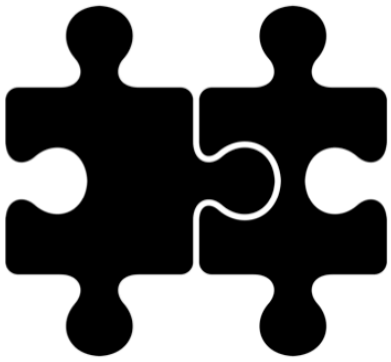
SELENIUM CHEAT SHEETS

C# Edition



by Dave Haeffner

Version 1.1.0

Table of Contents

1. Local Configuration
2. Cloud Configuration
3. Common Actions
4. Locators
5. Exception Handling
6. Waiting
7. Cookies
8. Dropdowns
9. File Transfers
10. Frames
11. JavaScript Alerts
12. Keyboard Keys
13. Multiple Windows
14. Screenshots on Failure

Chapter 1

Local Configuration

Chrome

1. Download the latest ChromeDriver binary from here
2. Add it to your system path (or tell Selenium where to find it)
3. Create an instance of Chrome

```
using OpenQA.Selenium.Chrome;
Driver = new ChromeDriver(VendorDirectory);
```

For more info see:

- Selenium Wiki page for ChromeDriver
- Google's ChromeDriver documentation

Firefox

1. Download the latest geckodriver binary from here
2. Add it to your path, or tell Selenium where to find it with a `FirefoxDriverService` object
3. Create an instance of Firefox, passing in the `FirefoxDriverService` object

```
using OpenQA.Selenium.Firefox;
var VendorDirectory = System.IO.Directory.GetParent(
                        System.AppDomain.CurrentDomain.BaseDirectory).
                        Parent.Parent.FullName
                        + @"\Vendor";
var Service = FirefoxDriverService.CreateDefaultService(VendorDirectory);
Driver = new FirefoxDriver(Service);
```

To use the legacy FirefoxDriver:

1. Specify the path to the geckodriver binary file with a `FirefoxDriverService` object (to avoid a lookup error)
2. Set `UseLegacyImplementation` to `true` in a `FirefoxOptions` object
3. Create an instance of Firefox, passing in both the `FirefoxDriverService` and `FirefoxOptions` objects (along with a timeout for browser instantiation)

```
using OpenQA.Selenium.Firefox;
var VendorDirectory = System.IO.Directory.GetParent(
                        System.AppDomain.CurrentDomain.BaseDirectory).
                        Parent.Parent.FullName
                        + @"\Vendor";
var Service = FirefoxDriverService.CreateDefaultService(VendorDirectory);
FirefoxOptions Options = new FirefoxOptions() { UseLegacyImplementation = true };
Driver = new FirefoxDriver(Service, Options, System.TimeSpan.FromSeconds(30));
```

For more info:

- the Selenium wiki page for FirefoxDriver
- the geckodriver documentation from Mozilla

Internet Explorer

Only available on Microsoft Windows.

1. Download the latest IEDriverServer from here
2. Tell Selenium where the IEDriverServer file is location
3. Enable the Privacy Mode in Internet Explorer's security settings
4. Create an instance of Internet Explorer

```
using OpenQA.Selenium.IE;
Driver = new InternetExplorerDriver(VendorDirectory);
```

For more info see:

- Selenium Wiki page for InternetExplorerDriver

Safari

Available out of the box as of version 2.21 of Selenium.

```
using OpenQA.Selenium.Safari;
Driver = new SafariDriver();
```

For more info see:

- Selenium wiki page for SafariDriver
- A tip on how to use SafariDriver if you run into issues

Chapter 2

Cloud Configuration

Sauce Labs

Initial Setup

1. Create an App.config file with the values you want for a specific browser/OS combination
2. Grab the values from App.config and store them in field variables
3. Specify the browser and operating system you want through Selenium's `DesiredCapabilities`
4. Create an instance of `RemoteWebDriver` using Sauce Labs' end-point -- providing your credentials and `DesiredCapabilities`
5. Store the instance in a field variable for use in your tests

```
<!-- filename: App.config -->
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="Host" value="saucelabs"/>
    <add key="BrowserName" value="Internet Explorer"/>
    <add key="BrowserVersion" value="11.0"/>
    <add key="Platform" value="Windows 7"/>
    <add key="ApplicationBaseUrl" value ="https://the-internet.herokuapp.com"/>
  </appSettings>
</configuration>
```

```
using OpenQA.Selenium.Remote;

public static string ApplicationBaseUrl;
private static string VendorDirectory;
private static string BrowserName;
private static string Host;
private static string BrowserVersion;
private static string Platform;

var configReader    = new AppSettingsReader();
Host                = (string)configReader.GetValue("Host", typeof(string));
BrowserName         = (string)configReader.GetValue("BrowserName", typeof(string));
BrowserVersion      = (string)configReader.GetValue("BrowserVersion", typeof(string));
Platform            = (string)configReader.GetValue("Platform", typeof(string));
ApplicationBaseUrl  = (string)configReader.GetValue("ApplicationBaseUrl", typeof(string
));

DesiredCapabilities caps = new DesiredCapabilities();
caps.SetCapability(CapabilityType.BrowserName, BrowserName);
caps.SetCapability(CapabilityType.Version, BrowserVersion);
caps.SetCapability(CapabilityType.Platform, Platform);
caps.SetCapability("username", System.Environment.GetEnvironmentVariable(
"SAUCE_USERNAME"));
caps.SetCapability("accessKey", System.Environment.GetEnvironmentVariable(
"SAUCE_ACCESS_KEY"));
Driver = new RemoteWebDriver(new Uri("http://ondemand.saucelabs.com:80/wd/hub"), caps);
```

For more info see:

- Sauce Labs Available Platforms page
- Sauce Labs Automated Test Configurator

Setting the Test Name

1. Pull the test name out of NUnit's `TestContext`
2. Pass the name to Sauce Labs using the `"name"` capability in `DesiredCapabilities`

```
caps.SetCapability("name", TestContext.CurrentContext.Test.Name);
```

Setting the Job Status

1. Check the test result after the test completes
2. Use the JavaScript executor to pass the result onto Sauce Labs
3. BONUS POINTS: Output the Sauce Labs job URL to the console

```
bool testPassed = TestContext.CurrentContext.Result.Outcome.Status.Equals(TestStatus.
Passed);
((IJavaScriptExecutor)Driver).ExecuteScript("sauce:job-result=" + (testPassed ?
"passed" : "failed"));
Console.WriteLine("https://saucelabs.com/beta/tests/" + ((RemoteWebDriver)Driver).
SessionId);
```

Chapter 3

Common Actions

Visit a page

```
Driver.Navigate().GoToUrl("http://the-internet.herokuapp.com");
```

Find an element

Works using locators, which are covered in the next section.

```
// find just one, the first one Selenium finds
Driver.FindElement(locator);

// find all instances of the element on the page
Driver.FindElements(locator);
// returns a collection
```

Work with a found element

```
// chain actions together
Driver.FindElement(locator).Click();

// store the element
IWebElement Element = Driver.FindElement(locator);
Element.click();
```

Perform an action

```
Element.Click();                  // clicks an element
Element.Submit();                 // submits a form
Element.Clear();                  // clears an input field of it's text
Element.SendKeys("input text");   // types text into an input field
```

Ask a question

Each of these returns a Boolean.

```
Element.Displayed;    // is it visible to the human eye?
Element.Enabled;      // can it be selected?
Element.Selected;     // is it selected?
```

Retrieve information

Each of these returns a String.

```
// by attribute name
Element.GetAttribute("href");

// directly from an element
Element.Text;
```

For more info see:

- Selenium IWebElement API Documentation

Chapter 4
Locators

Guiding principles

Good Locators are:

- unique
- descriptive
- unlikely to change

Be sure to:

1. Start with ID and Class
2. Use CSS selectors (or XPath) when you need to traverse
3. Talk with a developer on your team when the app is hard to automate
    1. tell them what you're trying to automate
    2. work with them to get more semantic markup added to the page

ID

```
Driver.FindElement(By.Id("username"));
```

Class

```
driver.findElement(By.ClassName("dues"));
```

CSS Selectors

```
Driver.FindElement(By.CssSelector("#username"));
Driver.FindElement(By.CssSelector(".dues"));
```

Approach	Locator	Description
ID	`#example`	`#` denotes an ID
Class	`.example`	`.` denotes a Class
Classes	`.flash.success`	use `.` in front of each class for multiple
Direct child	`div > a`	finds the element in the next child
Child/subschild	`div a`	finds the element in a child or child's child
Next sibling	`input.username + input`	finds the next adjacent element
Attribute values	`form input[name='username']`	a great alternative to id and class matches
Attribute values	`input[name='continue'][type='button']`	can chain multiple attribute filters together
Location	`li:nth-child(4)`	finds the 4th element only if it is an li
Location	`li:nth-of-type(4)`	finds the 4th li in a list
Location	`*:nth-child(4)`	finds the 4th element regardless of type
Sub-string	`a[id^='beginning_']`	finds a match that starts with (prefix)
Sub-string	`a[id$='_end']`	finds a match that ends with (suffix)
Sub-string	`a[id*='gooey_center']`	finds a match that contains (substring)
Inner text	`a:contains('Log Out')`	an alternative to substring matching

NOTE: Older browser (e.g., Internet Explorer 8) don't support CSS Pseudo-classes, so some of these locator approaches won't work on them (e.g., Location matches and Inner text matches).

For more info see:

- CSS vs. XPath benchmarks
- CSS & XPath Examples by Sauce Labs
- CSS Selector Game
- The difference between nth-child and nth-of-type
- w3schools CSS Selectors Reference
- w3schools XPath Syntax Reference
- How To Verify Your Locators

Chapter 5
Exception Handling

1. Try the action you want
2. Catch the relevant exception and return `false` instead

```
try {
  return Find(locator).Displayed;
} catch(OpenQA.Selenium.NoSuchElementException) {
  return false;
}
```

For more info see:

- Selenium WebDriverException API Documentation

Chapter 6
Waiting

Implicit Wait

- Only needs to be configured once
- Tells Selenium to wait for a specified amount of time before raising an exception (typically a `NoSuchElementException`)
- Less flexible than explicit waits

```
Driver.Manage().Timeouts().ImplicitlyWait(TimeSpan.FromSeconds(10));
```

Explicit Waits

- Recommended way to wait in your tests
- Specify an amount of time and an action
- Selenium will try the action repeatedly until either:
  - the action can be accomplished, or
  - the amount of time has been reached (and throw a `TimeoutException`)

```
WebDriverWait Wait = new WebDriverWait(Driver, System.TimeSpan.FromSeconds(10));
Wait.Until(ExpectedConditions.ElementIsVisible(locator));
return true;
```

For more info see:

- The case against using Implicit and Explicit Waits together
- Explicit vs. Implicit Waits

Chapter 7

Cookies

Retrieve a cookie

```
Driver.Manage().Cookies.GetCookieNamed("cookieName");
```

Add a cookie

```
Driver.Manage().Cookies.AddCookie(new Cookie("cookieName", "cookieValue"));
```

Delete a cookie

```
Driver.Manage().Cookies.DeleteCookieNamed("cookieName");
```

Delete all cookies

```
// Only deletes cookies for the domain Selenium visits
Driver.Manage().Cookies.DeleteAllCookies();
```

For more info see:

- Selenium ICookieJar API Documentation

Chapter 8

Dropdowns

1. Find the dropdown list
2. Select the item you want from the list by either its visible text or value number

```
Driver.Navigate().GoToUrl("http://the-internet.herokuapp.com/dropdown");
SelectElement Dropdown = new SelectElement(Driver.FindElement(By.Id("dropdown")));
Dropdown.SelectByText("Option 1");
```

Chapter 9

File Transfers

Upload

1. Find the form input field for uploading the file
2. Use `SendKeys` to input the full path of the file you want to upload
3. Submit the form

```
string File = "SomeFile.txt";
string FilePath = @"C:\Temp\" + File;
Driver.Navigate().GoToUrl("http://the-internet.herokuapp.com/upload");
Driver.FindElement(By.Id("file-upload")).SendKeys(FilePath);
Driver.FindElement(By.Id("file-submit")).Click();
```

Download with Selenium

1. Create a uniquely named temporary folder to store downloaded files
2. Configure a browser profile to download files without prompting
3. Create a new instance of Selenium and pass in the profile
4. Perform checks on the file after downloading to verify it is the correct type and size
5. Delete the file and folder when done

```
FolderPath = @"C:\Temp\" + System.Guid.NewGuid().ToString();
Directory.CreateDirectory(FolderPath);

FirefoxProfile Profile = new FirefoxProfile();
profile.SetPreference("browser.download.dir", FolderPath);
profile.SetPreference("browser.download.folderList", 2);
profile.SetPreference("browser.helperApps.neverAsk.saveToDisk",
                      "image/jpeg, application/pdf, application/octet-stream");
profile.SetPreference("pdfjs.disabled", true);
Driver = new FirefoxDriver(Profile);
```

Download without Selenium

1. Get the URL of the file you want to download
2. Perform a header (a.k.a. HEAD) request on the URL with an HTTP library
3. Check the content type and content length of the response to make sure the file is what you expected

```
Driver.Navigate().GoToUrl("http://the-internet.herokuapp.com/download");
string FileURL = Driver.FindElement(By.CssSelector(".example a")).GetAttribute("href");
var Request = (HttpWebRequest)WebRequest.Create(FileURL);
Request.Method = "HEAD";
WebResponse Response = Request.GetResponse();
Assert.That(Response.ContentType.Equals("application/octet-stream"));
Assert.Greater(Response.ContentLength, 0);
```

Chapter 10
Frames

In order to access elements in frames, you need to switch to them.

If the element you want is nested inside of 2 or more frames, you first need to switch to the parent frame, then the child frame.

```
Driver.SwitchTo().Frame("frame-top");
Driver.SwitchTo().Frame("frame-middle");
```

You can quickly switch back to the top of the page with a single command, rather than traversing backwards.

```
Driver.SwitchTo().DefaultContent();
```

Chapter 11

JavaScript Alerts

1. Switch to the alert window
2. Accept or dismiss the alert

```
IAlert Popup = Driver.SwitchTo().Alert();
Popup.Accept();
// or Popup.Dismiss();
```

For more info see:

- Selenium IAlert API Documentation

Chapter 12
Keyboard Keys

Option 1:

1. Find a target element
2. Send keys to that element

```
Driver.FindElement(By.Id("content")).SendKeys(Keys.Space);
```

Option 2:

1. Use the Selenium Action Builder to send keys to the element currently in focus

```
Actions Builder = new Actions(Driver);
Builder.SendKeys(Keys.Left).Build().Perform();
```

Chapter 13
Multiple Windows

Some browsers list window handles in the order opened, others alphabetically. Here's a ubiquitous approach to switching between windows:

1. Find and store the initial window handle
2. Trigger the new window to appear
3. Find all window handles and iterate through them, looking for the new window handle
4. Store the new window handle
5. Switch freely between the initial and new windows

```
Driver.Navigate().GoToUrl("http://the-internet.herokuapp.com/windows");
string FirstWindow = Driver.CurrentWindowHandle;
string SecondWindow = "";

Driver.FindElement(By.CssSelector(".example a")).Click();

var Windows = Driver.WindowHandles;
foreach(var Window in Windows)
{
    if (Window != FirstWindow)
        SecondWindow = Window;
}

Driver.SwitchTo().Window(FirstWindow);
Assert.That(Driver.Title != "New Window");

Driver.SwitchTo().Window(SecondWindow);
Assert.That(Driver.Title.Equals("New Window"));
```

Chapter 14

Screenshots on Failure

1. In the test teardown, check to see if the test failed
2. If it has, then capture a screenshot with Selenium, storing it to local disk

```
private void TakeScreenshot()
{
    string SaveLocation = @"C:\Temp\" +
                          "failshot_" +
                          TestContext.CurrentContext.Test.FullName +
                          ".png";
    ITakesScreenshot ScreenshotDriver = (ITakesScreenshot) Driver;
    ScreenshotDriver.GetScreenshot().SaveAsFile(SaveLocation, ImageFormat.Png);
}

[TearDown]
public void TearDown()
{
    if (TestContext.CurrentContext.Result.Outcome.Status.Equals(TestStatus.Failed))
        TakeScreenshot();

    Driver.Quit();
}
```