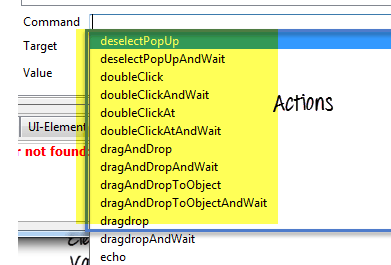
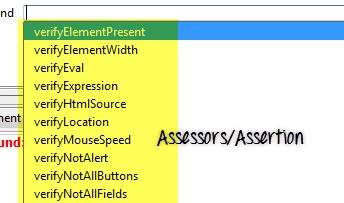
**Selenium Core Extensions (User-Extensions.js)**

To understand extensions, lets first understand the three pillars of selenium IDE

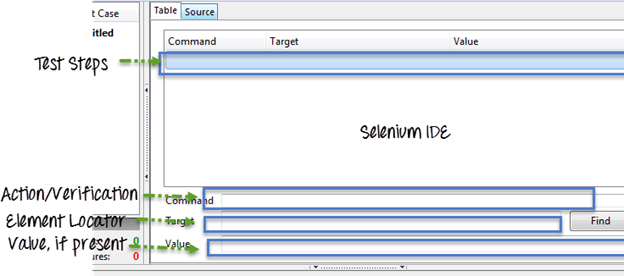
1. Action: What operation you are performing on UI Screen



1. Assessors/Assertion: What verification you do on data you get from UI



1. Locator Strategy: How can we find the element in UI.



Now, Selenium IDE has a very mature library with plenty of Actions, Assertion/Assessors and Locator Strategies.

But sometimes we need to add some more functionality to it for our project requirements. In that situation, we can expand this library by adding our custom extensions. These custom extensions are called ‘User Extension’.

For example, we need an Action which can convert the text to upper case before filling it in a web element. You cannot find this Action in the default Action library. In such case you can create your own ‘User Extension’. In this tutorial, we will learn how to create user extension to convert Text to Upper Case

## Requirement to create Selenium user extension:

To create user extension for Selenium IDE, we need to know the basic concept of[JavaScript](https://www.guru99.com/interactive-javascript-tutorials.html)and[Java](https://www.guru99.com/java-tutorial.html)Script prototype object concept.

To create your user extension, you need to create Java script methods and add them to the selenium object prototype and PageBot object prototype.

## How Selenium IDE recognizes User Extension?

After adding of User Extension to Selenium IDE when we start Selenium IDE, all of these extensions in javascript prototype get loaded, and Selenium IDE recognizes them by their name.

## How to Create User Extension

**Step 1) Action**– all actions are started by “do”, i.e. if the action is for upper case text than its name will be **doTextUpperCase.** When we add this action method in Selenium IDE, Selenium IDE will itself create a wait method for this action. So in this case when we create **doTextUpperCase**action, Selenium IDE will create a corresponding wait function as **TextUpperCaseAndWait**. It can accept two parameters

**Example: Upper Case Text Action**

Selenium.prototype.doTextUpperCase = function(locator, text) {

// Here findElement is itself capable to handle all type of locator(xpath,css,name,id,className), We just need to pass the locator text

var element = this.page().findElement(locator);

// Create the text to type

text = text.toUpperCase();

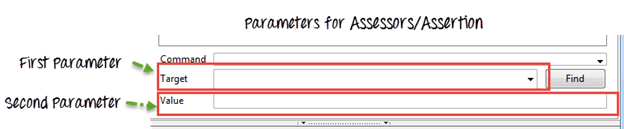
// Replace the element text with the new text

this.page().replaceText(element, text);

};

**Step 2) Assessors/Assertion-** All assessors registered in selenium object prototype will be prefixed

by “get” or “is” Ex. getValueFromCompoundTable , isValueFromCompoundTable .It can accept two parameters, one for target and other for value field in test case.



For each Assessor, there will be corresponding verification functions prefixed by “verify,” “assert” and the wait function prefix by “waitFor”

**Example: For Upper Case Text assessors**

Selenium.prototype.assertTextUpperCase = function(locator, text) {

// All locator-strategies are automatically handled by "findElement"

var element = this.page().findElement(locator);

// Create the text to verify

text = text.toUpperCase();

// Get the actual element value

var actualValue = element.value;

// Make sure the actual value matches the expected

Assert.matches(expectedValue, actualValue);

};

Selenium.prototype.isTextEqual = function(locator, text) {

return this.getText(locator).value===text;

};

Selenium.prototype.getTextValue = function(locator, text) {

return this.getText(locator).value;

};

**Step 3) Locator strategy-** If we wish to create our own function to locate an element then

we need to extend PageBot prototype with a function with prefix “locateElementBy.”

It will take two parameters, first will be the locator string and second will be the document

where it needs to be searched.

Example: For Upper Case Text Locator

// The "inDocument" is a document you are searching.

PageBot.prototype.locateElementByUpperCase = function(text, inDocument) {

// Create the text to search for

var expectedValue = text.toUpperCase();

// Loop through all elements, looking for ones that have

// a value === our expected value

var allElements = inDocument.getElementsByTagName("\*");

// This star '\*' is a kind of regular expression it will go through every element (in HTML DOM every element surely have a tag name like<body>,<a>,<h1>,<table>,<tr>,<td> etc. ). Here our motive is to find an element which matched with the Upper Case text we have passed so we will search it with all elements and when we get match we will have the correct web element.

for (var i = 0; i < allElements.length; i++) {

var testElement = allElements[i];

if (testElement.innerHTML && testElement.innerHTML === expectedValue) {

return testElement;

}

}

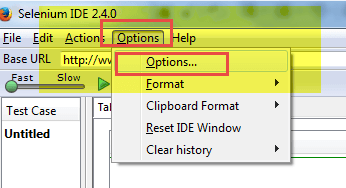
return null;

};

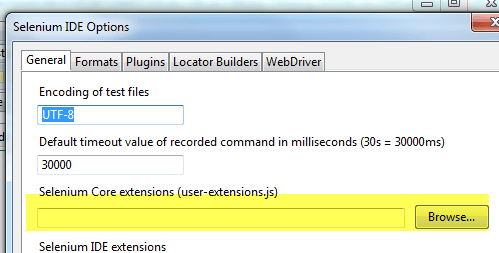
## How to use newly created core extension?

1. Go to Selenium IDE

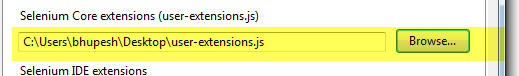
Click on Options -> Options…



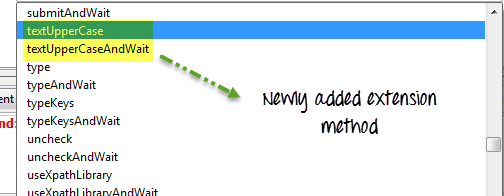
1. In General section select the location of the newly created Selenium Core Extension



1. Click OK and restart Selenium IDE



1. You will find the extension in the command list



## Here is a list of popular extensions/plug-in used in Selenium IDE

|  |  |
| --- | --- |
| **Name** | **Purpose** |
| Favorites | To mark a test suite as favorite and execute them in one click |
| Flex Pilot X | For Flex based automation |
| FlexMonkium | For Adobe Flex based recording and playback[Testing](https://www.guru99.com/software-testing.html)in Selenium IDE |
| File Logging | For saving logs in a file |
| Flow Control | To control test execution flow |
| Highlight Elements | To highlight a web control |
| Implicit Wait | To wait for an element for certain time limit |
| ScreenShot on Fail | Take a screenshot on failure |
| Test Results | Save[Test Case](https://www.guru99.com/test-case.html)result for a test suite in one click |

You can get these all and many more from SeleniumHQ official site’s download section

http://docs.seleniumhq.org/download/

## Summary:

* There is three part of Selenium IDE, Action, Assessors/Assertion, Locator strategy.
* User extension is created, when Selenium IDE is not fulfilling the current requirement.
* To create user extension it is required to add javascript to selenium’s object prototype.
* After creation of extension, it is required to add it in Selenium IDE and restart IDE.