# Exercises: Selenium Advanced and POM

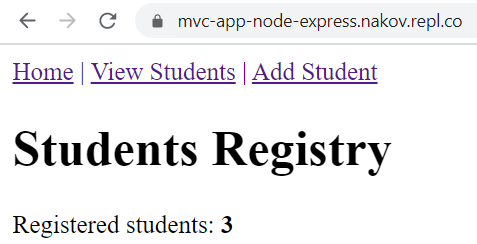
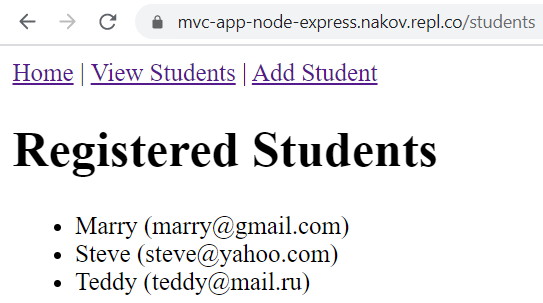
This document defines the homework assignments from the [QA Automation" Course @ Software University](https://softuni.bg/trainings/2550/qa-automation-may-2020).

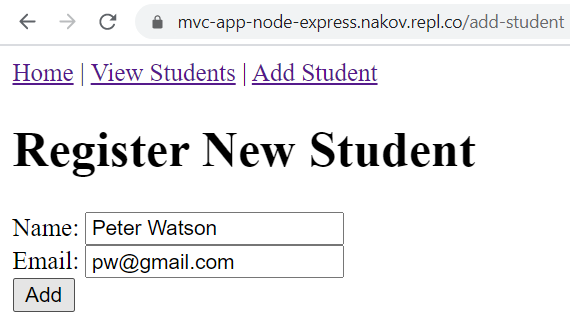
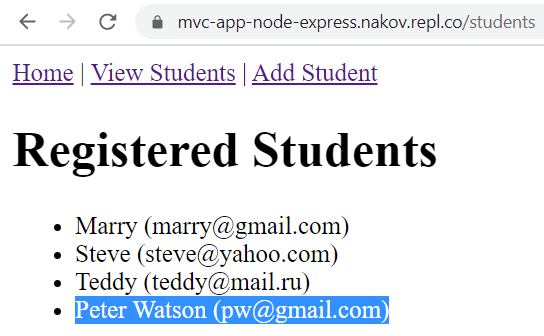
Please submit the homework a single zip / rar / 7z archive holding the source code and any other project assets.

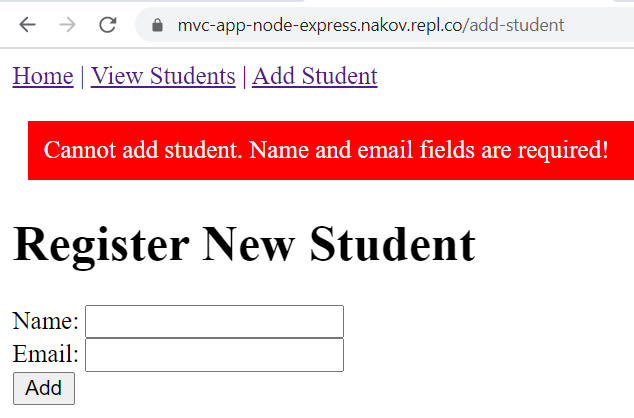
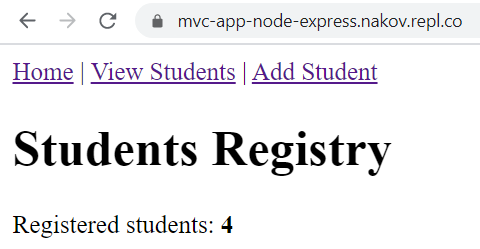
## Automated Testing of “Student Registry” App

Write **automated Selenium UI tests** for the following app, which holds a registry of users:

* Live app URL: <https://mvc-app-node-express.nakov.repl.co>
* Source code: <https://repl.it/@nakov/mvc-app-node-express>

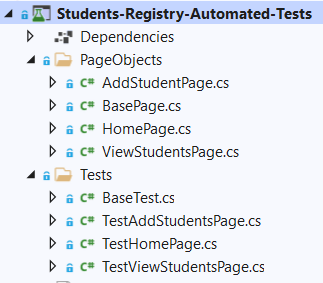
 

### Test Project Structure

Create a “NUnit Test Project” called “Students-Registry-Selenium-POM-Tests”. As we will use the Page Object Model, we will have **Page Object classes** and **test classes**. **Separate them to folders**. You should have the following **project structure**:



### Create Page Object Classes

Create Page Object **classes** **for each page** of the “Students Registry App”. You should have a **base PO class** with **common properties and methods** for all PO child classes. The other **PO classes** – the “HomePage”, the “ViewStudentsPage” and the “AddStudentPage” **classes** should **inherit** the “BasePage” base class.

#### Create the “BasePage” Page Object Class

The “BasePage” **class** is a **base class** for all **Page Object classes**. It should contain:

* Field: IWebDriver driver (protected and readonly)
* Constructor: BasePage(IWebDriver driver)
* Virtual property: PageUrl
* Properties: LinkHomePage, LinkViewStudentsPage, LinkAddStudentsPage, ElementTextHeading
* Method: Open() => driver.Url = this.PageUrl;
* Method: IsOpen() => driver.Url == this.PageUrl;
* Methods: GetPageTitle(), GetPageHeading()

The field of type IWebDriver should be protected, so that only child classes can access it but it should not be changed directly, so it is also readonly. The driver is accepted through the constructor. Also, it is a good idea to set an implicit wait for the driver. Additionally, we should have a virtual property PageURL, which will be different for each child class. Write the field, the constructor and the property like this:

Картина, която съдържа текст

Описанието е генерирано автоматично

Other properties of the PO class keep each of the links in the main menu, which are shared between all pages. Only the last property is different – it locates the page heading of the current page. All elements are located in the usual way with different locator strategies. The properties look like this:

Картина, която съдържа текст

Описанието е генерирано автоматично

Next, we have the Open() method, which is responsible for opening a page on a given page URL. The method is pretty simple:

Картина, която съдържа текст

Описанието е генерирано автоматично

We also have the IsOpen() **boolean** method, which **checks whether the current URL of the driver is the same as the page URL of our page**. If they are the same, then the user is on the right page and it is open. The method looks like this:

Картина, която съдържа текст

Описанието е генерирано автоматично

Our final **two methods** for this class **get the title** **and the heading text of the current page**. The GetPageTitle() method is the following:

Картина, която съдържа текст

Описанието е генерирано автоматично

The GetPageHeadingText() **returns the text** from the ElementPageHeading **property**:

Картина, която съдържа текст

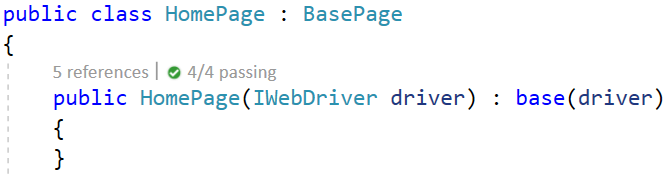
Описанието е генерирано автоматично

#### Create the “HomePage” Page Object Class

The “HomePage” **class** **inherits** the “BasePage” **class** (can use its properties and methods) and should contain:

* Constructor: HomePage(IWebDriver driver)
* Properties: inherited + PageUrl (assigned correctly) + ElementStudentsCount
* Methods: inherited + GetStudentsCount()

**Inherit** the “BasePage” class and **use its constructor** **by** **providing it with the driver**:



Next, **override** **the** PageURL **property** with the **URL** of the “Home” **page** of the app (<https://mvc-app-node-express.nakov.repl.co/>):



Then, **create** the ElementStudentsCount **property**, which **locates the count of registered students** on the page:

Картина, която съдържа текст

Описанието е генерирано автоматично

The property looks like this:



#### Create the “ViewStudentsPage” Page Object Class

The “ViewStudentsPage” **PO class** also **inherits** the “BasePage” **class**. In addition, it has:

* Properties: inherited + PageUrl (assigned correctly) + ListItemsStudents (of type ReadOnlyCollection<IWebElement>)
* Methods: inherited + GetStudentsList() (returns string[])

The whole class looks like this:

Картина, която съдържа текст

Описанието е генерирано автоматичноКартина, която съдържа текст

Описанието е генерирано автоматично

#### Create the “AddStudentPage” Page Object Class

The “AddStudentPage” class **inherits** the “BasePage” class and has:

* Properties: inherited + PageUrl (assigned correctly) + ElementErrorMsg
* Form field properties: FieldStudentName, FieldStudentEmail, ButtonAdd
* Methods: AddStudent(string name, string email), GetErrorMsg()

Write the **constructor** and the **properties** by yourself. Note that the ElementErrorMsg property **locates an error message**, which appears only when the “Register New Student” **form is filled with invalid data**:

Картина, която съдържа текст

Описанието е генерирано автоматично

The AddStudent(string name, string email) **fills the registration form**, using the **field properties**. It looks like this:

Картина, която съдържа текст

Описанието е генерирано автоматично

The GetErrorMsg() method **returns the error text** from the ElementErrorMsg **property**. Write the method on your own.

### Write Selenium POM Tests

#### Create the Tests Base Class

The “BaseTest” **class** is a **base class for all other test classes**. It has the OneTimeSetUp() and OneTimeTearDown() methods, which **initialize** and **quit** the ChromeDriver(). The class looks like this:

Картина, която съдържа текст

Описанието е генерирано автоматично

Картина, която съдържа текст

Описанието е генерирано автоматично

#### Create “Home” Page Tests

On our “Home” **page**, we should **test the page content** and the **page links**. Create the “TestHomePage” **tests** **class**, which should **inherit** the “BaseTest” class to **access the driver**:



The Test\_HomePage\_Content() test **will open the** “Home” **page** and **assert that it has a correct title**, **heading** and **students count**. To write the test method, follow these steps:

* **Instantiate** the “Home” **page** **with** **driver** and **open the page**:

Картина, която съдържа текст

Описанието е генерирано автоматично

* **Assert** the page **title** is correct (window title):



* **Assert** the page **heading** is correct (the top heading at the start of the page):



* **Invoke** the GetStudentsCount() method- it should **not throw any errors**:

Картина, която съдържа текст

Описанието е генерирано автоматично

The Test\_HomePage\_Links() test will check whether the “Home” **page** **links open the correct pages**. To write the test method, follow these steps:

* **Instantiate** the “HomePage” class with **driver**:

Картина, която съдържа текст

Описанието е генерирано автоматично

* **Go** to the “Home” **page**, **click on the** “Home” **page link** and **assert the** “Home” **page is open**:

Картина, която съдържа текст

Описанието е генерирано автоматично

* Do the **same steps** from the previous point to **test** the “AddStudentsPage” and the “ViewStudentsPage” **links**:

Картина, която съдържа текст, закрито, екранна снимка

Описанието е генерирано автоматично

#### Create “View Students” Page Tests

The “TestViewStudentsPage” **test class** should inherit the “BaseTest” **base class**. Write the following **test methods** in the class: the Test\_ViewStudentsPage\_Content() method to **check page content** and the Test\_ViewStudentsPage\_Links() method to **check links to other pages**.

The Test\_ViewStudentsPage\_Content() test method should:

* **Instantiate** the “ViewStudentsPage” class, **open** the “View Students” **page** and **check its title and heading**:

Картина, която съдържа текст

Описанието е генерирано автоматично

* **Invoke** the GetRegisteredStudents() method to **get all students on the page**:



* **Assert** that **each student record contains** “(” **and finishes with** “)”:

Картина, която съдържа текст

Описанието е генерирано автоматично

For the Test\_ViewStudentsPage\_Links() test method, **go** to the “View Students” **page** and **click on each of the links**. They should **open the correct pages**. Write the test by yourself, it is very similar to the Test\_HomePage\_Links() test.

#### Create “Add Student” Page Tests

The “TestAddStudentPage” inherits the “BaseTest” class and has the following **test methods**:

* Test\_TestAddStudentPage\_Content()
  + **Instantiate** the “AddStudentPage” **class** with driver
  + **Open** the “Add Student” **page**
  + **Assert** the **page title** and **heading** are correct
  + **Assert** the **form fields are empty**
  + **Assert** that the **form button has a correct text**
* Test\_TestAddStudentPage\_Links()
  + **Instantiate** the “AddStudentPage” **class** with driver
  + **Open** the “Add Student” page
  + **Assert** the “Home” **page link** opens the page
  + **Assert** the “Add Student” **page link** opens the page
  + **Assert** the “View Students” **page link** opens the page
* Test\_TestAddStudentPage\_AddValidStudent
  + **Instantiate** the “AddStudentPage” **class** with driver
  + **Open** the “Add Student” **page**
  + Generate a **unique student name and email**:
    - string name = "New student" + DateTime.Now.Ticks;
    - string email = "email" + DateTime.Now.Ticks + "@email.com";
  + **Invoke** the AddStudent(string name, string email) method
  + **Instantiate** the “ViewStudentsPage” class with driver
  + **Assert** the “View Students” page is open
  + **Assert** the **page contains the new student**
    - studentsPage.GetStudentsList() collection should include the new student
* Test\_TestAddStudentPage\_AddInvalidStudent()
  + **Instantiate** the “AddStudentPage” class with driver
  + **Open** the “Add Student” page
  + **Invoke** the AddStudent(string name, string email) method with **invalid data**, e.g. an empty name
  + **Assert** the “Add Student” page is still open
  + **Assert** that the error message contains the “**Cannot** **add** **student**” text
  + **Invoke** the GetErrorMsg() method

You already know how to write these **test cases** – **write them on your own**.

### Run Tests

**Run** all tests and **ensure** **they work correctly**.

This is how your **final set of tests** may look like:

Картина, която съдържа текст

Описанието е генерирано автоматично