Lab: Selenium WebDriver

1. Search for "Quality Assurance" in Wikipedia

- Select Create a new Console App (.NET Core) project in Visual Studio.

- Install Necessary NuGet Packages:
  - Selenium WebDriver
  - ChromeDriver
- Import necessary namespaces.
- Inside the Main method, create a new instance of ChromeDriver.
- Navigate to the Wikipedia homepage.
- Print the title of the main page to the console.
- Find the search input element by its ID.
- Click on the search box to focus it.
- Type "Quality Assurance" into the search box and press Enter.
- Print the title of the QA search results page to the console.
- Close the browser and end the session.

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

// Create a new instance of ChromeDriver
var driver = new ChromeDriver();

// Navigate to the Wikipedia homepage

driver.Navigate().GoToUrl("https://wikipedia.org");

// Print the title of the main page to the console
Console.WriteLine("Main page title: " + driver.Title);

// Find the search input element by its ID
var searchBox = driver.FindElement(By.Id("searchInput"));

// Click on the search box to focus it
searchBox.Click();

// Type "QA" into the search box and press Enter
searchBox.SendKeys("Quality Assurance" + Keys.Enter);

// Print the title of the QA search results page to the console
Console.WriteLine("Quality A page title: " + driver.Title);

// Close the browser and end the session
driver.Dispose();
```

2. First Test

- Create a new NUnit Test Project (.NET Core) project.
- Install Necessary NuGet Packages.
- Import necessary namespaces.
- Define the test class.
- Inside the class, declare a private IWebDriver variable named driver.
- Create a Setup method to initialize the ChromeDriver before each test.

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;

namespace FirstTest
{
    0 references
    public class FirstTest
    {
        private IWebDriver driver;

        [SetUp]
        0 references
        public void Setup()
        {
            driver = new ChromeDriver();
        }
```

- Write the test method.
- In the method, navigate to https://nakov.com.
- Retrieve the page title and verify it contains "Svetlin Nakov – Official Web Site".
- Find the search link by its class name and verify its text contains "SEARCH".
- Click on the search link.
- Find the search input element by its ID.
- Retrieve and verify the placeholder attribute of the search input is "Search this site".

```
[Test]
0 references
public void Test_NakovCom()
{
    driver.Url = "https://nakov.com";
    var windowTitle = driver.Title;
    Assert.That(windowTitle.Contains("Svetlin Nakov – Official Web Site"));
    Console.WriteLine(windowTitle);

    var searchLink = driver.FindElement(By.ClassName("smoothScroll"));
    Assert.That(searchLink.Text, Does.Contain("SEARCH"));
    Console.WriteLine(searchLink.Text);

    searchLink.Click();

    var message = driver.FindElement(By.Id("s"));
    var placeholderText = message.GetAttribute("placeholder");
    Assert.That(placeholderText, Is.EqualTo("Search this site"));
    Console.WriteLine(placeholderText);
}
```

- Define the teardown method to close the browser after each test.

```
[TearDown]
0 references
public void TearDown()
{
    driver.Dispose();
}
```

3. Practice Locators

You will receive two HTML files: **Locators.html** and **Thankyou.html**. These files will be in a folder called **SimpleForm**. Ensure both **Locators.html** and **Thankyou.html** are in the **same directory on your local machine**. For example, place them in **C:/Users/YourUsername/Desktop/SimpleForm/**.

- Create a New Project:

- Name your project (e.g., LocatorsPractice) and click Create.
- Install Necessary NuGet Packages:
- Install Selenium WebDriver:
- Search for Selenium.WebDriver.
- Install Selenium WebDriver for Chrome:
- In your test file, add the necessary using statements at the top.
- Initialize the WebDriver and navigate to the Locators.html file.

```
private string baseUrl = "file:///C:/Users/YourUsername/Desktop/SimpleForm/Locators.html";
```

- Write **OneTimeSetup** and **OneTimeTearDown** Method

3.1. Basic Locators

Practice locating elements by ID, Name, Tag Name, and Class Name, for example:

- Find the LastName by ID
- Find the Newsletter by Name
- Find the Official Softuni Page by TagName
- Find the information fields by ClassName

```
driver.FindElement(By.Id("lname"));
driver.FindElement(By.Name("newsletter"));
driver.FindElement(By.TagName("a"));
driver.FindElement(By.ClassName("information"));
```

3.2. Text Link Locators

Practice locating elements by Link Text and Partial Link Text, for example:

```
driver.FindElement(By.LinkText("Softuni Official Page"));
driver.FindElement(By.PartialLinkText("Official Page"));
```

3.3. CSS selectors

Practice locating elements using various CSS selectors, for example:

```
driver.FindElement(By.CssSelector("#fname"));
driver.FindElement(By.CssSelector("input[name='fname']"));
driver.FindElement(By.CssSelector("input[class*='button']"));
driver.FindElement(By.CssSelector("div.additional-info > p > input[type='text']"));
driver.FindElement(By.CssSelector("form div.additional-info input[type='text']"));
```

3.4. XPath Locators

Practice locating elements using various XPath expressions, for example:

```
driver.FindElement(By.XPath("/html/body/form/input[1]"));
driver.FindElement(By.XPath("//input[@value='m']"));
driver.FindElement(By.XPath("//input[@name='lname']"));
driver.FindElement(By.XPath("//input[@type='checkbox']"));
driver.FindElement(By.XPath("//input[@class='button']"));
driver.FindElement(By.XPath("//div[@class='additional-info']//input[@type='text']"));
```

4. Practice Locators Extended

4.1. Basic Locators

Extend the previous task. Add assertions to verify the located elements.

- Locate the "Last name" input field and verify its value
- Locate the "Newsletter" checkbox and verify it is not selected
- Locate the anchor tag and verify its text
- Locate the element with class name "information" and verify its background color

4.2. Text Link Locators

- Locate the link by its full text and verify its href attribute
- Locate the link by partial text and verify its displayed text

4.3. CSS selectors

- Locate the "First name" input field by ID and verify its value
- Locate the "First name" input field by name attribute and verify its value
- Locate the submit button by class name and verify its value attribute
- Locate the "Phone Number" input field by CSS selector and verify it is displayed
- Locate the "Phone Number" input field using a more specific CSS selector and verify it is displayed

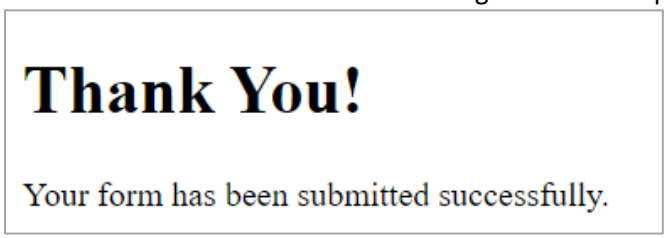
4.4. XPath Locators

- Locate the male radio button using absolute XPath and verify its value attribute
- Locate the male radio button using relative XPath and verify its value attribute
- Locate the last name input field using relative XPath and verify its value
- Locate the newsletter checkbox using relative XPath and verify its type attribute
- Locate the submit button using relative XPath and verify its value attribute
- Locate the phone number input field within additional info using relative XPath and verify it is displayed

5. Test Form Submission

Write a Test that does the following:

- Asserts the "Contact Form" title.
- Selects the male radio button and asserts its selection.
- Enters "Butch" as the first name and asserts the entered value.
- Enters "Coolidge" as the last name and asserts the entered value.
- Asserts the presence of the "Additional Information" section.
- Enters "0888999777" as the phone number and asserts the entered value.
- Selects the newsletter checkbox and asserts its selection.
- Clicks the submit button.
- Asserts the "Thank You!" message on the next page.



6. Headless Mode

Run some of your Tests in a headless browser. Check if there is difference in times.

```
var options = new ChromeOptions();
options.AddArgument("--headless=new");
options.AddArgument("--window-size=1920,1200");
driver = new ChromeDriver(options);
```