Selenium with C#

Learn how to write effective test scripts for web applications using Selenium with C#

Pallavi Sharma

www.bpbonline.com

First published: 2023

Published by BPB Online
WeWork
119 Marylebone Road
London NW1 5PU

UK | UAE | INDIA | SINGAPORE

ISBN 978-93-55518-187

www.bpbonline.com

Dedicated to

Neeal *and* ***Maisha***

My kids who continue to make me see the wonders of life

&

Dr Neelaksh *and* ***Dr Manju***

My parents because of them I am

About the Author

Pallavi has an experience of 15 years in Software Testing industry. She is a multi skilled professional who dons many hats from an IC to Project Manager to now a Founder at 5 Elements Learning. She also coaches people on open source test automation, and has taught more than 5000 people across globe. She is an instructor at Udemy, with more than 21K followers.

She is an active participant as a reviewer, and organizer for various international conferences like Selenium Official Conference Chicago, India, Agile Alliance, Scotland, Global Testing Retreat by Agile Testing Alliance, Selenium Summit, APISummit, Delhi Software Testing Conference, she keeps herself abreast with the latest developments in the software testing field. She also holds various global certifications in the field of automation.

She is a published author with BPB Publications and Lean Pub. She is an avid reader, writer and enjoys travelling. She also supports various NGOs and believes in the larger good.

About the Reviewer

Gaurav Gupta is a seasoned professional with over 12 years of experience in Test Automation. As a Test Automation Architect, he has honed my skills and gained profound knowledge in various programming languages and market-leading test automation tools.

Gaurav has a specific interest in test automation framework design for complex enterprise applications. Gaurav has designed test automation solutions for products in the healthcare and aerospace domains.

Gaurav is passionate about staying up to date with the latest trends and advancements in the test automation field. Leveraged machine learning concepts in test automation to develop automation suites of image processing ensuring that automation solutions are always cutting-edge and aligned with industry standards.

Acknowledgement

There are a few people I want to thank for the continued and ongoing support they have given me during the writing of this book. First and foremost, I would like to thank the people behind the Selenium project and community, who have contributed to the project and few who continue to keep the lamp lit. These are - Jason Huggins, Simon Stewart, Jim Evans, David Burns, Diego Molina, Manoj Kumar, Titus Fortner, Maaret Pyhäjärvi, to name a few!

I would like to thank my husband Ravinder for his unconditional love, and silly jokes. My parents for setting example to live life by giving. My children Neeal and Maisha who remind me to be a child at heart always. My sister Dr Mahak for her endless patience, and our conversations about life, environments, politics, humour and of course relatives.

I would like to thank Manish Jain, for giving me the opportunity with BPB publications, and helping me achieve my dream of becoming a writer. My thanks also extends to the editorial and technical team at BPB Publication for their continuous support.

I would like to thank all the readers who have taken an interest in my book and for their support in making it a reality. Your encouragement has been invaluable. I would also like to thank all the people who have given me the opportunity to coach them on Selenium, and helping me to learn along with them.

Finally my immense thanks to the various forms Almighty takes, and allows me to experience its endless wisdom, as I live.

Preface

Selenium automates browser. What you do with that power is entirely up to you. Selenium has been widely used in the field of test automation of web browser for decades now. Its first ever released was marked in April 2002. Twenty years later, it remains the most used end-to-end web automation tool, despite other solutions available in the market for web automation. Selenium automates browser, as a real end user and helps provide authenticity in the test automation process.

The popularity of Selenium is largely due to the fact that it provides compatibility across browsers, and operating systems. Selenium scripts can also be created using most popular programming languages like – Java, CSharp, Python, JavaScript etc. Selenium WebDriver, is a W3C standard which means that any browser in the market has to be compatible to Selenium. In this book, we have introduced Selenium, and its usage using CSharp as the programming language.

This book starts with introduction to Selenium, its three projects the Selenium IDE, Selenium WebDriver and Selenium Grid. We discuss in the book how to work with different types of web elements. What Selenium CSharp entities from the library are required and how they are used. We discuss advance test automation concepts of complex user action, management of browsers, and handling of data and object. The unit test framework for CSharp, NUnit is also discussed in the book which helps us write our first set of test cases. In the end, we learn how to set our tests to be executed in parallel, using the Selenium Grid.

Over the **14 chapters** of the book, you will learn the following:

Chapter 1: Introduction to the Selenium Project
Learn about the Selenium Project, Selenium IDE, Selenium WebDriver and Selenium Grid. Introduction of the Selenium CSharp Library. Finally learn how to set up the automation project in Visual Studio.

Chapter 2: Web Applications Used in the Book
Learn about the different Web Applications used in the book. Understand the different scenarios they provide to learn automation and practice.

Chapter 3: Browser Automation and More Using WebDriver
Learn about the IWebDriver interface for web browser. Understand the different browser drivers for different browsers, and how to set them up in the Visual Studio Project. Learn about the different IWebDriver methods and properties available to handle the browser.

Chapter 4: Handling Web Elements
Learn about the IWebElement interface. Understand the different properties and methods available in the interface which allows handling of the web elements for automation.

Chapter 5: Locate HTML Elements Using the By Class
Learn about the By Class, and how it is used to locate the web elements. Understand the different locator types available to help identify the web elements.

Chapter 6: Synchronization with Selenium
Learn about the concept and importance of introducing wait in automation scripts. Understand the different types of waits available in Selenium, where and how to use them.

Chapter 7: Working with HTML Elements - Part 1
Learn how to handle simple form web elements in this chapter. Web elements like textbox, button, radio button, checkbox etc are shown in this chapter. Understand methods to type text, click element and other similar actions to automate web elements of a web page.

Chapter 8: Working with HTML Elements - Part 2
Learn to work with web tables, and drop down web elements. Understand the different automation scenarios involving these elements, and how to handle them.

Chapter 9: Working with HTML Elements - Part 3
Learn to work with web elements like Alerts, Frames, IFrame, Windows. Understand how the different automations scenarios around these elements, and handle them.

Chapter 10: Actions, Options, and Capturing Screenshots
Learn about the management of different browsers, by using the Option Class. This chapter also covers Actions class, which explains handling complex user actions like double click, drag and drop etc.

Chapter 11: Unit Testing with NUnit

Learn about the need and importance of working with a unit test framework. Understand what NUnit is. How to write a test case, pass data and execute the test. Understand how to add assertions to the test to verify the actions.

Chapter 12: Learn How to Manage Objects Using a Page Object Model

Understand the need and importance of the object management in the test automation scripts. Learn the concept of Page Object Model design pattern and how it can be used to manage object information.

Chapter 13: Handling Test Data

Learn the necessity to manage data for test in the test automation scripts. Understand usage of excel and csv in data management for tests. And how we can integrate that with the CSharp scripts.

Chapter 14: Selenium Grid

Learn about Selenium Grid. Understand how it is used to run the tests in parallel. Understand setting the grid and executing test on it.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/tsfqmpb

The code bundle for the book is also hosted on GitHub at **https://github.com/bpbpublications/Selenium-with-C-Sharp**. In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Table of Contents

CHAPTER 1
Introduction to the Selenium Project

Selenium is a popular automation tool to drive the web browser. For more than a decade, it has been widely used by testers across the world to improve the work they perform. Web automation using Selenium is popularly used by organizations by creating an automation ecosystem around it using various other open-source or paid technologies. To understand and use Selenium, we must be aware of the programming language. This book targets the users of the CSharp language who would like to use Selenium to automate the web browser. This book also targets testers who use the CSharp language to build the automation process around their work and would want to use Selenium to automate the web browser. In this chapter, we will investigate the Selenium project, and understand the structure of the Selenium CSharp library. We will also see how we set up our project using Visual Studio. Finally, we will take a walk-through of the web application we will be using frequently to understand various automation concepts for tests.

Structure

This chapter is divided into the following sections-

- Selenium Project
- Understand the Selenium Library in CSharp
- Setting the Automation Project in Visual Studio

Objectives

After completing this chapter, you will be able to understand what a Selenium Project is and what the Selenium library of CSharp is. You will also understand how to set the system to get started with writing the first program in CSharp for automating the web browser.

Selenium project

Selenium, the stories behind how and why this name was chosen are many. The most famous is that Selenium the element is considered an antidote for Mercury. At the time of the creation of Selenium by Jason Huggins, while he was working with Thoughtworks, the automation tools from the Mercury organization were quite popular and at the same time, their licenses were expensive. So, as a cure for tools by Mercury, Selenium could be used. Selenium was then released as Selenium 1.0 and since then various releases of Selenium have been done. The latest version is Selenium 4.0. In the various releases of Selenium, the release 2.0 and 3.0 are considered to be milestones. It is in these releases Selenium married Webdriver, a solution created by Simon Stewart. With Selenium 3.0 release it became a W3C standard, which meant that any web browser which is compatible with the W3C standards will be compatible with Selenium. Since the 3.0 release, all used browsers like Chrome, and Firefox provide their own drivers using which Selenium can automate them.

Under the Selenium project, other components are also available. Selenium IDE, which is largely supported by Applitools, and Selenium Grid which allows one to execute tests in parallel using the grid structure. One of the major changes in Selenium from version 3.0 to 4.0 is a change in the implementation and structure of Selenium Grid. We will see that in the chapter, we will discuss how to set up and use Selenium Grid. To know and work with Selenium IDE, which could also assist with writing Selenium scripts, which we will discuss in detail in its chapter. The following table should help us understand where and how which Selenium component is used:

Selenium Project	Application
Selenium IDE	Selenium IDE is used to record and replay scenarios on a web browser. The recorded script can also be exported into various programming languages.

Selenium Project	Application
Selenium Webdriver	The latest version available is now Selenium 4.0, which supports automation of all major browsers like Edge, Chrome, Firefox, and Safari. The Selenium project is governed by various volunteers across the globe, and more details on it are available at - **https://www.selenium.dev/project/**. Selenium webdriver which is used primarily to automate browser actions can be implemented using different programming languages and be used on various operating systems.
Selenium Grid	Selenium Grid allows the execution of scripts for browser automation in parallel on different machine browser combinations. Various organizations like Sauce Labs, browser stack, and lambda test allows the execution of scripts on the cloud.

Table 1.1: *Projects in Selenium*

Selenium CSharp Library

The Selenium CSharp library provides various interfaces and classes to help automate user actions on browsers. The most important component of this library is the IWebDriver interface which provides a method to automate the browser. The IWebElement interface provides methods to work with web elements. And the By Class which helps us create locator methods to identify the object on which action is to be performed. The detailed Selenium dotnet library is available here - **https://www.selenium.dev/selenium/docs/api/dotnet/.**

It primarily consists of Interfaces and Classes.

Classes

Let us take a look at the following table, which lists the various classes available in Selenium (please note the following table is taken as it is from the preceding link, and no changes are made here)

	Class	Description
	By	Provides a mechanism by which to find elements within a document.
	Cookie	Represents a cookie in the browser.

	Class	Description
	DefaultFileDetector	Represents the default file detector for determining whether a file must be uploaded to a remote server.
	DriverOptions	Base class for managing options specific to a browser driver.
	DriverService	Exposes the service provided by a native WebDriver server executable.
	DriverServiceNotFoundException	The exception that is thrown when an element is not visible.
	ElementNotVisibleException	The exception that is thrown when an element is not visible.
	InvalidCookieDomainException	The exception that is thrown when the users attempt to set a cookie with an invalid domain.
	InvalidElementStateException	The exception that is thrown when a reference to an element is no longer valid.
	InvalidSelectorException	The exception that is thrown when an element is not visible.
	Keys	Representations of keys able to be pressed that are not text keys for sending to the browser.
	LogEntry	Represents an entry in a log from a driver instance.
	LogType	Class containing names of common log types.
	NoAlertPresentException	The exception that is thrown when an alert is not found.
	NoSuchElementException	The exception that is thrown when an element is not found.
	NoSuchFrameException	The exception that is thrown when a frame is not found.
	NoSuchWindowException	The exception that is thrown when a window is not found.
	NotFoundException	The exception that is thrown when an item is not found.
	Platform	Represents the platform on which tests are to be run.

	Class	Description
	Proxy	Describes proxy settings to be used with a driver instance.
	Screenshot	Represents an image of the page currently loaded in the browser.
	StaleElementReferenceException	The exception that is thrown when a reference to an element is no longer valid.
	UnableToSetCookieException	The exception that is thrown when the user is unable to set a cookie.
	UnhandledAlertException	The exception that is thrown when an unhandled alert is present.
	WebDriverException	Represents exceptions that are thrown when an error occurs during actions.
	WebDriverTimeoutException	Represents exceptions that are thrown when an error occurs during actions.
	XPathLookupException	The exception that is thrown when an error occurs during an XPath lookup.

Table 1.2: *Classes in Selenium*

Now, let us try and explore one of the classes and take a look at the By Class. The By class has some constructors, properties, methods, and operators. The detailed information of the By class can be explored once you click on the link of By.

Methods

The methods which we find when we look at the class are as follows:

	Name	Description
	ClassName	Gets a mechanism to find elements by their CSS class.
	CssSelector	Gets a mechanism to find elements by their cascading style sheet (CSS) selector.
	Equals	Determines whether the specified Object is equal to the current Object. (Overrides `Object.Equals(Object)`.)
	Finalize	Allows an Object to attempt to free resources and perform other cleanup operations before the Object is reclaimed by garbage collection. (Inherited from Object.)

	Name	Description
◆	**FindElement**	Finds the first element matching the criteria.
◆	**FindElements**	Finds all elements matching the criteria.
◆	**GetHashCode**	Serves as a hash function for a particular type. (Overrides `Object.GetHashCode()`.)
◆	**GetType**	Gets the Type of the current instance. (Inherited from Object.)
◆ S	**Id**	Gets a mechanism to find elements by their ID.
◆ S	**LinkText**	Gets a mechanism to find elements by their link text.
◆	**MemberwiseClone**	Creates a shallow copy of the current Object. (Inherited from Object.)
◆ S	**Name**	Gets a mechanism to find elements by their name.
◆ S	**PartialLinkText**	Gets a mechanism to find elements by a partial match on their link text.
◆ S	**TagName**	Gets a mechanism to find elements by their tag name.
◆	**ToString**	Gets a string representation of the finder. (Overrides `Object.ToString()`.)
◆ S	**XPath**	Gets a mechanism to find elements by an XPath query. When searching within a WebElement using xpath be aware that WebDriver follows standard conventions: a search prefixed with "//" will search the entire document, not just the children of this current node. Use ".//" to limit your search to the children of this WebElement.

Table 1.3: Methods of By class

Now, when you click on the By class and let us say you wish to explore the ID method, click on the ID method and we will find the method definition:

```
public static By Id(

      string idToFind

)
```

So, this method returns a By-object, which is identified by a string, which was the ID value of the object passed.

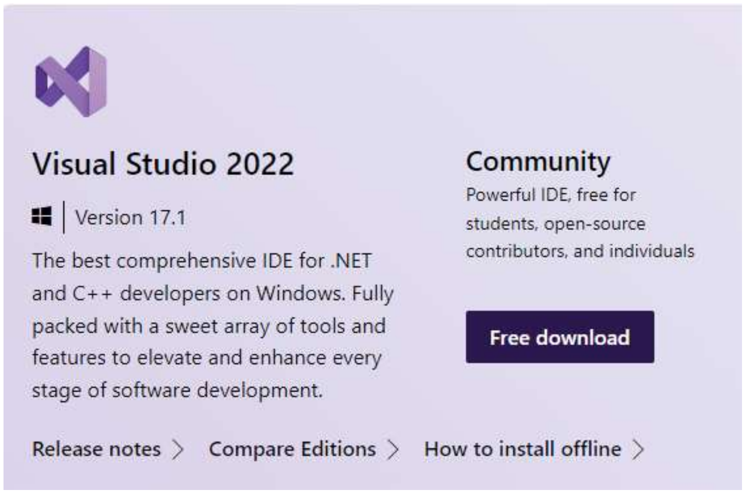
In this way, we can explore other classes of Selenium and the various methods available in those classes. This will help us write better code. In the same way, we can explore the interfaces of Selenium.

Setup Project in Visual Studio

Visual Studio is generally the default IDE which we use while working on the dotnet projects. And we need to understand here is t Selenium with CSharp project setup would be like a dotnet project setup. We will proceed in this book with the understanding that you are well familiar with CSharp as a programming language and are comfortable with using Visual Studio as an IDE. To set up and download Visual Studio in your system, we will follow these steps:

1. Download the Visual Studio community edition from here:

   https://visualstudio.microsoft.com/downloads/



Visual Studio 2022

Version 17.1

The best comprehensive IDE for .NET and C++ developers on Windows. Fully packed with a sweet array of tools and features to elevate and enhance every stage of software development.

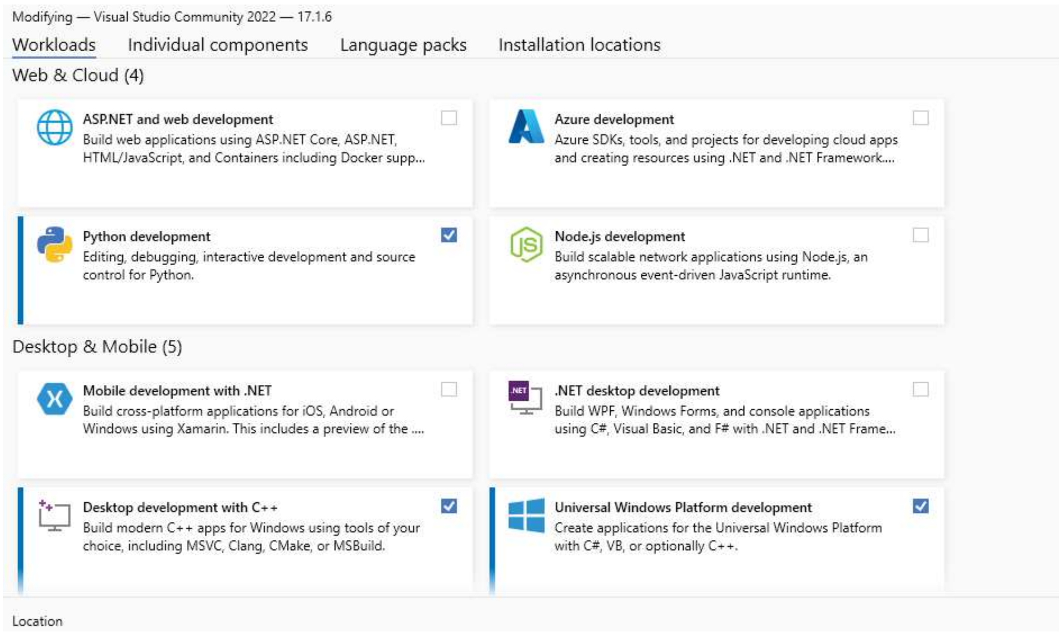
Community

Powerful IDE, free for students, open-source contributors, and individuals

Figure 1.1: *Visual Studio Community Edition*

2. After you have downloaded the Visual Studio Community Edition, the next step will be to install it on the system. Once you run the installer, Visual

Studio will start, and you will be required to set up some required packages. Select the one required for the dotnet development project:



Modifying — Visual Studio Community 2022 — 17.1.6

Workloads | Individual components | Language packs | Installation locations

Web & Cloud (4)

- [ ] **ASP.NET and web development** — Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker supp...
- [ ] **Azure development** — Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET and .NET Framework...
- [✓] **Python development** — Editing, debugging, interactive development and source control for Python.
- [ ] **Node.js development** — Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

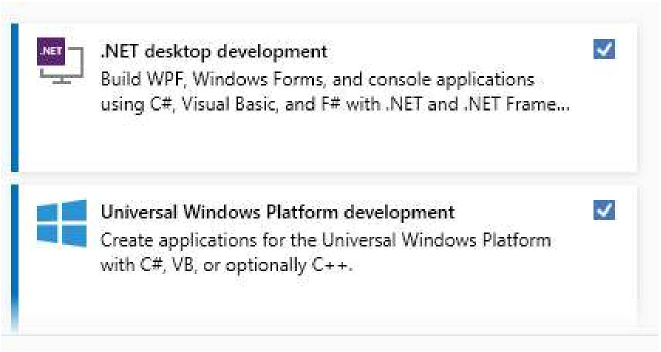
Desktop & Mobile (5)

- [ ] **Mobile development with .NET** — Build cross-platform applications for iOS, Android or Windows using Xamarin. This includes a preview of the ....
- [ ] **.NET desktop development** — Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET and .NET Frame...
- [✓] **Desktop development with C++** — Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.
- [✓] **Universal Windows Platform development** — Create applications for the Universal Windows Platform with C#, VB, or optionally C++.

Location

Figure 1.2: Components for Visual Studio

3. We will need to select the .Net desktop development and Universal Windows Platform development as shown in *Figure 1.3*:



- [✓] **.NET desktop development** — Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET and .NET Frame...
- [✓] **Universal Windows Platform development** — Create applications for the Universal Windows Platform with C#, VB, or optionally C++.

Figure 1.3: Component Selection

4. We then click on the extreme right side at the lower section of the modify button and allow the installation to take place:

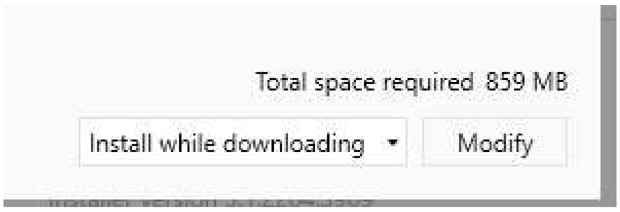


***Figure 1.4**: Allow component installation*

With the preceding steps, our Visual Studio IDE will be set up and ready to use. Our next step is project creation.

To set up the project in Visual Studio, we will need to choose the type of Project we wish to create. Since this book is aimed to use Selenium using Csharp for the automation of web testing processes, it will be advisable to select an NUnit Project type. In the upcoming chapter, we will discuss NUnit and eventually move on to that by adding relevant libraries. Currently, as we begin with our first steps on Selenium, we will select a project type of ConsoleApp, and add the required NuGet packages to set up Selenium. In the following steps, let us see how we do that:

1. After launching Visual Studio, we will first select **Create New Project:**

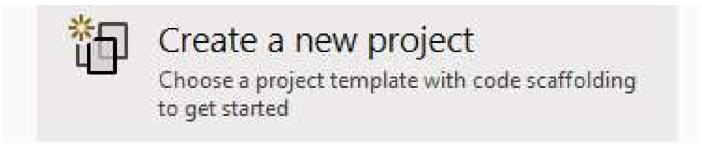

Create a new project
Choose a project template with code scaffolding to get started

***Figure 1.5**: Create New Project*

2. We can now from the list, select Console App (.Net Framework):

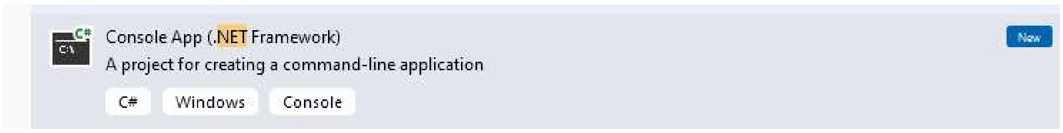
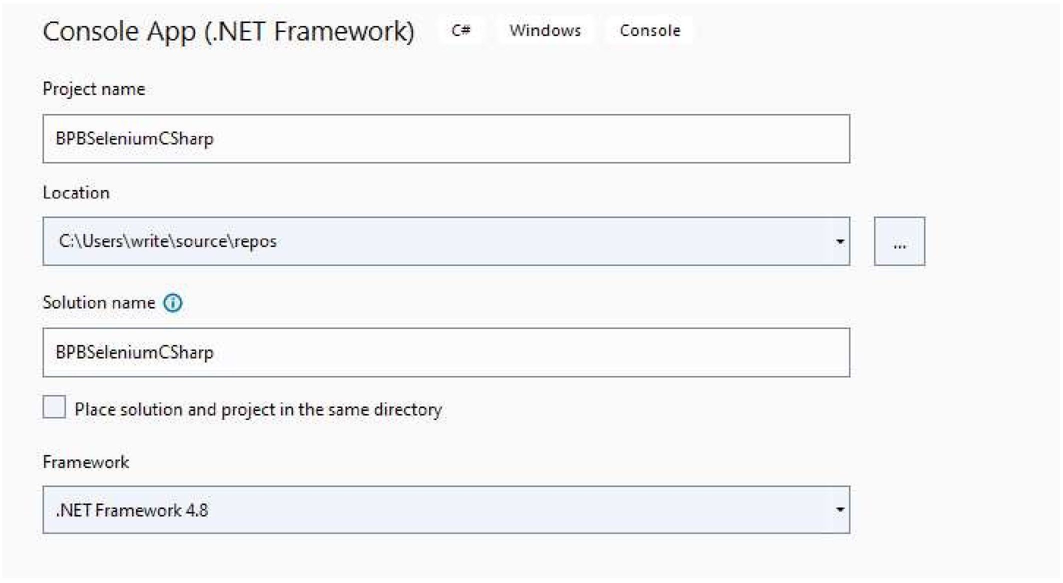

Console App (.NET Framework)
A project for creating a command-line application
C#
Windows
Console
New

***Figure 1.6**: Project Selection*

3. Provide the Project with a name, select 4.8 and .Net framework, and click on the **Create button**:



Console App (.NET Framework) C# Windows Console

Project name

BPBSeleniumCSharp

Location

C:\Users\write\source\repos

Solution name

BPBSeleniumCSharp

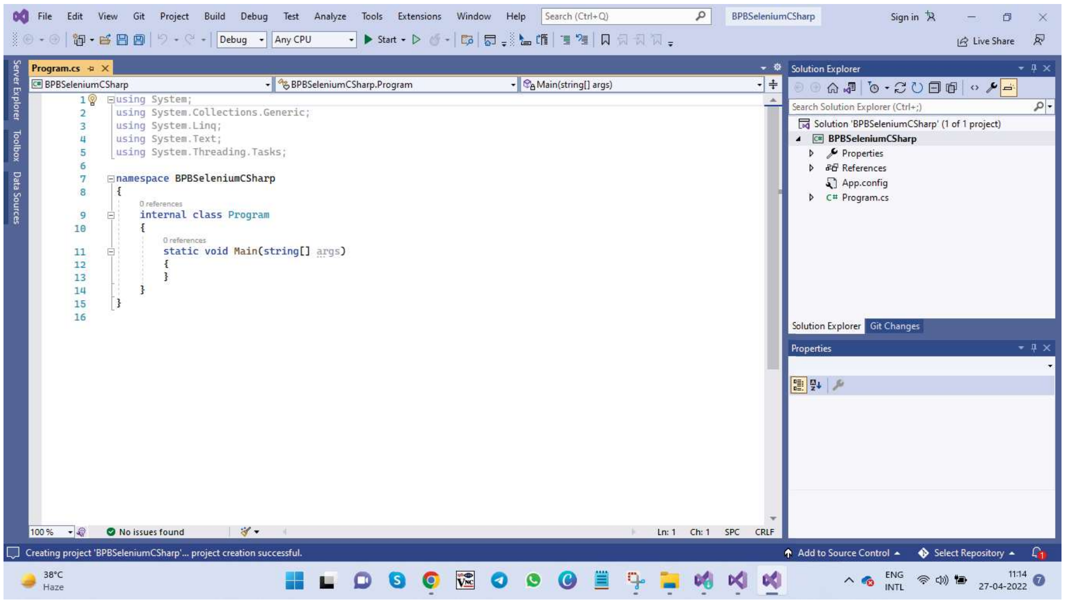
☐ Place solution and project in the same directory

Framework

.NET Framework 4.8

Figure 1.7 *Project Creation*

4. Once done, you should be able to see this screen, and on the right-hand side, it shows the project tree:



BPBSeleniumCSharp | BPBSeleniumCSharp.Program | Main(string[] args)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace BPBSeleniumCSharp
{
    0 references
    internal class Program
    {
        0 references
        static void Main(string[] args)
        {
        }
    }
}
```

Solution 'BPBSeleniumCSharp' (1 of 1 project)
- BPBSeleniumCSharp
  - Properties
  - References
  - App.config
  - Program.cs

Creating project 'BPBSeleniumCSharp'... project creation successful.

Figure 1.8: *CSharp Project*

5. Our next step here would be to add the relevant NuGet packages which would be required before writing down our first Selenium Script. For this, right click on the Solution, and select the Manage Nuget packages:

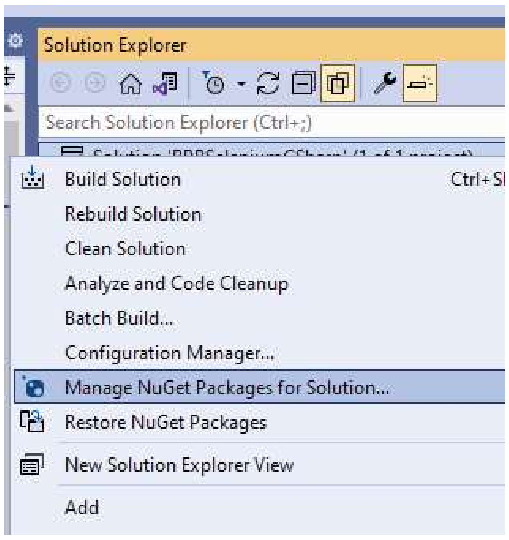
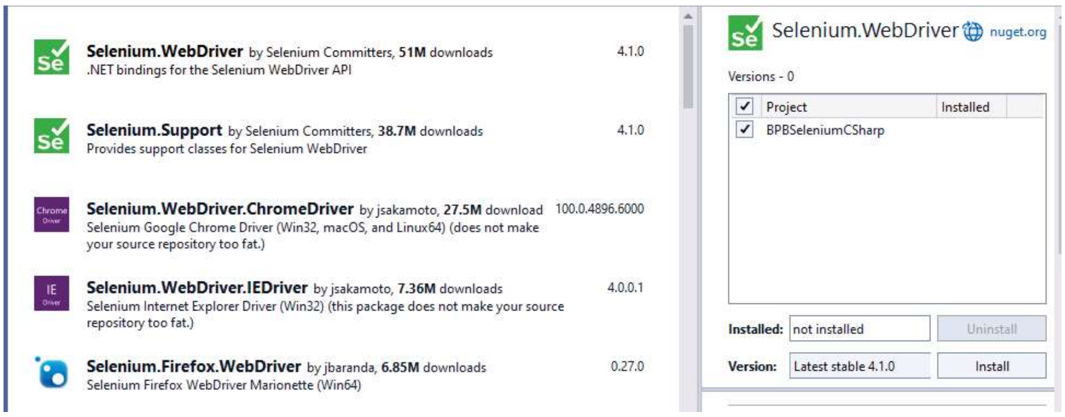


***Figure 1.9**: NuGet Packages*

6. Click on **Browse**, and search for Selenium, chose Selenium Webdriver, the first option, and on the window on the right-hand side, select Solution, which will ensure that any project created in the solution will have the Selenium WebDriver NuGet available. Click on **Install** as shown in *Figure 1.10*:



Selenium.WebDriver by Selenium Committers, **51M** downloads 4.1.0
.NET bindings for the Selenium WebDriver API

Selenium.Support by Selenium Committers, **38.7M** downloads 4.1.0
Provides support classes for Selenium WebDriver

Selenium.WebDriver.ChromeDriver by jsakamoto, **27.5M** download 100.0.4896.6000
Selenium Google Chrome Driver (Win32, macOS, and Linux64) (does not make your source repository too fat.)

Selenium.WebDriver.IEDriver by jsakamoto, **7.36M** downloads 4.0.0.1
Selenium Internet Explorer Driver (Win32) (this package does not make your source repository too fat.)

Selenium.Firefox.WebDriver by jbaranda, **6.85M** downloads 0.27.0
Selenium Firefox WebDriver Marionette (Win64)

Selenium.WebDriver nuget.org

Versions - 0

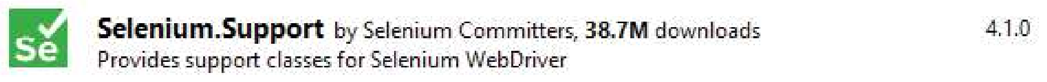
✓	Project	Installed
✓	BPBSeleniumCSharp	

Installed: not installed Uninstall

Version: Latest stable 4.1.0 Install

***Figure 1.10**: Install Selenium NuGet*

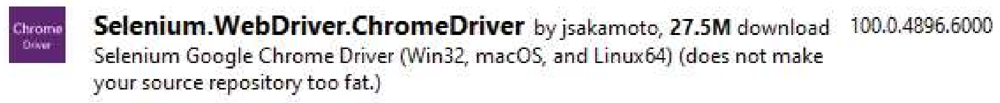
7. In a similar manner, our next NuGet to install would be Selenium Support:



Selenium.Support by Selenium Committers, **38.7M** downloads 4.1.0
Provides support classes for Selenium WebDriver

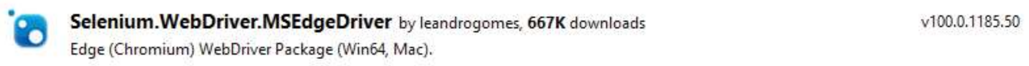
***Figure 1.11**: Selenium Support NuGet*

8. And after these two, we will install the NuGet for ChromeDriver, GeckoDriver, and EdgeDriver, as shown in *Figure 1.12*:



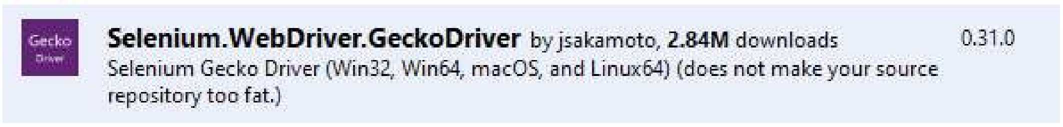
Selenium.WebDriver.ChromeDriver by jsakamoto, **27.5M** download 100.0.4896.6000
Selenium Google Chrome Driver (Win32, macOS, and Linux64) (does not make your source repository too fat.)

***Figure 1.12**: Selenium ChromeDriver Nuget*



Selenium.WebDriver.MSEdgeDriver by leandrogomes, **667K** downloads v100.0.1185.50
Edge (Chromium) WebDriver Package (Win64, Mac).

***Figure 1.13**: Selenium Edge Driver Nuget*



Selenium.WebDriver.GeckoDriver by jsakamoto, **2.84M** downloads 0.31.0
Selenium Gecko Driver (Win32, Win64, macOS, and Linux64) (does not make your source repository too fat.)

***Figure 1.14**: Selenium Browser Drivers NuGets*

Once all this is complete, we are now in a good position to write our first .cs file to open the web browser for Chrome, Firefox, and Edge, which we will explore in the next chapter, after we have a walk-through of our web application, which we will use at various places to understand the automation concepts.

Conclusion

In this chapter, we explored what is Selenium and the three projects of Selenium. We then moved on to understanding how to navigate through the Selenium dotnet library and its importance. We finally saw how to install and set up Visual Studio as the IDE for our automation project for testing using Selenium. We installed the crucial NuGet packages which we will require to get started.

In the next chapter, we will take a walk-through of the web application which we will be using to understand the different concepts. We will also take a look at some other websites we will explore in the book to automate scenarios.