**SETTING UP AUTOMATION FRAMEWORK WITH SELENIUM (C#)**

The following step is required to set up an automation framework using selenium Webdriver with C#.

**Step 1: Download and Install Visual Studio**

This is can be downloaded from the Microsoft website : <https://visualstudio.microsoft.com/downloads/>. There are three versions available – community, professional and enterprise. The version of interest is the community as it is the free version while the other two needs to be purchased. Download the community version of visual studio (2017) and add the minimal workload required as appropriate. Windows related workloads are added for a start. Others can be added when required.

**Step 2: Installing Specflow Extension**

This is installed for every instance of visual studio downloaded. Invariably, it is installed once for community, professional and enterprise.

Steps:

1.Open Visual Studio

2. Go to Tools

3. Select Extensions & Updates

4. Click Online tab

5. Search for Specflow (type Specflow in the text box)

6. Click download (Specflow package id displayed)

7. Close visual studio on completion of download

8. Follow instruction on screen

Specflow should now be installed i.e. integrated with Visual Studio.

To confirm Specflow is installed, repeat steps 1 to 3 above and on step 4 click installed. Specflow should be on the list of installed packages.

**Step 3: Creating a Test Project**

Steps

1. Open Visual Studio

2. Click File

3. Select New

4. Select Project

5. Select Test (test framework under C# is used)

6. Select Unit Test Project (.NET framework)

7. Fill project details (name)

8. Browse to select the location where you intend to save your folders if needed

9. Click create directory for solution

10. Click Ok

A solution is created. The project is under solution. A solution will have 1 or more projects but at least one project.

To refresh your code or rebuild solution when changes are made, right click solution and select build solution.

**Step 4: Adding NuGet Packages**

NuGet packages needs to be added for every project individually. Each instance of project needs its own individual sets of NuGet packages.

The following NuGet packages are required:

Nunit

1. Nunit
2. Nunit.Console
3. Nunit.Runner
4. Nunit3TestAdapter

SpecFlow

1. Specflow
2. Specflow.Nunit

Selenium

1. SeleniumWebDriver
2. Selenium.Support

Browser

1. Selenium.WebDriver.IEDriver
2. Selenium.Firefox.WebDriver
3. Selenium.WebDriver.ChromeDriver

Make sure the project is open before commencing to install the NuGet packages.

Steps

1. Open Project to which NuGet packages is to be added to.

2. Click tools

3. Select NuGet packages manager

4. Select Manage NuGet packages for solution.

There are two preinstalled packages. It is a good practice to ensure both packages are there. They are MSTest.TestAdapter and MSTest.TestFramework.

5. Select browse and make sure you have internet connection (online)

6. Type in search text box NuGet package to be installed e.g. Nunit

7. Check the project that the package is required for if there are more than one package under the solution.

8. Click install

9. Package is installed and can be confirmed from the output window at the bottom of the screen.

10. Repeat steps 6 to 9 for all other NuGet package to be installed as in the list above.

**Step 5: Creating Folder Structure**

Folders required for automation is created at the project level. There are four folder levels that needs to be created. They are

* Features
* PageObjects
* StepDefinitions
* Utilities

**Features -** Specflow feature files are stored here. Feature files basically contains the step by step process of how to test a requirement in Gherkin format. This is the core of the test. It is basically the test steps in Gherkin format.

**PageObjects** - Every requirement to be tested will be an element on the webpage or application. Each of this element is viewed as being an object. An instance of an object is created for each element and stored in the PageObjects folder. PageObject will be different classes that is created for each of the page or element being inspected.

**StepDefinitions** - Detailed steps of how to achieve the test for each step in the features files is stored in StepDefinitions folder.

**Utilities** - This is also referred to as the helper class. All class or file that is not in the three above is stored here. This is more often the folder where the web browser is stored.

To create the folders above, the following steps is required.

Steps

1. Right click the project in question

2. Select Add

3. Select New Folder

4. Name the folder as appropriate i.e. Features

5. Repeat steps 1 to 4 for the other three folders

All done

**Step 6: Creating Feature Files**

Steps

1. Right click on the feature folder

2. Click Add

3. Click New Item

4. Click SpecFlow Feature File (type in name of file – advised to be as close as possible to what is being tested) e.g. Login, Registration etc.

5. Click Add (a C# class is automatically created – this can be viewed by clicking the filename under features in the explorer window).

6. Update description

7. Save

8. Build solution

Each feature to be tested is created as a scenario in the feature file. However, there can be more than one scenario in a feature file. Each scenario name must be unique.

**Step 7: Creating Step Definitions**

There are two methods that can be used to create step definition for the steps in the feature file.

**Method 1**

1. Open the feature file of the test in question (the Gherkin steps in the feature file will be in purple). This implies that there is no corresponding steps definition for the steps yet in the StepDefinitions folder.

2. Right click on any of the steps in purple in the scenario

3. Click on Generate Step Definitions (all steps with no matching step definition will be displayed)

4.Check either all or part the steps displayed depending on what is required to generate step definitions for them

5. Click generate

6. Navigate to the location of your StepDefinitions folder

7. Click save (change the file name but advisable to use the same filename as the feature file e.g. Login, Registration)

8. Step definitions is created

9. To confirm, Go to StepDefinitions folder in the solution explorer and click on the step definition file e.g. RegistrationSteps.cs or LoginSteps.cs

When adding more steps, method 1 cannot be used as it will overwrite the existing steps in the file. The steps below is to be used in that scenario.

Steps

1. Right click on the new step whose step definition is to be generated.

2. Click Go To Definition (a popup window “Go To Binding” appears)

3. Click yes to copy to clipboard

4. Open StepDefinitions file and paste in file

**Method 2**

Steps

1. Right click StepDefinitions folder in solution explorer

2. Click Specflow step definitions (create a step definition file)

3. Put in name of step definition file

4. Open the features whose step definitions is to be generated from the solution explorer (feature folder).

5. Right click on any purple statement

6. Click generate step definition

7. Click generate

8. Click filename for step definition file created in step 2

9. Click Save (warning file name already exists

10. Click Yes

**Step 8: Creating Hooks (Utilities)**

There are a few things that needs to be done before and after a requirement can be tested. There are two main classes of that which are

Before Feature and After Feature – Things that needs to be done before steps in the feature file can be executed and after feature file is executed.

Before Scenario and After Scenario – Things that needs to be done before scenario is executed and after scenario e.g. invoking a web browser and closing the web browser.

Hooks needs to be created to achieve those objectives.

[BeforeFeature] [AfterScenario]

public static void BeforeFeature() public static void AfterScenario()

{ {

Driver.Quit();

} }

[BeforeScenario] [AfterFeature]

public static void BeforeSenario() public static void AfterFeature

{ {

Driver =newChromeDriver(); Driver.Quit();

} }

Of interest is the BeforeScenario and AfterScenario. This can be achieved by creating hooks. The following steps is applicable to create hooks.

Steps

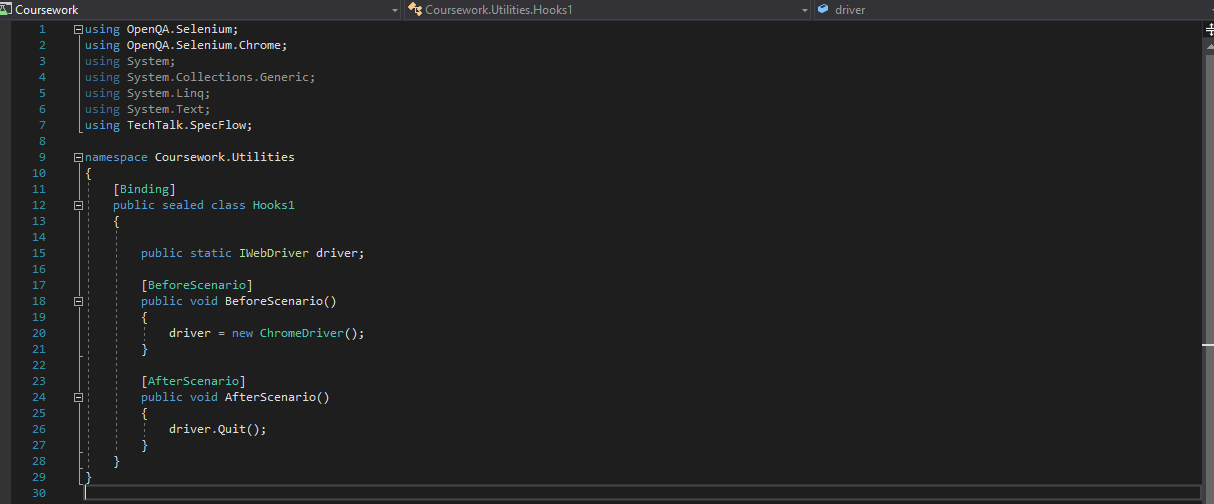
1. Right click utilities in solution explorer

2. Select new items

3. Select Specflow Hooks (change filename to Hooks or whatever you wish)

4. Click Add

The page overleaf shows a typical hook to open and close a web browser

Hook in C# for opening and closing a web browser - Chrome

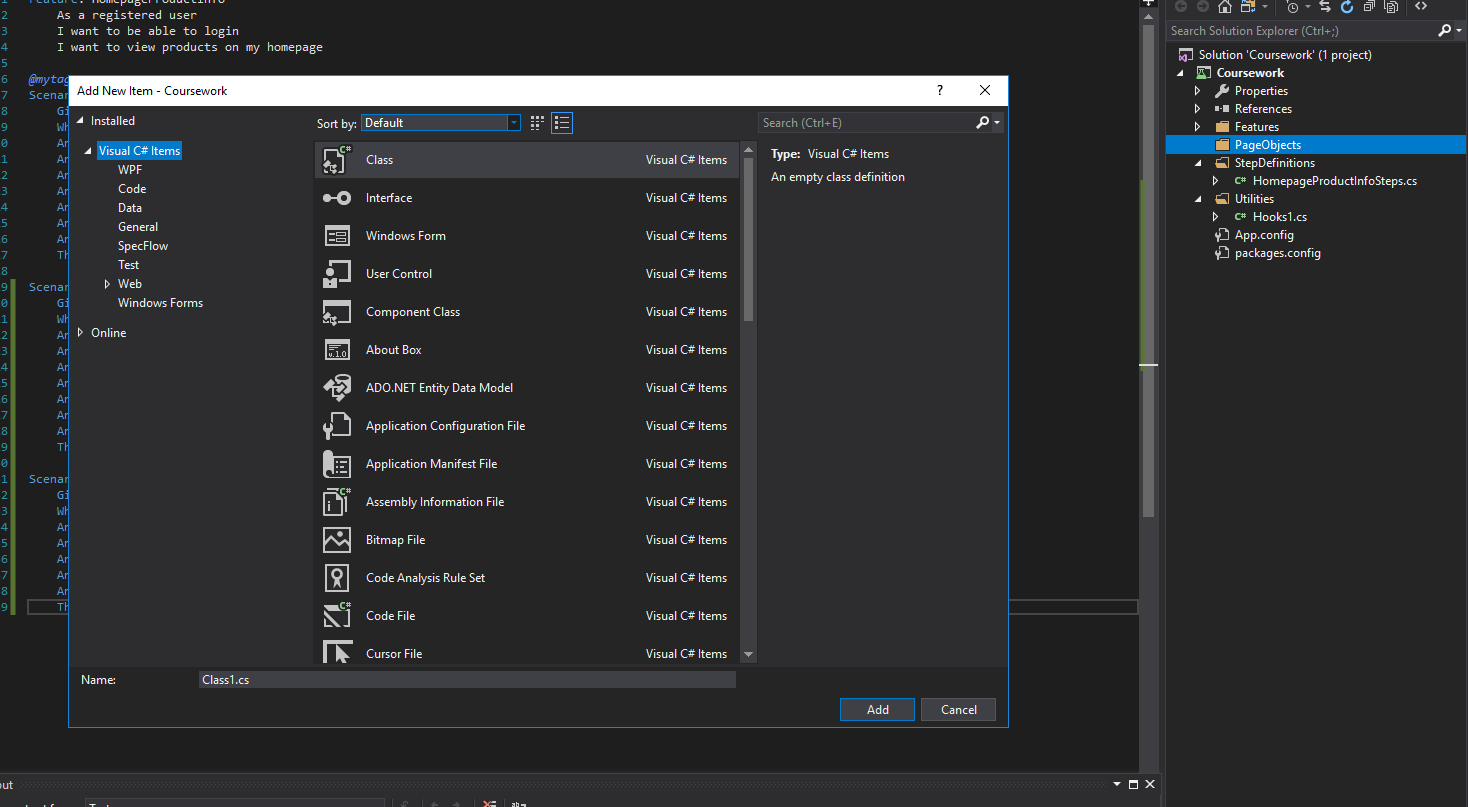
**Step 9 : Creating PageObject Files**

To create page object file, the steps below is required:

1. Right click PageObject folder in the solution explorer

2. Click Add

3. Select New Item

4. Select Class

5. Update name as required

6. Click Add (PageObject file is created).

Manual testing of a webpage or application entails the actual test being done by an individual manually by navigating to the website or application and manually testing as required. For example, to test the login functionality of giftrete website, an individual need to manually navigate to the site and follow the login steps. At the completion of the process, the result is one of two – either successful or failure login. Either way, there will be confirmation of the result by way of “welcome message” for successful login and “error message” for failed login.

The above process can be automated. In order to automate the process, we need to identify the site to be tested and then identify the functionality to be tested. Every functionality to be tested consists of elements which are stored as object on the page. The above example of manual login of giftrete website, the manual test is as detailed below:

1. Click login

2. Type email address

3. Type password

4. Click secure login

There will confirmation of successful or failed login on completion of process.

To automate the process, we need to identify the elements needed to automate the process ( login tab, email address field, password field, secure login button). To be able to automate the process, there needs to be a way of interacting with the elements. This can be achieved by inspecting the identified elements above.

**Inspecting an Element**

Steps

1. Right click element to be inspected

2. Select “Inspect Element” (a window pop up on the right of the screen showing Document Object Model (DOM) of the element.

3. Right click on the blue highlighted line in the DOM

4. Select Copy

5. Choose from the list of options of the type of copy – Copy Outer HTML, Copy Selector, Copy XPath.

6. Paste in your PageObject file in your project as needed.

The task of automation using falls in two parts – locating and inspecting the element and the second part is the action that needs to be carried out on the element.

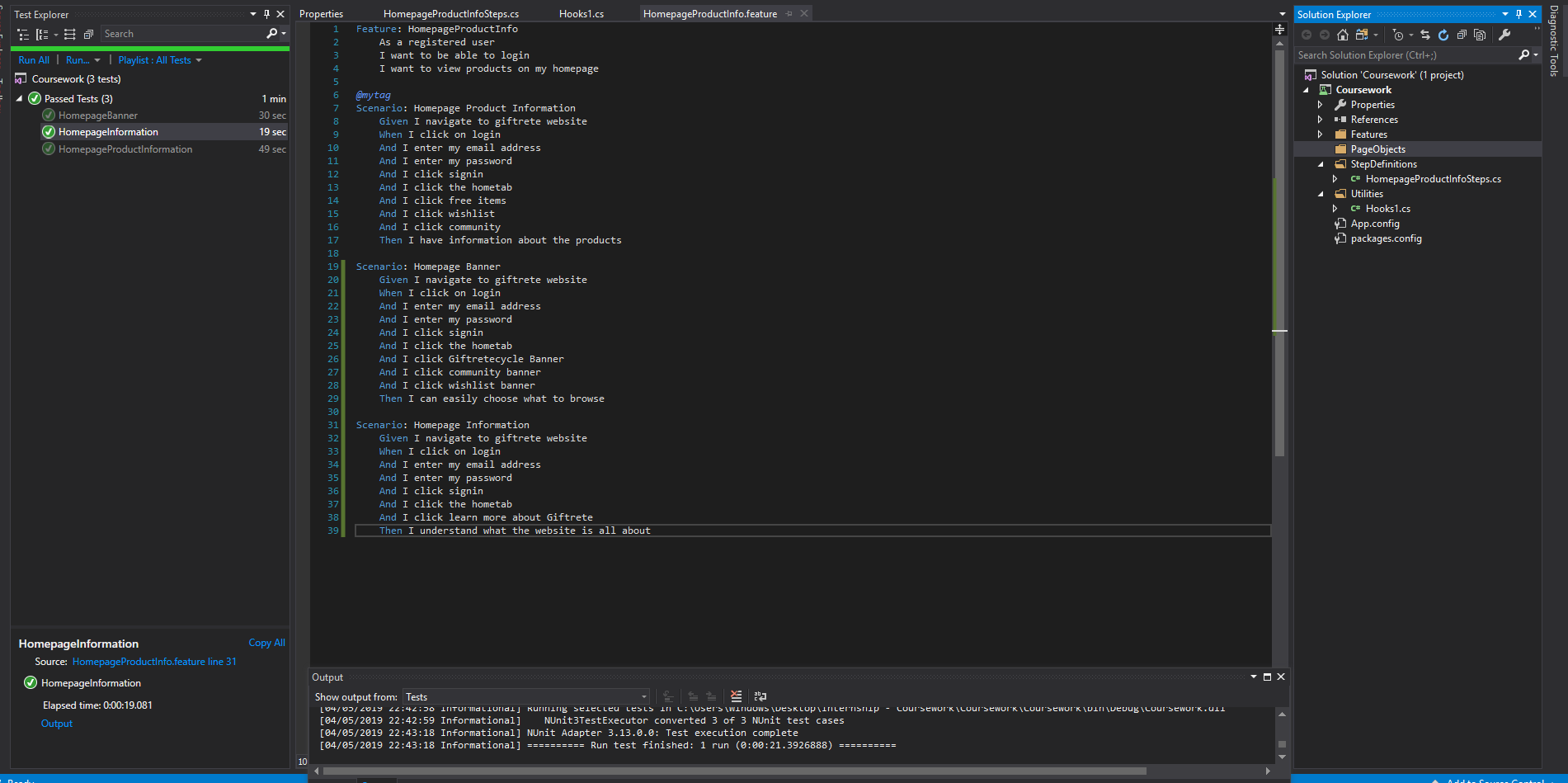
The example above, the elements and the action required on them is stated below.

Login (Button or Link) – Click

Email address (Field) -Type / Sendkeys

Password (Field) – Type / Sendkeys

Secure Login (Button) – Click



Automation Coursework – Passed Test