The Page Object Test Pattern

How to make the tests less brittle

# Page Objects

A test pattern to make GUI tests less brittle.

TestCode: [PageObjects.7z (see Files)](https://thermofisher.sharepoint.com/sites/EPU20-MigrationTeam/Shared%20Documents/Prototyping-SandBoxes/PageObjects.7z)

# Why use Page Objects

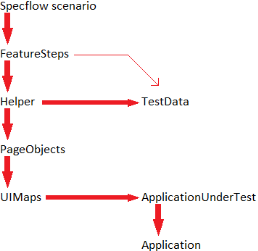
The page object pattern was at first used to test against web pages and web app GUI's. A page object was a wrapper around a HTML page, so that the tests could use a functional interface, rather than parsing HTML in the test.

For app testing, page objects can be used to wrap the code for the GUI test library. A set of classes make up a test using page objects (see image below). Each of the different layers provides a service to the layer above. The test describes What To Test (functionality). Each layer provides an abstraction for How To Test (pressing buttons and filling in text-boxes).

Using page objects, the tests are easier to maintain when code changes. And thus help write less brittle tests.

# Layers of a Page Object test

Example, when using SpecFlow to test an application, the layers look like this:



The red arrows can be translated as "uses". When reversing the arrows (except the arrow pointing to the Application) the arrows can be translated to "offers a service". E.g. Helpers use test data and page objects, UIMaps offer a service to PageObjects.

# Description of the different layers

The layers are described as follows:

* **Scenario:** SpecFlow scenarios are tests described in the Gherkin language.  
  Given a certain situation, When some action Then expect result.
* **FeatureSteps:** Each line in the scenario is represented by a method. So one for Given, one for When and one for Then.
* **Helpers:** Helper classes expose the functionality that is used in the test. For Example, a Login screen is there to log on to the program or service. The helper in this example usually exposes the complete login functionality. LoginWithUserCredentials(username, password);
* **TestData:** TestData is data needed to run a test, but has no impact on the outcome of the test. For Example: When testing login functionality, the scenario will provide a number of username/password combinations. However, other tests might assume an application where the user is already successfully logged on. The user credentials needed to log on can be stored as TestData.
* **PageObjects:** The page object represents a web page, part of a web page, a usercontrol, a screen or some other GUI object. The page object exposes functionality of the specific page, like EnterUsername, EnterPassword or ClickLoginButton. The page object wraps the GUI object, so that the helper class is not concerned by whether the page is HTML, a White library object or something else.
* **UIMaps:** This layer maps specific actions onto the different GUI items. For example, using the White library, the UIMap exposes a Button object, wrapping the White query like this.loginWindow.Get<Button>("Login").Click();
* **ApplicationUnderTest:** This is the part that starts the application or retrieves the webpage. It also handles the closing of the application.

# TestCode

The code link on top of this page contains a c# solution, implementing the different layers. It is an example where the application shows a log in screen, and the scenario tries to log on using different usernames and passwords. The resulting texts in the messagebox is used in the assert.

It uses Specflow, TestStack white and NUnit.