C# Test Setup Example

This script provides an example of how you might configure your own automated tests to run in the Sauce Labs browser cloud. The sample test uses environment variables for authentication, assigns a tag and build number for test result management, and reports Pass/Fail status to the Sauce Labs dashboard.

What You'll Need

- You should have a Sauce Labs account
- You should have a Selenium environment with the C# bindings set up on a local machine where you'll launch your tests. If you need help with setting this up, check out the ReadMe in the C# sample script repository on GitHub.
- You should review the Best Practices for Running Tests

Setup Example Script

The sample test opens the browser, navigates to the saucelabs.demo web app, and then closes the browser.

To run this script against your own app using your Sauce Labs credentials:

1. Set your authentication credentials as environment variables to connect the test to your Sauce Labs account.
2. Enter the URL for the web app you want to test in the place of `saucedemo.com`.

Once you've been able to run the test against your web app, check out the Platform Configurator to see more of the desired capabilities you can use when testing with Sauce.

Clone the following script from saucelabs-training on GitHub:

SauceExamples/Web.Tests/OnboardingTests/InstantSauceTest4.cs (source from Sauce Labs Training C#)

```
using NUnit.Framework;
using NUnit.Framework.Interfaces;
using OpenQA.Selenium;
using OpenQA.Selenium.Remote;
using System;
using System.Collections.Generic;
using System.Threading;

namespace Web.Tests.OnboardingTests
{
    /*
     * These scripts are simply for demonstration purposes.
     * They should not be used as an example of good practices for how to do test automation.
     */
    [TestFixture]
    [Category("InstantSauceTest"), Category("NUnit"), Category("Instant")]
    [Parallelizable]
    public class InstantSauceTest4
    {
        private IWebDriver _driver;
        private readonly string _sauceUserName =
            Environment.GetEnvironmentVariable("SAUCE_USERNAME", EnvironmentVariableTarget.User);
        private readonly string _sauceAccessKey =
            Environment.GetEnvironmentVariable("SAUCE_ACCESS_KEY", EnvironmentVariableTarget.User);
        private IJavaScriptExecutor _javascriptExecutor;

        [Test]
        public void BestPractices()
        {
            /*
             * Commenting is one of the most powerful ways to debug your failed tests.
             * Using the sauce:context command below will allow you to place
             * comments inside of Sauce Labs logs that you can read and analyze.
             * Comment your important methods and your automation will drastically improve
             */
            _javascriptExecutor.ExecuteScript("sauce:context=Open SauceDemo.com");
            _driver.Navigate().GoToUrl("https://www.saucedemo.com");

            _javascriptExecutor.ExecuteScript("sauce:context=Sleep for 10000ms");
            Thread.Sleep(10000);
            Assert.IsTrue(true);
        }

        [SetUp]
        public void ExecuteBeforeEveryTest()
        {
            DesiredCapabilities caps = new DesiredCapabilities();
            caps.SetCapability("browserName", "Safari");
            caps.SetCapability("platform", "macOS 10.13");
            caps.SetCapability("version", "11.1");
            caps.SetCapability("username", _sauceUserName);
            caps.SetCapability("accessKey", _sauceAccessKey);
            caps.SetCapability("name", TestContext.CurrentContext.Test.Name);
            //Tags are an excellent way to control and filter your test automation
            //in Sauce Analytics. Get a better view into your test automation.
            var tags = new List<string> { "demoTest", "sauceDemo" };
            caps.SetCapability("tags", tags);
            /*
             * One of the most important things that you can do to get started
             * is to set timeout capabilities for Sauce based on your organizations needs
             */
            //How long is the whole test allowed to run?
            caps.SetCapability("maxDuration", 3600);
            //Selenium crash might hang a command, this is the max time allowed to wait for a Selenium command
            //600sec is a great start for majority of engineers
            caps.SetCapability("commandTimeout", 600);
            //How long can the browser wait before a new command?
            //1000sec is max and is a good timeout duration for most engineers
            caps.SetCapability("idleTimeout", 1000);
            /*
             * Setting a build name is one of the most fundamental pieces of running
             * successful test automation. Builds will gather all of your tests into a single
             * 'test suite' that you can analyze for results.
             * You should always group your tests into builds.
             */
            caps.SetCapability("build", "SauceDemo");

            _driver = new RemoteWebDriver(new Uri("http://ondemand.saucelabs.com:80/wd/hub"),
                caps, TimeSpan.FromSeconds(600));

            _javascriptExecutor = ((IJavaScriptExecutor)_driver);
        }


        [TearDown]
        public void CleanUpAfterEveryTestMethod()
        {
            var passed = TestContext.CurrentContext.Result.Outcome.Status == TestStatus.Passed;
            _javascriptExecutor.ExecuteScript("sauce:job-result=" + (passed ? "passed" : "failed"));
            _driver?.Quit();
        }
    }
}
```

Example Only

The code in this topic is presented as an example only, since your tests and testing environments may require specialized scripting. This information should be taken only as an illustration of how you could set up your tests with Sauce Labs, and is not directly supported by Sauce.