Getting Started with WebDriver in C# Using Visual Studio

Posted Feb 13th, 2019

WRITTEN BY

Matthew Heusser

TOPICS

Programming languages | Frameworks | Get Started/Guide | Selenium

This article will demonstrate how to get WebDriver working with C# using Visual Studio.

Getting started with WebDriver's C# bindings in Visual Studio is easy -- if you know how to connect the pieces together. Once the pieces are in place, development is a snap. In this article we'll show you how to get the various parts and pieces, plus write and run one simple test.

This article is one in a series showing how to get WebDriver working in various editors and language platforms.

For an overview of how WebDriver works, please see the section "WebDriver Overview" in the blogpost "Getting Started with WebdriverSelenium for Java in Eclipse" here.

The Components You'll Need

To create and run WebDriver tests in C# using Visual Studio you'll need the following components:

- Visual Studio
- A test framework (We'll use NUnit; there are many you can use)
- WebDriver's C# bindings
- The ChromeDriver executable

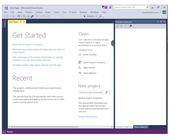
Getting Visual Studio

We expect the reader is using Visual Studio for work. If you don't have Visual Studio yet, take heart. The non-commercial version,Visual Studio Community, is available at no cost. Visual Studio Professional has a modest fee; it is also available on a monthly rental basis.This article uses Visual Studio Community, but you can use exactly the same approach with the Enterprise Edition!

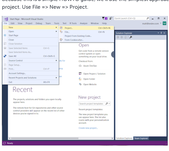
For this particular post we'll skip the mechanics of acquiring and installing due to the wide range of ways to get the product.

Creating The Project

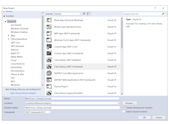
Once you've got Visual Studio installed and started, the next step is to create the project to work on. As with all editors and toolsets, there are multiple ways to accomplish the same basic goal of getting a test project built. If you're working in a team environment, please do your part to maintain sanity and a good team environment: follow the same approach the team uses!



Because this is a simple HOWTO guide, we'll use the simplest approach: A Class Library project. Use File => New => Project.



If this is the first time you've used this installation of Visual Studio you'll see a short delay while project templates are constructed. At this point you'll see the template selection dialog.

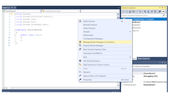


Choose the C# class library (.NET Framework) option. (Discussing the .NET Standard and Core options are beyond the scope of this article.) Give your project a good name and click OK. Visual Studio will create the project and add one class file to it.

Adding WebDriver and NUnit to Visual Studio

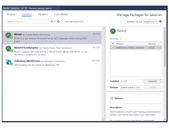
At this point we need to add WebDriver and our test framework NUnit to the project. Visual Studio has a great built in dependency management tool called NuGet. NuGet pulls dependencies like WebDriver from central repositories and adds them to your project. You'll need an internet connection to proceed.

Right-click the project, then select Manage NuGet Packages.



In the package management screen, first click "Browse" in the top left (instead of looking at installed package.) Then search for WebDriver. As of this writing, you will find a wide variety of packages with similar names. Some of them support Webdriver2 and others WebDriver3. Today you will install Selenium.Webdriver.Nunit3 and the NUnit3 Test Adapter. The version number of the Selenium Webdriver should be at 3.x, as of today the version is 3.141.0. If the version is a double-digit number or 2.X, then it will be incompatible. If you wish to use other browsers such as Internet Explorer or FireFox, then you'll need to install those drivers as well.

Once you have installed all three packages, click on "Installed." Your results should look like this:



Adding The Driver EXECUTABLE

Your C# code will not talk to the browser directly. Instead, Webdriver will call a .exe file that will drive the browser automation. Download Chromedriver, unzip it and put it in your windows path. You can tell this is successful by calling "cmd <enter>" to get to the command line in windows, typing "Chromedrive.exe." You will see a message like "Starting ChromeDriver ..."

Note for Internet Explorer, you can download the packages in NuGet and skip installing an intermediate executable.

Writing Your First Test

WebDriver doesn't know how to do anything other than talk to the browser driver. As a result, you'll need some sort of test framework to execute your tests, make assertions, and report test status. We'll use NUnit, which is popular, free, and easy to learn and use. There are many other test frameworks for the .NET platform. NUnit test cases are nothing more than class files added to the Visual Studio class library project. You can rename the initial "Class1.cs" file added to the project by default, or you can add another complete class by right-clicking the project and selecting Add Class.

A Simple Test

Below is a complete test case that starts a browser locally, executes a very simple test, then closes out the browser instance. The example is extremely simple and doesn't follow normal practices like using Page Object Patterns. This is example code, not production code!

```
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;


namespace WebDriver_CSharp_Example
{

    [TestFixture]
    public class Chrome_Sample_test
    {
        private IWebDriver driver;
        public string homeURL;


        [Test(Description="Check SauceLabs Homepage for Login Link")]
        public void Login_is_on_home_page() {

            homeURL = "https://www.SauceLabs.com";
            driver.Navigate().GoToUrl(homeURL);
            WebDriverWait wait = new WebDriverWait(driver,
System.TimeSpan.FromSeconds(15));
                wait.Until(driver =>
driver.FindElement(By.XPath("//a[@href='/beta/login']"));
                IWebElement element =
driver.FindElement(By.XPath("//a[@href='/beta/login']"));
            Assert.AreEqual("Sign In", element.GetAttribute("text"));


        }


        [TearDown]
        public void TearDownTest()
        {
            driver.Close();
        }


        [Setup]
        public void SetupTest()
        {
            homeURL = "http://SauceLabs.com";
            driver = new ChromeDriver();

        }


    }
}
```

Running The Test

When you first create or open a project, Visual Studio doesn't know what tests are in it. You need to first populate the list of tests. Do this by opening Visual Studio's Test Explorer via Tests => Windows => Test Explorer. Select Run All Tests to build the project then automatically discover and run all tests—in this case, one.

You'll see (hopefully!) a green test in the Explorer window which means your test passed.

Wrapping It All Up

In this post you learned a bit about the different versions of Visual Studio, where to find the free Community version, how to create a basic project and add the various WebDriver pieces necessary for C# WebDriver tests, and we showed you an end-to-end test.

Good luck with your explorations of WebDriver!

Related resources

Khan Academy Ensures a World-Class Online Education

Ultimately, Sauce Labs lets Khan Academy develop more educational content without sacrificing ...

Khan Academy: Ensuring a World-Class Online Educational Experience

Khan Academy uses Sauce Labs to execute their entire end-to-end test suite up to 24 times per day.

WHITE PAPER

DZone Refcard - Getting Started with Appium

ARTICLE

Selenium Tips: CSS Selectors