Contents

[C# and Selenium Code Snippets 1](#_Toc20228614)

[GetVisibleElementByVisibleText 1](#_Toc20228615)

[GetTableRow 2](#_Toc20228616)

[GetColumn 2](#_Toc20228617)

[GetFilePathWithEnvironment 3](#_Toc20228618)

[RemoveSpacesAndNewLines 4](#_Toc20228619)

[WaitForWithCondition 5](#_Toc20228620)

[WaitForElements 6](#_Toc20228621)

[ClickOnSearchResultsForSelect2Dropdown 7](#_Toc20228622)

[DateTime.Parse and Teneary Operator 7](#_Toc20228623)

[Linq Query 9](#_Toc20228624)

[Postman Code Snippets 9](#_Toc20228625)

[//To write to the console 9](#_Toc20228626)

[//Check Status Code 9](#_Toc20228627)

[//Check TpiRegistrationDetailId has a value 10](#_Toc20228628)

[//Set TpiRegId environment var for use by later script(s) 10](#_Toc20228629)

[//Check NumberOfRowsAffected = 1 10](#_Toc20228630)

[//Check TpiRegistrationDetailId has a value > 0 11](#_Toc20228631)

[//Get a value from the REQUEST and set the environment variable to that value 11](#_Toc20228632)

[//Request with Environment Variable in URL 11](#_Toc20228633)

[// Check DateAccepted populated with a date 11](#_Toc20228634)

[//Check response item INCLUDES a value 11](#_Toc20228635)

# C# and Selenium Code Snippets

### GetVisibleElementByVisibleText

**Xpath to find element by Text:**

//\*[contains(text(), 'blah')]

And can be parameterised and put in a method like this:

public IWebElement GetVisibleElementByVisibleText(string searchText, bool retry = true, int? waitSeconds= null)

{

int seconds = waitSeconds ?? WebDriverTimeout;

WebDriverWait wait = new WebDriverWait(Driver, TimeSpan.FromSeconds(seconds));

try

{

return wait.Until(ExpectedConditions.ElementIsVisible(By.XPath(".//\*[contains(text(),'" + searchText + "')]")));

}

catch (WebDriverTimeoutException e)

{

if (retry)

{

try

{

return wait.Until(ExpectedConditions.ElementIsVisible(By.XPath(".//\*[contains(text(),'" + searchText + "')]")));

}

catch (Exception)

{

//failed again, throw original error

throw new WebDriverTimeoutException($"Timeout searching for element containing visibleText: {searchText}. {e.Message}", e);

}

}

else

{

throw;

}

}

}

### GetTableRow

protected IWebElement GetTableRow(By by, int nRow)

{

var tableElement = Driver.FindElement(by);

Assert.IsNotNull(tableElement, $"Didn't find the table given by: {by}");

Assert.IsTrue(tableElement.TagName == "table", "Element is not a table!");

var webElements = tableElement.FindElements(By.TagName("tr"));

return !webElements.Any() ? null : webElements[nRow];

}

### GetColumn

protected IWebElement GetColumn(IWebElement webElementRow, int nColumn)

{

Assert.IsTrue(webElementRow.TagName == "tr", "Element is not a row!");

return webElementRow.FindElements(By.TagName("td"))[nColumn];

}

### GetFilePathWithEnvironment

What Kryz wrote:

var dir = new DirectoryInfo(AppContext.BaseDirectory);

var solutionPath = dir.Parent.Parent.Parent.FullName;

var finalPath = [$@"{solutionPath}\packages\Selenium.WebDriver.ChromeDriver.2.37.0\driver\win32](mailto:$@%22%7bsolutionPath%7d\packages\Selenium.WebDriver.ChromeDriver.2.37.0\driver\win32)";

NOTE: before being able to access the file in the folder whose path you are trying to get with the following code, you need to right click on the file and select Properites. Then set

Copy to Output Directory = Copy always

Can be used like this:

public string GetFilePath()

{

var baseDir = new DirectoryInfo(AppContext.BaseDirectory);

var solutionPath = baseDir.Parent.Parent.Parent.FullName;

var finalPath = $@"{solutionPath}\Portal.Web.AcceptanceTests\TestData\";

return finalPath;

}

And if you wanted to include the environment:

public string GetFilePathWithEnvironment()

{

var dir = new DirectoryInfo(AppContext.BaseDirectory);

var solutionPath = dir.Parent.Parent.Parent.FullName;

var finalPath = $@"{solutionPath}\Portal.Web.AcceptanceTests\TestData\ {new RunSettings().EnvironmentKey}\";

return finalPath;

}

In order to find out where your base directory corresponds to, run the code in debug mode and see what value is assigned to the variable. You can now see how many .Parent you need. Bear in mind that you want to get to the directory ABOVE Portal.Web.AcceptanceTests (for this example) so that you can come back down the leg that starts with Portal.Web.AcceptanceTests

Could parameterise it like this:

public static string GetFilePath(string targetFolder)

{

var dir = new DirectoryInfo(AppContext.BaseDirectory);

var solutionPath = dir.Parent.Parent.Parent.FullName;

var finalPath = $@"{solutionPath}\Portal.Web.AcceptanceTests\{targetFolder}\";

return finalPath;

}

Here is an alternative approach:

public static string AssemblyDirectory

{

get

{

string codeBase = Assembly.GetExecutingAssembly().CodeBase;

UriBuilder uri = new UriBuilder(codeBase);

string path = Uri.UnescapeDataString(uri.Path);

return Path.GetDirectoryName(path);

}

}

var environmentTestDataDirectory = Path.Combine(Path.Combine(AssemblyDirectory, targetFolder, new RunSettings().EnvironmentKey));

AssemblyDirectory is a property(?) and the get (function?) will return the value determined by the code in the brackets when AssemblyDirectory is referenced.

### RemoveSpacesAndNewLines

/// <summary>

/// Removes text and spaces from the given text to make string comparison easier.

/// </summary>

/// <param name="textToClean">text you want spaces and new lines removed from</param>

/// <returns>string</returns>

public string RemoveSpacesAndNewLines(string textToClean)

{

var cleanedText = Regex.Replace(textToClean, @"\s", "");

return cleanedText = Regex.Replace(cleanedText, @"\n", "");

}

Alternatively:

/// <summary>

/// This method extends the String class.

/// The string object that uses it (this) will both call the method and implicitly pass itself in as the parameter

/// Example: errorText.RemoveSpacesAndNewLines()

/// </summary>

/// <param name="textToClean">the calling string that is to be cleaned</param>

/// <returns>string </returns>

public static string RemoveSpacesAndNewLines(this string textToClean)

{

var cleanedText = Regex.Replace(textToClean, @"\s", "");

return Regex.Replace(cleanedText, @"\n", "");

}

### WaitForWithCondition

/// <summary>

/// Waits for a custom condition to be fulfilled.

/// </summary>

/// <param name="by">Used to locate the element to be waited for</param>

/// <param name="condition">The condition that must be fulfilled, e.g. x => x.Text.Contains("Send")</param>

/// <param name="waitSeconds">Timeout in seconds </param>

/// <param name="retry">Whether to retry, defaults to TRUE</param>

/// <param name="waitForVisibility">Whether to wait for visiblity or just presence, defaults to FALSE</param>

/// <returns>IWebElement</returns>

protected IWebElement WaitForWithCondition(By by, Func<IWebElement, bool> condition, int waitSeconds = 20,

bool retry = true, bool waitForVisibility = false)

{

try

{

var wait = new WebDriverWait(Driver, new TimeSpan(0, 0, waitSeconds));

Func<IWebDriver, IWebElement> waitForElement = x =>

{

var elements = WaitForElements(by, waitSeconds, retry, waitForVisibility);

var element = elements.FirstOrDefault(e => e != null && condition(e) && e.Displayed);

return element;

};

return wait.Until(waitForElement);

}

catch (Exception e)

{

if (retry)

{

try

{

WaitForWithCondition(by, condition, waitSeconds, false); //only retry once

}

catch (Exception)

{

//failed again, throw original error

throw new WebDriverTimeoutException($"Cannot find element via: " + by + " Failing method = " + UtilityMethods.GetCurrentMethod(), e);

}

}

else

{

throw new WebDriverTimeoutException($"Cannot find element via: " + by + " Failing method = " + UtilityMethods.GetCurrentMethod(), e);

}

}

return null;

}

### WaitForElements

/// <summary>

/// Waits for ALL the element specified via the given By to be present/visible

/// Depending on whether or not waitForVisibility = TRUE

/// NOTE: This seems to be getting used incorrectly in places where only 1 element is expected.

/// </summary>

/// <param name="by">Used to locate the element to be waited for</param>

/// <param name="waitSeconds">Timeout</param>

/// <param name="retry">Whether to retry, defaults to TRUE</param>

/// <param name="waitForVisibility">Whether to wait for visibility or just presence, defaults to FALSE</param>

/// <returns>List of web elements</returns>

protected IReadOnlyCollection<IWebElement> WaitForElements(By by, int? waitSeconds = null, bool retry = true, bool waitForVisibility = false)

{

int seconds = waitSeconds ?? WebDriverTimeout;

var wait = new WebDriverWait(Driver, TimeSpan.FromSeconds(seconds));

try

{

if (waitForVisibility)

{

return wait.Until(ExpectedConditions.VisibilityOfAllElementsLocatedBy(by));

}

return wait.Until(ExpectedConditions.PresenceOfAllElementsLocatedBy(by));

}

catch (Exception e)

{

if (retry)

{

try

{

WaitForElements(by, seconds, retry: false); //Call itself, but don't do another retry

}

catch (Exception)

{

//failed again, throw original error

throw new WebDriverTimeoutException($"Cannot find element via: " + by + " Failing method = " + UtilityMethods.GetCurrentMethod(), e);

}

}

else

{

throw new WebDriverTimeoutException($"Cannot find element via: " + by + " Failing method = " + UtilityMethods.GetCurrentMethod(), e);

}

}

return null;

}

## ClickOnSearchResultsForSelect2Dropdown

/// <summary>

/// Selects the results from a Select2 style dropdown.

/// Locator would be something like By.XPath("//\*[@id='select2-selectBasket-results']")

/// </summary>

/// <param name="resultsElementLocator">Locator for the results list</param>

/// <param name="searchResultText">Text of the result you want to select.</param>

/// <param name="retry">Whether or not to retry on failure to find element</param>

public void ClickOnSearchResultsForSelect2Dropdown(By resultsElementLocator, string searchResultText, bool retry = true)

{

try //Sometimes get a stale element error here

{

IWebElement searchResults = WaitForWithCondition(

resultsElementLocator,

x => x.Text.Contains(searchResultText), waitSeconds: 5, retry: true, waitForVisibility: true);

//Just in case it is visible but not clickable

WaitForElementToBeClickable(searchResults, timeoutSeconds: 5);

searchResults.Click();

}

catch (Exception e)

{

if (retry)

{

IWebElement searchResults = WaitForWithCondition(

resultsElementLocator,

x => x.Text.Contains(searchResultText), waitSeconds: 5, retry: true, waitForVisibility: true);

//Just in case it is visible but not clickable

WaitForElementToBeClickable(searchResults);

searchResults.Click();

}

else

{

throw;

}

}

}

## DateTime.Parse and Teneary Operator

Getting date from database and comparing it to value on front end

Complicated version using string.Split() and **ternary operator**

var dateOnDB = GetAccessToDateDateFromDatabase(server, dbInstance, databaseName, emailAddress);

char[] splitchar = { '/'};

string[] dateArray = dateOnDB.Split(splitchar);

//dateOnDB is is mm/dd/yyy format

var compareMM = dateArray[0].Length <2 ? $"0{dateArray[0]}" : dateArray[0];

var compareDD = dateArray[1].Length < 2 ? $"0{dateArray[1]}" : dateArray[1];

var compareYYYY = dateArray[2].Substring(0,4);

var compareDate = $"{compareDD}/{compareMM}/{compareYYYY}";

return \_selectedAccessUntilDate == compareDate;

Simpler version using **DateTime.Parse()**

var dateOnDB = GetAccessToDateDateFromDatabase(server, dbInstance, databaseName, emailAddress);

var dt = DateTime.Parse(dateOnDB);

var dtString = dt.ToString("dd/MM/yyyy");

return \_selectedAccessUntilDate == dtString;

## Linq Query

using System;

using System.Linq;

public class Program

{

public static void Main()

{

// Data source

string[] names = {"Bill", "Steve", "James", "Mohan" };

// LINQ Query

var myLinqQuery = from item in names

where item.Contains('e')

select item;

// Query execution

foreach (var item in myLinqQuery)

Console.Write(item + " ");

}

}

# Postman Code Snippets

To view the Postman Console, select View->Show Postman Console from top menu

### //To write to the console

console.log("jsonData.TpiRegistrationDetailId = "+ jsonData.TpiRegistrationDetailId)

### //Check Status Code

pm.test("Status code is 201", function () {

pm.response.to.have.status(201);

});

### //Check TpiRegistrationDetailId has a value

pm.test("Check TpiRegistrationDetailIdTpiRegistrationDetailId not zero", function(){

var jsonData = pm.response.json();

pm.expect(jsonData.TpiRegistrationDetailId !== 0);

});

### //Set TpiRegId environment var for use by later script(s)

pm.test("If TpiRegistrationDetailId not zero, then set TpiRegId environment var", function ()

{

var jsonData = pm.response.json();

if (jsonData.TpiRegistrationDetailId > 0)

{

pm.environment.set("TpiRegId", jsonData.TpiRegistrationDetailId)

pm.expect(jsonData.TpiRegistrationDetailId).to.eql(pm.environment.get("TpiRegId"));

}

});

### //Check NumberOfRowsAffected = 1

pm.test("Check NumberOfRowsAffected", function ()

{

var jsonData = pm.response.json();

pm.expect(jsonData.NumberOfRowsAffected).to.eql(1);

});

//Notes:

//pm.test function is used to write test specifications inside the Postman test sandbox. So a bit like a [TestFixture] annotatoin in selenium

//pm.expect is like an Assert

### //Check TpiRegistrationDetailId has a value > 0

pm.test("Check TpiRegistrationDetailIdTpiRegistrationDetailId geater than zero", function(){

var jsonData = pm.response.json();

pm.expect(jsonData.TpiRegistrationDetailId).to.be.above(0);

});

### //Get a value from the REQUEST and set the environment variable to that value

var jsonInputData = JSON.parse(request.data);

pm.environment.set("TOU\_UserId", jsonInputData.UserId);

### //Request with Environment Variable in URL

[http://porapp01qa:81/api/termsofuse/isaccepted/{{TOU\_UserId}}](http://porapp01qa:81/api/termsofuse/isaccepted/%7b%7bTOU_UserId%7d%7d)

## // Check DateAccepted populated with a date

pm.test("Check DateAccepted populated with a date", () => {

var jsonData = pm.response.json();

aPastDate = new Date('2019-03-12').getTime();

isoFormatDate = new Date(Date.parse(jsonData.DateAccepted)).getTime(); // Converting to integer from date format

pm.expect(isoFormatDate).to.be.above(aPastDate);

});

## //Check response item INCLUDES a value

pm.test("Check Second FileName", function () {

var jsonData = pm.response.json();

pm.expect(jsonData[1].FileName).to.include("CD\_BASKET\_M&C\_AGE\_UK");

});