Developer Tips: C# Selenium with MSTest Basics

Selenium is a great tool for testing your user interface (UI). There are plenty of [great tutorials](https://www.swtestacademy.com/selenium-webdriver-csharp-nunit/) on the web that I encourage you to review. This article is going to cover some basic setup steps, a simple .NET Core 2.1 code sample, and promote additional resources such as an intermediate session you may attend at Dog Food Con 2019 to learn how to incorporate your Selenium tests into your DevOps pipeline.

# Browser settings

1. Assumptions: Chrome/Firefox (64-bit)/IE11/Edge (Win10 or higher).
2. Most of these settings have to be done for IE11 as the modern browsers do this by default or the alternative usually still works.
   1. Always open pop-ups in a new tab.
   2. Turn off pop-up blockers.
   3. IE11: Enable Protected Mode for all security zones.
   4. Disable save password prompts.
   5. When prompted to AutoComplete, click "No".
   6. Set zoom to 100%.
3. Restart the browsers

# Windows settings

1. [Disable the logon screen saver](https://support.microsoft.com/en-us/help/185348/how-to-change-the-logon-screen-saver-in-windows) so while you're sitting back watching your automated tests run the screen saver does not kick on ruining your test.
2. Restart the compute

# Setup the WebDrivers

1. IE 11
   1. <https://www.seleniumhq.org/download/>
   2. Under "The Internet Explorer Driver Server" section > click "32 bit Windows IE"
      1. 64 bit should also work, but some consultants I worked with recommended the 32 bit over 64 bit as of 12/2018
   3. Extract "IEDriverServer.exe" from the zip to c:\Selenium.WebDrivers
2. Microsoft Edge (EdgeHtml)
   1. <https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/>
   2. Edge version 18 or great, then run the following in command prompt as an administrator
      1. DISM.exe /Online /Add-Capability /CapabilityName:Microsoft.WebDriver~~~~0.0.1.0
   3. Edge version less than 18, then do the following
      1. Under "Downloads" > Microsoft Edge (EdgeHtml) > click the top Release #####
      2. Save "MicrosoftWebDriver.exe" to c:\Selenium.WebDrivers
3. Microsoft Edge (Chromium)
   1. Since this version is in Preview I did not download and test but here are the steps
   2. <https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/>
   3. Under "Downloads" > Microsoft Edge (Chromium) > for the top Release ##### click x64
   4. Extract "msedgedriver.exe " from the zip to c:\Selenium.WebDrivers
4. Chrome
   1. <https://sites.google.com/a/chromium.org/chromedriver/>
   2. Under "All versions available in Downloads" next to the "Latest stable release" click the ChromeDriver #.## link > Click "chromedriver\_win32.zip"
   3. Extract "chromedriver.exe" from the zip to c:\Selenium.WebDrivers
5. Firefox
   1. <https://github.com/mozilla/geckodriver/releases>
   2. Under the latest release v#.##.# under "Assets" click the geckodriver-\*-win64.zip
   3. Extract "geckodriver.exe" from the zip to c:\Selenium.WebDrivers

# Create the Application

The full source code is in <https://github.com/penblade/Tips/tree/master/Tips.Selenium>.

1. Create a new project > MSTest Test Project (.NET Core)
2. Install the following NuGet packages
   1. DotNetSeleniumExtras.WaitHelpers by SeleniumExtras.WaitHelpers (v3.11.0)
      1. Used for StalenessOf checks
   2. Selenium.Support by Selenium Committers (v3.141.0)
   3. Selenium.WebDriver by Selenium Committers (v3.141.0)

# BrowserType



# WebDriverFactory

Create a factory to get the correct browser web driver.



# WebDriverExtensions

I toiled over deciding if I wanted to create class objects vs. extension methods off the IWebDriver. After going back and forth multiple times I settled on the [extension methods](https://www.danylkoweb.com/Blog/developer-tips-c-extension-methods-on-interfaces-H8). After some further research, I decided to follow the [naming convention](https://stackoverflow.com/questions/2700675/c-sharp-naming-convention-for-extension-methods-for-interface) of not including the "I" in front of the extensions class. I decided to keep the code in the OpenQA.Selenium namespace except for the actual test class, so developers would not have to add another using path.







# WebDriverTest

Now that the extensions have been setup, let's add our test. We'll open a web page, click a link, wait until page load, and then scroll to an element.

For the test, I want to loop through the list of browsers as part of the same step to verify I have setup the environment correctly. Once you start looking into adding tests through the pipeline you'll want to look into using a test runner and specify the browser to test at run time via a configuration file.



# Resources

1. The full source code is in <https://github.com/penblade/Tips/tree/master/Tips.Selenium>
2. [Introduction To Selenium Webdriver With C# In Visual Studio 2015](https://www.c-sharpcorner.com/UploadFile/093731/introduction-to-selenium-webdriver-with-C-Sharp-in-visual-studio/)
3. [Migrating A Selenium Project From .NET Framework To .NET Core](https://medium.com/maestral-solutions/migrating-selenium-project-from-net-framework-to-net-core-32a56589fe7c)
4. [Most Complete Selenium WebDriver C# Cheat Sheet](https://www.automatetheplanet.com/selenium-webdriver-csharp-cheat-sheet/)
5. [DogFoodCon 2018 – Addition by Abstration: A conversation about abstraction and dynamic programming – by Jerren Every](https://www.danylkoweb.com/Blog/review-dog-food-con-2018-NZ#addbyabstract)
   1. Jerren discussed standard refactoring techniques with Test-Driven Development along with concepts like Page Objects that abstract away logic from data.
6. Here are some additional [Notes](https://github.com/penblade/Tips/blob/master/Tips.Selenium/docs/Notes.docx) on a couple issues I dealt with on setup with potential solutions
   1. My Win10 Chrome kept displaying a Windows Defender message
   2. You're supposed to add your Selenium driver path to the environment variable %PATH%, but I hit the max character limit

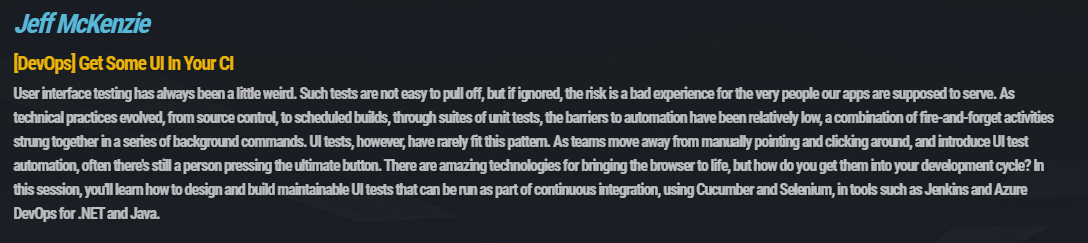
# DogFoodCon 2019

DogFoodCon is my favorite conference to attend each year which is why I was ecstatic when I was approached to become a volunteer tech blogger. I've written a few articles covering the sessions I've attended so far at [DogFoodCon 2017](https://www.danylkoweb.com/Blog/review-dog-food-conference-2017-L4) and [DogFoodCon 2018](https://www.danylkoweb.com/Blog/review-dog-food-con-2018-NZ).

Unlike those articles that reviewed a session I've attended, I'll be writing a few articles like this one that will provide some context (when possible) into one of the sessions with the goal to spur some interest and lead those of you with interest into attending the session. Let us know in the comments your thoughts on this type of article. Thanks!

If you followed the source code above or downloaded and ran it yourself, you'll see that the Selenium code brought up the following session.

[DogFoodCon](https://dogfoodcon.com) > [Sessions](https://dogfoodcon.com/sessions/) > [Jeff McKenzie](https://dogfoodcon.com/speakers#sz-speaker-b912632d-9770-4a0c-a028-9ec5c9a2e023) – [[DevOps] Get Some UI In Your CI](https://dogfoodcon.com/sessions/#sz-session-126695)



I had a delightful interview with Jeff McKenzie where we discussed the highlights of his session. While there may be a short intro to Selenium, this is an intermediate level session so there is assumed basic knowledge of how to create a Selenium test in Java or C#. We compared and contrasted techniques we've both implemented in our pipelines. Jeff will be providing an overview of the following workflows:

1. Java > Gherkins > Selenium > Jenkins
   1. 2 years of experience on various projects
2. C# .NET > SpecFlow > Selenium > Azure DevOps
   1. 5 years of experience on various projects

My team currently works with the workflow: C# .NET > MSTest (Page Object Model) > Selenium > Azure DevOps. I found a lot of common ground and can't wait to compare his process fully with ours.

Jeff McKenzie has been working with Azure DevOps in its various forms since its inception and over the last couple years has translated that knowledge and experience to the Jenkins workflow as well. Testing is one of his passions as evidenced by his dedication to following [Test-Driven Development (TDD)](https://en.wikipedia.org/wiki/Test-driven_development) (15 years) and [Behavior-Driven Development (BDD)](https://en.wikipedia.org/wiki/Behavior-driven_development) (7 years).

By attending this session, you'll learn tried and true patterns to develop the mentioned workflows for testing UI in your DevOps processes.

# Conclusion

I've provided the basics required to setup Selenium including the environment for Win10 and browsers. The sample code demonstrates how to implement common calls to handle waits, find elements, and wait until page load as extension methods off the IWebDriver to enhance and stream line the process. Next steps would be to learn about the [Page Object models](https://www.swtestacademy.com/page-object-model-c/). I also plugged a DogFoodCon 2019 session that will teach you how to add your Selenium tests into your DevOps pipeline.

*Did I cover Selenium basics? Did I miss any? Did the setup instructions work for you? Do you like the extension methods implementation or do you prefer creating class objects instead?*

*Did you like the plug for the DogFoodCon session? Would the conference/session get as much attention if it was its own article? Do you like the format of this article? What can I do to improve these articles?*