HTML Object Spy Nested Flow

Creation Date: 2/9/2022 2:21:00 PM

Table Of Contents

Introduction

This document describes the HTML object spy which is integral part of the Nested Flow tool and it is used to perform below tasks

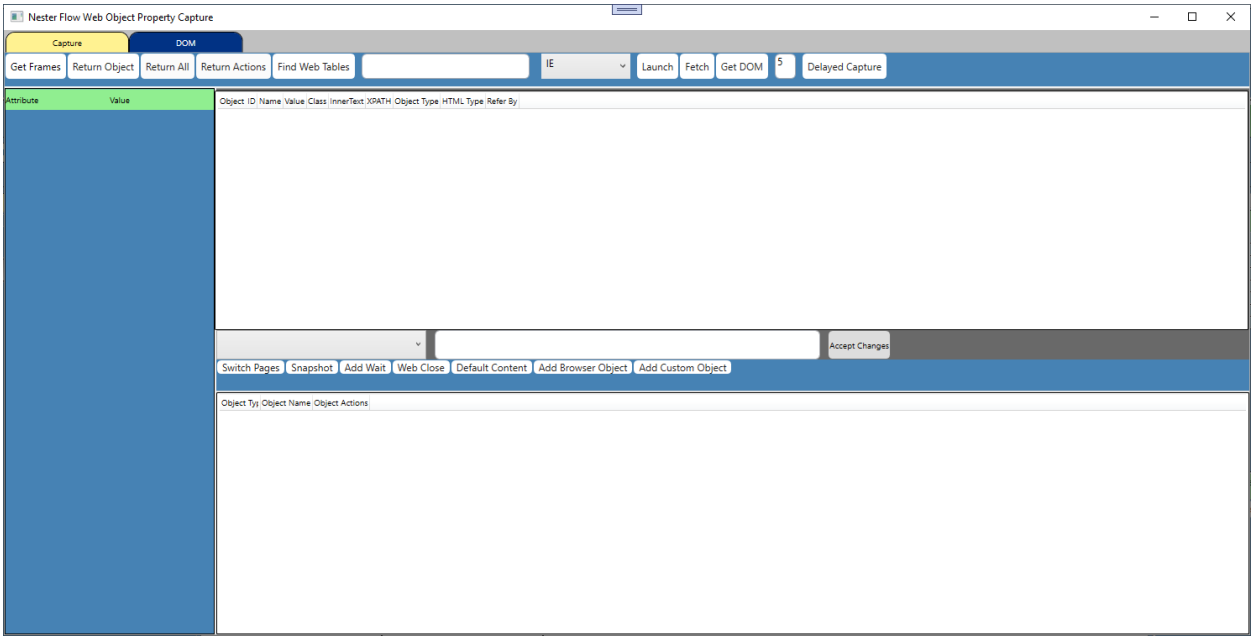
- Capture a single Item
- Capture list of items
- Capture list of items and associated actions
- Capture Object actions as C# script

The tool uses multiple strategies to capture objects:

- Capture Internet explorer objects with mouse right click using .NET framework provided native methods
- Capture any other browser objects with mouse double right click using Selenium (This expects the browser to be opened from the spy itself)

In the upcoming sections we will go over each of these topics in detail

Tool Layout



The Tool has 2 separate menu bars.

The top bar is for helping with object search whereas the bottom bar is to simulate object actions (The tool doesn't support traditional recording instead provides a big list of Selenium webdriver functions to simulate object actions)

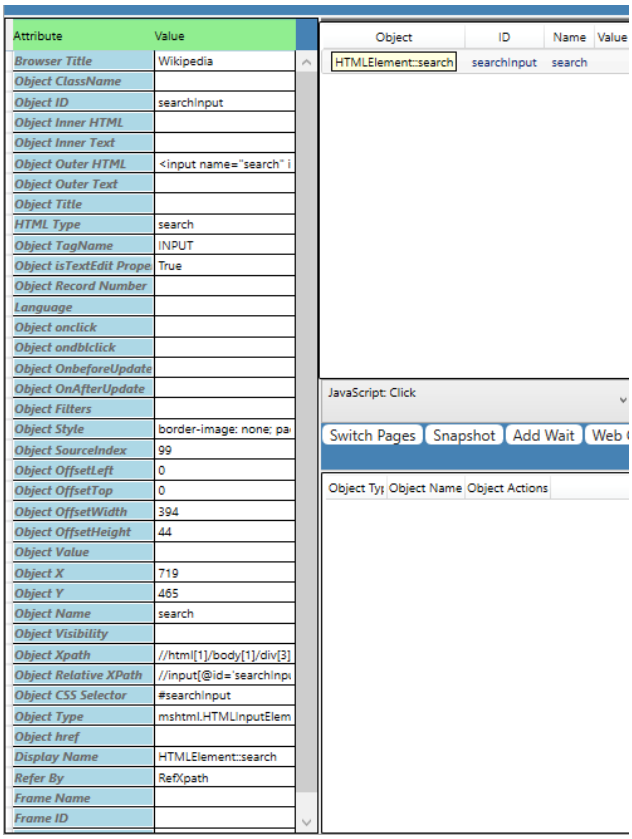
Top Bar Menu Functions:

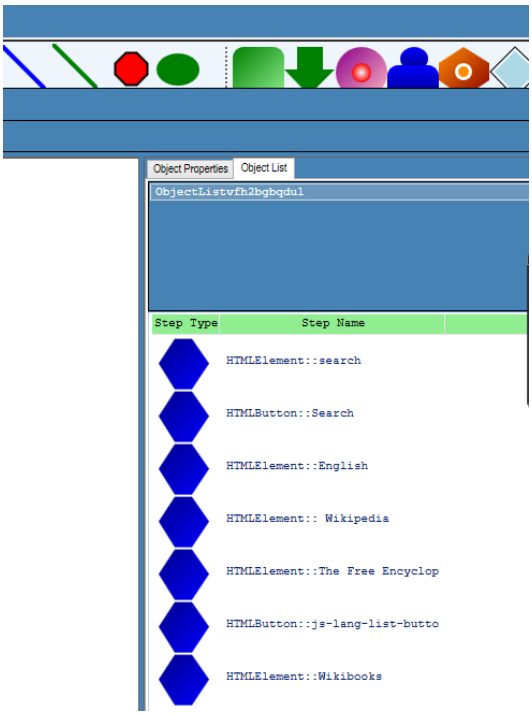
Get Frames → Get All frames in the webpage opened using **Launch** button

Return Object → Returns one single object which is currently displayed in the property grid

Return All → Returns all objects on the grid to Test creation Panel

Return Actions → Returns all object definitions and actions to test case draw panel object list or creates action scripts


Attribute
Value
Browser Title
Wikipedia
Object ClassName
Object ID
searchInput
Object Inner HTML
Object Inner Text
Object Outer HTML
<input name="search" i
Object Outer Text
Object Title
HTML Type
search
Object TagName
INPUT
Object isTextEdit Prope
True
Object Record Number
Language
Object onclick
Object ondblclick
Object OnbeforeUpdate
Object OnAfterUpdate
Object Filters
Object Style
border-image: none; pa
Object SourceIndex
99
Object OffsetLeft
0
Object OffsetTop
0
Object OffsetWidth
394
Object OffsetHeight
44
Object Value
Object X
719
Object Y
465
Object Name
search
Object Visibility
Object Xpath
//html[1]/body[1]/div[3]
Object Relative XPath
//input[@id='searchInpu
Object CSS Selector
#searchInput
Object Type
mshtml.HTMLInputElem
Object href
Display Name
HTMLElement::search
Refer By
RefXpath
Frame Name
Frame ID
Object
ID
Name
Value
HTMLElement::search
searchInput
search
JavaScript: Click
Switch Pages
Snapshot
Add Wait
Web
Object Ty
Object Name
Object Actions


Object Properties
Object List
ObjectListvfh2bgbqdul
Step Type
Step Name
HTMLElement::search
HTMLButton::Search
HTMLElement::English
HTMLElement:: Wikipedia
HTMLElement::The Free Encyclop
HTMLButton::js-lang-list-butto
HTMLElement::Wikibooks

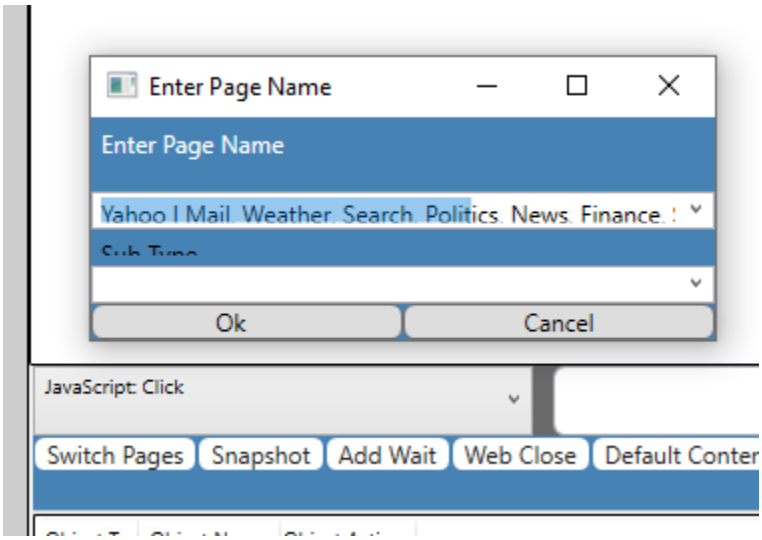
Find Web Tables → Get All web Tables in the webpage opened using **Launch** button

→ This is for launching the browser using Selenium from inside the tool. The initial text box is for entering the URL. Second LOV is to choose type of browser. Supported types are:

- IE → Internet Explorer
- CHROME → Google Chrome
- FF → FireFox
- EDGE → Microsoft chromium-based Edge
- EDGE IE MODE → Microsoft Edge in IE Mode

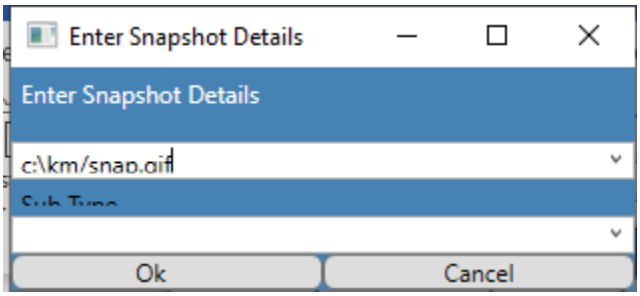
Bottom Bar Menu Functions:

Switch Pages → Moves control to Web Page



Just enter Page title in the text box and click on **Ok** button

Snapshot → Creates snapshot of the web page

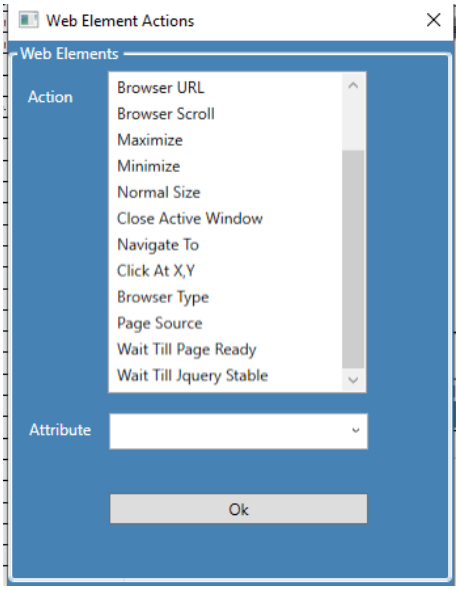


Add Wait → creates a system Delay

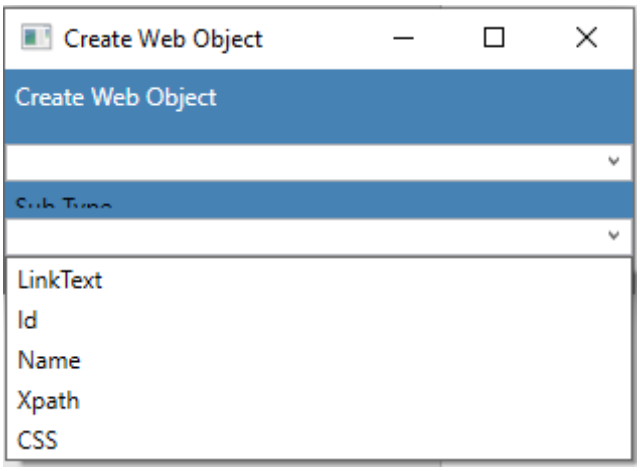
Web Close → Creates a closure of web page

Default Content → Moves to default content (first page or default frame)

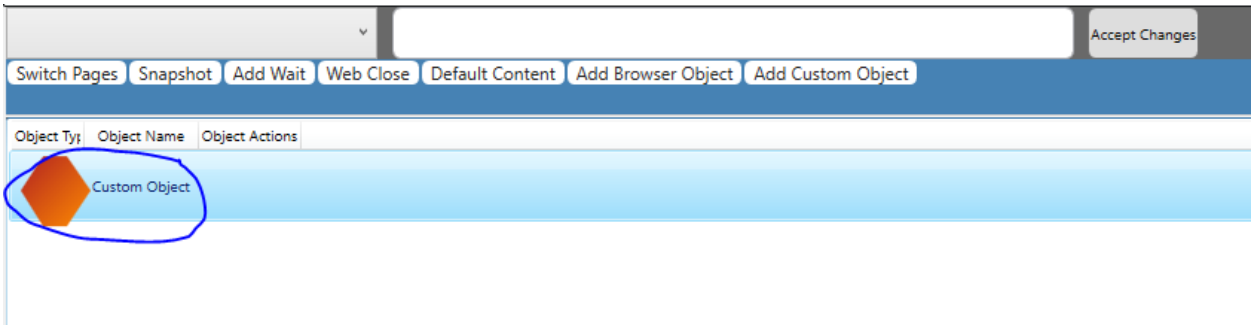
Add Browser Object → Creates a browser action



Add Custom Object → Creates dynamic web object. Just provide identified by and locator value. This function is useful when there is an object created in the run time and for some reason cannot be used with object spy



Bottom Pane:



This section records the actions simulated. These can be returned to the test case builder panel or to test script creator

Example Of a Framed Page with IE

In this example we will examine the ways of recording automation of a framed web page using Internet Explorer.

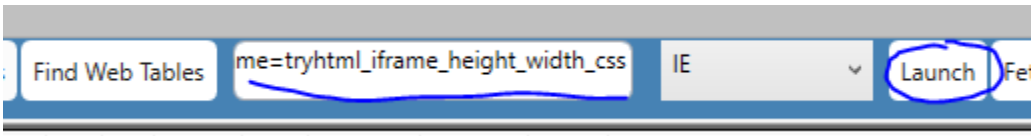
Note: Most modern applications (web pages) have discontinued support for IE as after the advent of HTML5 the entire browser landscape is captured by Google Chrome, MS Edge, Firefox etc. as most features are not supported by IE. IE is still used in legacy applications where a web page opens a form-based windows or java app.

The web page we are considering here is:

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_iframe_height_width_css

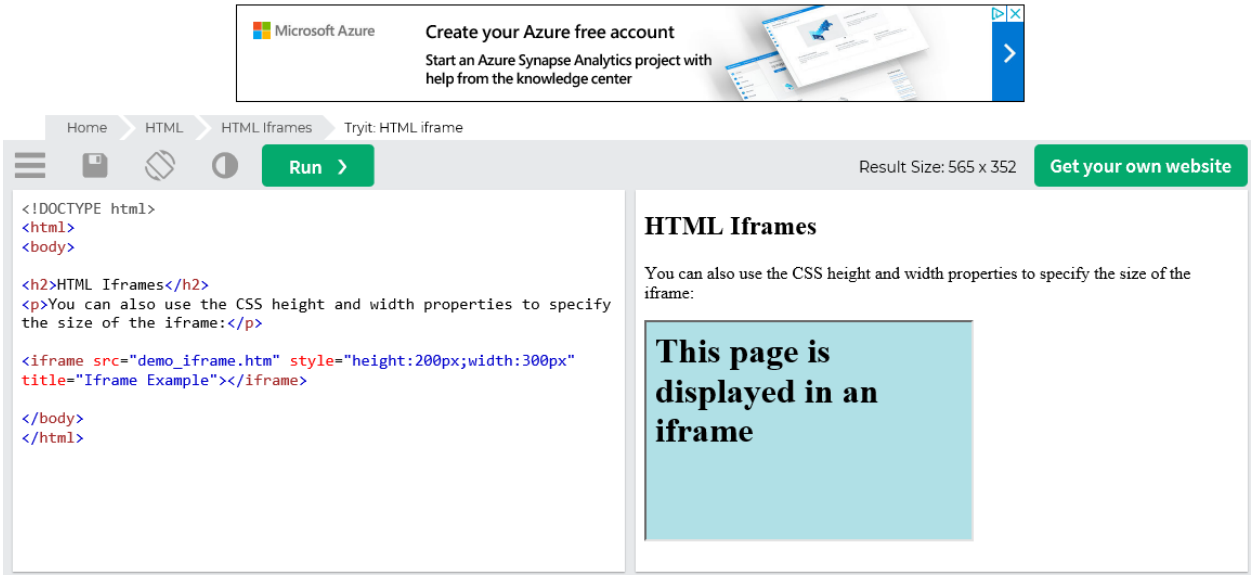
IE pages are automatically captured by the object spy whether they are opened by selenium or not but for actions recording we still need it to be invoked through selenium.

Steps:

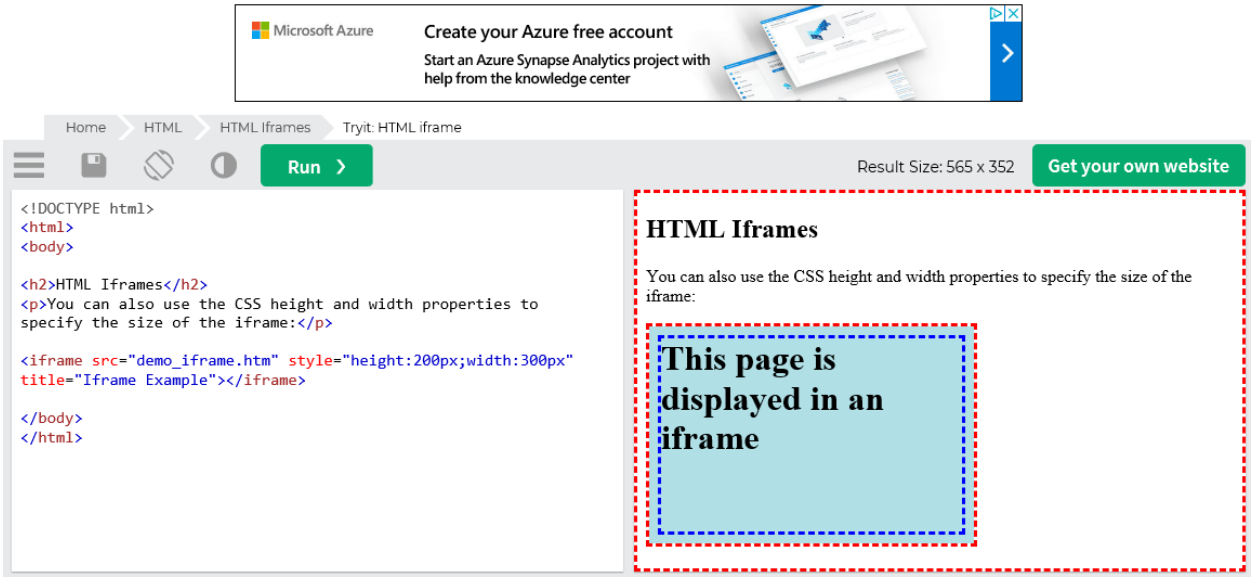


Enter URL in the text box choose IE as the browser and click on **Launch**

Site opens successfully

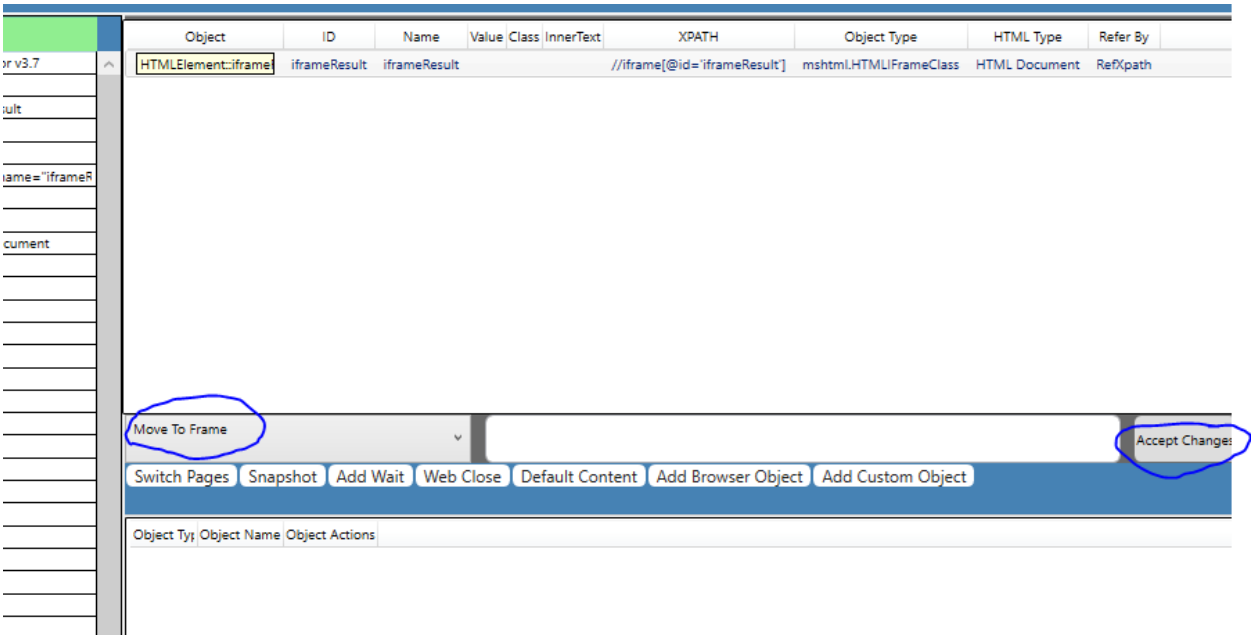


Tool will automatically identify the objects as mouse moves over any IE browser opens.

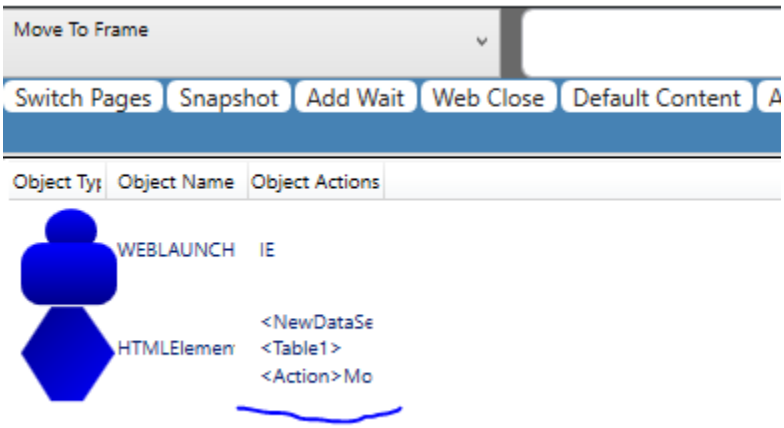


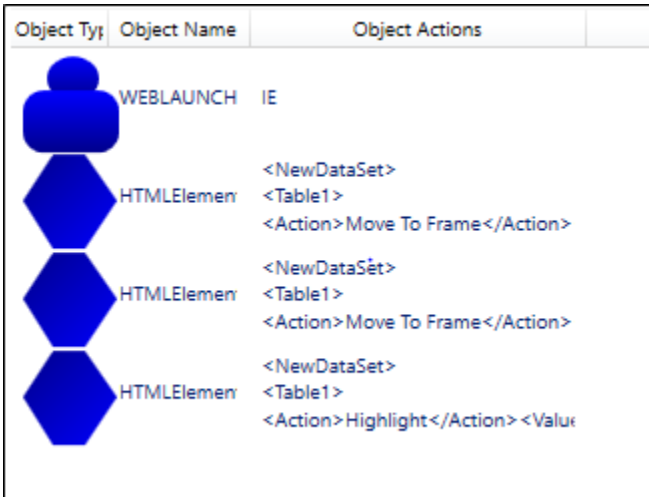
All iframes are identified as red border and all other items are shown as blue.

For creating a script to access the text in blue border, we need to move to 2 iframes one after another



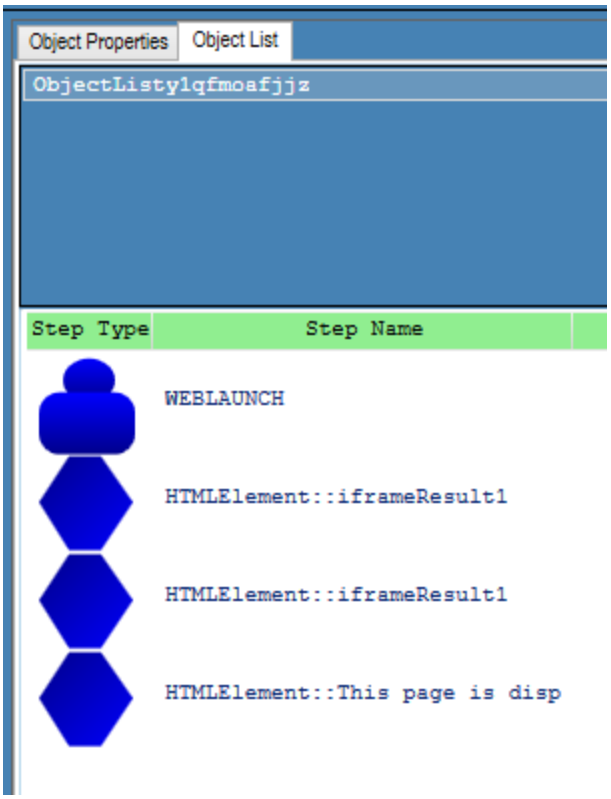
Choose **Move To Frame** as action and click on **Accept Changes**.

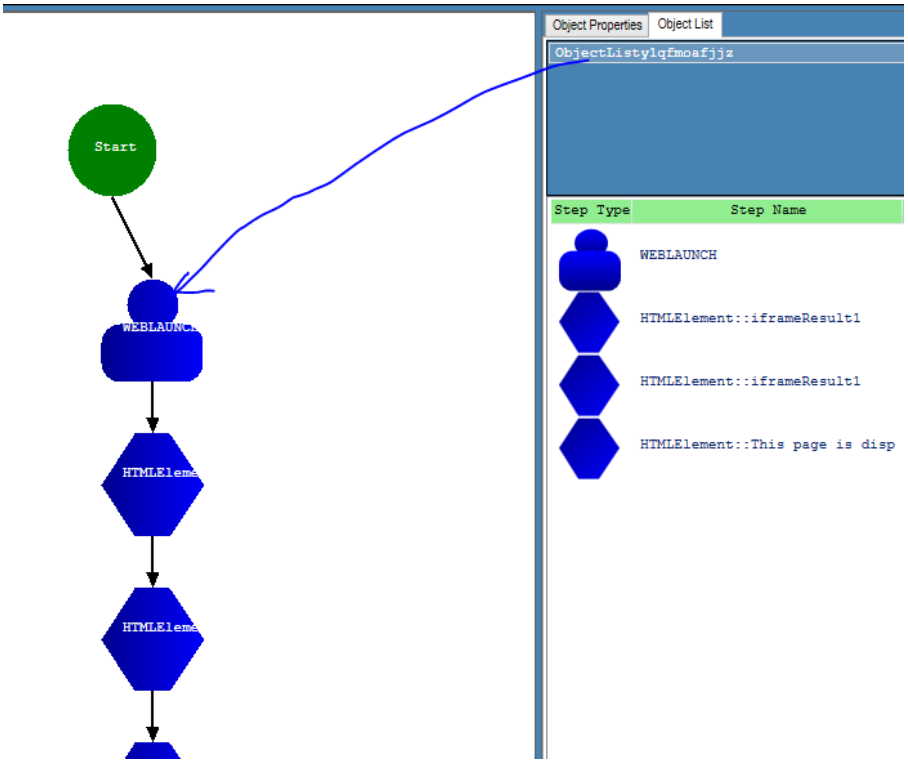




Click on **Return Actions**

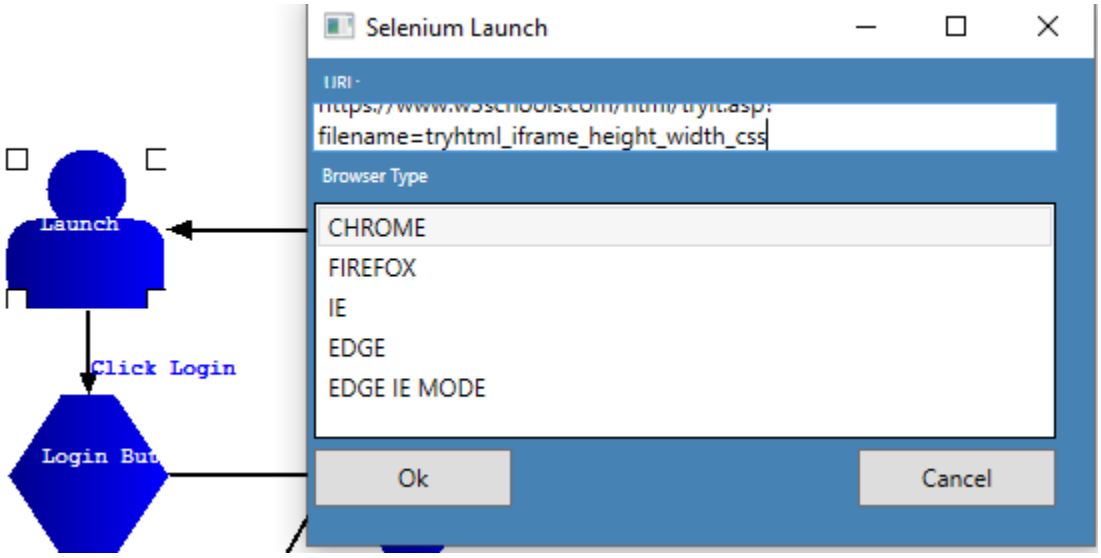
Actions returned to test case builder panel





Drag actions to panel.

In the Web Launch component right click and choose **Selenium Launch Parameters** option



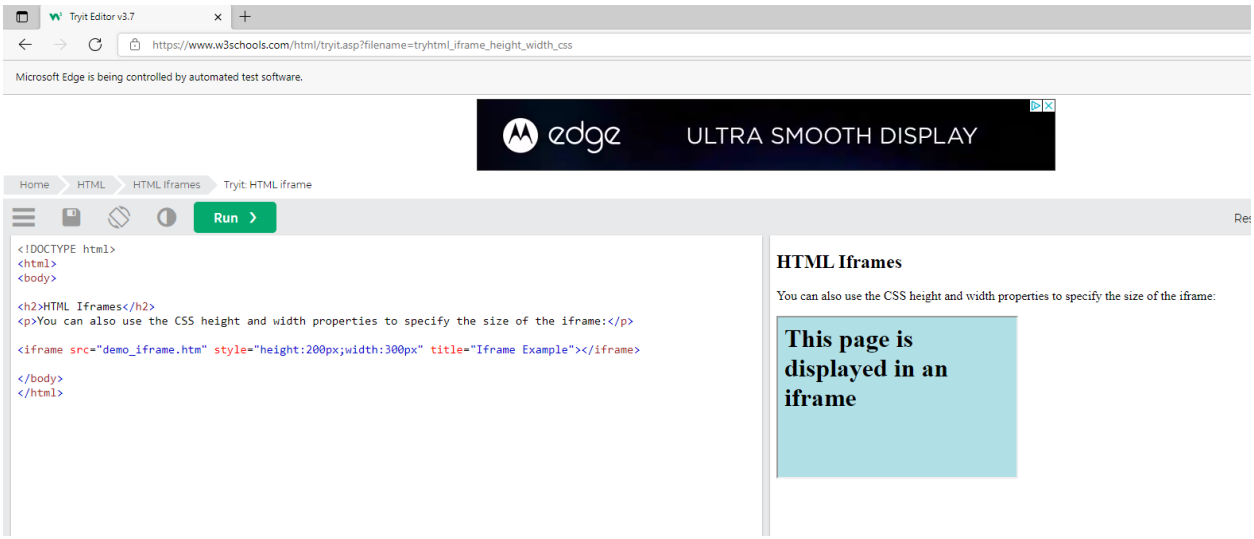
This way whichever browser you record scenario with, you can execute it with any other supported browser

Example Of a Framed Page with Other Browsers

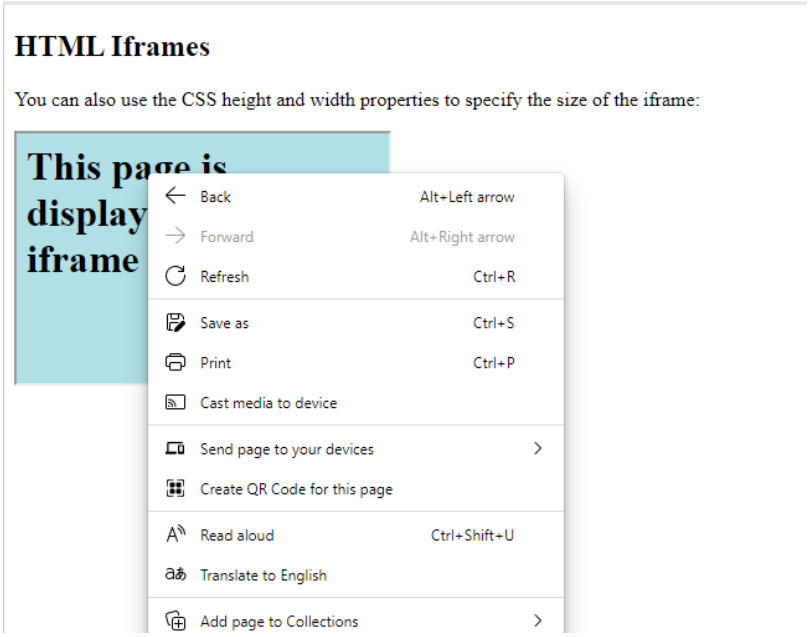
The object spy works with other browsers (Chrome, Firefox, Edge) only if they are triggered from inside the tool. Let us examine the same example with Microsoft Edge (other browsers work the same way)

As shown below, enter the URL in the textbox, choose browser as EDGE and click on **Launch** button.

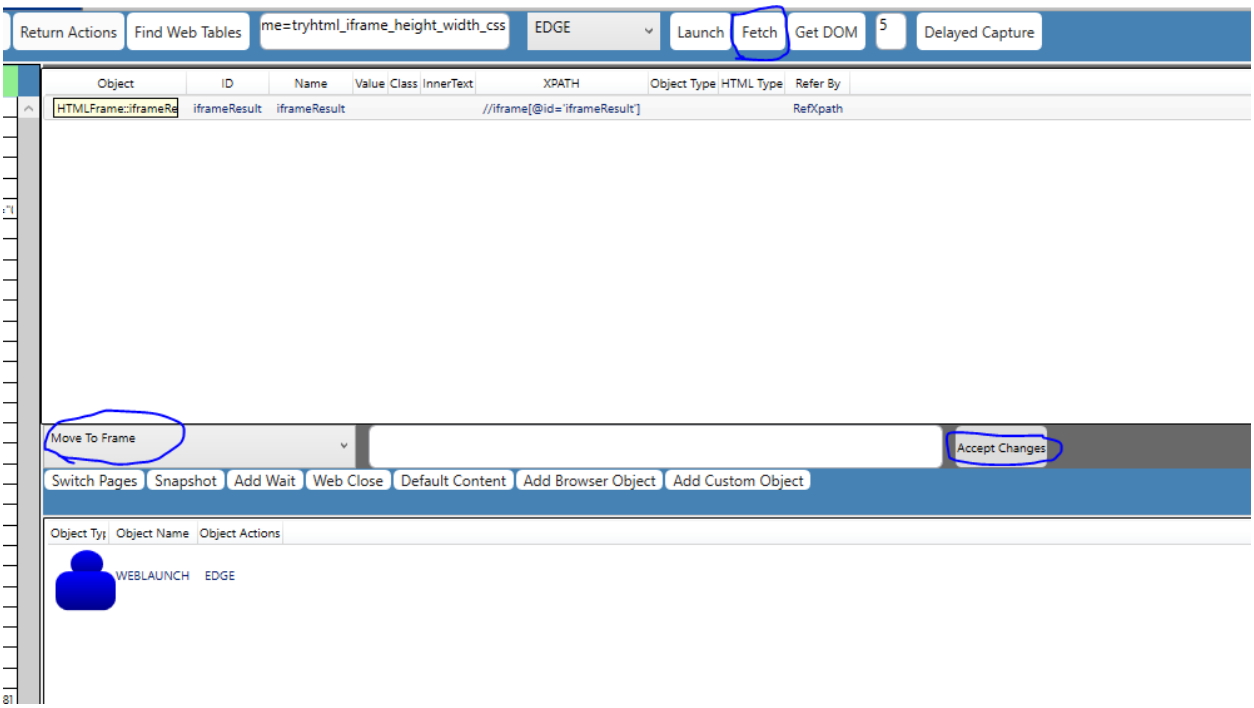
Page opens on edge



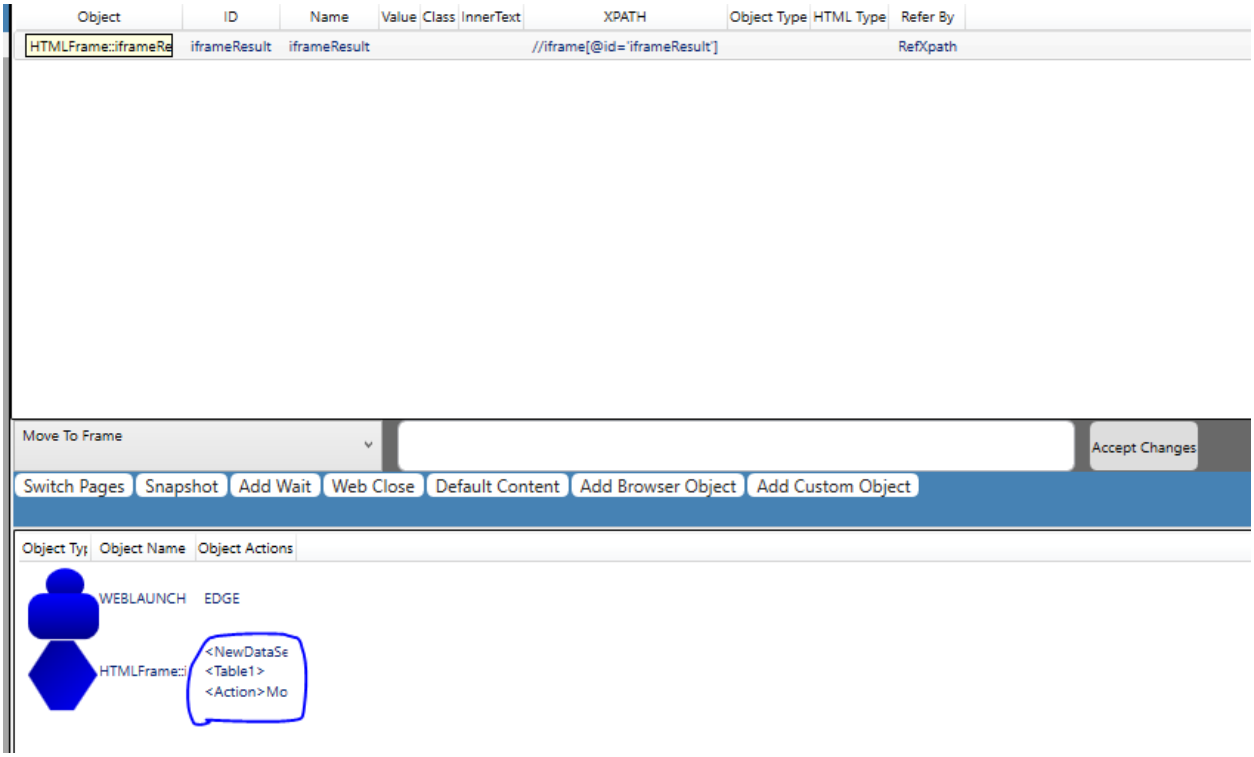
Right click on the text we want to capture on webpage. Instead of capturing it shows default right click menu as it is inside a frame and not visible yet to the object spy



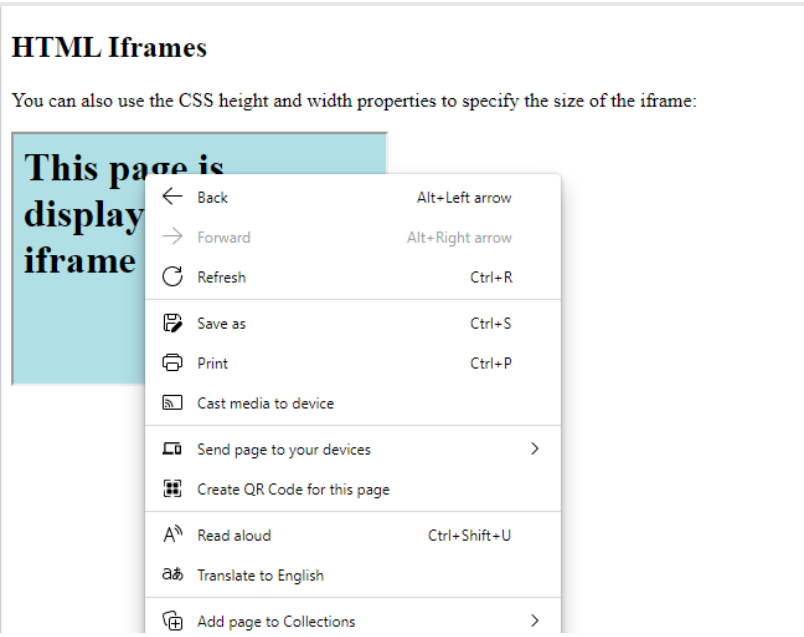
Click on the text and click on Fetch button in the spy



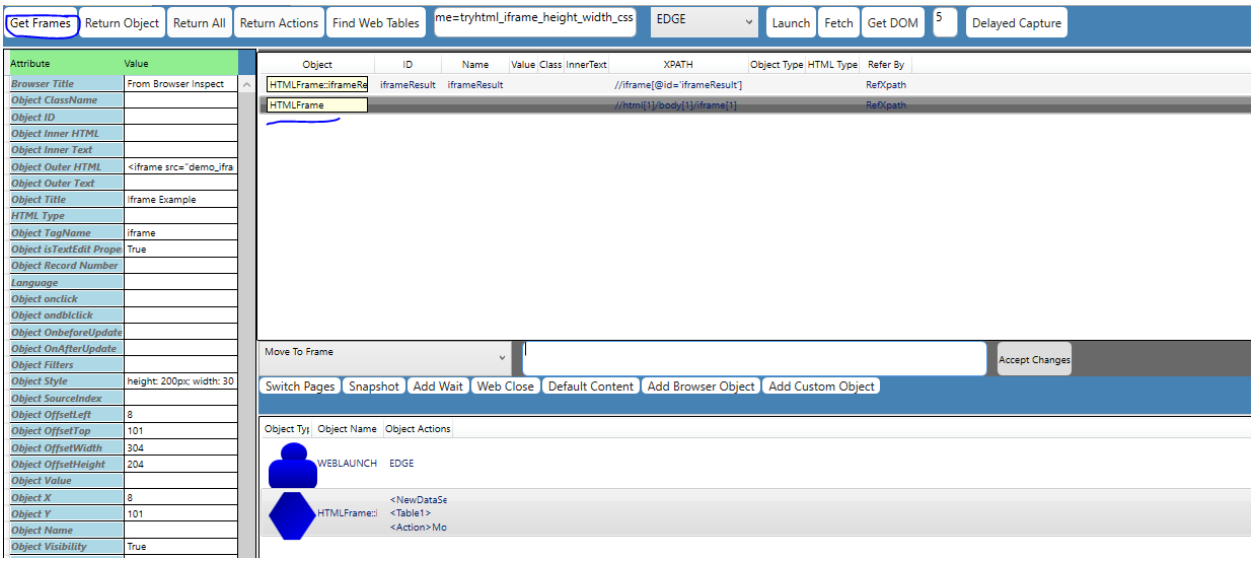
It will capture the frame (parent frame) in which the object is. By default, action is chosen as **Move To Frame** and click on **Accept Changes** button. Automatically focus will be moved to that frame



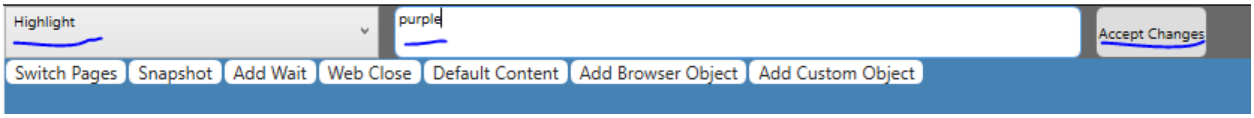
Once again right click on the text we want to capture on webpage. Instead of capturing it shows default right click menu of Edge again which means it is inside another frame.

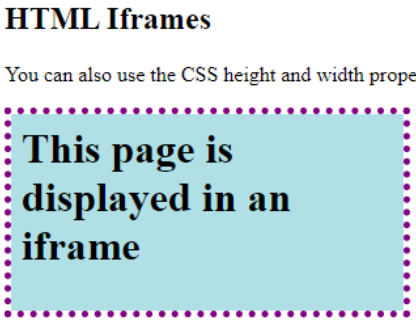


Click on **Get Frames** button. All frames on the current frame are displayed

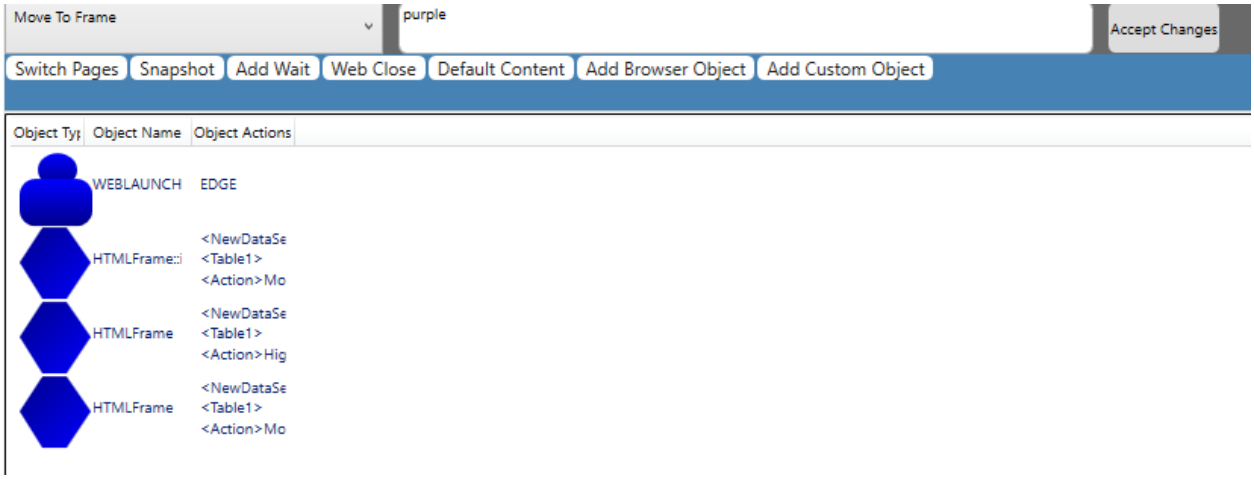


Try highlighting the frame. You can optionally give a color to highlight it with.



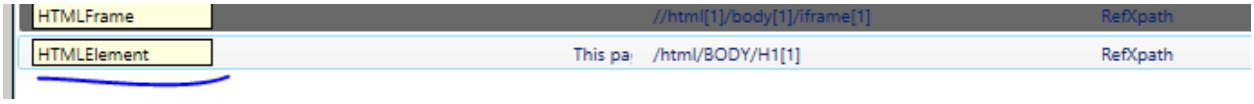


It is evident that the text is inside this frame. Hence, perform a **Move To Frame** action.

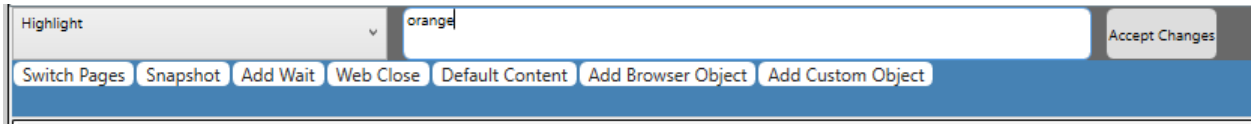


This time, right click on the text will not display default context menu of Edge which means Object spy is seeing it now. Double right click on the object.

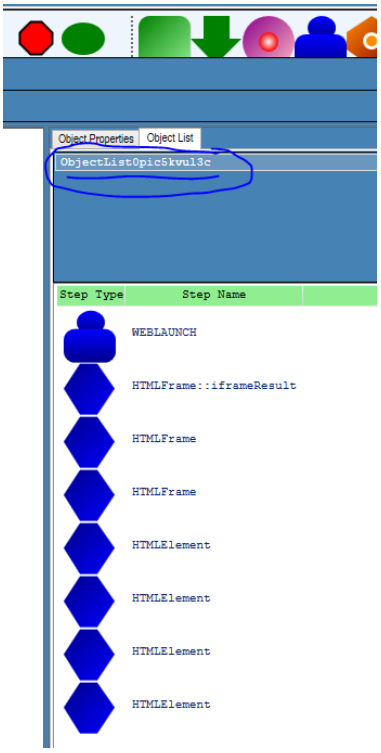
Object is captured.



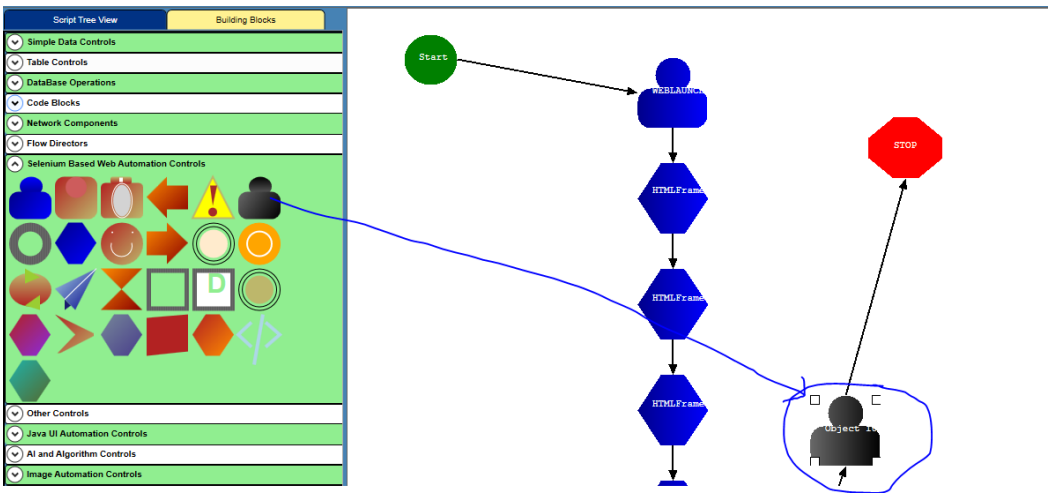
Highlight the object.



Click on **Return Actions** button

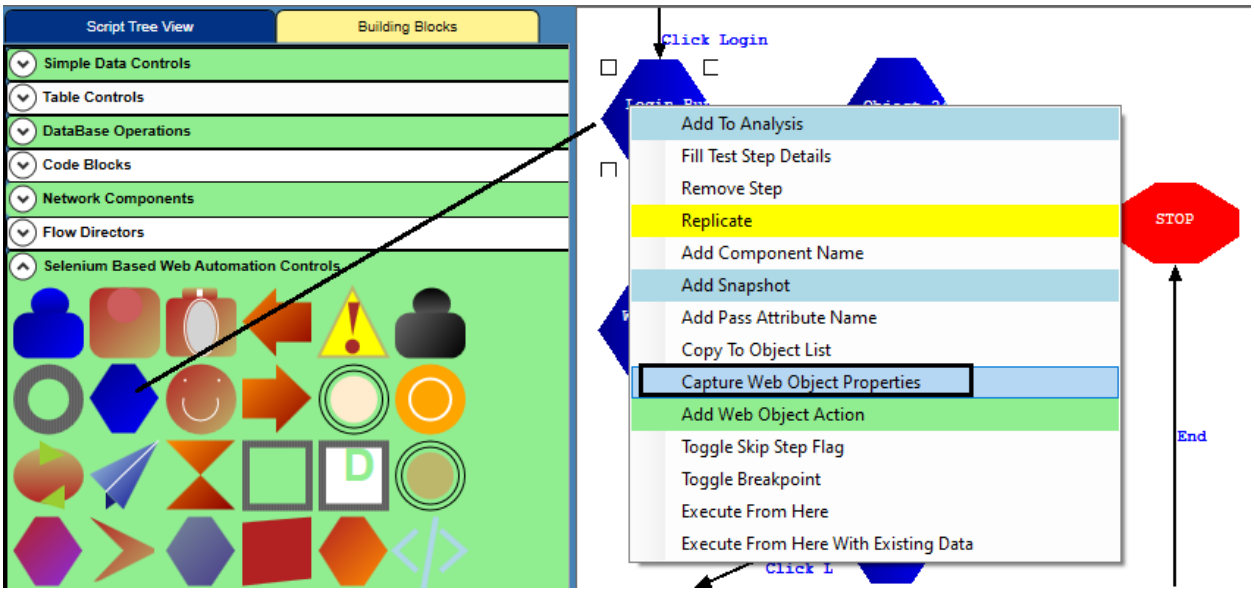


All the actions are returned to the object list. Drag the file to the Test Case Builder. Add a Start and end Block. You have a script ready to be executed now (we optionally added webpage close method before the end of script)

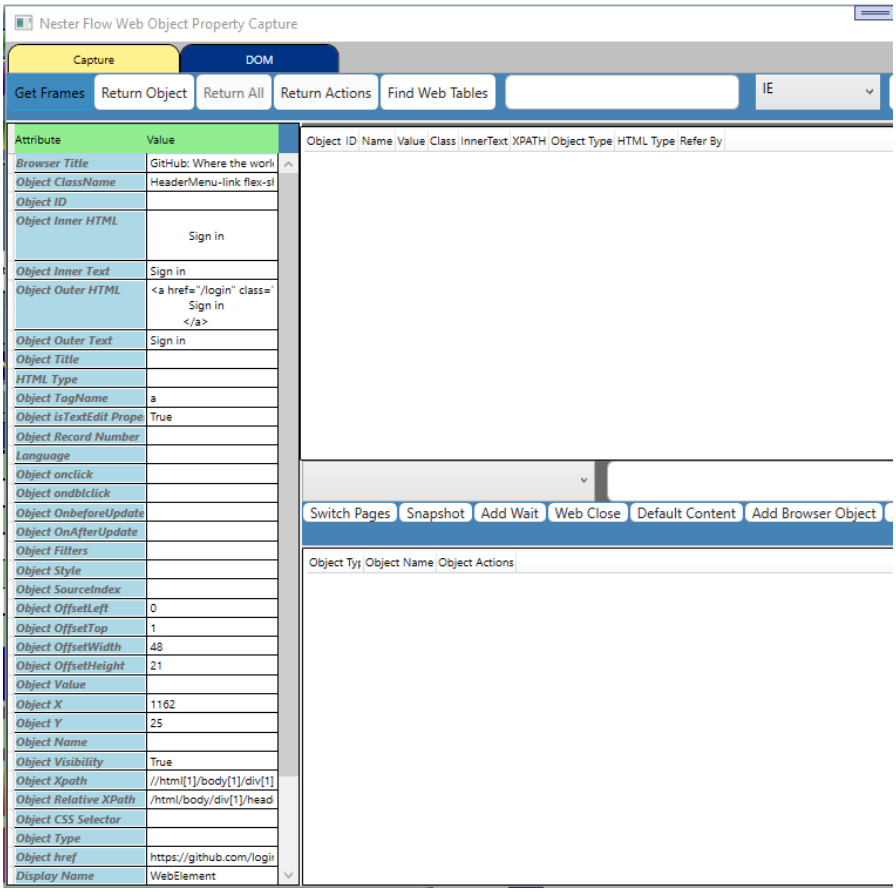


Return Single Object

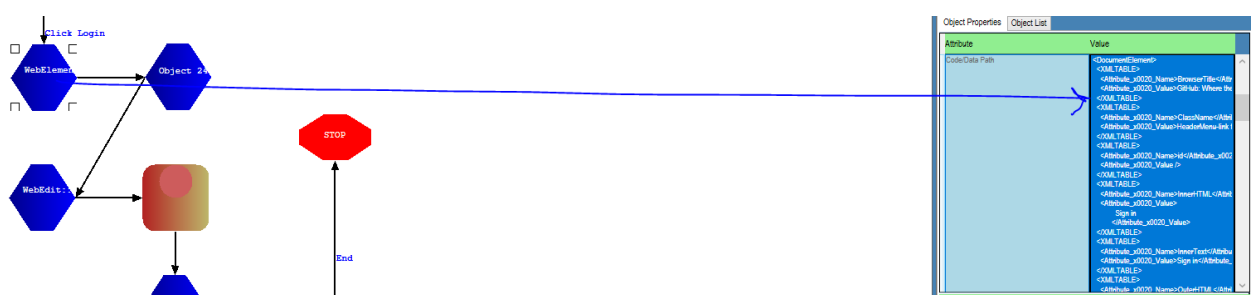
If we already built a flow and either want to assign an object to the web object step or update the step with latest update to the object properties, right click on the step and choose **Capture Web Object Properties** option



HTML Object Spy opens and if the object already has properties assigned, it will display them on property grid.

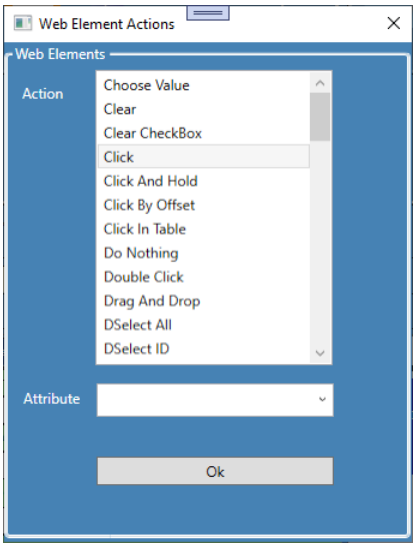


Spy a different object (the most recently captured or the most recently selected item will be shown on property grid). Click on **Return Object** button. Click **Yes** or **No** in the confirmation window (it will not make any difference as this is single object return mode)



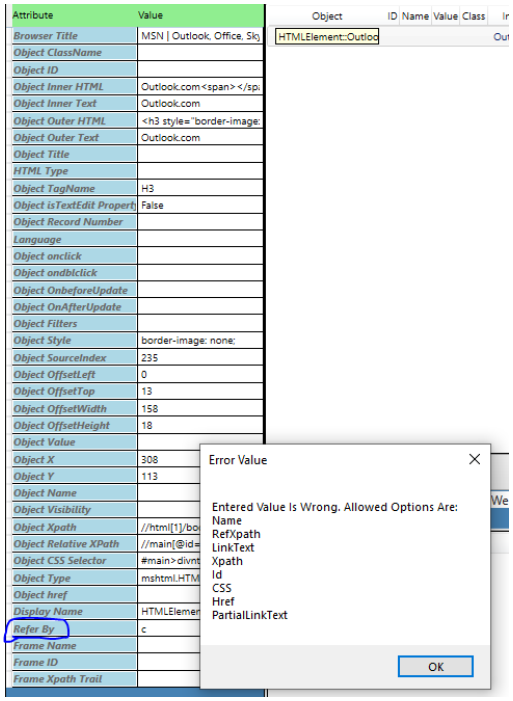
The object property reflects the selected value

Also, when clicking on any web object step on Test Case Builder, **Add Web Object Action** option will enable the user to choose a right object action to be performed with recently chosen action in clicked state.


Web Element Actions
Web Elements
Action
Choose Value
Clear
Clear CheckBox
Click
Click And Hold
Click By Offset
Click In Table
Do Nothing
Double Click
Drag And Drop
DSelect All
DSelect ID
Attribute
Ok

Object Spy – Object Identification Strategy

The object spy allows the right object identification strategy to be chosen.



The property grid provides a field called **Refer By** which allows following values:

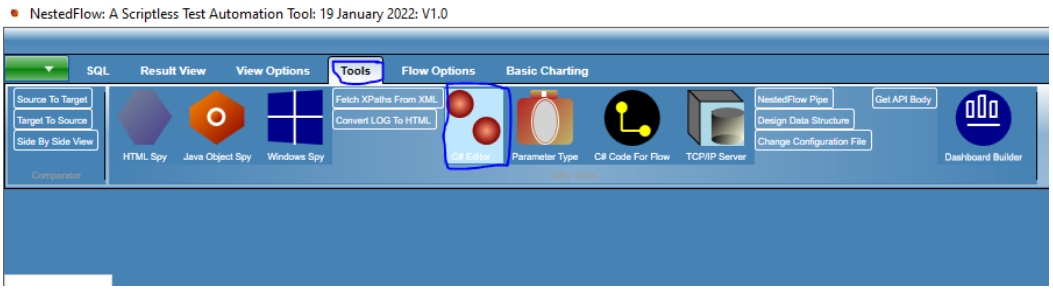
- **Name**: Object is identified using object Name
- **RefXpath**: This is default value. Allows object identification using shorter Xpath value the tool provides
- **LinkText**: Allows object identification using innerText
- **Xpath**: Allows object identification using Absolute Xpath of the object shown by the tool
- **Id**: Object is identified using object Id
- **CSS**: Object is identified using CSS selector value provided by the tool
- **Href**: Object is identified using Href value if existing
- **PartialLinkText**: Allows object identification using partial innerText

Create Code Out of Object Spy Actions

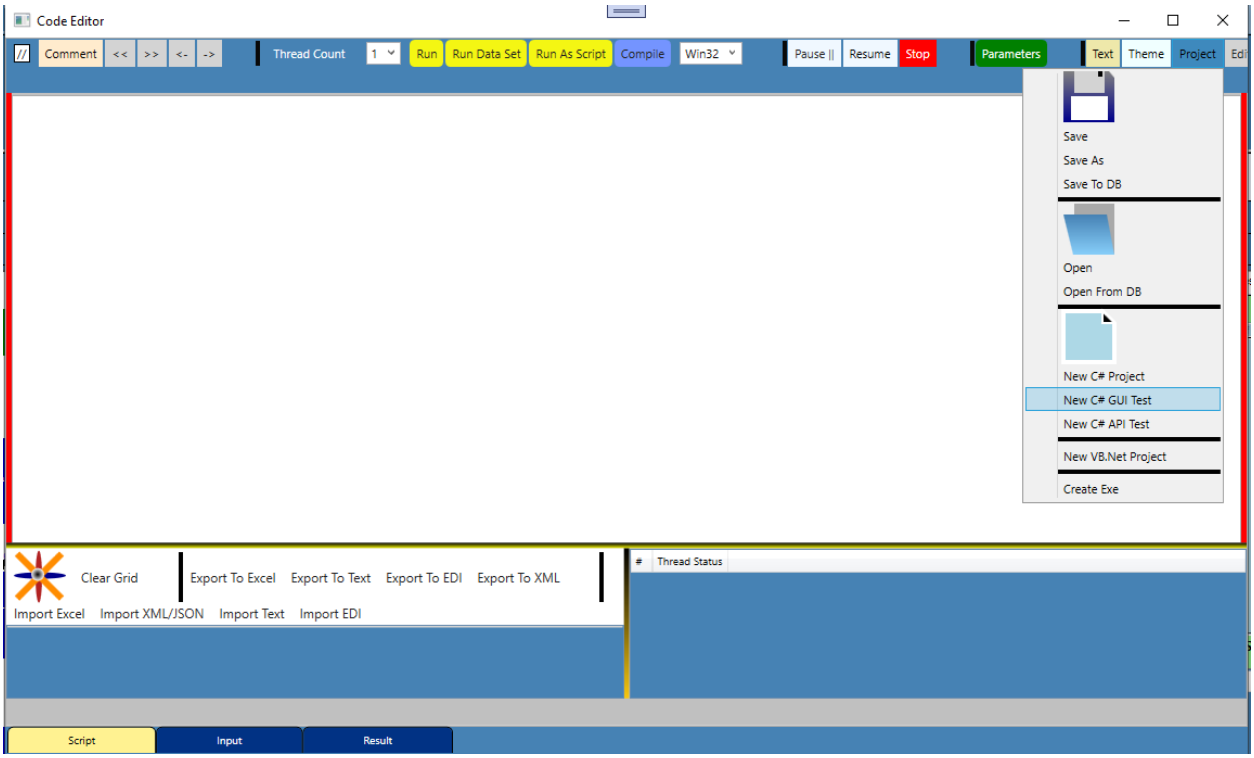
Nested Flow framework exposes all its functions through DLL. Using which fully scripted code can also be created using Visual Studio or using the Code editor available in the tool.

This methodology only copies selenium object and actions to Code it will ignore other functions in spy like Move to frame and other functions. Those functions need to be manually coded in the editor (There will be a separate document for a coded framework)

Click on **Tool** Ribbon Menu and click on **C# Editor**.



Click on Project → New C# GUI Test



This will create a barebone UI testing script:

```
New Project
using System;
using System.Xml;
using System.Data;
using System.Data.SqlClient;
using System.Windows.Forms;
using System.Drawing;

using System.IO;

using OpenQA.Selenium;

using OpenQA.Selenium.Support.UI;

using OpenQA.Selenium.IE;

using OpenQA.Selenium.Firefox;

using OpenQA.Selenium.Chrome;

using OpenQA.Selenium.Interactions;

using OpenQA.Selenium.Remote;

//Return value from Eval Code will be assigned to column
//DataTable T1: Whatever Data in Bottom Grid Of Query 1
//string s1   : Oracle User Name
//string s2   : Oracle Password
//string s3   : Second User Name
//string s4   : Second Password

namespace NSUpdateTable
{
    public class CSUpdateTable
    {
       //Return value from Eval Code will be assigned to column
       public object UpdateTable(DataTable T1,string s1,string s2,string s3,string s4,ref string s5)
        {

                XMLParser.LogFileActions.CreateLogFile();//GetLogFile
                XMLParser.Web_Automation_Framework.SeleniumFunctions SF = new XMLParser.Web_Automation_Framework.SeleniumFunctions();

                string webtype                                          = "";
                string pid                                          = "";
                SF.InitializeWebDriver("CHROME",pid,out webtype);
                bool retval;
                SF.LaunchDriver("https://<URL>/", out retval);            return (T1);
        }
    }
}
```

Update the URL and browser (default is CHROME)

Put cursor between browser definition and return Table statements as shown below

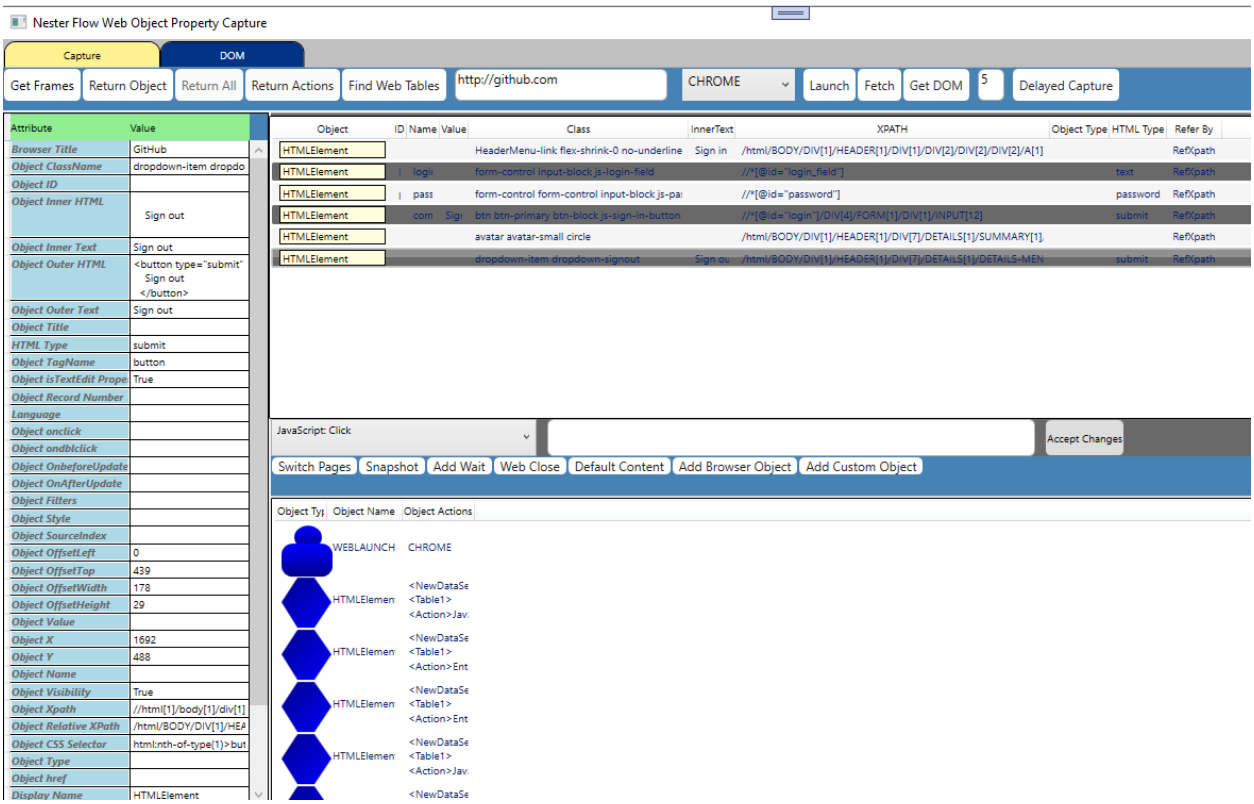
```
public object UpdateTable(DataTable T1,string s1,string s2,string s3,string s4,ref string s5)
 {

        XMLParser.LogFileActions.CreateLogFile();//GetLogFile
        XMLParser.Web_Automation_Framework.SeleniumFunctions SF = new XMLParser.Web_Automation_Framework.SeleniumFunctions();

        string webtype                                          = "";
        string pid                                          = "";
        SF.InitializeWebDriver("EDGE",pid,out webtype);
        bool retval;
        SF.LaunchDriver("http://github.com", out retval);

        return (T1);
 }
```

Right Click and choose **Web Object Spy** menu.

```
namespace NSUpdateTable
{
    public class CSUpdateTable
    {
       //Return value from Eval Code will be assigned to column
       public object UpdateTable(DataTable T1,string s1,string s2,string s3,string s4,ref string s5)
        {

                XMLParser.LogFileActions.CreateLogFile();//GetLogFile
                XMLParser.Web_Automation_Framework.SeleniumFunctions SF = new XMLParser.Web_Automation_Framework.SeleniumFunctions();

                string webtype                                          = "";
                string pid                                          = "";
                SF.InitializeWebDriver("EDGE",pid,out webtype);
                bool retval;
                SF.LaunchDriver("http://github.com", out retval);

                re
        }
    }
}
```

Web Object Spy

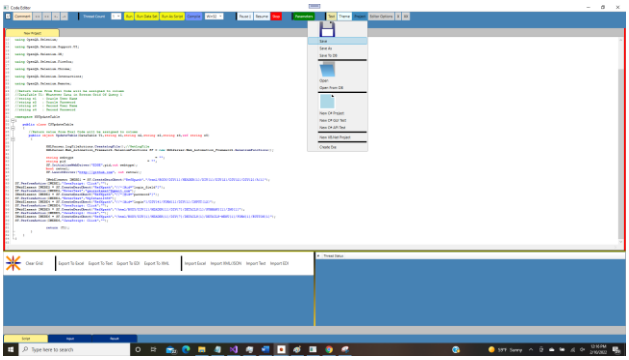
Record Actions and Click on **Return Actions** button

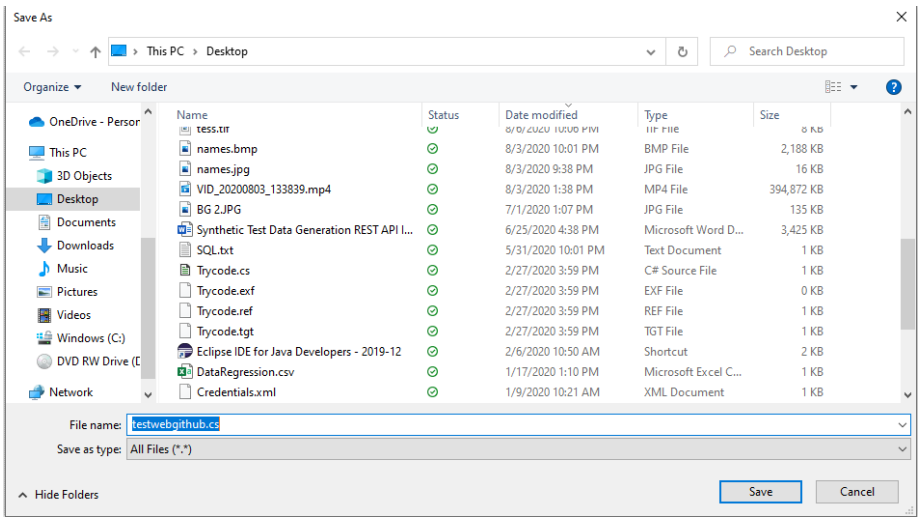


Recorded script returns to editor

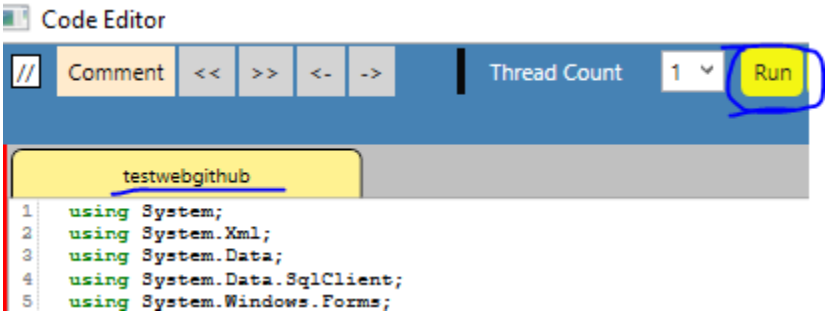
```
using OpenQA.Selenium.Support.UI;

using OpenQA.Selenium.IE;

using OpenQA.Selenium.Firefox;

using OpenQA.Selenium.Chrome;

using OpenQA.Selenium.Interactions;

using OpenQA.Selenium.Remote;

//Return value from Eval Code will be assigned to column
//DataTable T1: Whatever Data in Bottom Grid Of Query 1
//string s1   : Oracle User Name
//string s2   : Oracle Password
//string s3   : Second User Name
//string s4   : Second Password

namespace NSUpdateTable
{
    public class CSUpdateTable
    {
        //Return value from Eval Code will be assigned to column
        public object UpdateTable(DataTable T1,string s1,string s2,string s3,string s4,ref string s5)
        {

                XMLParser.LogFileActions.CreateLogFile();//GetLogFile
                XMLParser.Web_Automation_Framework.SeleniumFunctions SF = new XMLParser.Web_Automation_Framework.SeleniumFunctions();

                string webtype                                          = "";
                string pid                                            = "";
                SF.InitialiseWebDriver("EDGE",pid,out webtype);
                bool retval;
                SF.LaunchDriver("http://github.com", out retval);

                IWebElement IWEBE1 = SF.CreateDescObject("RefXpath","/html/BODY/DIV[1]/HEADER[1]/DIV[1]/DIV[2]/DIV[2]/DIV[2]/A[1]");
SF.PerformAction(IWEBE1,"JavaScript: Click","");
IWebElement IWEBE2 = SF.CreateDescObject("RefXpath","//*[@id="login_field"]");
SF.PerformAction(IWEBE2,"EnterText",[redacted],
IWebElement IWEBE3 = SF.CreateDescObject("RefXpath","//*[@id="password"]");
SF.PerformAction(IWEBE3,"EnterText",[redacted],
IWebElement IWEBE4 = SF.CreateDescObject("RefXpath","//*[@id="login"]/DIV[4]/FORM[1]/DIV[1]/INPUT[12]");
SF.PerformAction(IWEBE4,"JavaScript: Click","");
IWebElement IWEBE5 = SF.CreateDescObject("RefXpath","/html/BODY/DIV[1]/HEADER[1]/DIV[7]/DETAILS[1]/SUMMARY[1]/IMG[1]");
SF.PerformAction(IWEBE5,"JavaScript: Click","");
IWebElement IWEBE6 = SF.CreateDescObject("RefXpath","/html/BODY/DIV[1]/HEADER[1]/DIV[7]/DETAILS[1]/DETAILS-MENU[1]/FORM[1]/BUTTON[1]");
SF.PerformAction(IWEBE6,"JavaScript: Click","");

                return (T1);
        }
    }
}
```

Save the script as .cs





Click on **Run** button.



Script will execute in coded mode.