “Browser Commands” option (value “browser\_commands”) and “Wait Commands” option (value “wait\_commands”).

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| selectOptionByValue | sl\_Commands | “browser\_commands”, false |  |

Users want to select “Wait Commands” option (value “wait\_commands”) and “WebElement Commands” option (value “webelement\_commands”).

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| selectOptionByValue | sl\_Commands | “^w(ai|ebelemen)t\_commands”, true |  |

* 1. **verifyAnyOptionNotSelected (not implemented yet)**

|  |  |
| --- | --- |
| Description | Verify if no option is selected |
| Object | TestObject object – represent a web element |
| Input | 1. int timeout – system will wait at most timeout (seconds) to return result |
| Output | result – **true** if no option is selected; otherwise, **false** |

**Example** **Scenario** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to verify if no option is selected. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyAnyOptionNotSelected | sl\_Commands | 5 | var\_result |

* 1. **verifyAnyOptionSelected (not implemented yet)**

|  |  |
| --- | --- |
| Description | Verify if any option is selected |
| Object | TestObject object – represent a web element |
| Input | 1. int timeout – system will wait at most timeout (seconds) to return result |
| Output | boolean result – **true** if any option is selected; otherwise, **false** |

**Example** **Scenario** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to verify if any option is selected. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyAnyOptionSelected | sl\_Commands | 5 | var\_result |

* 1. **verifyOptionNotPresentByIndex (not implemented yet)**

|  |  |
| --- | --- |
| Description | Verify if the options at the given indices do NOT present. Index starts from 0. |
| Object | TestObject object – represent a web element |
| Input | 1. String index\_range – index range of options to verify if **NOT** presenting   Example: 2 - index 2  2,3 - index 2 and 3   * 1. - index 2 to 5 (2, 3, 4, 5)  1. int timeout – system will wait at most timeout (seconds) to return result |
| Output | boolean result – **true** if all options at given indices do **NOT** present; otherwise, **false** |

**Example** **Scenario 1** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to verify if options at index 0 and index 2 do **NOT** present. The output (**false**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionNotPresentByIndex | sl\_Commands | “0,2”, 5 | var\_result |

**Scenario 2** – With the above select element, users want to verify if option at index 4 and index 5 do **NOT** present. The output (**false**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionNotPresentByIndex | sl\_Commands | “4,5”, 5 | var\_result |

**Scenario 3** – With the above select element, users want to verify if option at index 5 and index 6 do **NOT** present. The output (**true**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionNotPresentByIndex | sl\_Commands | “5,6”, 5 | var\_result |

* 1. **verifyOptionNotPresentByLabel**

|  |  |
| --- | --- |
| Description | Verify if the option with the given label (displayed text) does NOT present |
| Object | TestObject object – represent a web element |
| Input | 1. String label – displayed text of the option to verify if **NOT** presenting 2. boolean isRegex – **true** if label is regular expression, **false** if not 3. int timeout – system will wait at most timeout (seconds) to return result |
| Output | boolean result – **true** if option with the given label does **NOT** present; otherwise, **false** |

**Example** **Scenario 1** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to verify if “Frame Commands” option does **NOT** present. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionNotPresentByLabel | sl\_Commands | “Frame Commands”, false, 5 | var\_result |

**Scenario 2** – With the above select element, users want to verify if “Wait Commands” option and “WebElement Commands” option do **NOT** present. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionNotPresentByLabel | sl\_Commands | “^W(ai|ebelemn)t Commands”, true, 5 | var\_result |

* 1. **verifyOptionNotPresentByValue**

|  |  |
| --- | --- |
| Description | Verify if the option with the given value does NOT present |
| Object | TestObject object – represent a web element |
| Input | 1. String value – value of the option to verify if **NOT** presenting 2. boolean isRegex – **true** if value is regular expression, **false** if not 3. int timeout – system will wait at most timeout (seconds) to return result |
| Output | boolean result – **true** if option with the given value does **NOT** present; otherwise, **false** |

**Example** **Scenario 1** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to verify if option with value “browser\_commands” does **NOT** present. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionNotPresentByValue | sl\_Commands | “browser\_commands”, false, 5 | var\_result |

Users want to verify if options with values “table\_commands” and “frame\_commands” do **NOT** present. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionNotPresentByValue | sl\_Commands | “^(table|frame)\_commands$”, true, 5 | var\_result |

* 1. **verifyOptionNotSelectedByIndex**

|  |  |
| --- | --- |
| Description | Verify if the options at the given indices are NOT selected. Index starts from 0. |
| Object | TestObject object – represent a web element |
| Input | 1. String index\_range – index range of the options to be verify if **NOT** being selected   Example: 2 - index 2  2,3 - index 2 and 3  2-5 - index 2 to 5 (2, 3, 4, 5)   1. int timeout – system will wait at most timeout (seconds) to return result |
| Output | boolean result – **true** if all options in the given index range are **NOT** selected; otherwise, **false** |

**Example** **Scenario** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to verify if “Navigation Commands” option (index 1) **NOT** selected. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionNotSelectedByIndex | sl\_Commands | “1”, 5 | var\_result |

Users want to verify if “Browser Commands” option (index 0) and “Wait Commands” option (index 3) are **NOT** selected. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionNotSelectedByIndex | sl\_Commands | “0,3”, 5 | var\_result |

Users want to verify if all options from index 0 to index 3 are **NOT** selected. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionNotSelectedByIndex | sl\_Commands | “0-3”, 5 | var\_result |

* 1. **verifyOptionNotSelectedByLabel**

|  |  |
| --- | --- |
| Description | Verify if the option with the given label (displayed text) is NOT selected |
| Object | TestObject object – represent a web element |
| Input | 1. String label – displayed text of the option to verify if **NOT** being selected 2. boolean isRegex – **true** if label is regular expression, **false** if not 3. int timeout – system will wait at most timeout (seconds) to return result |
| Output | boolean result – **true** if option with the given label is **NOT** selected; otherwise, **false** |

**Example** **Scenario** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to verify if “Browser Commands” option is **NOT** selected. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionNotSelectedByLabel | sl\_Commands | “Browser Commands”, false, 5 | var\_result |

Users want to verify if “Wait Commands” and “WebElement Commands” options are **NOT** selected. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionNotSelectedByLabel | sl\_Commands | “^W(ai|ebElemen)t Commands”, true, 5 | var\_result |

* 1. **verifyOptionNotSelectedByValue**

|  |  |
| --- | --- |
| Description | Verify if the option with the given value is NOT selected |
| Object | TestObject object – represent a web element |
| Input | 1. String value – value of the option to verify if **NOT** being selected 2. boolean isRegex – **true** if value is regular expression, **false** if not 3. int timeout – system will wait at most timeout (seconds) to return result |
| Output | boolean result – **true** if the option with the given value is **NOT** selected; otherwise, **false** |

**Example** **Scenario** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to verify if option with value “browser\_commands” is **NOT** selected. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionNotSelectedByValue | sl\_Commands | “browser\_commands”, false, 5 | var\_result |

Users want to verify if options with value “wait\_commands” and “webelement\_commands” are **NOT** selected. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionNotSelectedByValue | sl\_Commands | “^w(ai|ebelemen)t\_commands”, true, 5 | var\_result |

* 1. **verifyOptionPresentByIndex (not implemented yet)**

|  |  |
| --- | --- |
| Description | Verify if the options at the given indices present. Index starts from 0. |
| Object | TestObject object – represent a web element |
| Input | 1. String index\_range – index range of options to verify if presenting   Example: 2 - index 2  2,3 - index 2 and 3  2-5 - index 2 to 5 (2, 3, 4, 5)   1. int timeout – system will wait at most timeout (seconds) to return result |
| Output | boolean result – **true** if all options at the given indices present; otherwise, **false** |

**Example** **Scenario 1** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to verify if options at index 0 and index 3 present. The output (**true**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionPresentByIndex | sl\_Commands | “0,3”, 5 | var\_result |

**Scenario 2** – With the above select element, users want to verify if option at index 0 and index 5 present. The output (**false**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionPresentByIndex | sl\_Commands | “0,5”, 5 | var\_result |

* 1. **verifyOptionPresentByLabel**

|  |  |
| --- | --- |
| Description | Verify if the option with the given label (displayed text) presents |
| Object | TestObject object – represent a web element |
| Input | 1. String label – displayed text of the option to verify if presenting 2. boolean isRegex – **true** if label is regular expression, **false** if not 3. int timeout – system will wait at most timeout (seconds) to return result |
| Output | result – **true** if the option with the given label presents; otherwise, **false** |

**Example** **Scenario 1** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to verify if “Browser Commands” option presents. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionPresentByLabel | sl\_Commands | “Browser Commands”, false, 5 | var\_result |

**Scenario 2** – With the above select element, users want to verify if “Wait Commands” option and “WebElement Commands” option present. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionPresentByLabel | sl\_Commands | “^W(ai|ebelemen)t Commands$”, true, 5 | var\_result |

* 1. **verifyOptionPresentByValue**

|  |  |
| --- | --- |
| Description | Verify if the option with the given value presents |
| Object | TestObject object – represent a web element |
| Input | 1. String value – value of the option to verify if presenting 2. boolean isRegex – **true** if value is regular expression, **false** if not 3. int timeout – system will wait at most timeout (seconds) to return result |
| Output | boolean result – **true** if the option with the given value presents; otherwise, **false** |

**Example** **Scenario 1** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to verify if options with value “browser\_commands” presents. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionPresentByValue | sl\_Commands | “browser\_commands”, false, 5 | var\_result |

**Scenario 2** – With the above select element, users want to verify if option with value “wait\_commands” and “webelement\_commands” present. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionPresentByValue | sl\_Commands | “^w(ai|ebelemen)t\_commands$”, true, 5 | var\_result |

* 1. **verifyOptionSelectedByIndex**

|  |  |
| --- | --- |
| Description | Verify if the options in the given index range are selected. Index starts from 0. |
| Object | TestObject object – represent a web element |
| Input | 1. String index\_range – index range of the options to verify if being selected   Example: 2 - index 2  2,3 - index 2 and 3  2-5 - index 2 to 5 (2, 3, 4, 5)   1. int timeout – system will wait at most timeout (seconds) to return result |
| Output | boolean result – **true** if all options in the given index range are selected; otherwise, **false** |

**Example** **Scenario** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to verify if “Navigation Commands” option (index 1) is selected. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionSelectedByIndex | sl\_Commands | “1”, 5 | var\_result |

Users want to verify if “Browser Commands” option (index 0) and “Wait Commands” option (index 3) are selected. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionSelectedByIndex | sl\_Commands | “0,3”, 5 | var\_result |

Users want to verify all options from index 0 to index 3 are selected. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionSelectedByIndex | sl\_Commands | “0-3”, 5 | var\_result |

* 1. **verifyOptionSelectedByLabel**

|  |  |
| --- | --- |
| Description | Verify if the option with the given label (displayed text) is selected |
| Object | TestObject object – represent a web element |
| Input | 1. String label – displayed text of the option to verify if being selected 2. boolean isRegex – **true** if label is regular expression, **false** if not 3. int timeout – system will wait at most timeout (seconds) to return result |
| Output | boolean result – **true** if the option with the given label is selected; otherwise, **false** |

**Example** **Scenario** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to verify if “Browser Commands” option is selected. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionSelectedByLabel | sl\_Commands | “Browser Commands”, false, 5 | var\_result |

Users want to verify if “Wait Commands” option and “WebElement Commands” option are selected. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionSelectedByLabel | sl\_Commands | “^W(ai|ebElemen)t Commands$”, true, 5 | var\_result |

* 1. **verifyOptionSelectedByValue**

|  |  |
| --- | --- |
| Description | Verify if the option with the given value is selected |
| Object | TestObject object – represent a web element |
| Input | 1. String value – value of the option to verify if being selected 2. boolean isRegex – **true** if value is regular expression, **false** if not 3. int timeout – system will wait at most timeout (seconds) to return result |
| Output | boolean result – **true** if the option with the given value is selected; otherwise, **false** |

**Example** **Scenario** – There is a select element with the following html code:

<select id="selenium\_commands" class="input-xlarge" name="selenium\_commands" multiple="multiple">

<option value="browser\_commands">Browser Commands</option>

<option value="navigation\_commands">Navigation Commands</option>

<option value="switch\_commands">Switch Commands</option>

<option value="wait\_commands">Wait Commands</option>

<option value="webelement\_commands">WebElement Commands</option>

</select>

Users want to verify if option with value “browser\_commands” is selected. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionSelectedByValue | sl\_Commands | “browser\_commands”, false, 5 | var\_result |

Users want to verify if options with value “wait\_commands” and value “webelement\_commands” are selected. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyOptionSelectedByValue | sl\_Commands | “^w(ai|ebelemen)t\_commands”, true, 5 | var\_result |

1. **Utility Action**
   1. **callTestCase**

|  |  |
| --- | --- |
| Description | Call another test case |
| Object | N/a |
| Input | 1. tc\_id – ID of the test case to be called |
| Output | N/a |

**Example** **Scenario** – Users want to call “Login” test case. This test case resides inside folder “Common Test Cases”.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| callTestCase |  | “Test Cases/Common Test Cases/Login” |  |

* 1. **concatenate**

|  |  |
| --- | --- |
| Description | Concatenate all strings end-to-end into one string |
| Object | N/a |
| Input | 1. String[] string\_list – list of strings to concatenate |
| Output | String result – the end-to-end joined string of the input |

**Example** **Scenario** – Users want to concatenate two strings “KMS” and “ Technology” into one string “KMS Technology”. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| concatenate |  | [“KMS”, “ Technology”] | var\_result |

* 1. **delay**

|  |  |
| --- | --- |
| Description | Delay the execution for the specified seconds |
| Object | N/a |
| Input | 1. int seconds – the amount of seconds to delay |
| Output | N/a |

**Example** **Scenario** – Users want to delay the execution for 10 seconds.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| delay |  | 10 |  |

* 1. **modifyObjectProperty**

|  |  |
| --- | --- |
| Description | Modify existing property or add new property to test object. Use when test object has attributes changing in runtime. This keyword does not modify the object saved in Object Repository, instead, it creates another test object, modifies and returns this test object. Hence, users must use a variable to store the returned object. |
| Object | TestObject – represent a web element |
| Input | 1. String propertyName - name of the property, for example, xpath, id, name, ... If the property already exists in the object, the keyword will modify its related artifacts; if not, the keyword will add new property.  2. String matchCondition - condition to match property name with property value, for example, equals, not equals, ... In case the property already exists, input **null** to this argument to keep the old value of match condition.  3. String propertyValue - value of the property. In case the property already exists, input **null** to this argument to keep the old property value.  4. boolean isActive - **true** if the property is checked (used to find the test object); otherwise, **false.** In case the property already exists, input **null** to this argument to keep the old value. |
| Output | TestObject – the modified object |

**Example** **Scenario** – Users want to modify property of an object and click on this modified object.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| modifyObjectProperty | btn\_Temp | “xpath”; “equals”; “//\*[@type=’button’]”; true | var\_ModifiedBtn |
| click | var\_ModifiedBtn |  |  |

* 1. **removeObjectProperty**

|  |  |
| --- | --- |
| Description | Remove existing property of test object. Use when test object has attributes changing in runtime. This keyword does not modify the object saved in Object Repository, instead, it creates another test object, modify and return this test object. Hence, users must use a variable to store the returned object. |
| Object | TestObject – represent a web element |
| Input | 1. String propertyName - name of the property, for example, xpath, id, name, ... |
| Output | TestObject – the modified object |

**Example** **Scenario** – Users want to remove property of an object and click on this modified object.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| removeObjectProperty | btn\_Temp | “class” | var\_ModifiedBtn |
| click | var\_ModifiedBtn |  |  |

* 1. **takeScreenshot**

|  |  |
| --- | --- |
| Description | Take screenshot of the browser |
| Object | N/a |
| Input | N/a |
| Output | N/a |

**Example** **Scenario**  – Users want to take a screenshot of the browser at current state

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| takeScreenshot |  |  |  |

* 1. **verifyEqual**

|  |  |
| --- | --- |
| Description | Verify if two numbers are equal |
| Object | N/a |
| Input | 1. number1 – first number  2. number2 – second number |
| Output | boolean result – **true** if number1 equals number2; otherwise, **false** |

**Example** **Scenario 1** – Users want to verify if number 10 equals number 10, the result (**true**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyEqual |  | 10; 10 | var\_result |

**Scenario 2** – Users want to verify if number 10 equals number 11, the result (**false**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyEqual |  | 10; 11 | var\_result |

* 1. **verifyGreaterThan**

|  |  |
| --- | --- |
| Description | Verify if the first number is greater than the second number |
| Object | N/a |
| Input | 1. number1 – first number  2. number2 – second number |
| Output | boolean result – **true** if number1 is greater than number2; otherwise, **false** |

**Example** **Scenario 1** – Users want to verify if number 10 is greater than number 9, the result (**true**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyGreaterThan |  | 10; 9 | var\_result |

**Scenario 2** – Users want to verify if number 10 is greater than number 11, the result (**false**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyGreaterThan |  | 10; 11 | var\_result |

**Scenario 3** – Users want to verify if number 10 is greater than number 10, the result (**false**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyGreaterThan |  | 10; 10 | var\_result |

* 1. **verifyGreaterThanOrEqual**

|  |  |
| --- | --- |
| Description | Verify if the first number is greater than or equal the second number |
| Object | N/a |
| Input | 1. number1 – first number  2. number2 – second number |
| Output | boolean result – **true** if number1 is greater than or equal number2; otherwise, **false** |

**Example** **Scenario 1** – Users want to verify if number 10 is greater than or equal number 9, the result (**true**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyGreaterThanOrEqual |  | 10; 9 | var\_result |

**Scenario 2** – Users want to verify if number 10 is greater than or equal number 10, the result (**true**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyGreaterThanOrEqual |  | 10; 10 | var\_result |

**Scenario 3** – Users want to verify if number 10 is greater than or equal number 11, the result (**false**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyGreaterThanOrEqual |  | 10; 11 | var\_result |

* 1. **verifyLessThan**

|  |  |
| --- | --- |
| Description | Verify if the first number is less than the second number |
| Object | N/a |
| Input | 1. number1 – first number  2. number2 – second number |
| Output | boolean result – **true** if number1 is less than number2; otherwise, **false** |

**Example** **Scenario 1** – Users want to verify if number 10 is less than number 9, the result (**false**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyLessThan |  | 10; 9 | var\_result |

**Scenario 2** – Users want to verify if number 10 is less than number 10, the result (**false**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyLessThan |  | 10; 10 | var\_result |

**Scenario 3** – Users want to verify if number 10 is less than number 11, the result (**true**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyLessThan |  | 10; 11 | var\_result |

* 1. **verifyLessThanOrEqual**

|  |  |
| --- | --- |
| Description | Verify if the first number is less than or equal the second number |
| Object | N/a |
| Input | 1. number1 – first number  2. number2 – second number |
| Output | boolean result – **true** if number1 is less than or equal number2; otherwise, **false** |

**Example** **Scenario 1** – Users want to verify if number 10 is less than or equal number 9, the result (**false**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyLessThanOrEqual |  | 10; 9 | var\_result |

**Scenario 2** – Users want to verify if number 10 is less than or equal number 10, the result (**true**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyLessThanOrEqual |  | 10; 10 | var\_result |

**Scenario 3** – Users want to verify if number 10 is less than or equal number 11, the result (**true**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyLessThanOrEqual |  | 10; 11 | var\_result |

* 1. **verifyMatch**

|  |  |
| --- | --- |
| Description | Verify if two strings match each other, the second string can be a regular expression |
| Object | N/a |
| Input | 1. String string1 – first string  2. String string2 – second string  3. boolean isRegex – **true** if string2 is regular expression; otherwise, **false** |
| Output | boolean result – **true** if string1 matches string2; otherwise, **false** |

**Example** **Scenario 1** – Users want to verify if string “KMS Technology” matches string “KMS Tech.\*”, not using regular expression. The output (**false**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyMatch |  | “KMS Technology”; “KMS Tech.\*”;false | var\_result |

**Scenario 2** – Users want to verify if string “KMS Technology” matches string “KMS Tech.\*”, using regular expression. The output (**true**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyMatch |  | “KMS Technology”; “KMS Tech.\*”; true | var\_result |

* 1. **verifyNotEqual**

|  |  |
| --- | --- |
| Description | Verify if two numbers are NOT equal |
| Object | N/a |
| Input | 1. number1 – first number  2. number2 – second number |
| Output | boolean result – **true** if number1 is **NOT** equal number2; otherwise, **false** |

**Example** **Scenario 1** – Users want to verify if number 10 is **NOT** equal number 10, the result (**false**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyNotEqual |  | 10; 10 | var\_result |

**Scenario 2** – Users want to verify if number 10 is **NOT** equal number 11, the result (**true**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyNotEqual |  | 10; 11 | var\_result |

* 1. **verifyNotMatch**

|  |  |
| --- | --- |
| Description | Verify if two strings do NOT match each other, the second string can be a regular expression |
| Object | N/a |
| Input | 1. String string1 – first string  2. String string2 – second string  3. boolean isRegex – **true** if string2 is regular expression; otherwise, **false** |
| Output | result – **true** if string1 does **NOT** match string2; otherwise, **false** |

**Example** **Scenario 1** – Users want to verify if string “KMS Technology” does not match “KMS Tech.\*”, not using regular expression. The output (**true**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyNotMatch |  | “KMS Technology”; “KMS Tech.\*”;false | var\_result |

**Scenario 2** – Users want to verify if string “KMS Technology” does not match “KMS Tech.\*”, using regular expression. The output (**false**) is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyNotMatch |  | “KMS Technology”; “KMS Tech.\*”; true | var\_result |

1. **Web Element Action**
   1. **check**

|  |  |
| --- | --- |
| Description | Check a toggle-button (check-box/radio-button) |
| Object | TestObject object – represent a web element |
| Input | N/a |
| Output | N/a |

**Example** **Scenario** – Users want to check the check-box “chkbx\_KeepLogin”.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| check | chkbx\_KeepLogin |  |  |

* 1. **focus**

|  |  |
| --- | --- |
| Description | Move the focus to the specified element; for example, if the element is an input field, move the cursor to that field |
| Object | TestObject object – represent a web element |
| Input | N/a |
| Output | N/a |

**Example** **Scenario** – Users want to focus on text-box “txtbx\_Username”.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| focus | txtbx\_Username |  |  |

* 1. **getAttribute**

|  |  |
| --- | --- |
| Description | Get attribute value of a web element |
| Object | TestObject object – represent a web element |
| Input | 1. String attribute – name of the attribute |
| Output | String value – value of the attribute |

**Example** **Scenario** – Users want to get value of the attribute “type” of the text-box “txtbx\_Username” which has the following html code:

<input type="text" value="" name="user">. The output (“text”) is stored into variable var\_output.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| getAttribute | txtbx\_Username | “type” | var\_output |

* 1. **getText**

|  |  |
| --- | --- |
| Description | Get the visible (i.e. not hidden by CSS) innerText of the web element, including sub-elements, without any leading or trailing whitespace. |
| Object | TestObject object – represent a web element |
| Input | N/a |
| Output | String text – innerText of the web element |

**Example** **Scenario** – Users want to get text value of the link “lnk\_Register” which has the following html code:

<a href="register.htm">Register</a>. The output (“Register”) is stored into variable var\_output.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| getText | lnk\_Register |  | var\_output |

* 1. **setText**

|  |  |
| --- | --- |
| Description | Set the value of an input field, as though you type it in. It also clears the previous value of the input field |
| Object | TestObject object – represent a web element |
| Input | 1. String value – the text to type |
| Output | N/a |

**Example** **Scenario** – The input filed “txtbx\_Username” already has text “user1”. Users want to set text value to “user2”. After the action is executed, input field will have text “user2”, not “user1user2”.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| setText | txtbx\_Username | “user2” |  |

* 1. **uncheck**

|  |  |
| --- | --- |
| Description | Uncheck a toggle-button (check-box/radio-button) |
| Object | TestObject object – represent a web element |
| Input | N/a |
| Output | N/a |

**Example:** **Scenario** – Users want to uncheck the check-box “chkbx\_KeepLogin”.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| uncheck | chkbx\_KeepLogin |  |  |

* 1. **verifyElementChecked**

|  |  |
| --- | --- |
| Description | Verify if the given web element is checked |
| Object | TestObject object – represent a web element |
| Input | 1. int timeout – system will wait at most timeout (seconds) to return result |
| Output | boolean result – **true** if the element is checked; otherwise, **false** |

**Example** **Scenario** – Users want to verify if the check-box “chkbx\_KeepLogin” is checked. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyElementChecked | chkbx\_KeepLogin | 5 | var\_result |

* 1. **verifyElementEnabled (not implemented yet)**

|  |  |
| --- | --- |
| Description | Verify if the given web element is enabled |
| Object | TestObject object – represent a web element |
| Input | 1. int timeout – system will wait at most timeout (seconds) to return result |
| Output | boolean result – **true** if the element is enabled; otherwise, **false** |

**Example** **Scenario** – Users want to verify if the check-box “chkbx\_KeepLogin” is enabled. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyElementEnabled | chkbx\_KeepLogin | 5 | var\_result |

* 1. **verifyElementNotChecked**

|  |  |
| --- | --- |
| Description | Verify if the given web element is NOT checked |
| Object | TestObject object – represent a web element |
| Input | 1. int timeout – system will wait at most timeout (seconds) to return result |
| Output | boolean result – **true** if the element is **NOT** checked; otherwise, **false** |

**Example** **Scenario** – Users want to verify if the check-box “chkbx\_KeepLogin” is **NOT** checked. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyElementNotChecked | chkbx\_KeepLogin | 5 | var\_result |

* 1. **verifyElementNotEnabled (not implemented yet)**

|  |  |
| --- | --- |
| Description | Verify if the given web element is NOT enabled |
| Object | TestObject object – represent a web element |
| Input | 1. int timeout – system will wait at most timeout (seconds) to return result |
| Output | result – **true** if the element is NOT enabled; otherwise, **false** |

**Example** **Scenario** – Users want to verify if the check-box “chkbx\_KeepLogin” is NOT enabled. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyElementNotEnabled | chkbx\_KeepLogin | 5 | var\_result |

* 1. **verifyElementNotPresent**

|  |  |
| --- | --- |
| Description | Verify if the given web element does NOT present. |
| Object | TestObject object – represent a web element |
| Input | 1. int timeout – system will wait at most timeout (seconds) to return result |
| Output | boolean result – **true** if the element does **NOT** present; otherwise, **false** |

**Example** **Scenario** – Users want to verify if the check-box “chkbx\_KeepLogin” does **NOT** present. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyElementNotPresent | chkbx\_KeepLogin | 5 | var\_result |

* 1. **verifyElementNotVisible (not implemented yet)**

|  |  |
| --- | --- |
| Description | Verify if the given web element is NOT visible |
| Object | TestObject object – represent a web element |
| Input | 1. int timeout – system will wait at most timeout (seconds) to return result |
| Output | boolean result – **true** if the element is **NOT** visible; otherwise, **false** |

**Example** **Scenario** – Users want to verify if the check-box “chkbx\_KeepLogin” is **NOT** visible. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyElementNotVisible | chkbx\_KeepLogin | 5 | var\_result |

* 1. **verifyElementPresent**

|  |  |
| --- | --- |
| Description | Verify if the given web element presents |
| Object | TestObject object – represent a web element |
| Input | 1. int timeout – system will wait at most timeout (seconds) to return result |
| Output | boolean result – **true** if the element presents; otherwise, **false** |

**Example** **Scenario** – Users want to verify if the check-box “chkbx\_KeepLogin” presents. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| verifyElementPresent | chkbx\_KeepLogin | 5 | var\_result |

* 1. **waitForElementClickable**

|  |  |
| --- | --- |
| Description | Wait until the given web element is clickable within timeout. |
| Object | TestObject object – represent a web element |
| Input | 1. int timeout – how many seconds to wait (maximum) |
| Output | boolean result – **true** if the element is clickable; otherwise, **false** |

**Example** **Scenario** – Users want to wait until the check-box “chkbx\_KeepLogin” is clickable within 5 seconds. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| waitForElementClickable | chkbx\_KeepLogin | 5 | var\_result |

* 1. **waitForElementNotPresent**

|  |  |
| --- | --- |
| Description | Wait for the given element to NOT present (disappear) within the given time in second unit |
| Object | TestObject object – represent a web element |
| Input | 1. int timeout – system will wait at most timeout (seconds) to return result |
| Output | N/a |

**Example** **Scenario** – Users want to wait for the check-box “chkbx\_KeepLogin” to **NOT** present (disappear) within 10 seconds.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| waitForElementNotPresent | chkbx\_KeepLogin | 10 |  |

* 1. **waitForElementPresent**

|  |  |
| --- | --- |
| Description | Wait for the given element to present (appear) within the given time in second unit |
| Object | TestObject object – represent a web element |
| Input | 1. int timeout – system will wait at most timeout (seconds) to return result |
| Output | N/a |

**Example** **Scenario** – Users want to wait for the check-box “chkbx\_KeepLogin” to present (appear) within 10 seconds.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| waitForElementPresent | chkbx\_KeepLogin | 10 |  |

* 1. **waitForElementVisible**

|  |  |
| --- | --- |
| Description | Wait until the given web element is visible within timeout. |
| Object | TestObject object – represent a web element |
| Input | 1. int timeout – how many seconds to wait (maximum) |
| Output | boolean result – **true** if the element is visible; otherwise, **false** |

**Example** **Scenario** – Users want to wait until the check-box “chkbx\_KeepLogin” is visible within 5 seconds. The output is stored into variable var\_result.

|  |  |  |  |
| --- | --- | --- | --- |
| Item | Object | Input | Output |
| waitForElementVisible | chkbx\_KeepLogin | 5 | var\_result |