C# NUnitNew

Objectives

- Why do we need Unit test?
- AAA of Testing
- Nunit
- MSTest
- Annotation
- Create Test Product
- Add Dependencies
- Add Test Methods
- UnitTestProductRepo
- Test case
- UnitTestProductManager

TIME FOR CASE STUDY

Case Study - Continued

- As a Shopon developer, I want to unit test ProductManager methods which I am programming.

MSTestDemo.zip

NOTE: Refer to MSTestDemo Application from following link

Thought

- Write Unit test to Controller/Manager class.

Knowledge Byte

- In the software development process Unit Tests basically test individual parts (also called as Unit) of code (mostly methods) and make it work as expected by programmer.
- A Unit Test is a code written by any programmer which test small pieces of functionality of big programs.
- Generally the tests cases are written in the form of functions that will evaluate and determine whether a returned value after performing Unit Test is equals to the value you were expecting when we wrote the function.
- The main objective in unit testing is to isolate a unit part of code and validate its correctness and reliable.
- There are three different test frameworks that are supported by .NET Core for unit testing - **MSTest**, **xUnit**, and **NUnit**. These frameworks allow us to test our code in a consistent way.

Why do we need Unit test?

- Unit testing can increase confidence and certainty in changing and maintaining code in the development process.
- Unit testing always has the ability to find problems in early stages in the development cycle.
- Codes are more reusable, reliable and clean.
- Development becomes faster.
- Easy to automate.
- Read more at - http://agiledata.org/essays/tdd.html

AAA of Testing

- We have AAA pattern to write Unit Test cases:

Arrange all the necessary preconditions and inputs.
Act on the object or method under test.
Assert that the expected results have occurred.

NUnit

- NUnit is an open source unit test framework for all .NET languages. It is initially ported from JUnit. The current released version of NUnit is 3 which has been completely rewritten with many new features.

- To know more on version history, please do visit https://nunit.org/

MSTest

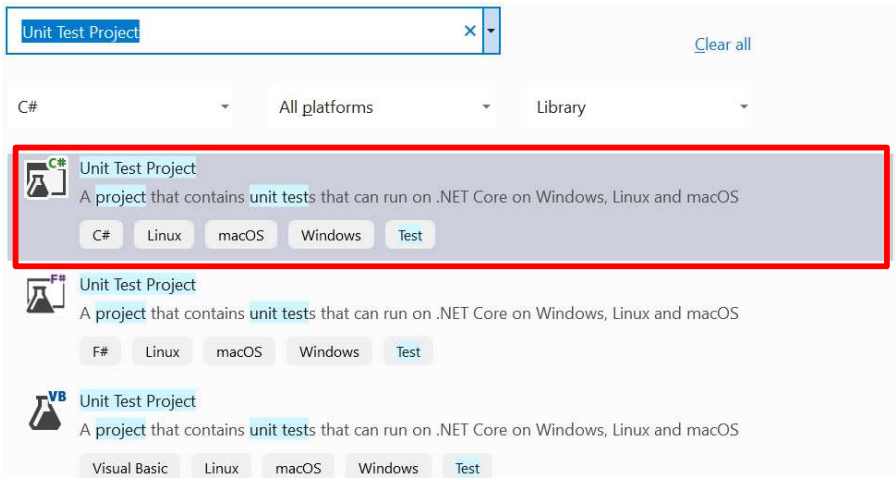
- MSTest is the default test framework that comes along with Visual Studio. In the earlier days, it started as a command-line tool for executing tests. It is also referred to as Visual Studio Unit Testing Framework; however, the name MSTest is more synonymous with the developers.

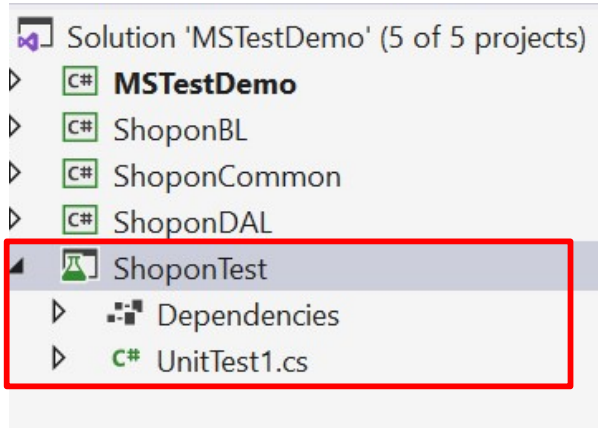
Annotation

ANNOTATION	DESCRIPTION
[TestInitialize]	Marks a method that should be called before each test method. One such method should be present per test class.
[TestCleanup]	Marks a method that should be called after each test method. One such method should be present per test class.
[TestClass]	Marks a class that contains tests.
[TestMethod]	Marks a method, i.e., an actual test case in the test class.
[Ignore]	Marks a test method or test class that should be considered for execution, i.e., it is ignored.
[TestCategory]	Specify the category for the test
[ClassInitialize]	Methods that will be called only once before executing any of the test methods present in that class.
[ClassCleanup]	Methods that will be called only once after executing the test methods present in that class.

Create Test Product

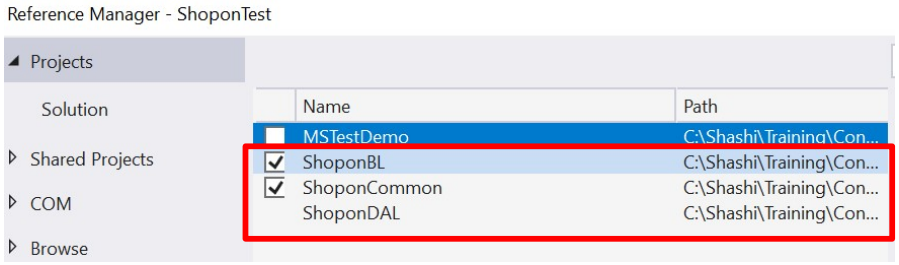
- To create test project, right click on the solution and **Add New Project**, select **Unit Test Project**

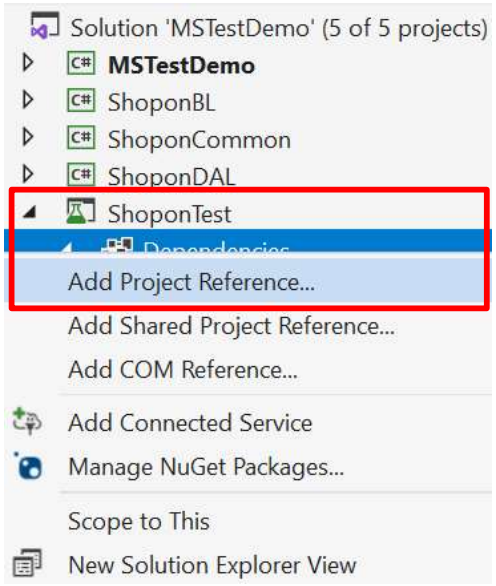




Add Dependencies

- To access the method from other project(s), we must add references of those projects in our test project.
- To add dependencies, right click on Dependencies in **Test project** and select **Add Project Reference**
- Select projects that should be added as reference





Add Test Methods

- In MSTest, all test class should be decorated by [**TestClass**] and all test method(s) by [*TestMethod*]

UnitTestProductRepo

- ProductRepo has contract(IProductRepo) and implementation(ProductRepoImpl)
- Method

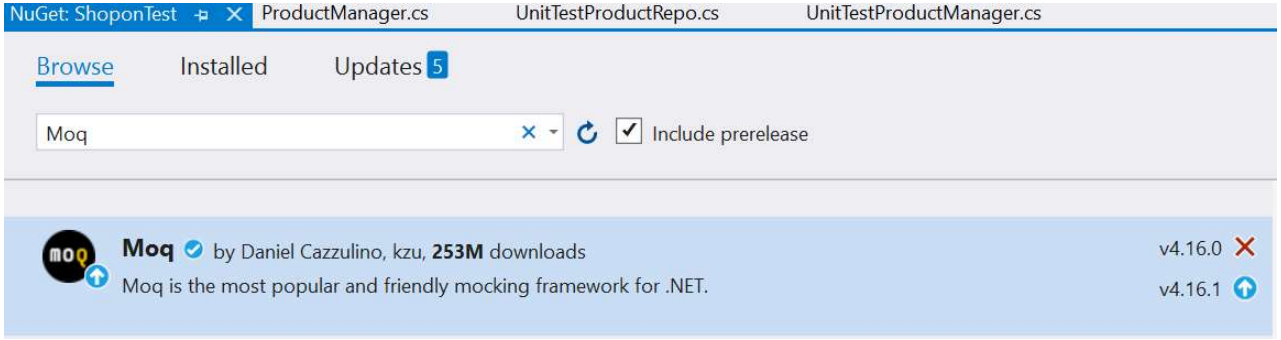
```
IEnumerable<Product> GetProducts();

Product Get(int id);

IEnumerable<Product> Get(string key);
```

Test case

```
[TestClass]
0 references
public class UnitTestProductRepo
{
    private IProductRepo productRepo = null;

    [TestInitialize]
    0 references
    public void Init()
    {
        this.productRepo = new ProductRepoImpl();
    }

    [TestMethod]
    | 0 references
    public void TestGetProducts()
    {
        var products = productRepo.GetProducts();
        var expectedCount = 10;
        var actualCount = products.Count();
        Assert.AreEqual(expectedCount, actualCount);
    }

    [TestMethod]
    | 0 references
    public void TestGetProductWhichExists()
    {
        var pid = 1;
        var product = productRepo.Get(pid);
        var expectedPrice = 25699;
        var actualPrice = product.Price;
        Assert.AreEqual(expectedPrice, actualPrice);
    }

    [TestMethod]
    | 0 references
    public void TestGetProductDoesnotExists()
    {
        var pid = 1001;
        var product = productRepo.Get(pid);
        Assert.IsNull(product);
    }
}
```

UnitTestProductManager

- As ProductManager has dependent on ProductRepository for data, we have to mock the implementation of it.
- To mock the repository, we must first install the required package.



Note: Moq should be installed to our Test project only.

Test Cases

```
[TestClass]
0 references
public class UnitTestProductManager
{
    private IProductManager productManager = null;
    private Mock<IProductRepo> productRepo = new Mock<IProductRepo>();

    [TestInitialize]
    0 references
    public void Init()
    {
        productManager = new ProductManager(productRepo.Object);
    }

    [TestMethod]
    | 0 references
    public void TestGetProducts()
    {
        List<Product> products = GetTestProducts().ToList();
        productRepo.Setup(x => x.GetProducts()).Returns(products);
        var actualProduct = productManager.GetProducts();
        Assert.IsNotNull(products);
    }
    [TestMethod]
    | 0 references
    public void TestGetProductById()
    {
        int id = 1;
        var product = new Product()
        {
            Id = 1,
            ProductName = "Apple I-Pad Mini",
            Price = 25699,
            ImageUrl = "images/Apple/01.jpg"
        };
        productRepo.Setup(x => x.Get(id)).Returns(product);
        var actualProduct = productManager.GetProductById(id);
        Assert.IsNotNull(actualProduct);
        Assert.AreEqual(actualProduct.Id, 1);
    }
```

```
1 reference | 1/1 passing
private IEnumerable<Product> GetTestProducts()
{
    return new List<Product>()
    {
        new Product()
        {
            Id = 1,
            ProductName = "Apple I-Pad Mini",
            Price = 25699,
            ImageUrl = "images/Apple/01.jpg"
        },
        new Product()...,
        new Product()...,
        new Product()...,
        new Product()...
    };
}
}
```

Next Step

Thank YOU

- Any Questions?

US – CORPORATE HEADQUARTERS

1248 Reamwood Avenue,
Sunnyvale, CA 94089
Phone: (408) 743 4400

343 Thornall St 720
Edison, NJ 08837
Phone: (732) 395 6900

UK

57 Rathbone Place,
4th Floor, Holden House,
London, W1T 1JU , UK

89 Worship Street Shoreditch,
London EC2A 2BF, UK
Phone: (44) 2079 938 955

INDIA

Mumbai
4th Floor, Nomura
Powai , Mumbai 400 076
Phone: +91 (22) 3051 1000

Pune
5th floor, Amar Paradigm Baner Road
Baner, Pune 411 045
Phone: +91 (20) 6604 6000

Bangalore
4th Floor, Kabra Excelsior,
80 Feet Main Road, Koramangala 1st Block,
Bengaluru (Bangalore) 560034
Phone: +91 (80) 4666 1666

www.xoriant.com