# Where we are

You should have 7 tests by now. They have a high amount of duplication – the same locators and steps repeated over and over. Let’s fix that.

Please watch the video on Frameworks before starting this task.

# Task 1 – Page Objects

The first thing we should do is implement the Page Object pattern. We do this by creating a class for each page that contains all information and logic related to that page:

1. Our steps use 5 pages: Home, News, Have your say, Search results, Lorem ipsum. Let’s create a class for each of them with a descriptive name, e.g. NewsPage;
2. Move all IWebElements that you use in your tests to private (!) fields of these new classes;
3. Create public (or private if useful) methods that represent actions that you take on these pages. For example, to do a search on the news page, create a method Search(string toSearch) in the NewsPage class;
4. Refactor all your tests to use these new Page Objects;
5. Use PageFactory (PageFactory.InitElements(driver, page)) to initialize your Page Object. This can be called from its constructor. For this to work, all your private IWebElement fields need an attribute [FindsBy(How = How.Xpath, Using = “xpath”)];
6. Page Factory is a simple initialization method, which may not do all that you want it to. Implement your own initialization where this happens.

# Task 2 – Improve your Xpaths

Your Xpaths (especially if you simply copied them from Chrome) are likely quite long and you don’t understand what they do by looking at them. Let’s try a few things to fix that:

1. Xpaths for buttons can be shortened by using button text. You can even create a method that returns Xpath of element number X with a certain text, like GetTextXpath(String elementLabel, int elementNumber);
2. Xpaths for the “Have your say” form boxes can all be the same, except for the name of the input field.

You can create a GetFormXpath(String fieldName) method. You can then use this method to create a FillForm(Dictionary<string, string> values) method for your Page Object, which will fill this form (and in fact, any similarly implemented form) with values based only on fieldname-value pairs stored in the dictionary.

Rather than adding directly to your Page Object, the better approach is to wrap all Form logic in a class and then add this class to the Page Object for the page that has a form.

1. For all other Xpaths – try to find a unique attribute value somewhere close to the target element (or in the target element itself) that you can bind to. For example, top stories have a parent element with attribute data-entityid=’container-top-stories#X’, where X is the number of the story.

# Task 3 – add Business Logic layer

1. Add classes that will contain steps for “Have your say” and Lorem Ipsum. They will contain methods that work with multiple pages to accomplish tasks related to these two parts of functionality;
2. Create one method in each that implements a logical action that we can do with these parts of functionality: grab sample Lorem Ipsum, and submit a question to BBC;
3. Move logic into these methods until your 4 tests from Task 2 contain only 2 method calls each (plus a few operations with values). The first one will only take an integer value and return a Lorem Ipsum string of that size. The second will fill the form, based on a Dictionary<string, string> of form values and a bool true/false value of whether to press Submit or take a screenshot.

# Voila

We have a framework. 😊 As you can see, each test became much shorter than it was, and you can add new tests using methods from BL layer extremely quickly.