# Lab Module 2 – Build a Domain Specific Language

## Introduction

Writing Web UI Automation Test is quite intuitive and easy to write once you’ve seen a few examples using Selenium to locate elements, perform action and make some assertions.

However, maintaining these UI Automated test *– which are brittle by nature –* can be harder over time to the point where it becomes impractical to maintain.

In this module we will explore a few patterns to introduce a strong separation of concerns between the scripts *- which are the specifications or scenarios of what the tests should do –* and specialised objects abstracting the detailed interaction with the Pages using Selenium of our Application or System Under Test (SUT).

## Pre-requisites

Docker for Windows/Mac/Ubuntu, Visual Studio 2015, 2017 or 2019 and a GitHub account.

## Objectives

After completing this lab, you will be able to:

* Refactor existing brittle UI Selenium Tests with a lightweight DSL using Page Objects
* Understand how to apply SOLID, DRY and DAMP principle to maximise, optimise maintainability & readability of your test harness
* Although this example focuses on Web UI Automation, these principles apply for any type of interfaces whether Desktop or Mobile.

Scenario

Our new e-commerce website is about to release to production and some UI Selenium tests were written in a rush due to tight deadline by a contractor who left the organisation. You are now in charge on maintain these and extends these as a Automation Quality Engineer.

## Setting up

All the demos use the open source ASP.NET Core 2.2 reference application eShopOnWeb.

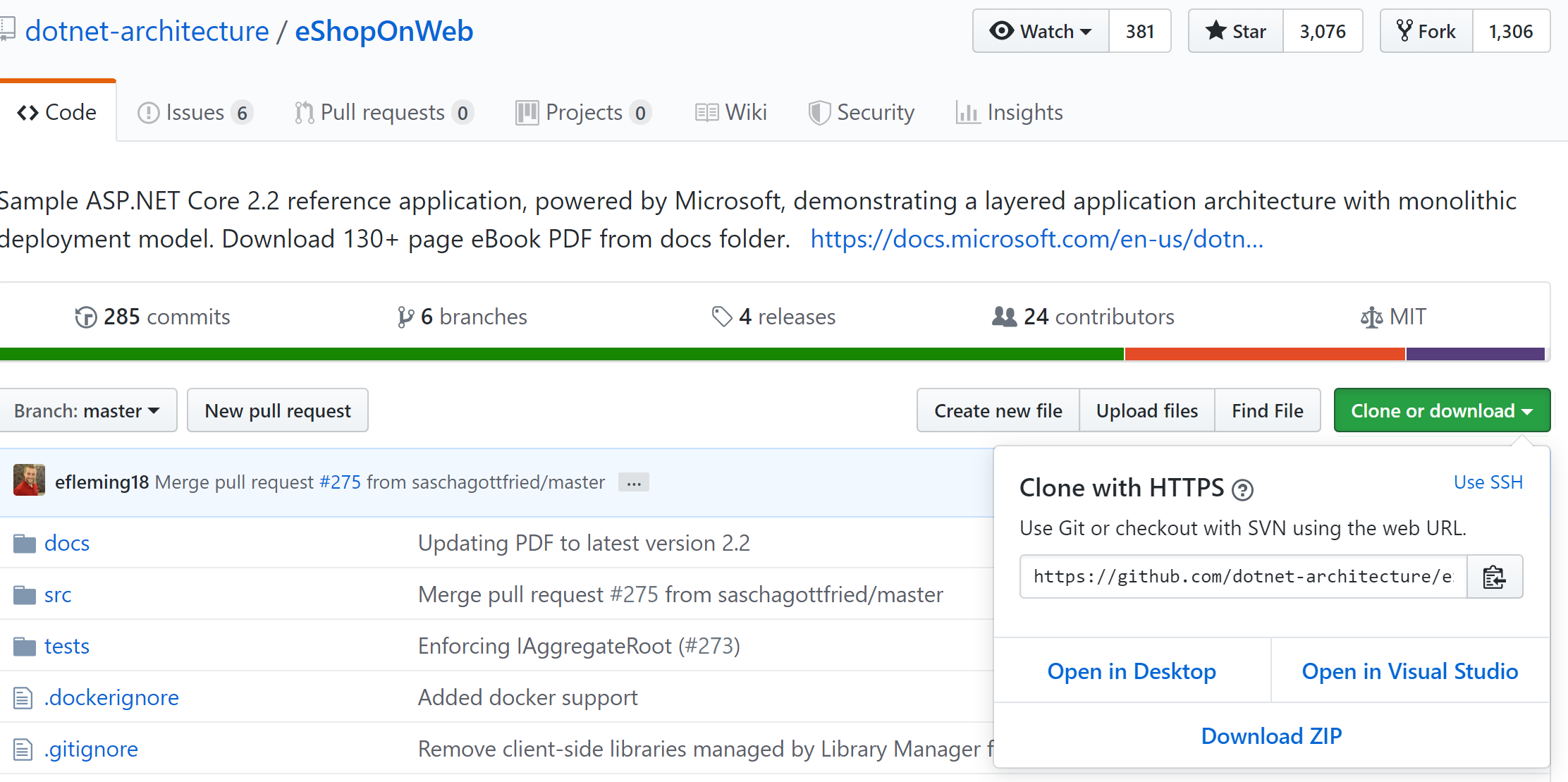
1. To set it up and getting it running, please execute the following commands:

git clone <https://github.com/dotnet-architecture/eShopOnWeb>  
cd eShopOnWeb  
docker-compose build  
docker-compose up

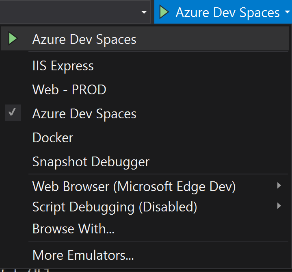
You can now access the application from <http://localhost:5106>

1. Alternatively, if you have an Azure Subscription, the solution can be deployed in Azure DevTest AKS environment. If you don’t have a AKS DevTest Cluster run the following Azure CLI command using [Azure Cloud Shell](https://docs.microsoft.com/en-us/azure/cloud-shell)

az aks create -g <resourceGroupName> -n <AKSClusterName> --location <region> --disable-rbac --generate-ssh-keys

Then clone the repository and open eShopOnWeb.sln with Visual Studio

And finally deploy using Azure DevTest from the list of execution environments as below:



### Exercise 1: Introducing Page Objects

#### Introduction

1. Encapsulate the page interaction into Page Objects
2. Break down test in separate scenarios flow

#### Tasks 1.1: Create Page Objects