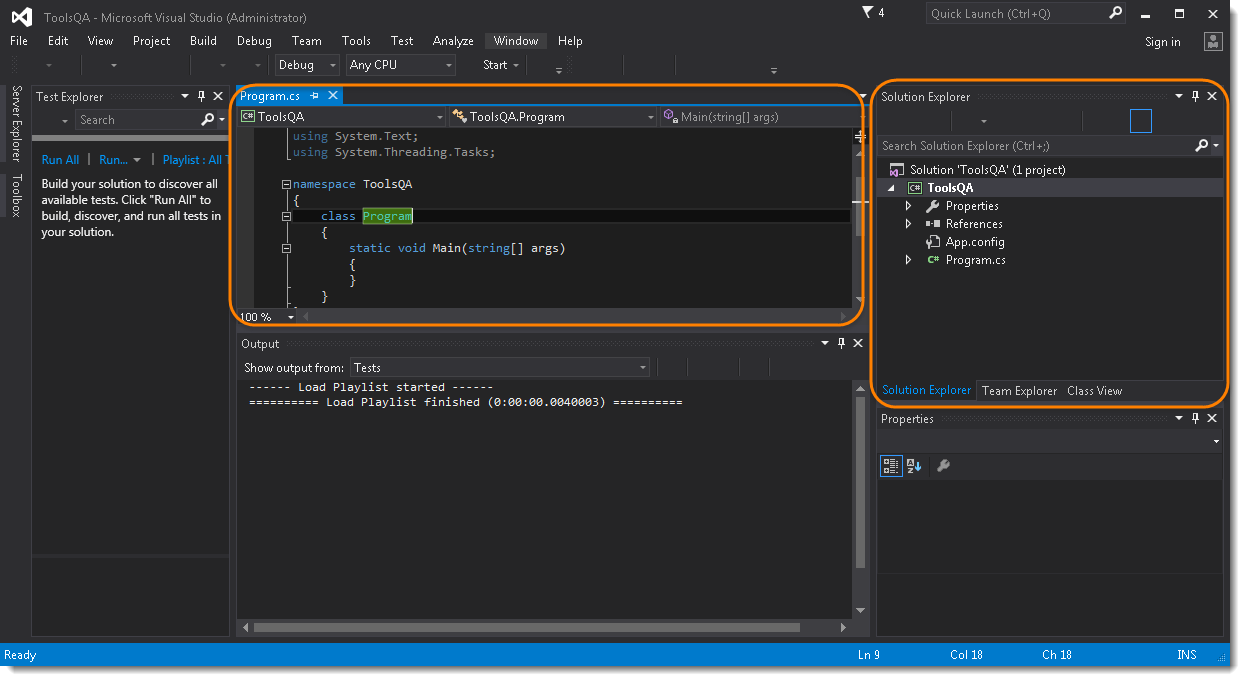
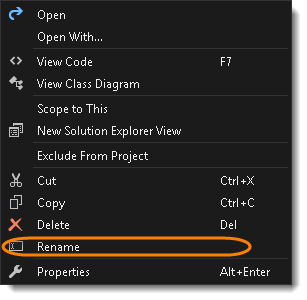
You can download the software from the [***https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx.***](https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx)

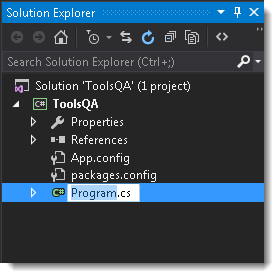
Before moving on, let’s just first change the name of the test case, which got created by default by the Visual Studio at the time of the Project Creation. As of now, the name of the test case is *Program*, lets just change this to something more meaningful, like ***FirstTestCase***.



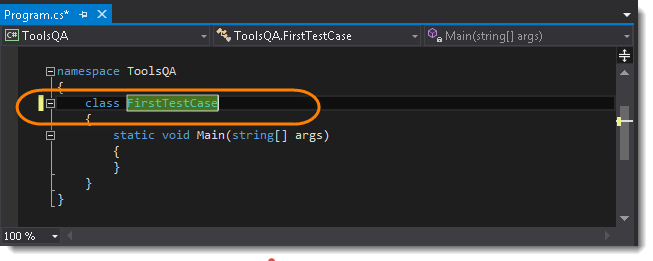
1) In the ***Solution Explorer****window*, which is on the right side of the *Visual Studio* in the above image. *Right Click* on the ***Program.cs*** and *Select****Rename***.



2) Notice that the text *Program.cs* is selected by default, now just *type* the new test case name ‘***FirstTestCase***‘.

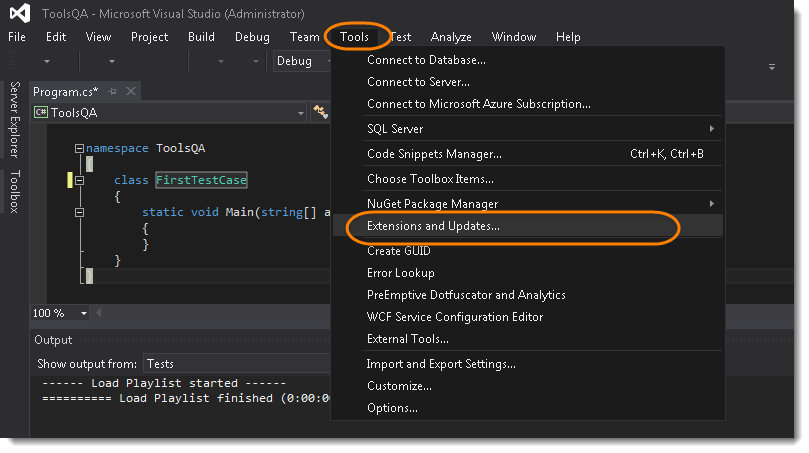


3) The new name will start reflecting everywhere in the project or code window.



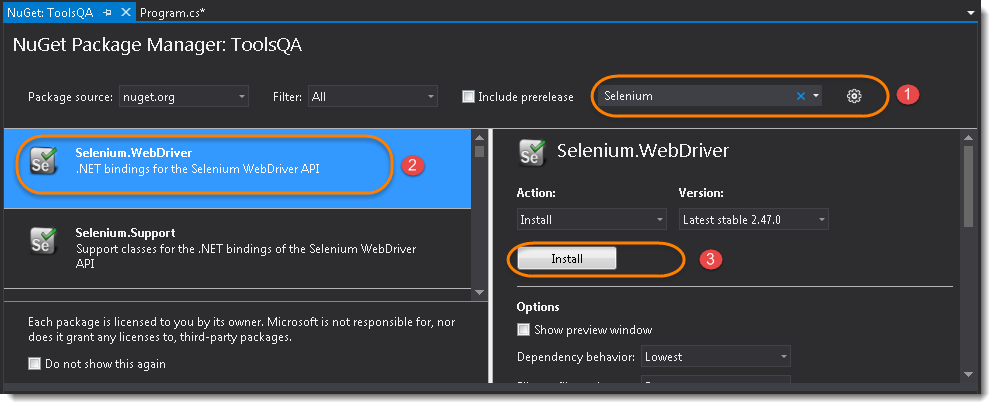
**Steps to Download Selenium WebDriver**

1) Go to ***Tools >> Nuget Package Manager >> Manage Nuget Packages for Solution..***.***.***

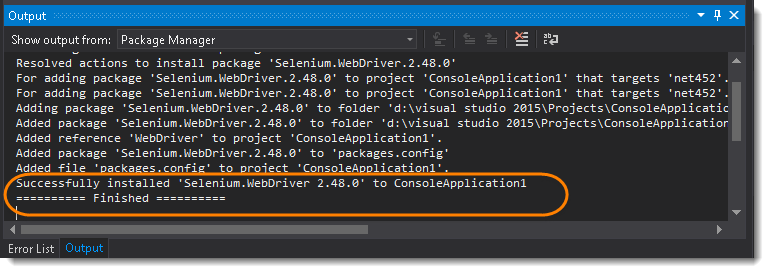


***Note****: Above screenshot is wrongly taken, please select Nuget Package Manager >> Manage Nuget Packages for Solution…*

2) In the *Seach Box*, search for ***Selenium.WebDriver*** or ***Selenium***. This will take few seconds to populate the *Selenium*. Once done, just *select* ***Selenium.WebDriver*** and *click* on ***Install*** to start the installation process.



3) Once Visual Studio is finished with the successful installation of the *Selenium WebDriver*, it will generate the output logs.



Now that we have the ***Selenium DLLs*** referenced in our project it time to test the configuration by running a quick test.

**Steps to Write the First Selenium C Sharp Test**

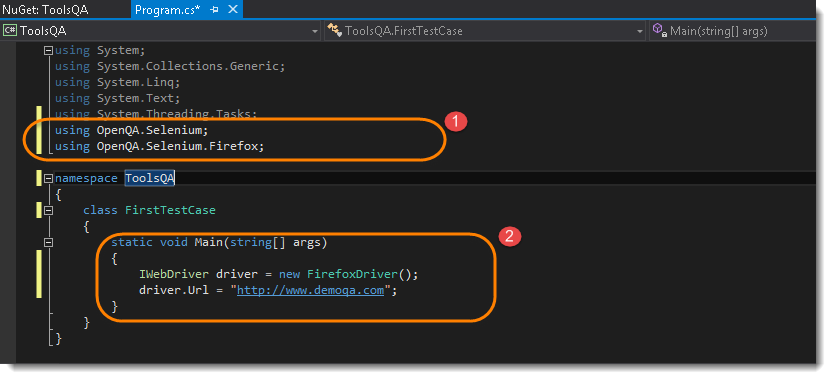
1) At the top of your project code after the last ‘***using***’ namespace add the following *Selenium namespaces*:

|  |  |
| --- | --- |
| 1  2 | using OpenQA.Selenium;  using OpenQA.Selenium.Firefox; |

2) Add the following code in your***static void Main*** section:

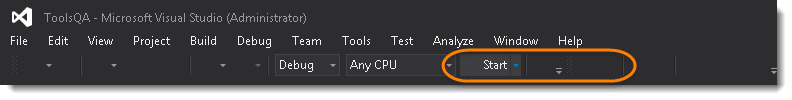
|  |  |
| --- | --- |
| 1  2 | IWebDriver driver = new FirefoxDriver();  driver.Url = "http://www.demoqa.com"; |

Now the Code window of the project should look like this:



***Note***:*We will come back to the understanding of the code later, once we start with Selenium Tutorial, but as of now just understand that the above code is creating an instance of Firefox driver and opening an URL in it.*

3) Run the test by clicking on the ***Start***button given on the top bar.

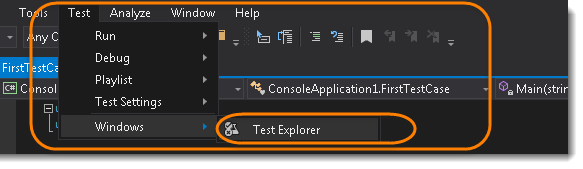


Notice that the Visual Studio started a Console application and just followed that it initiated a Firefox driver and opened the website.

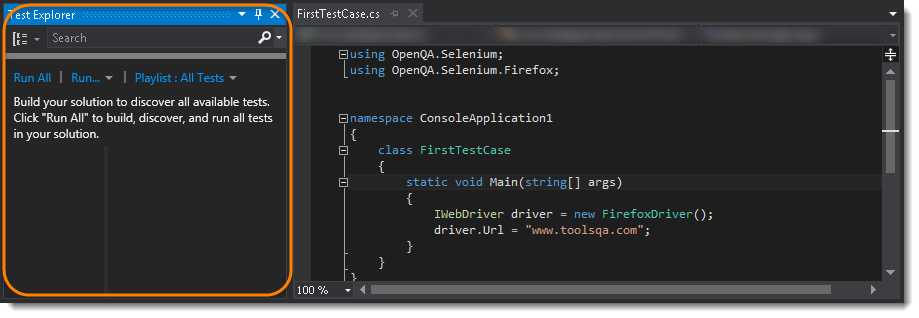
# How to write Selenium Test using NUnit Framework

we will learn the usage of ***NUnit Framework*** and we will also see how to download and install NUnit & NUnit adapter. But before that let’s just understand that why we need to use the NUnit Framework for Selenium test cases.

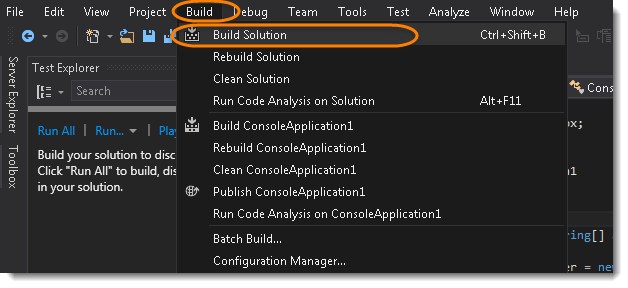
1) Go to ***Test >> Windows >> Test Explorer***.



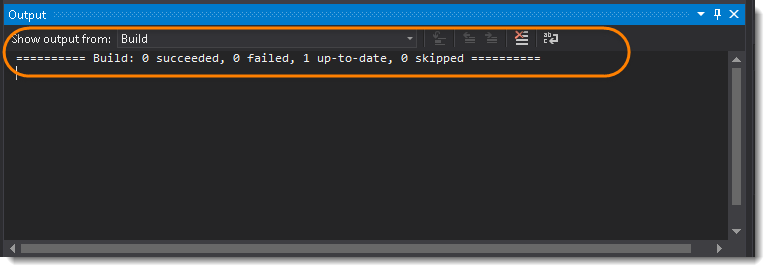
2) On the left hand side of the Visual Studio, a new window will appear if already not opened and this window is the ***Test Explorer*** Window.



3) Go to ***Build >> Build Solution***.



Notice that the Test Explorer window is still empty and Visual Studio is not able to find any test to run for the build. Even though if you are following the previous two tutorials, you will relies that we got one ***FirstTestCase*** in the project. Even in the ***Output window*** it says***0 succeeded and 0 failed***.



Now let’s just install Nunit Framework and Nunit Adapter and see what happens after that.

## What is NUnit Framework?

NUnit is a unit testing framework for performing unit testing based on the .NET platform. It is a widely used tool for unit testing and is preferred by many developers today. NUnit is free to use. NUnit does not create any test scripts by itself. You have to write test scripts by yourself, but NUnit allows you to use its tools and classes to make unit testing easier. The points to be remembered about NUnit are listed below:

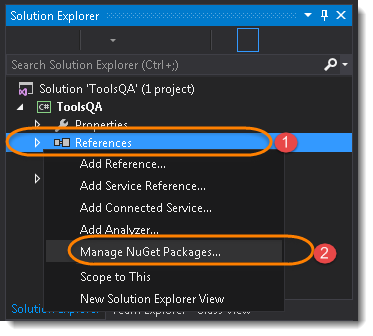
* NUnit is not an automated GUI testing tool.
* It is not a scripting language, all tests are written in .NET supported languages, e.g., C#, VC, VB.NET, J#, etc.

***What is NUnit Adapter?***

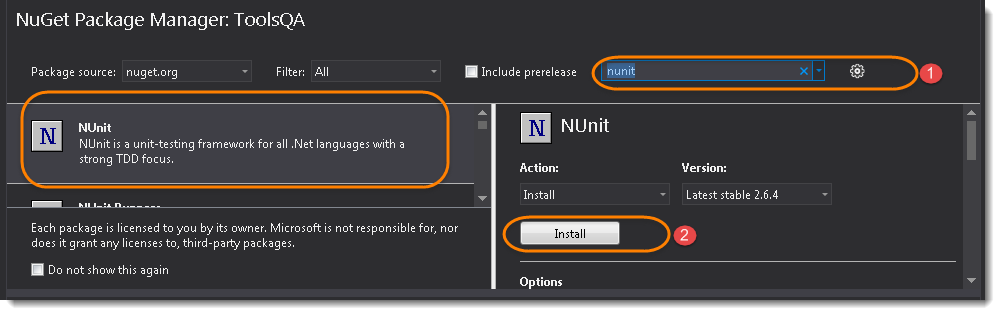
The NUnit Test Adapter allows you to run NUnit tests inside Visual Studio.

## Steps to Download and Install Nunit Framework using the NuGet Packages?

1) Go to ***References >> Manage Nuget Packages…*** from the Solution Explorer windowfrom the right side.

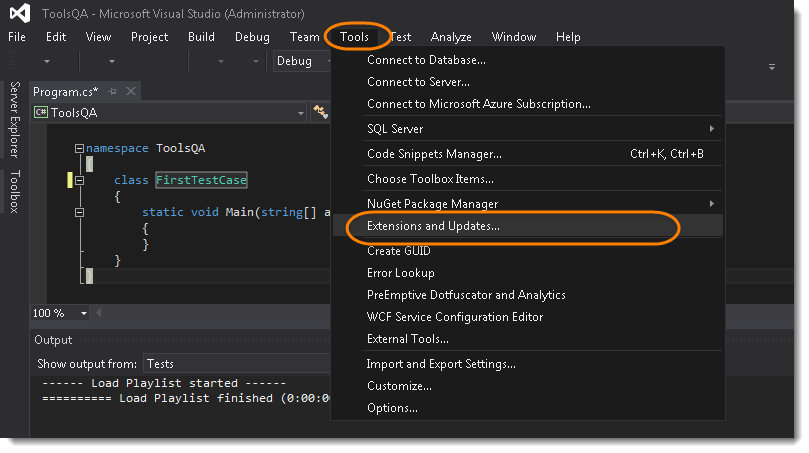


***Note***: Same can be done by going to ***Tools >>Nuget Package Manager***.

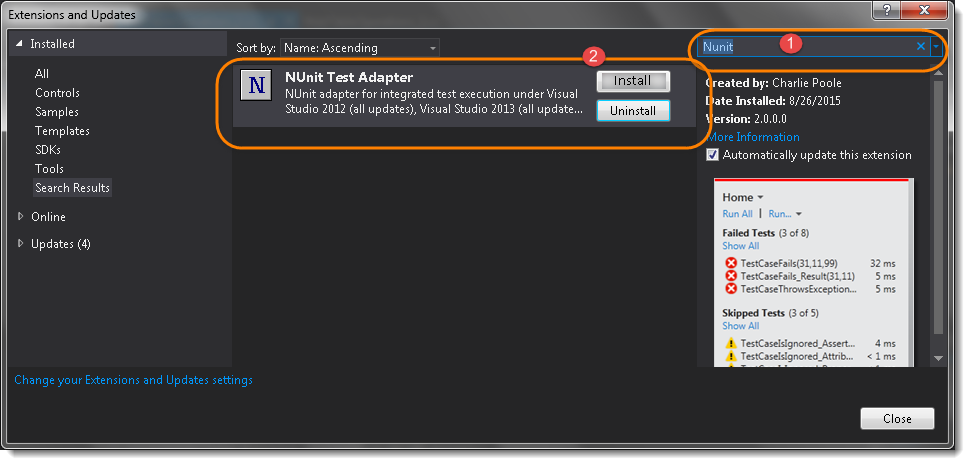
2) Type ***Nunit*** in the Search box. This will take few seconds to find the right tool.  Locate (search for) ***NUnit*** in the center panel and highlight it by selecting it and Click ***Install***.  


## Steps to Download and Install NUnit Adapter using the Extension Manager?

1) From within Visual Studio, select ***Tools >> Extensions and Updates..***



2) Search for NUnit Test Adapter and once found click on Install button.



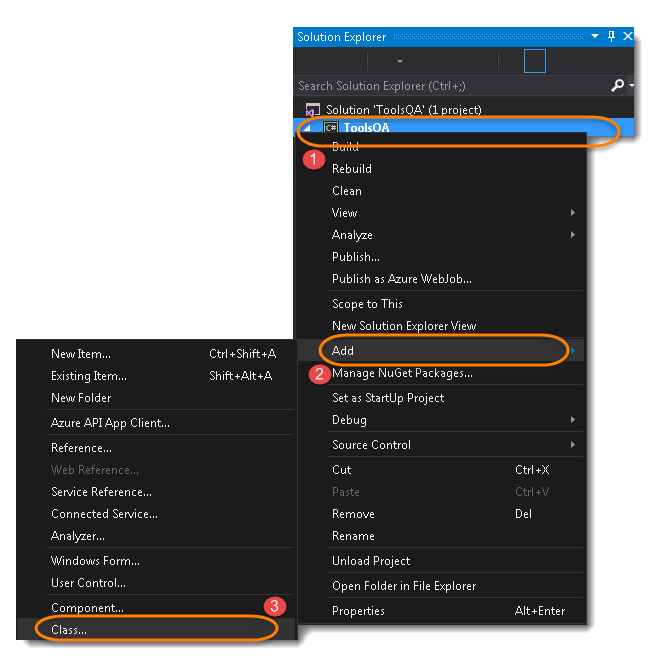
***Note***: Same can be downloaded from the Nuget package as well, below is the difference between Nuget and Extension.

## How to choose between Extension and NuGet package

The Extension will apply to Visual Studio itself, and will work for all projects you use. All users of your solution need to install the Extension.  
The Package will apply to the solution, and will work for any other user too, as it follows the solution.

## Steps to Create a NUnit Test

The first thing need to do is to create a new class  and write test code in that.

1) Right Click on the Project on the Solution Explorer and select ***Add >> Class***.

2) Give your Class name ‘***NUnitTest***‘ and click on ***Finish*** button. This will bring up totally a sweet class creation window.

3) Write the selenium code for the following steps:

* Start a FireFox Browser
* Open Website in the browser started by Selenium
* Close the Browser

***Code:***

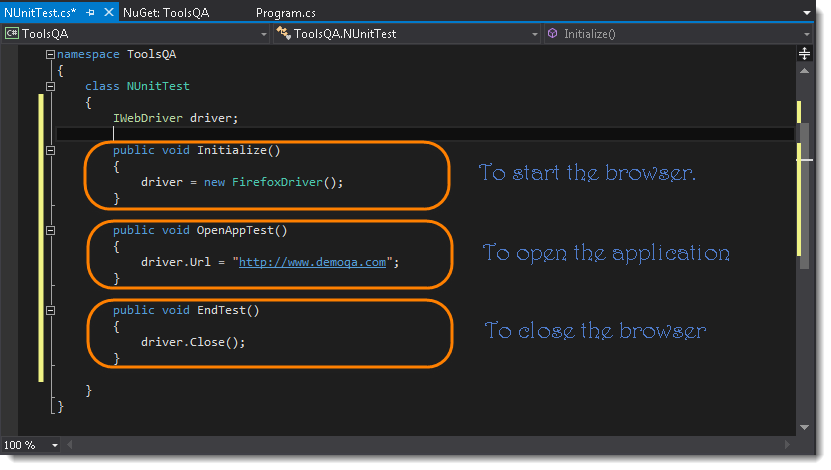


|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | using OpenQA.Selenium;  using OpenQA.Selenium.Firefox;  using NUnit.Framework;    namespace ToolsQA  {      class NUnitTest      {          public void TestApp()          {              IWebDriver driver = new FirefoxDriver();              driver.Url = "http://www.demoqa.com";              driver.Close();          }      }  } |

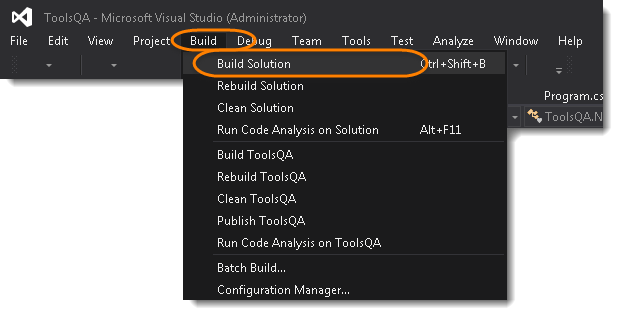
4) Divide the code in to three different parts:

* Start a FireFox Browser : ***Initialize***
* Open Website in the browser started by Selenium : ***OpenTestApp***
* Close the Browser : ***EndTest***

***Code:***



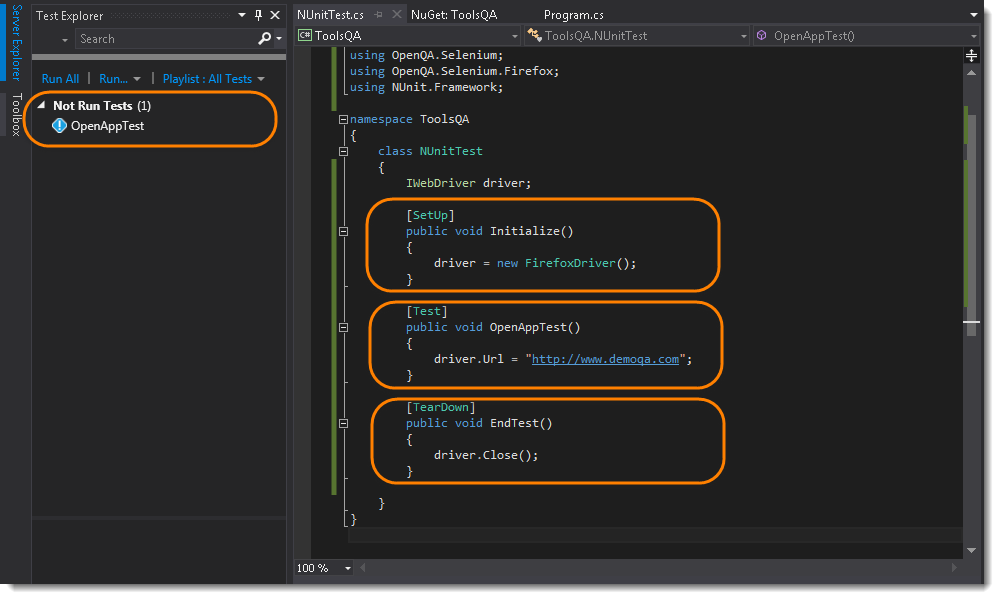
5) Now ***Build*** the solution again by doing ***Build >> Build Solution***.



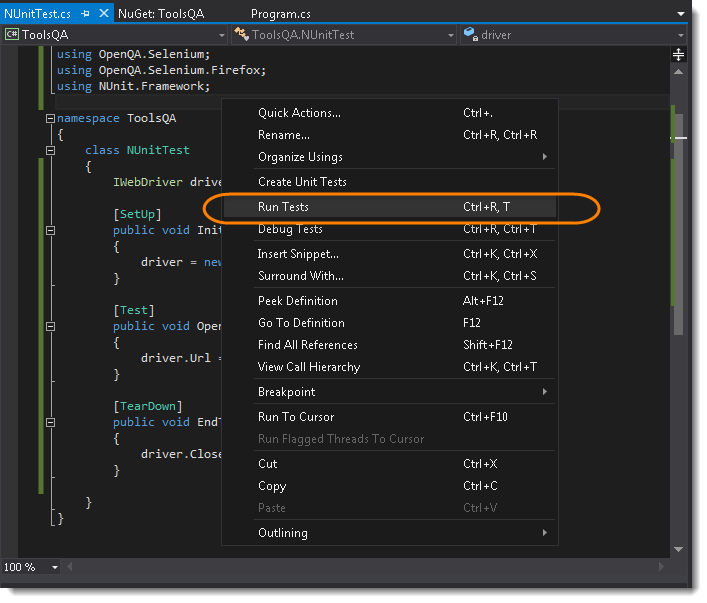
Take a look at the Test Explorer window, it is still Empty and not found any test to run for the build.

6) Now use NUnit Annotations to define your test. Use***[SetUp], [Test] & [TearDown]*** for the three methods created above for each action. And again run the Build

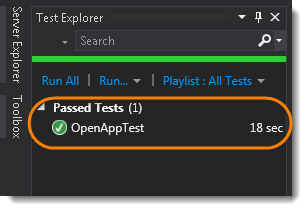
***Code:***



***Note***: Noticed that the Test is appeared in the Test Explorer window. So like this you can define unlimited number of Tests in the single class file but the SetUp method will run once before the every Test and TearDown method will also run once after every Test. Each test can be named differently like this.

7) Its the time to run the NUnit test. To run the test, Right Click in the Code window and select ***Run Tests***.  


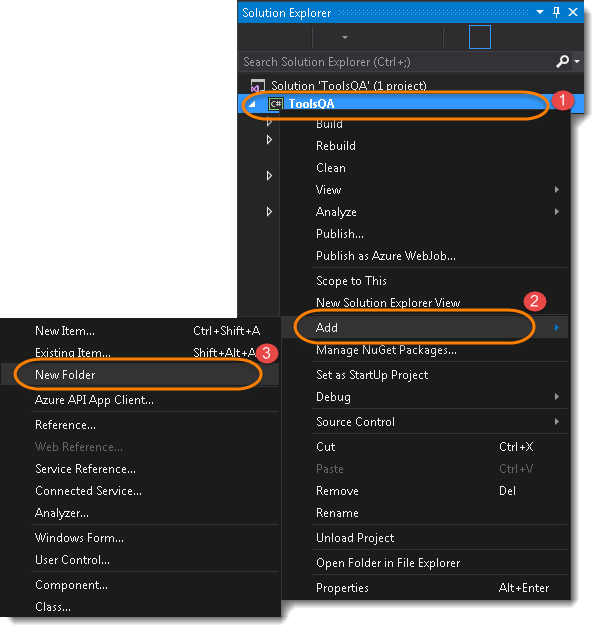
Selenium will start a Firefox browser, open the website and close it. Once the test run is finished, in the Test Explorer window the text above the OpenAppTest will change from ***Not Run Tests*** to ***Passed Tests***.



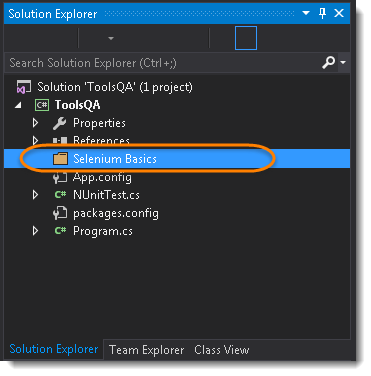
## Steps to Create a New Package

Now we are moving to learning Selenium, we will be doing lot of practice for Selenium Code and so it makes sense to create a different package for all the Selenium Test. To achieve that, we need to create a new folder in to the project solution.

1) Right Click on the Project and select ***Add >> New Folder*** and name it ***Selenium Basics***.

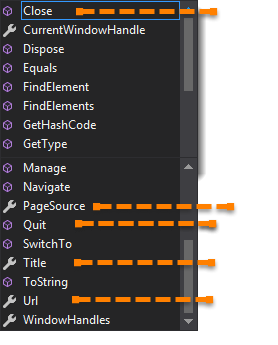


The***Solution Explorer*** will look like this now.

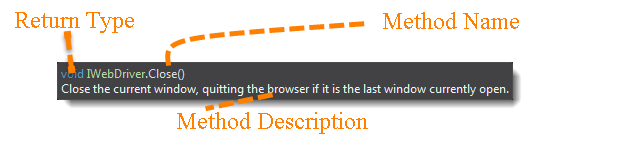


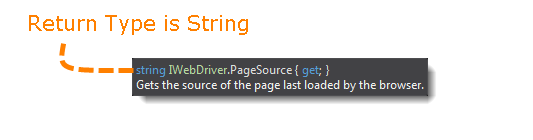
The very first question which comes to my mind and has been asked in many interviews is ***What is Selenium IWebDriver***? Is it an Automation Tool? Is it a Class? Is it an Interface or what actually it is? To answer this question we need to understand the ***Advance OOPs concepts*** first and then we would be able to visualize the ***IWebDriver Implementation***. For the sake of simplicity, we will avoid this IWebDriver Implementation topic for now and will cover this in later chapters. As of now we start with all the methods we get from IWebDriver.

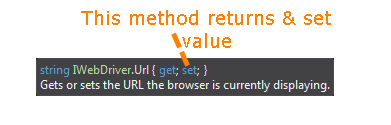
Now the next question is, How to access the methods of IWebDriver? To check what all we have in IWebDriver, create a driver object from IWebDriver and press ***dot key.*** This will list down all the methods of IWebDriver.



Let’s just start discussing the ***Orange colored*** methods of ***IWebDriver***but before that try to understand the syntax of the suggestions display by ***Visual Studio for IWebDriver***.







**Method:** A method is a collection of statements that are grouped together to perform an operation.

* ***Method Name:*** To access any method of any class, we need to create an object of class and then all the public methods will appear for the object.
* ***Getter & Setter :*** To get and set the value.
* ***Return Type:*** Method can returns a value or returning nothing (void). If the ***void*** is mentioned after the method, it means the method is returning no value. And if it is returning any value, then it must display the type of the value.
* ***Description*** : This is the short summary of the working of the method.

Now it would be very easy to understand the IWebDriver commands in the below chapter. The very first thing you like to do with Selenium is to ***Opening*** a new browser, ***Perform*** few tasks and ***Closing*** the browser. Below are the numbers of commands you can apply on the Selenium opened browser.

## URL Command

***string IWebDriver.Url { get; set;}***– This method ***Load***a new web page in the current browser window. This method get and set the value and can be used both ways.

**Command**– ***driver.Url = “appUrl”;  //To open the Url***

***Command*** – ***String Url = driver.Url;***  ***// To get the opened Url***

Where **appUrl**is the website address to load. It is best to use a fully qualified URL.

***Example:***



|  |  |
| --- | --- |
| 1 | driver.Url = ("http://www.demoqa.com"); |

## Title Command

***string IWebDriver.Title{ get;}***– This method fetches the ***Title*** of the current page. Accepts nothing as a parameter and returns a String value.

***Command – driver.Title;***

As the return type is String value, the output must be stored in String object/variable.

***Example:***



|  |  |
| --- | --- |
| 1 | String Title = driver.Title; |

## Page Source Command

***string IWebDriver.PageSource{ get;}*** – This method returns the ***Source Code***of the page. Accepts nothing as a parameter and returns a String value.

***Command – driver.PageSource;***

As the return type is String value, the output must be stored in String object/variable.

***Example:***



|  |  |
| --- | --- |
| 1 | String PageSource = driver.PageSource; |

## Close Command

***void IWebDriver.Close()*** – This method **Close** only the current window the IWebDriver is currently controlling. Accepts nothing as a parameter and returns nothing.

***Command – driver.Close();***

Quit the browser if it’s the last window currently open.

***Example:***



|  |  |
| --- | --- |
| 1 | driver.Close(); |

## Quit Command

***void IWebDriver.Quit()***  – This method **Closes** all windows opened by the IWebDriver. Accepts nothing as a parameter and returns nothing.

***Command – driver.Quit();***

Close every associated window.

***Example:***



|  |  |
| --- | --- |
| 1 | driver.Quit(); |

### Practice Exercise – 1

1. Launch a new Firefox browser.
2. Open Store.DemoQA.com
3. Get Page Title name and Title length
4. Print Page Title and Title length on the Console.
5. Get Page URL and Page Url Length
6. Print Page URL and Page Url Length on the Console.
7. Get Page Source (HTML Source code) and Page Source length
8. Print Page Length on Console.
9. Close the Browser.

### Solution

C#



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58 | using NUnit.Framework;  using System;  using OpenQA.Selenium;  using OpenQA.Selenium.Firefox;  namespace ToolsQA.Selenium\_Basics  {        class DriverCommands      {          [Test]          public void Test()          {              IWebDriver driver = new FirefoxDriver();                //Launch the ToolsQA Website              driver.Url = "http://www.demoqa.com";                // Storing Title name in String variable              String Title = driver.Title;                // Storing Title length in Int variable              int TitleLength = driver.Title.Length;                // Printing Title name on Console              Console.WriteLine("Title of the page is : "+Title);                // Printing Title length on console              Console.WriteLine("Length of the Title is : " + TitleLength);                // Storing URL in String variable              String PageURL = driver.Url;                // Storing URL length in Int variable              int URLLength = PageURL.Length;                // Printing URL on Console              Console.WriteLine("URL of the page is : " + PageURL);                // Printing URL length on console              Console.WriteLine("Length of the URL is : " + URLLength);                // Storing Page Source in String variable              String PageSource = driver.PageSource;                // Storing Page Source length in Int variable              int PageSourceLength = driver.PageSource.Length;                // Printing Page Source on console              Console.WriteLine("Page Source of the page is : " + PageSource);                // Printing Page SOurce length on console              Console.WriteLine("Length of the Page Source is : " + PageSourceLength);                //Closing browser              driver.Quit();          }      }  } |