**My First Selenium IDE Script – Selenium Tutorial #3**

Before jumping on to the creation of Selenium IDE script, let us take a moment to introduce elementary information about the application under test (AUT).

As a specimen, we would be using “Gmail” – an email service designed by Google. I believe because of its unbounded popularity, it needs no more introductions. The URL we would be using is “https://accounts.google.com”. I have also created dummy credentials to represent test data.

**Creating First Selenium IDE Script**

So let us now create our first script using Selenium IDE.

**The entire script creation process can be classified into 3 chunks:**

**Process #1:** **Recording** – Selenium IDE aids the user to record user interactions with the browser and thus the recorded actions as a whole are termed as Selenium IDE script.

**Process #2:** **Playing back** – In this section, we execute the recorded script so as to verify and monitor its stability and success rate.

**Process #3:** **Saving** – Once we have recorded a stable script, we may want to save it for future runs and regressions.

Let us now see their implementation.

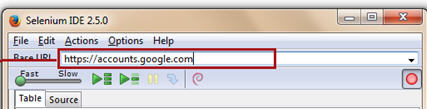
**Process #1: Recording a test script**

**Scenario**

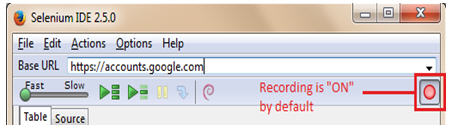
* Open “https://accounts.google.com”.
* Assert Title of the application
* Enter a valid username and password and submit the details to login.
* Verify that the user is re-directed to the Home page.

**Step 1 –**Launch the Firefox and open Selenium IDE from the menu bar.

**Step 2 –** Enter the address of application under test (“https://accounts.google.com”) inside the Base URL textbox.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-IDE-script-1.jpg)

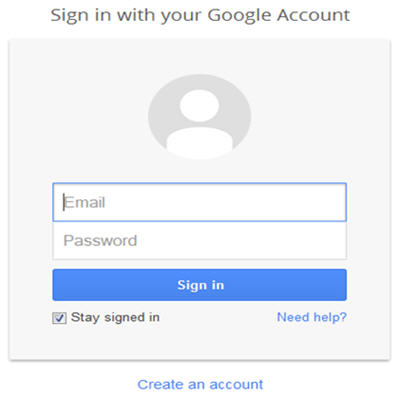
**Step 3 –** By default, the Record button is in ON state. Remember to tune it ON if it is in OFF state so as to enable the recording mode.

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-IDE-script-2.jpg)

**Step 4 –** Open the application under test (https://accounts.google.com) in the Firefox.

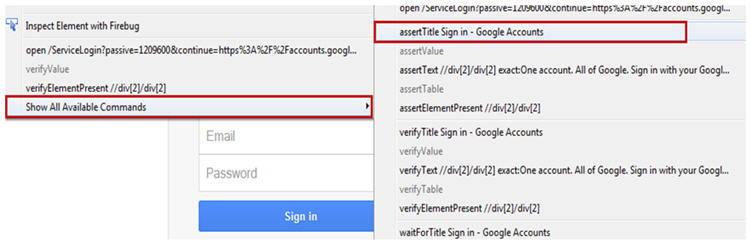
[seleniumsmita@gmail.com](mailto:seleniumsmita@gmail.com)

admin@123

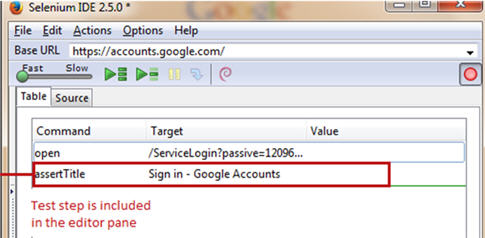
[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-IDE-script-3.jpg)

**Step 5 –** Verify if the application title is correct. To do so, right click anywhere on the page except the hyperlinks or images. The right click opens the Selenium IDE context menu listing few of the commands. To get an entire list, select “Show Available Commands” option. This will open another menu containing rest of the available and applicable commands. Select “assertTitle Sign in – Google Accounts” option to verify the page title.

***(Click to view enlarged image)***

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-IDE-script-4.jpg)

As soon as we click on “assertTitle Sign in – Google Accounts” option, a test step would be included /appended in the Selenium IDE editor.

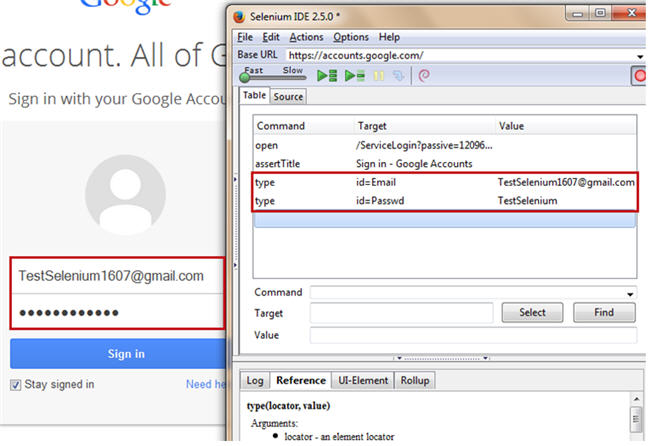
[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-IDE-script-5.jpg)

**Step 6 –** Enter a valid username in the “Email” Textbox of Gmail.

**Step 7 –** Enter a valid password in the “Password” Textbox of Gmail.

The simulation of the same user actions can be seen in the Selenium IDE test editor.

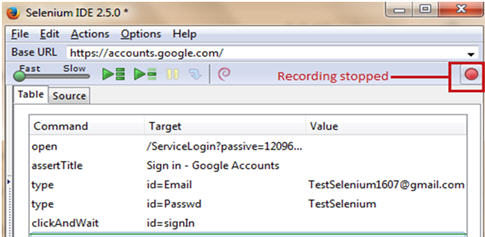
Notice that for the ease of understanding, I have already created test credentials. I would strictly advise the readers to create their own credentials instead of using these.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-IDE-script-6.jpg)

**Step 8 –** Click on the “Sign in” button to complete the login process.

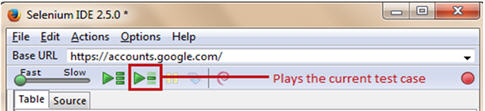
User should be re-directed to the home page provided the credentials entered are correct.

**Step 9 –** At the end, we would end the recording session by tuning the record button into OFF state. Below is the recorded script.

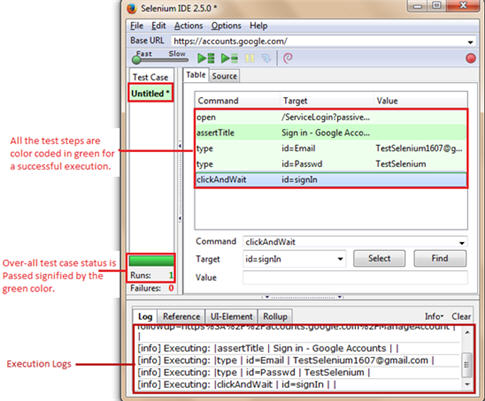
[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-IDE-script-9.jpg)

**Process #2: Playing back / executing a test script**

Now that we have created our first Selenium IDE script, we would want to execute it to see if the script is stable enough. Click on the playback button to execute the script.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-IDE-script-10.jpg)

Post execution, all the test steps would be color coded in green for the successful run. The same would be evitable from the test case pane.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-IDE-script-11.jpg)

For unsuccessful execution or test case failure, the failed test step would be highlighted in red. And the test case pane would mark the test case execution as failure.

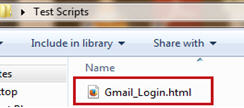
**Process #3: Saving a test script**

Once, we have played back the script, now it’s time to save the created test script.

**Step 1 –** To save the test script, Click on the File menu and select “Save Test Case” option.

**Step 2 –** The system will prompt us to browse or enter the desired location to save our test case and to provide the test script name. Furnish the test name as “Gmail\_Login” and click on the “Save” button.

The test script can be found at the location provided in the above step. Notice that the test script is saved in HTML format.

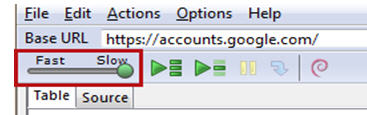
[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-IDE-script-12.jpg)

**Using Common features of Selenium IDE**

**Setting Execution speed**

While testing web applications, we come across several scenarios where an action performed may trigger a page load. Thus we must be cognizant enough while dealing such scenarios.

So to avoid failures while playing back these test scenarios, we can set the execution speed to be minimal. Refer the following figure for the same.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-IDE-script-13.jpg)

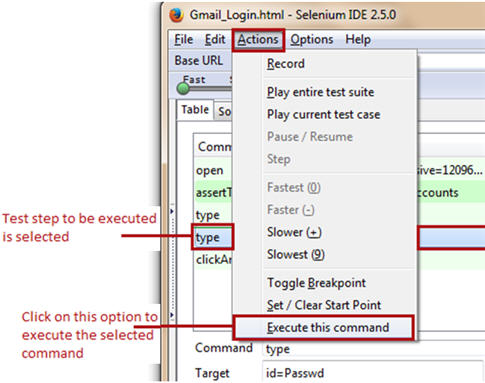
**Using “Execute this command” option**

Selenium IDE allows the user to execute a single test step within the entire test script without executing the entire test script. “Execute this command” is the option which makes this obtainable.

“Execute this command” option can be used at times when we want to debug/see the behavior of a particular test step.

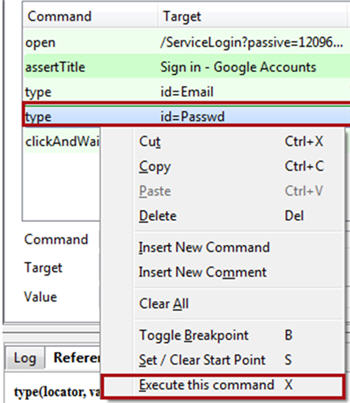
**“Execute this command” option can be used in the following four ways:**

**#1.** Using Actions tab from the Menu bar

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-IDE-script-14.jpg)

**#2.** Using short cut key (“X”)

**#3.** Right click the test step and select “Execute this command”

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-IDE-script-15.jpg)

**#4.** Double click the test step

In all the above cases, user is expected to select the test step which he / she want to execute.

**Steps to be followed:**

**Step 1 –**Launch the web browser and open the target URL (“https://accounts.google.com”), Select the test step that we desire to execute. Remember to open correct web page to mitigate the chances of failure.

**Step 2 –** Press “X” to execute the selected test step. Alternatively, one can use other ways too.

**Step 3 –** Notice that the selected test step is executed. The test step would be color coded in green for success or red for failure. At the same time, the test step would be simulated into an action on the web browser.

Note that the user is responsible to bring the script before executing the test step and Firefox in context. There is a probability of failure if the user has not opened the legitimate web page.

**Using Start point**

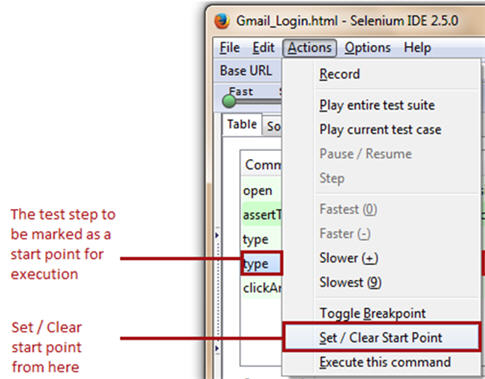
Selenium IDE allows the user to specify a start point within a test script. The start point points to the test step from where we wish to start the test script execution.

Start point can be used at times when we do not desire to execute the entire test script starting from the beginning rather we customize the script to execute from a certain step.

**Start point can be set and clear in the following three ways:**

**#1.** Using Actions tab from the Menu bar

------------

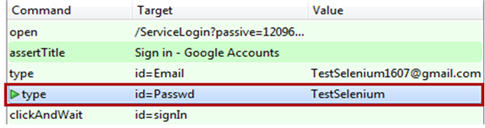
[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-IDE-script-16.jpg)

**#2.** Using short cut key (“S”)

**#3.** Right click the test step and select “Set/Clear Start Point”. Menu similar to above image will be displayed.

In all the above cases, user is expected to select the test step from where he wants to start the execution prior to setting start point.

As soon as the user has marked the test step to indicate the start point, an icon gets affixed to it.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-IDE-script-18.jpg)

Now whenever we execute the test script, it execution would be started from the start point i.e. fourth line (type | id=Passwd | TestSelenium) of the test script.

**Notes**

* There can be one and only one start point in a single script.
* The start point can be cleared in the same way it was set.
* User is responsible to bring the script after applying start point and Firefox in context. There is a probability of failure if the user has not opened the legitimate web page.

**Using Break point**

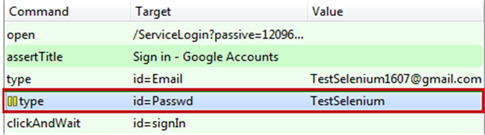
Selenium IDE allows the user to specify break points within a test script. The break points indicate Selenium IDE where to pause the test script.

Break points can be used at times when we desire to break the execution in smaller logical chunks to witness the execution trends.

**Break point can be set and clear in the following three ways:**

* Using Actions tab from the Menu bar
* Right click the test step and select “Toggle Breakpoint”.
* Using short cut key (“B”)

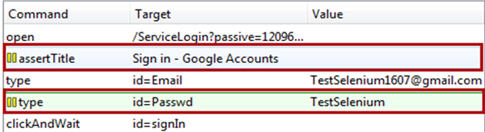
As soon as the user has marked the test step to indicate the break point, an icon gets affixed to it.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-IDE-script-19.jpg)

Now whenever we execute the test script, the execution pauses at the break point i.e. fourth line (type | id=Passwd | TestSelenium) of the test script.

**Apply multiple breakpoints**

Selenium IDE allows user to apply multiple breakpoints in a single test script. Once the first section of the test script is executed, the script pauses as and when the breakpoint is reached. To execute the subsequent test steps, user is required to execute each of the test steps explicitly.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-IDE-script-20.jpg)

In the above test script, the execution pauses at the line “assertTitle | Sign in – Google Accounts”. After explicitly executing this test step, the control moves to the next test step in sequence “type | id=Email | TestSelenium1607@gmail.com”. Thus, user needs to explicitly execute this test step. The similar trend is followed for rest of the subsequent steps.

Thus, this feature lets the user to spend more time executing each step and reviewing the outcomes of the previously executed test step.

**Notes**

* There can be as many break points as you wish in a single script.
* The break point can be cleared in the same way it was set.

**Using Find Button**

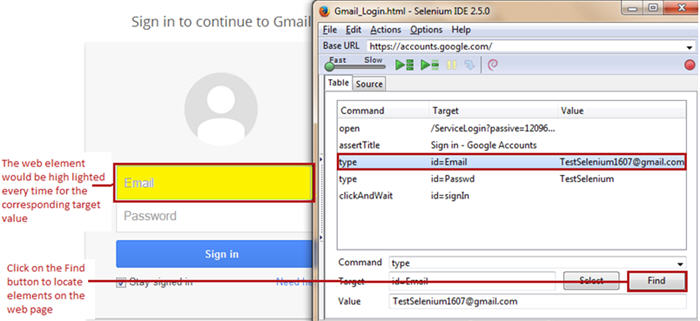
One of the most crucial aspects of Selenium IDE test scripts is to find and locate web elements within a web page. At times, there are web elements which have analogous properties associated with them, thus making it challenging for a user to identify a particular web element uniquely.

To address this issue, Selenium IDE provides Find button. The Find Button is used to ascertain that locator value provided in the Target test box is indeed correct and identifies the designated web element on the GUI.

Let us consider the above created Selenium IDE test script. Select any command and notice the target text box. Click on the Find button present just beside the Target text box.

Notice that the corresponding web element would be highlighted in yellow with a fluorescent green border around it. If no or wrong web element is highlighted, then the user is required to rectify the issue and would need to impose some other locator value.

***(Click to view enlarged image)***

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-IDE-script-21.jpg)

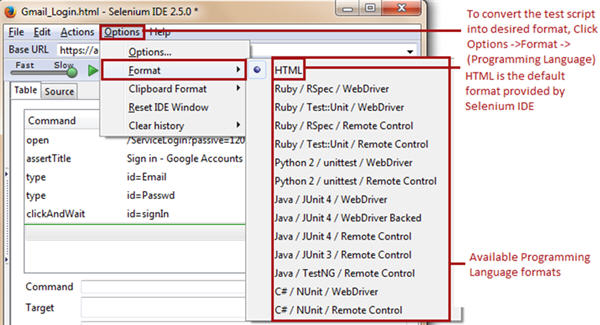
Thus, this procedure makes the user assured about the target value being used and that it corresponds to the correct web element on the GUI.

**Using Other Formats**

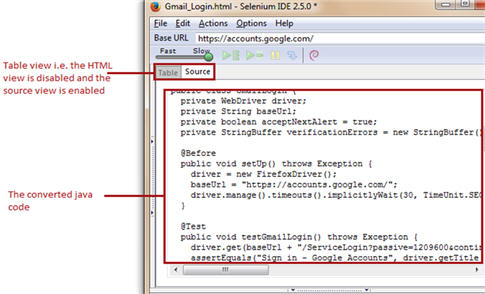
**Converting Selenium IDE test scripts to Other Programming Languages**

Selenium IDE supports conversion test scripts into set of programming languages from a default type (HTML). The converted test scripts cannot be played back using Selenium IDE until and unless it is reverted back to HTML. Thus the conversion is beneficial and constructive only when we are executing it from other tools of Selenium Suite.

**Step 1 –** Click on the options tab under the menu bar and select the programming language format under format option in order to convert the test script into our desired format.

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-IDE-script-22.jpg)

**Step 2** – As soon as we select our Desired Programming language format (“Java / JUnit4 / WebDriver” in our case), a prompt box appears which says “Changing format is now marked experimental! If you continue, recording and playback may not work, your changes may be lost and you may have to copy and paste the test in a text editor to save. It is better to make a copy of your test cases before you continue. Do you still want to proceed?” Click “OK” to continue.

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-IDE-script-23.jpg)

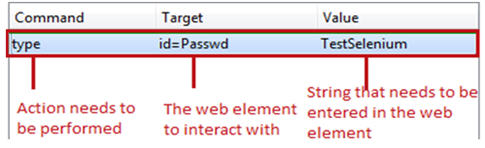
Thus, the above converted code can be executed by using WebDriver.

Mark that editing or modifying Selenium IDE test scripts from Source View is not advisable. If done so, the tool might introduce several repercussions. Several known bugs are already associated with it.

**Selenium IDE Commands**

Each Selenium IDE test step can chiefly be split into following three components:

* Command
* Target
* Value

[](http://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2014/10/Selenium-IDE-script-24.jpg)

**Types of Selenium IDE commands**

There are three flavors of Selenium IDE commands. Each of the test step in Selenium IDE falls under any of the following category.

* Actions
* Accessors
* Assertions

**Actions**

Actions are those commands which interact directly with the application by either altering its state or by pouring some test data.

For Example, “type” command lets the user to interact directly with the web elements like text box. It allows them to enter a specific value in the text box and as when the value is entered; it is showed on the UI as well.

Another example is “click” command. “click” command lets the user to manipulate with the state of the application.

In case of failure of an action type command, the test script execution halts and rest of the test steps would not be executed.

**Accessors**

Accessors are those commands which allows user to store certain values to a user defined variable. These stored values can be later on used to create assertions and verifications.

For example, “storeAllLinks” reads and stores all the hyperlinks available within a web page into a user defined variable. Remember the variable is of array type if there are multiple values to store.

**Assertions**

Assertions are very similar to Accessors as they do not interact with the application directly. Assertions are used to verify the current state of the application with an expected state.

**Forms of Assertions:**

**#1. assert** – the “assert” command makes sure that the test execution is terminated in case of failure.

**#2. verify** – the “verify” command lets the Selenium IDE to carry on with the test script execution even if the verification is failed.

**#3. waitFor** – the “waitFor” command waits for a certain condition to be met before executing the next test step. The conditions are like page to be loaded, element to be present. It allows the test execution to proceed even if the condition is not met within the stipulated waiting period.

**Commonly used Selenium IDE commands**

| **Command** | **Description** | **#Argu ments** |
| --- | --- | --- |
| open | Opens a specified URL in the browser. | 1 |
| assertTitle, VerifyTitle | Returns the current page title and compares it with the specified title | 1 |
| assertElementPresent, verifyElementPresent | Verify / Asserts the presence of an element on a web page. | 1 |
| assertTextPresent, verifyTextPresent | Verify / Asserts the presence of a text within the web page. | 1 |
| type, typeKeys,  sendKeys | Enters a value (String) in the specified web element. | 2 |
| Click, clickAt,  clickAndWait | Clicks on a specified web element within a web page. | 1 |
| waitForPageToLoad | Sleeps the execution and waits until the page is loaded completely. | 1 |
| waitForElement Present | Sleeps the execution and waits until the specified element is present | 1 |
| chooseOkOnNext Confirmation,  chooseCancelOn NextConfirmation | Click on ”OK” or “Cancel” button when next confirmation box appears. | 0 |

**Conclusion**

In this tutorial, we tried to make you acquainted with the creation of Selenium IDE scripts. We also briefed you about the usage of various Selenium features.

**Here are the cruxes of this article.**

* Test script in Selenium IDE can be created using Record and Playback feature.
* The script creation mechanism can be divided into 3 processes – **Recording, Playing back** and **Saving** the test script.
* Selenium IDE allows the user to execute a single test step within the test script without executing the entire test script. “**Execute this command**” is the option which makes this obtainable.
* User is leveraged to set the execution speed from the option within the toolbar.
* User can define any test step as a **Start point**. Thus, the execution will always initiate from that particular test step only.
* User can set multiple **Break points** to pause the execution at a certain test step.
* User can find and verify if the provided target value corresponds to the correct web element within the web page using **Find**
* Changing the source view to other formats is not recommended as there is a probability of loss of data.
* Remember to keep a copy of HTML test script before converting the test script into other non HTML formats.
* There are majorly three types of commands – **Actions, Accessors**and**Assertions**.
* Actions directly interact with the application and alter its state.
* Accessors are used to store an elements property in a user defined variable.
* Assertions are used to check if a specified condition is met or not.
* Assertions can further be categorized as **verify, assert**and**waitFor**commands**.**
* Verify makes sure that the test script execution is never halted if even if the verification fails.
* Assert lets no further execution of the test script in case of failure.
* WaitFor waits for a stipulated period for a certain condition to meet.
* **Some of the Selenium IDE commands which are used commonly are:**
  + open
  + assertTitle / VerifyTitle
  + AssertForElementPresent / VerifyForElementPresent
  + AssertForTextPresent / VerifyForTextPresent
  + type / typeAndWait / sendKeys
  + click /clickAt / clickAndWait
  + waitForPageToLoad
  + waitForElementPresent
  + chooseOkOnNextConfirmation / chooseCancelOnNextConfirmation