Configure Selenium using NUnit in Visual Studio

Before writing the test script with the help of NUnit, we will understand **NUnit Framework**, downloading and installing the NUnit Framework in Visual Studio

* **What is NUnit Framework?**
* **Download the NUnit3 Test Adapter**
* **Adding the References for NUnit**
* **Create a NUnit class**
* **Write Selenium test script using NUnit**
* **Execute the test script**

What is NUnit Framework?

**NUnit** is a **unit testing** framework for all **.Net languages**. It is open-source software, and it has been completely redesigned to take advantage of many new features .Net language.

It also supports a wide range of .Net languages platforms.

Download the NUnit3 Test Adapter

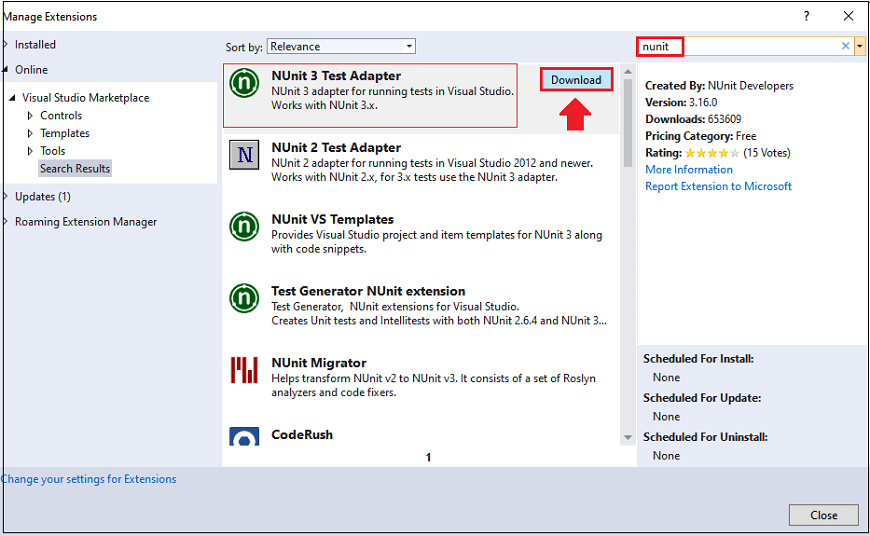
The **NUnit3 Test Adapter** is an extension that allows us to run the NUnit test inside the Visual Studio.

To download the NUnit3 Test Adapter in the Visual Studio, follow the below steps:

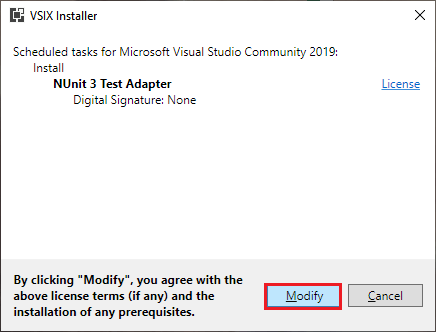
* First, go to the **Extensions** and select the **Manage Extensions** option in the Visual Studio.

Selenium with C# Tutorial

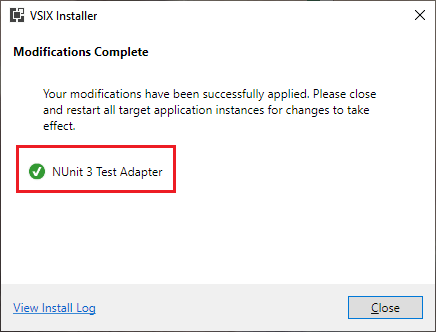
* The **Manage Extensions** window will appear on the screen, where we will search for **NUnit,** and click on the **Download** button for **NUnit 3 Test Adapter** as we can see in the below screenshot:



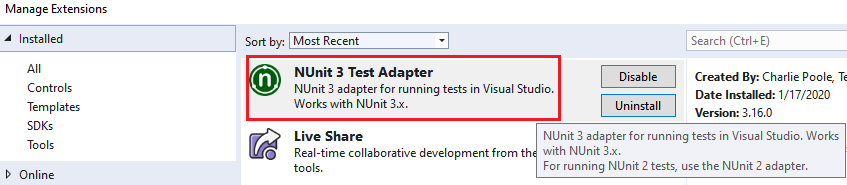
* Once we clicked on the Download button, it will ask us to close our Visual Studio to **modify** the changes, and then click on the Modify button as we can see in the below screenshot:



* Once we clicked on the Modify button, it gets start installing and modifying the NUnit 3 Test Adapter.
* When all the modifications are completed, we got the below window:



* As we can see in the Manage Extensions window, the **NUnit 3 Test Adapter** is successfully installed.

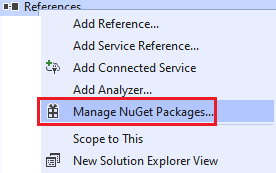


* Once we are done with the NUnit 3 Test Adapter installation process, we will add the references for NUnit.

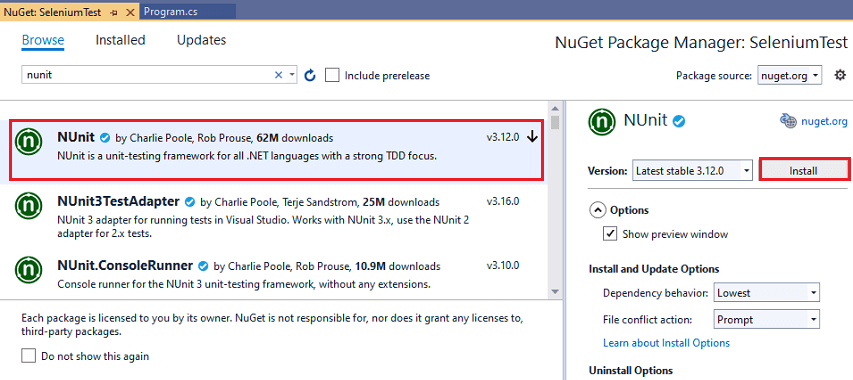
Adding the References for NUnit

To add the reference for NUnit, follow the below process:

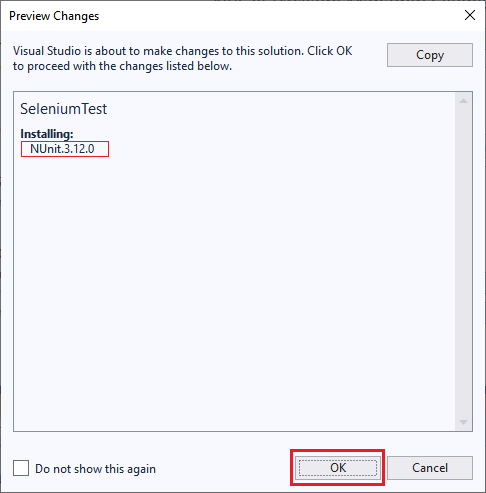
* Right-click on the **References** in **Solution Explorer**, and select the **Manage NuGet Packages** option in the given pop-up menu.



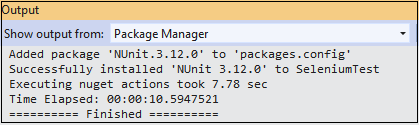
* In the **Manage NuGet packages** window, we will search for **NUnit** in the search field, and select the first option as **NUnit,** and click on the **Install** button as we can see in the below screenshot:



* Once we clicked on the Install button, **Preview changes** window will appear on the screen where we clicked on the **OK**



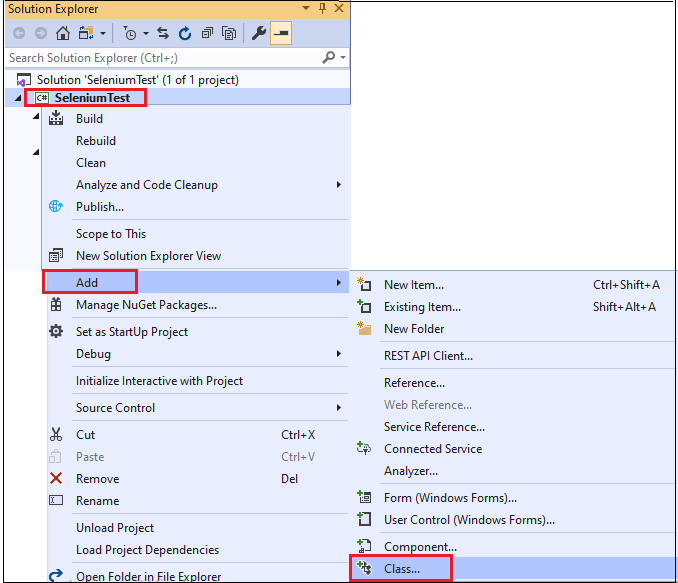
* Once the package got successfully installed, we will get the below output message.



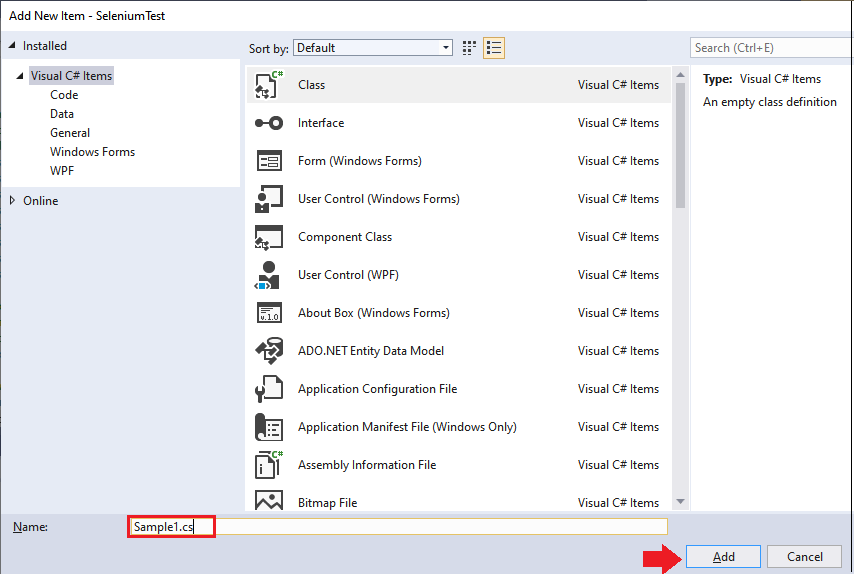
Create NUnit Class

To create a NUnit class in the Visual studio, follow the below process:

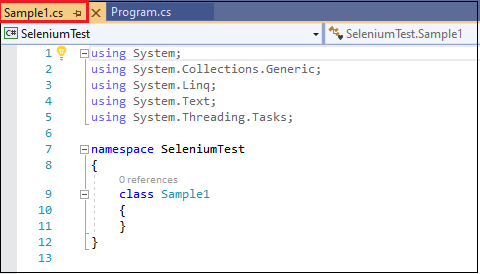
* Right-click on the **SeleniumTest** project, then go to **Add** and select **Class** option from the given pop-up menu.



* The **Add New Item** window will appear on the screen, where we will provide the **class name** as **Sample1,** and click on the **Add** button as we can see in the below screenshot:



* Once we clicked on the Add button, below window will appear on the screen:



* After creating the class successfully, we are ready to write our test script using NUnit.

Write Selenium Test Script using NUnit

To write a Selenium test scripts using NUnit, follow the below process:

For our testing purpose, we will perform **Login operation on the Facebook** Application.

|  |  |  |  |
| --- | --- | --- | --- |
| **Steps** | **Actions** | **Input** | **Expected Result** |
| **1.** | Open the Google Chrome browser. |  | The Google Chrome browser should be opened. |
| **2.** | Navigate to the Facebook login page. | [https://www.facebook.com](https://www.facebook.com/) | The Facebook login page must be displayed. |
| **3.** | Identify the **username** text box and pass the value. | Username=xyz11@gmail.com | The username text box should be identified, and the value should be entered. |
| **4.** | Identify the **password** text box and pass the value. | Password=####### | The Password text box should be identified, and the value should be entered. |
| **5.** | Identify the **Log in** button and click it. |  | The Login button should be identified and clicked. |
| **6.** | Close the browser. |  | The browser should be closed. |

We are creating our test script step by step to give you a complete understanding of how we write the test script with the help of the NUnit framework in Visual studio.

**Step1**

To access the Google Chrome browser, we will create the **IWebDriver** as a global variable.

Here the sample code:

1. //create the reference for the browser
2. IWebDriver driver = **new** ChromeDriver();

Note:  
Global variable: The Global variable is declared outside any function which is accessible to all functions throughout the program.

We will divide our code into different parts while writing the code using NUnit with the help of NUnit test methods like:

* For Open the browser**: Initialize**
* Perform browser operations**: ExecuteTest**
* Close the browser**: EndTest**

1. The syntax **for** the NUnit test method:
2. **public** **void** MethodName()

**Example:**

1. **public** **void** Initialize()
2. {
3. //open the browser
4. }
5. **public** **void** ExecuteTest()
6. {
7. //perform browser operations
8. }
9. **public** **void** EndTest()
10. {
11. //close the browser
12. }

**Step2**

In the next step, we will navigate to the given URL under the **Initialize()** method.

Here the sample code:

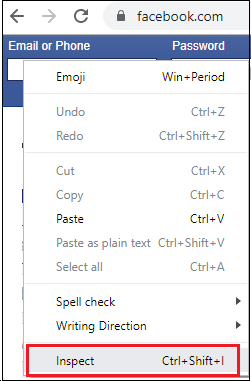
1. **public** **void** Initialize()
2. {
3. //navigate to URL
4. driver.Navigate().GoToUrl("https://www.facebook.com/");
5. //Maximize the browser window
6. driver.Manage().Window.Maximize();
7. Thread.Sleep(2000);
8. }

**Step3**

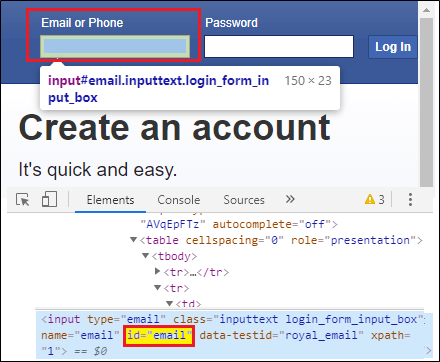
In this step, we will identify the **username text box** of the **Facebook** login page under the **ExecuteTest() method.**

Follow the below process:

* Right-click on the **username** text box, and click on the **Inspect** Element as we can see in the below screenshot:



* The developer tool window will open having all the specific codes, which are used in the development of the username text box.



* Copy the value of its **id** attribute as "**email**" and paste it into the code.

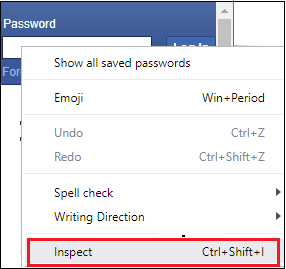
Here the sample code:

1. **public** **void** ExecuteTest()
2. {
3. //identify the username text box
4. IWebElement ele = driver.FindElement(By.Id("email"));
5. //enter the username value
6. ele.SendKeys("xyz11@gmail.com");
7. Thread.Sleep(2000);
8. Console.Write("username value is entered \n");

**Step4**

After that, we will identify the **password** text box of the Facebook login page, so for this follow the below process:

* Right-click on the **password** text box, and click on the **Inspect** Element option in the pop-up menu as we can see in the below image:



* It will launch a developer tool window that contains all the specific codes, which are used in the development of the password text box.



* Copy the value of its **name** attribute as "**pass**" and paste it into the code.

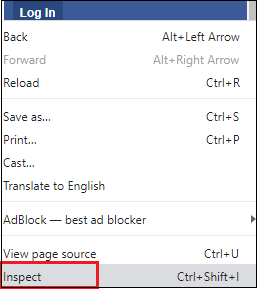
Here the sample code:

1. //identify the password text box
2. IWebElement ele1 = driver.FindElement(By.Name("pass"));
3. //enter the password value
4. ele1.SendKeys("########");
5. Console.Write("password is entered");

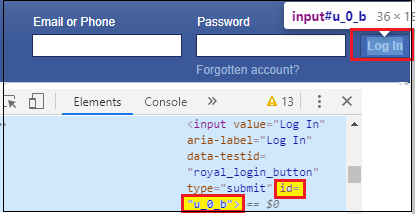
**Step5**

Once we identified the username or password textbox, we will find the **Log in** button and perform click operation.

* Right-click on the **Log in** button, and select the **Inspect** option from the given pop-up menu as we can see in the below image:



* The developer tool window will be launched with having all the specific codes, which are used in the development of the log in button as we can see in the below screenshot:



* Copy the value of its **Id** attribute "**u\_0\_b**" and paste it into the code.

Here the sample code:

1. //click on the Log in button
2. IWebElement ele2 = driver.FindElement(By.Id("u\_0\_b"));
3. ele2.Click();
4. Thread.Sleep(3000);
5. Console.Write("login button is clicked")

**Step7**

In the last step of our test script, we will close the browser under the **EndTest()** method.

Here, the sample code for closing the browser:

1. **public** **void** EndTest()
2. {
3. //close the browser
4. driver.Close();
5. }

After combining all steps together, our script will look like this:

**using** OpenQA.Selenium;

**using** OpenQA.Selenium.Chrome;

**using** System;

**using** System.Collections.Generic;

**using** System.Linq;

**using** System.Text;

**using** System.Threading.Tasks;

**using** System.Threading;

**namespace** SeleniumTest

{

**class** Sample1

{

//create the reference for the browser

IWebDriver driver = **new** ChromeDriver();

**public** **void** Initialize()

{

//navigate to URL

driver.Navigate().GoToUrl("https://www.facebook.com/");

//Maximize the browser window

driver.Manage().Window.Maximize();

Thread.Sleep(2000);

}

**public** **void** ExecuteTest()

{

//identify the username text box

IWebElement ele = driver.FindElement(By.Id("email"));

//enter the username value

ele.SendKeys("xyz11@gmail.com");

Thread.Sleep(2000);

Console.Write("username value is entered");

//identify the password text box

IWebElement ele1 = driver.FindElement(By.Name("pass"));

//enter the password value

ele1.SendKeys("########");

Console.Write("password is entered");

//click on the Login button

IWebElement ele2 = driver.FindElement(By.Id("u\_0\_b"));

ele2.Click();

Thread.Sleep(3000);

Console.Write("login button is clicked");

}

**public** **void** EndTest()

{

//close the browser

driver.Close();

}

}

}

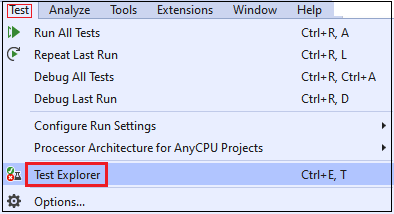
Note: In the above code, use your Facebook id at place: xyz11@gmail.com and password: ########

Execute the Test Script

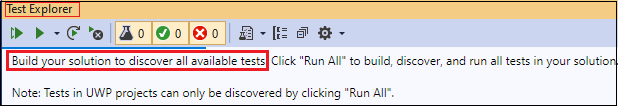
ADVERTISEMENT

To run the test scripts, follow the below process:

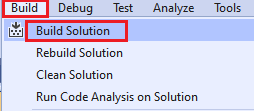
* Go to the **Test** in the toolbar and select **Test Explorer** as we can see in the below screenshot:



* The **Test Explorer** window will appear on the screen, to view all the available tests, we have to build our solution.



* To build the solution, go to **Build** and select **Build Solution** from given pop-up menu as we can see in the below screenshot:

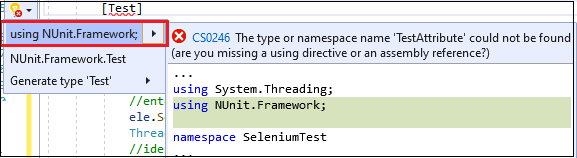


* Once we have done with the Build solution, we cannot see our Tests on the test explorer window because we did not add the **NUnit attributes**.
* The **NUnit** provides some inbuilt attributes, but here we are using **SetUp, Test, and TearDown**
* So, we will add **Test** attribute in our code just like below:

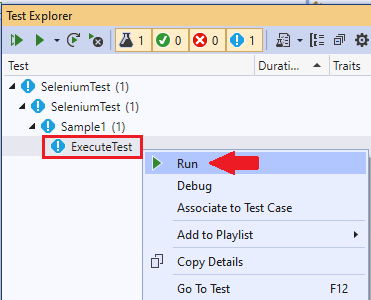
Note:  
Test attribute: It is used to mark the methods as callable from the NUnit test runner.

1. [Test]
2. **public** **void** ExecuteTest()

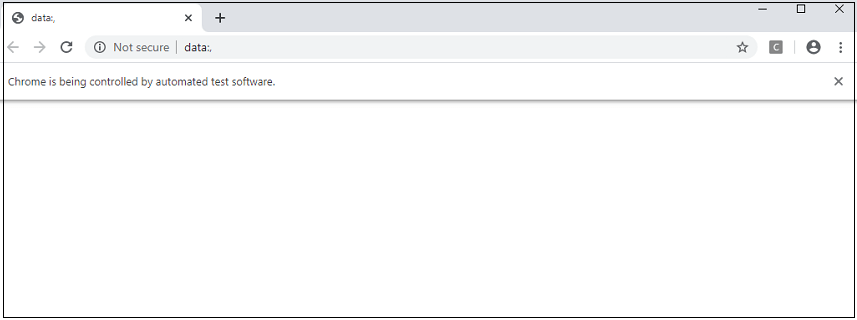
* While adding the **Test attribute,** it will give the below error.
* To overcome this error, we will use the reference of the **NUnit framework** as we can see in the below image:



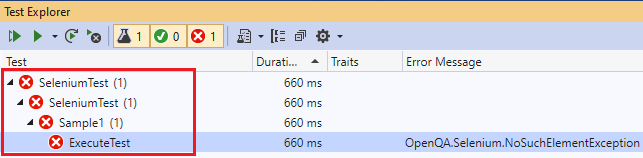
* Now, we go to the **Test Explorer** and Build the solution once again, and we will get the **ExecuteTest** on the **Test Explorer**
* After that, we will **Run** the selected test as we can see in the below image:



* After running the test script, It will able to open the browser as we can see in the below screenshot:



* The test will **Fail** because we have not called the **Initialize** method anywhere in the code.
* So we need to tell NUnit Framework by adding two more attributes called **SetUp and TearDown** into the code.

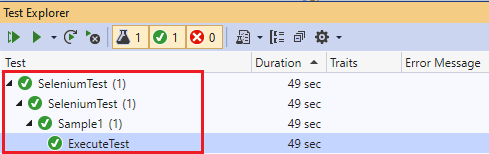


Note:  
SetUp: The SetUp attribute is used to identify a method to be called immediately; each test runs.  
TearDown: this attribute is used to identify a method that is to be called immediately after each test is executed. And this method is guaranteed to be called, even if an exception is thrown.

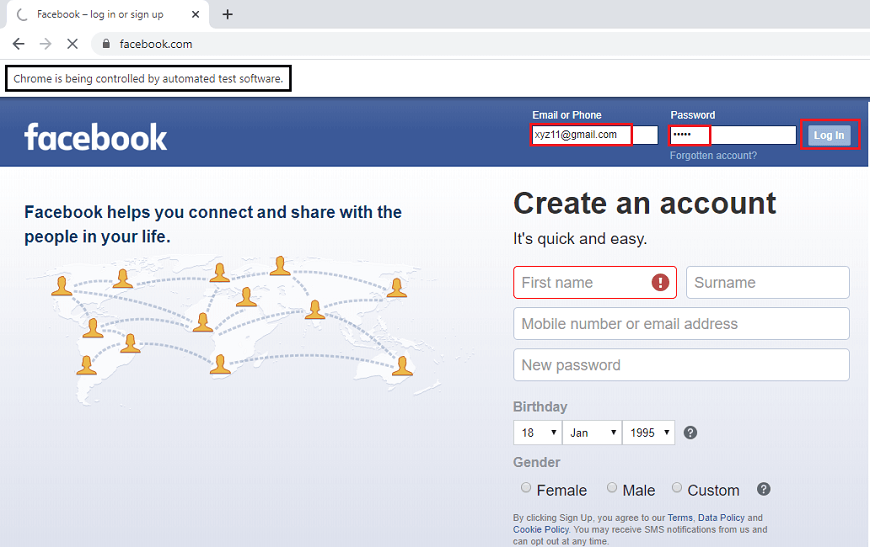
Our final code will look like this after adding all three attributes:

1. **using** OpenQA.Selenium;
2. **using** OpenQA.Selenium.Chrome;
3. **using** System;
4. **using** System.Collections.Generic;
5. **using** System.Linq;
6. **using** System.Text;
7. **using** System.Threading.Tasks;
8. **using** System.Threading;
9. **using** NUnit.Framework;
10. **namespace** SeleniumTest
11. {
12. **class** Sample1
13. {
14. //create the reference for the browser
15. IWebDriver driver = **new** ChromeDriver();
16. [SetUp]
17. **public** **void** Initialize()
18. {
19. //navigate to URL
20. driver.Navigate().GoToUrl("https://www.facebook.com/");
21. //Maximize the browser window
22. driver.Manage().Window.Maximize();
23. Thread.Sleep(2000);
24. }
25. [Test]
26. **public** **void** ExecuteTest()
27. {
28. //identify the username text box
29. IWebElement ele = driver.FindElement(By.Id("email"));
30. //enter the username value
31. ele.SendKeys("xyz11@gmail.com");
32. Thread.Sleep(2000);
33. Console.Write("username value is entered \n");
34. //identify the password text box
35. IWebElement ele1 = driver.FindElement(By.Name("pass"));
36. //enter the password value
37. ele1.SendKeys("########");
38. Console.Write("password is entered \n");
39. //click on the Login button
40. IWebElement ele2 = driver.FindElement(By.Id("u\_0\_b"));
41. ele2.Click();
42. Thread.Sleep(3000);
43. Console.Write("login button is clicked \n");
44. }
45. [TearDown]
46. **public** **void** EndTest()
47. {
48. //close the browser
49. driver.Close();
50. }
51. }
52. }

* After that, once again, do the build solution and run the selected test.
* Our test got passed in the Test Explorer window as we can see in the below screenshot:



* And the above test script will launch the Google Chrome browser and automate all the test scenarios as we can see in the below screenshot:



* Or if we want to see the output, we will pass some value and see the output by clicking on the **output link**.
* And we can see the output for all the operations that we perform on the console as we can see in the below screenshot:

