**Problem:** Design a robust framework to automate test cases for Walmart.ca and validate a successful delivery to production

**Approach**: The idea is to separate the logic from page objects and make the testing framework extendable and scalable for future. For this purpose **WalmartSmokeTest** Solution has 2 projects **Walmart.Framework** and **Walmart.Testss**

**Future Enhancements** **/Scope**

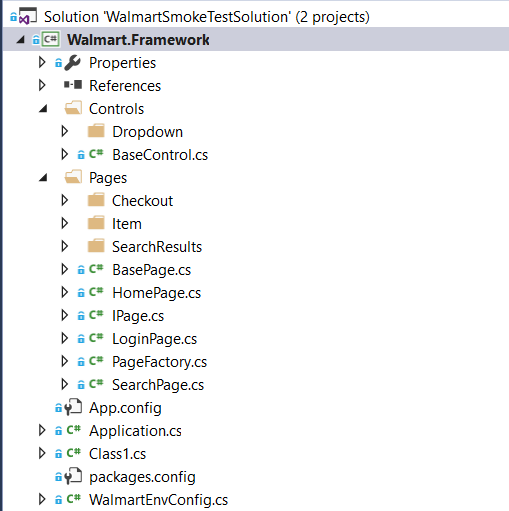
In Order to validate inventory updates, we may create another project as a part of same solution. This project will be designed to perform API testing.

1. **Watmart.Framework** project contains Pages Folder to hold all the Page Object classes and locators in order to help identify a control on UI. The Page objects are created based on all the different webpages a user wants to navigate to for example: SearchPage.cs, CheckoutPage.cs. These page objects classes contain methods to locate various objects like button or text editors etc.

**Control** folder contain subfolders for defining global controls like tables, dropdown control, bento-controls , ASP Controls etc

**Application.cs** is a class which defines the behavior of the application .all the one time action are recommended to define here like Launch the application for the first time each test starts executing. TearDown actions or any other application level actions should be written in this class.

**WalmartEnvConfig.cs** is created to read **App.config** file from **Walmart.Testss.dll.config** file (discussed later)



1. **Walmart.Testss** project contains following files and folders
2. Scenarioes : This folder is created to separate different test cases based on the their navigation or scenario type

Example: SeachItems.cs class file in which the search item related script is written, is kept inside SeachItems Folder to enhance the readability and help engineer located the script easily

1. App.config: this config file is created to define the environment variables required to execute the test script. For Example, the browser script should run , timeout, username, password, connection string etc.
2. Addins : This folder is created to redesign/enhance the nunit features for reporting purposes and redefine the behavior or different attributes. For Example. Instead of using **[Test**] attribute, a custom attribute **[Regression]** or **[NotReady]** are created. When any project is built, NUnit- Listener starts to look for **nunit.nuget.addins** addins file to identify the tests to execute.

Addins Folder further contains 2 sub-folders

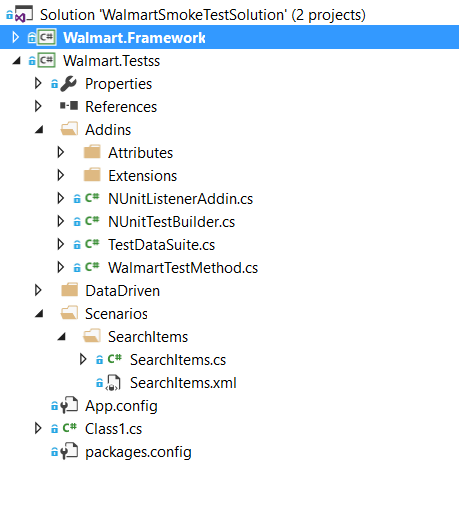
1. Attributes: to create any custom NUnit Attributes like Regression Attibute and NotReady Attribute. These attributes are created to give more meaningful looking tag to each test
2. Extension : this folder is created to hold any extension methods.

TestDataSuite.cs: Designed to redefine the way that NUnit finds, and executes the tests. For Example: instead of using [Test] to identify a test method we have created customAttribute [Regression] which will help identify the test method.

NUnitTestBuilder.cs: Designed to redefine how test attribute behaves. This helps to dynamically build the tests based on the data which is not known at compile time like directory location etc.

WalmartTestMethod

NUnitListenerAddin.cs:: this is class which Nunit looks for in order to find any extension attributes created



**Reporting::**

The idea is to use **ReportPortal** for creating interactive, real-time analysis and rich in user experience test report. Report Portal is easy to integrate with CI/CD Tools like Jenkins or TeamCity.

Some of the advantages are as follows:

1. Interactive reports
2. Real-time analysis or test scripts
3. Easy to collaborate with Cloud deployment
4. Fast traceability of defects
5. Manage all the report (history and present) at one place.

Pre-Requites for installation of ReportPortal

1. Docker Engine
2. Linux Virtual Machine

**Continuous Integration:**

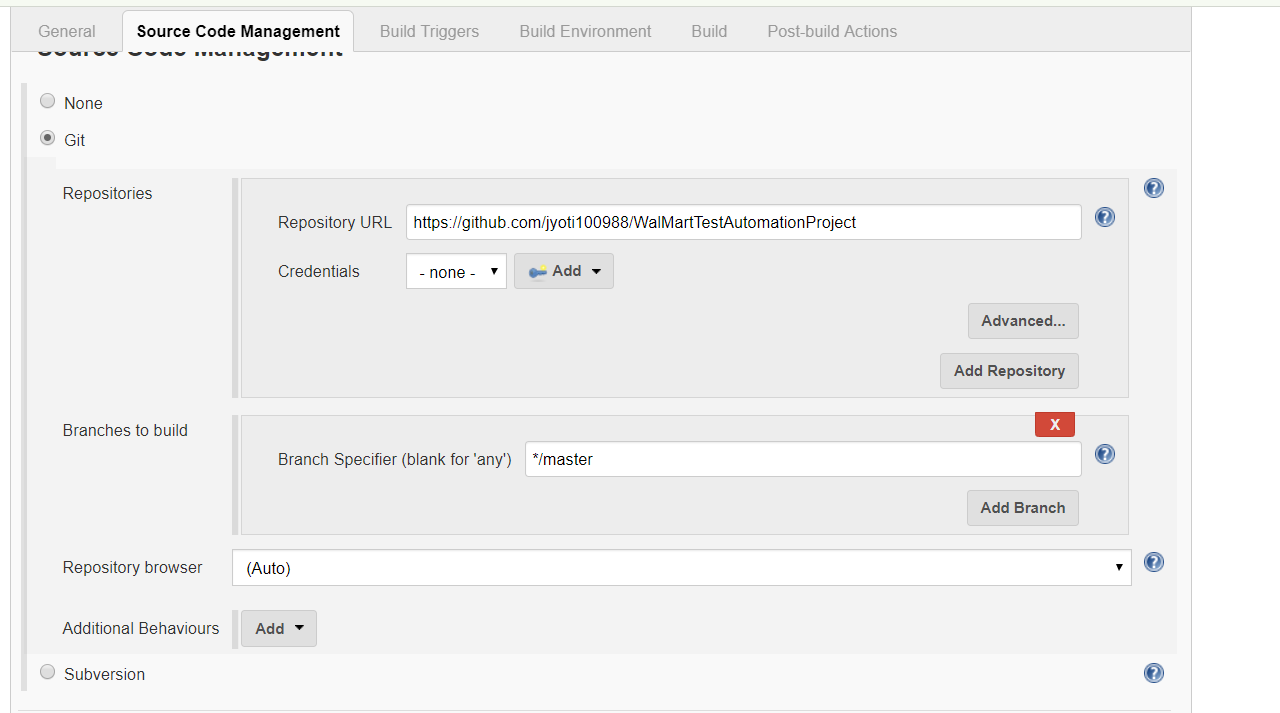
Current Project supports CI and CD using Jenkins.

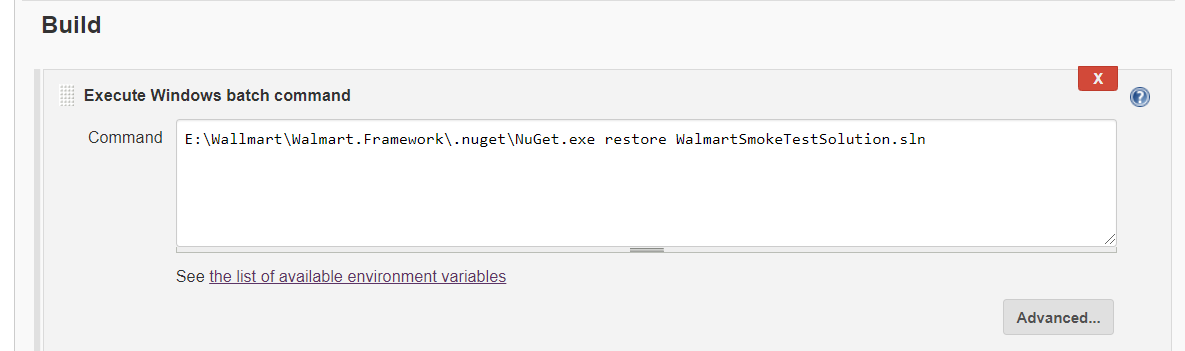
In Order to build this project from Jenkins (hosted locally on port 8080), we need to install MSBuild Plugin in Jenkins and created a free-style project for demo purpose instead of pipeline project.

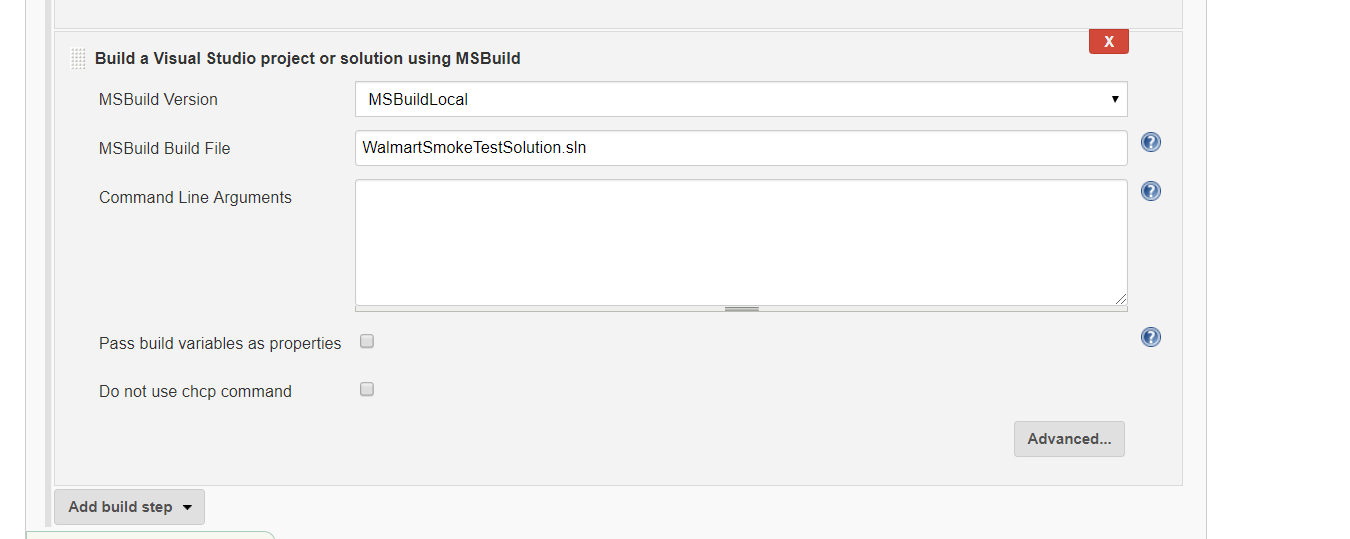
Note: MSBuild is the build engine from .Net Projects.

do the following configurations as shown in pictures below:

1. Install MSBuild Plugin from Global Configurations tools in Jenkins









**Parallelization:**

In order to achieve parallel execution to enable cross-browser testing or running multiple scripts at the same time, [Parallelizable] attribute is used on the top of test class.

**WaitMethods:**

**TryUntil** is a Coypu Framework method which is used until a specific condition return true.

**WaitLoading**(): Custom wait method written to wait until the page loading si completed using Javascript document.readyState == 'complete'

References:

1. <https://reportportal.io/>
2. <https://www.red-gate.com/simple-talk/dotnet/.net-tools/testing-times-ahead-extending-nunit/>