# Exercises: Selenium Basics

This document defines the homework assignments from the [QA Automation" Course @ Software University](https://softuni.bg/trainings/2550/qa-automation-may-2020).

Please submit the homework a single zip / rar / 7z archive holding the source code and any other project assets.

## Search for “QA” in Wikipedia

* Write a **test** that **opens a browser** and navigates to <https://wikipedia.org>.
* Then type “**QA**” in the search box and click **[Enter]**.
* **Assert** that the browser opens the following URL: <https://en.wikipedia.org/wiki/QA>.

Картина, която съдържа текст

Описанието е генерирано автоматично 

### Hints and Guidelines

Create a new **C# NUnit Project** in Visual Studio:

Картина, която съдържа текст

Описанието е генерирано автоматично

Give a meaningful name for your project. It will hold **Selenium tests**, so its name could be “SeleniumTests” or “NUnitProjectSelenium” or “SeleniumAutomatedTests”:

Картина, която съдържа текст

Описанието е генерирано автоматично

Install the NuGet packages “Selenium.WebDriver” and “Selenium.WebDriver.Chrome”:

Картина, която съдържа текст

Описанието е генерирано автоматично

**Rename** the main C# class in your project. Use a meaningful name like “SeleniumTests”:

Картина, която съдържа текст

Описанието е генерирано автоматично Картина, която съдържа текст

Описанието е генерирано автоматично

Картина, която съдържа текст

Описанието е генерирано автоматично

**Remove the namespace**. It is unnecessary in small projects.

Картина, която съдържа текст

Описанието е генерирано автоматично

Import the required **namespaces for Selenium**:

Картина, която съдържа текст

Описанието е генерирано автоматично

Initialize the ChromeDriver in the SetUp() method and release the driver in the TearDown() method:

Картина, която съдържа текст

Описанието е генерирано автоматично Картина, която съдържа текст

Описанието е генерирано автоматично

Now, it’s time to start **writing the Selenium test**. Define the test method. Choose a meaningful name:

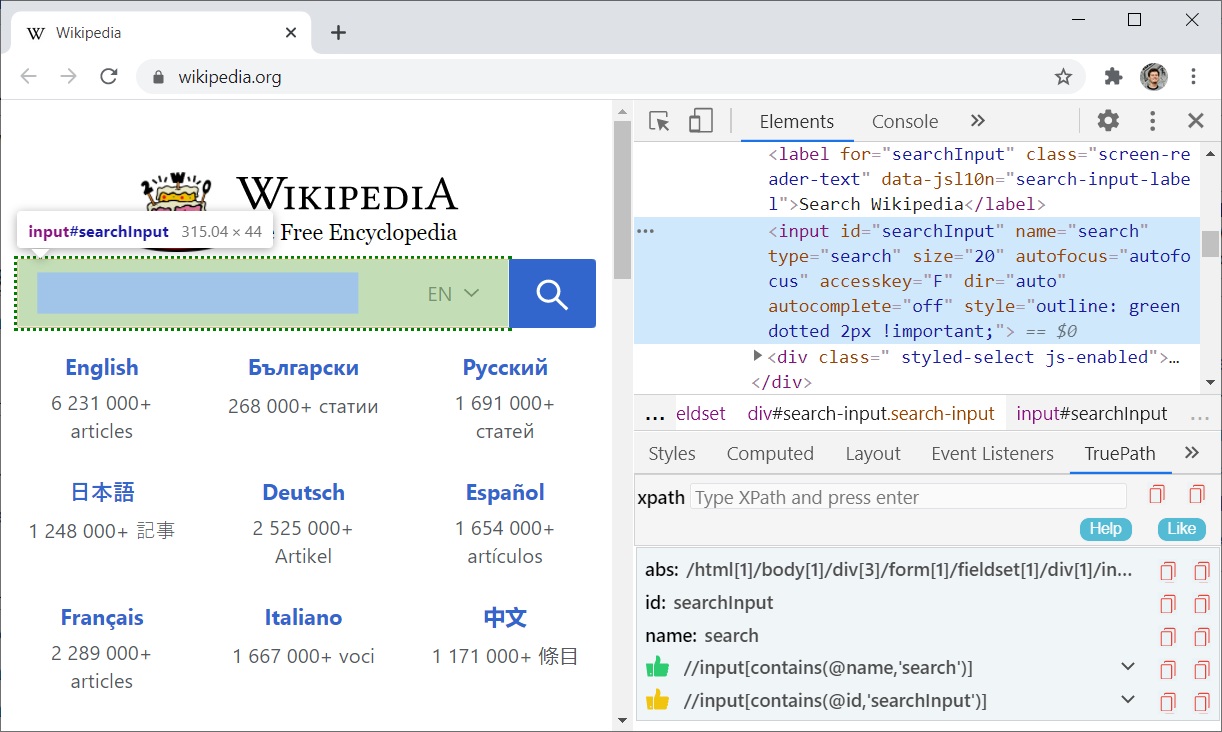
Картина, която съдържа текст

Описанието е генерирано автоматично

Navigate to the target Web page: <https://wikipedia.org>:



Find the **locator** for the **search box**. You may use **XPath locator** or **CSS locator** or other way to locate the element. **Inspect the element** in the Web browser’s developer tools:

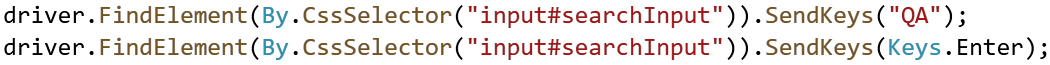


You may also use the [**TruePath** add-on](https://chrome.google.com/webstore/detail/truepath/mgjhkhhbkkldiihlajcnlfchfcmhipmn), to get some automatic suggestions for the element locator. This input text box is easy to locate. It has non-dynamic id="searchInput", name="search" and it is also the only input element of type="search" on the page.

We can **click on the search box** with the following Selenium command:



Then, we can **type “QA”** in the search box and click **[Enter]**:



The browser should be **redirected** to the “search results” page: <https://en.wikipedia.org/wiki/QA>.

Finally, we need to **assert** that the search result page for “QA” is **loaded**:



The last step is to **run the test** and make sure that it **passes correctly**, as expected:

Картина, която съдържа текст

Описанието е генерирано автоматично

## Automated Test for the “Summator of Numbers” App

Write **automated Selenium UI tests** for the following app, which sums two numbers:

* Live app URL: <https://sum-numbers.nakov.repl.co>
* Source code: <https://repl.it/@nakov/sum-numbers>

Картина, която съдържа текст

Описанието е генерирано автоматично Картина, която съдържа текст

Описанието е генерирано автоматично

### Hints and Guidelines

The **first test** should be the straightforward case (sum two valid integers). Use the following steps:

* Open the target web app URL with the Selenium Web driver.
* Find the **first text field** and type some number, e. g. “**15**”.
* Find the **second text field** and type some number, e. g. “**7**”.
* Find the **[Calculate]** button and **click** it.
* Find the result box and take the text from it.
* Assert that the text is as expected, e. g. “**Sum: 22**”.

This is how the test source code may look like:

Картина, която съдържа текст

Описанието е генерирано автоматично

**Run the test** and make sure it passes correctly:

Картина, която съдържа текст

Описанието е генерирано автоматично

The **second test** should check that when we **sum invalid numbers** (like “**hello**” and “”), the result will be “*invalid input*”. It should be very similar to the previous test:

Картина, която съдържа текст, лице, екранна снимка

Описанието е генерирано автоматично

The **third test** should check whether the **[Reset] button** works correctly. This is how it can be tested:

* **Open** the Summator app.
* Fill the form and **click [Calculate]**.
* Assert that all fields are **non-empty**.
* **Click [Reset]** to clear all fields.
* Assert that all fields are **empty**.

This is how the code of the test may look like:

Картина, която съдържа текст

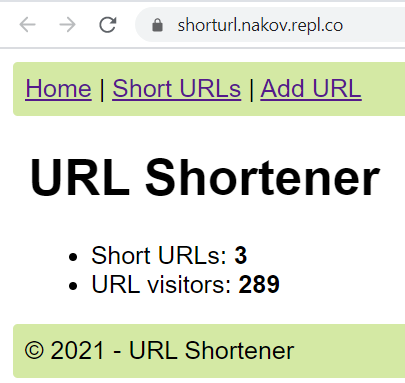
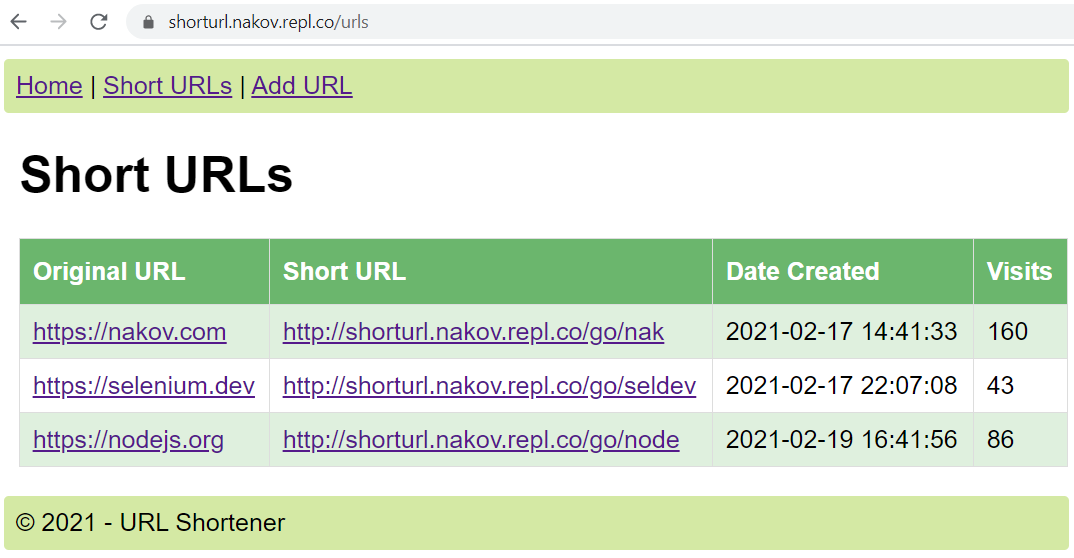
Описанието е генерирано автоматично

## Automated Test for the “URL Shortener” App

You are assigned to **write automated tests** for the “**URL Shortener**” Web app: <https://shorturl.nakov.repl.co>.

The app holds a **collection of URLs**, accessible by a **short code**. It supports the following operations:

* **Add a new URL** 🡪 generates a short URL by given URL + short code
* **Redirect** from a short URL to the original URL (by clicking a short URL from the “Short URLs” table)
* **View statistics**: URL | Short URL | Date Created | Visits (from the “Short URLs” table)

Your task is to write **automated UI tests** to cover the following test scenarios:

### Test Home Page

Open the **home page** and assert that its title holds the text “URL Shortener”.

### Test “Short URLs” Page

Open the **“Short URLs” page** and ensure it holds a **table** with **short URLs** and the first table rows holds the URL “https://nakov.com” and the short URL “https://shorturl.nakov.repl.co/go/nak”.

**Hints**:

1. **Open** the “Short URLs” page: <https://shorturl.nakov.repl.co/urls>.
2. **Assert** the **page title** (CSS selector main > h1) is “Short URLs”.
3. Find **all table cells** (CSS selector table tr > td).
4. Assert the first cell found holds <https://nakov.com>.
5. Assert the second table cell holds “https://shorturl.nakov.repl.co/go/nak”

### Test “Add URL”

Create new short URL from the [Add URL] (valid data & invalid data).

**Hints**:

* The “**invalid data**” test scenario is easier. Open the “Add URL” page, fill the form with invalid data, click the [Create] button and assert the red error message “Invalid URL!” is shown on the page.
* The “**valid data**” scenario is more complex. Open the “Add URL” page, fill the form with valid data. Ensure your testing data is unique. This will make your test repeatable (it will produce stable results if you run it many times consequently, regardless of what’s in the database). You can use DateTime.Now.Ticks to generate unique number, different at each test execution. After successful form submission, your URL should be listed in the “Short URLs” table. Assert this by finding all elements by link text. Use your uniquely generated URL as link text. Assert the matching elements are exactly 1.

### Test “Visit URL”

To test the “visit URL” functionality, you should implement two automated tests:

* **Visit existing short URL** and ensure that the visitors counter increases.
* **Visit non-existing short URL** and ensure an error is displayed.

**Hints**:

* When you click on certain URL, it will open in a new tab. You may need to switch the Selenium driver to operate on the new tab, e. g. by driver.SwitchTo().Window(driver.WindowHandles[1]).
* To return back to the previous tab, you may use the following code: driver.SwitchTo().Window(driver.WindowHandles[0]).

Design and implement the test cases yourself.

## Data-Driven Tests for the “Number Calculator” App

Write **automated Selenium tests** for the “Number Calculator” app, which executes different **arithmetic operations on two numbers**:

* Live app URL: <https://number-calculator.nakov.repl.co/>

Картина, която съдържа текст

Описанието е генерирано автоматичноКартина, която съдържа текст

Описанието е генерирано автоматично

Write at least **3 tests for each** of the following **test cases**. Test with:

* Valid integers
* Valid decimal numbers
* Exponential numbers
* Infinity
* Invalid inputs
* Invalid operations

### Hints and Guidelines

First, create **fields** for the **driver** and **web elements**:

Картина, която съдържа текст

Описанието е генерирано автоматично

Let’s now create the Setup() method. **Initialize** the driver as ChromeDriver() with the app URL:

Картина, която съдържа текст

Описанието е генерирано автоматично

Then, **locate** the **web elements** on the page by their Ids:

Картина, която съдържа текст

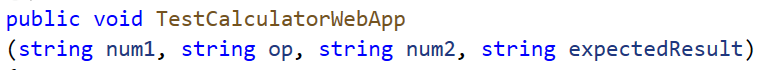
Описанието е генерирано автоматично

Now write the TearDown() method, as well. It should just **quit the driver**:

Картина, която съдържа текст

Описанието е генерирано автоматично

Then, **create the test method** itself. It should **accept** strings for the **first number**, the **operator**, **the second number** and the **expected result**:



The next steps are:

* **Click** the [Reset] button
* If given **attribute values are not** **empty string**, **send values** to the corresponding **fields**
* Click the [Calculate] button
* **Assert** the **expected** and **actual result text** are equal

The method may look like this:

Картина, която съдържа текст

Описанието е генерирано автоматично

At the end, write the **different test cases** with the [TestCase] attribute and the method **argument values**. For example, tests with **valid integers** may be the following:

Картина, която съдържа текст

Описанието е генерирано автоматично

Write **other test cases**, as well. At the end, your test cases may look like these:

Картина, която съдържа текст

Описанието е генерирано автоматичноКартина, която съдържа текст

Описанието е генерирано автоматично

Note that each **test case** is **executed as a separate test**, but code of test method is not repeated, but **reused**. Also, in this way **Selenium tests** are really **fast** and clear.