—

Moq, NSubstitute, AutoFixture & more

PERFECT COMBO FOR UNIT TESTING IN .NET

Agenda

1. AUTOMATED TESTING
2. DEMO PROJECT
3. .NET TESTING FRAMEWORKS
4. .NET MOCKING LIBRARIES
5. IMPROVING THE ARRANGE PART
6. IMPROVING THE ASSERT PART

1

DEV TESTS: UNIT TESTS, INTEGRATION TESTS, ETC...

Automated Testing

About Automated Testing

Why testing?

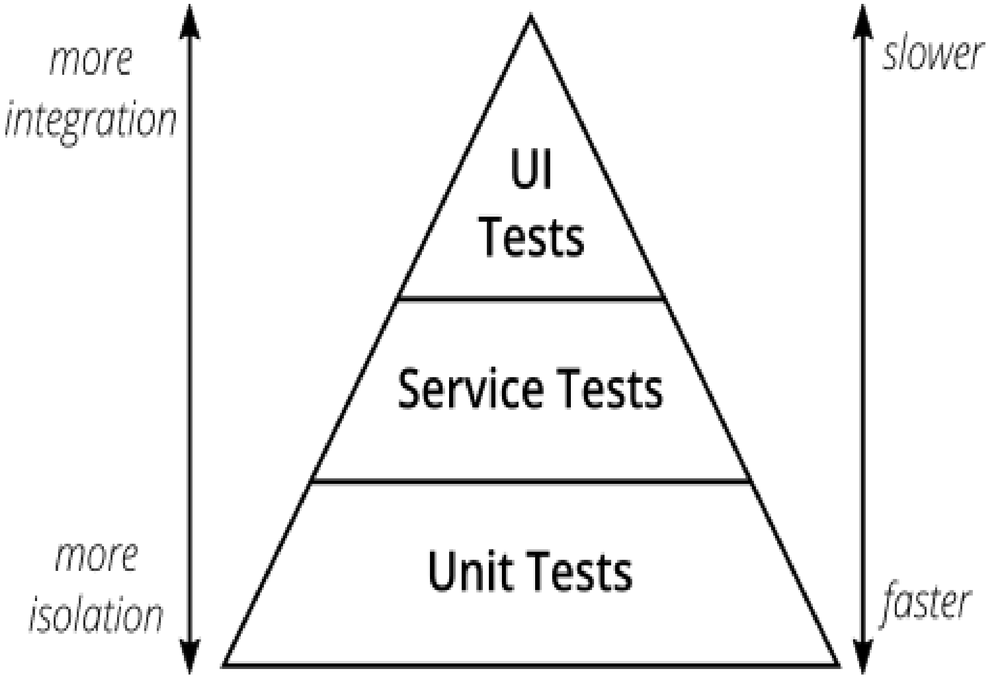
- It gives high-quality code at any time as the code is being tested continuously
- Helps in reducing software errors
- Speed up the delivery process
- Helps in increasing software development life cycle efficiency

Unit tests

A unit test is a test that exercises individual software components or methods, also known as "unit of work“.
They do not test infrastructure concerns (they use mocking objects to avoid infrastructure concerns, to avoid interacting with databases, file systems, and network resources).

Integration tests

An integration test is done to demonstrate that different pieces of the system work together.
They usually require resources like database instances and hardware to be allocated for them.
The integration tests do a more convincing job of demonstrating the system works (especially to non-programmers) than a set of unit tests can.



2

GETTING FAMILIAR WITH THE PROJECT USED FOR TESTING

Demo Project

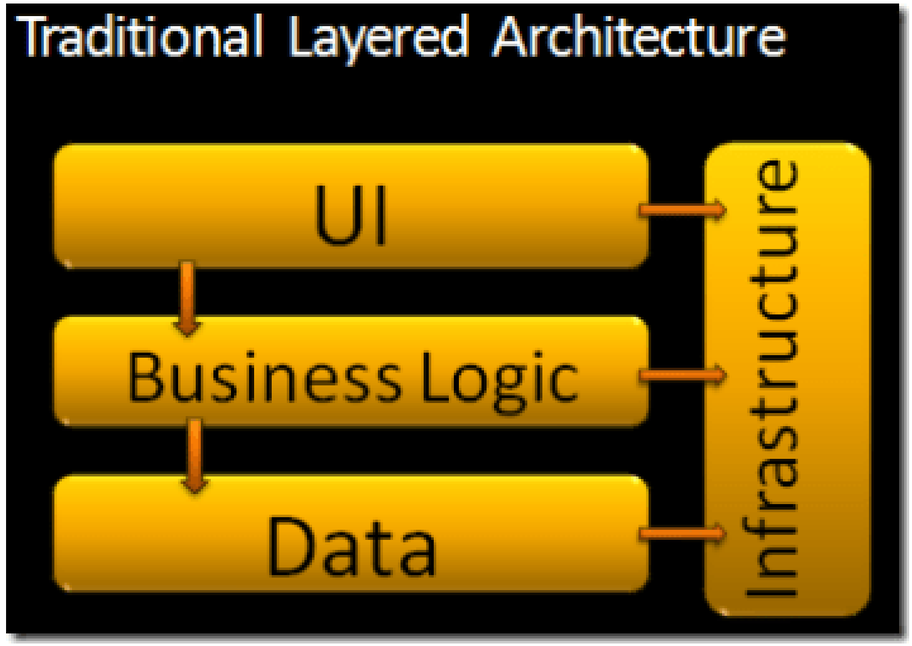
About the Demo Project

What the app does?

API that allows creating users (saving the new user in JSON file) and another API endpoint that allows getting some user's info from a file.

Architecture?

DotNet.UnitTestingFrameworks.API - "Presentation layer" Azure Functions
DotNet.UnitTestingFrameworks.BLL - "Business logic layer"
DotNet.UnitTestingFrameworks.DAL - "Data access layer



3

MSTEST, NUNIT, XUNIT

.NET Testing Frameworks

MSTest

MSTest is the default test framework that is shipped along with Visual Studio.
The initial version of MSTest (V1) was not open-source; however, MSTest V2 is open-source.
The project is hosted on GitHub.

NUnit

NUnit is an open-source testing framework ported from JUnit.
The latest version of NUnit is NUnit3 that is rewritten with many new features and has support for a wide range of .NET platforms.

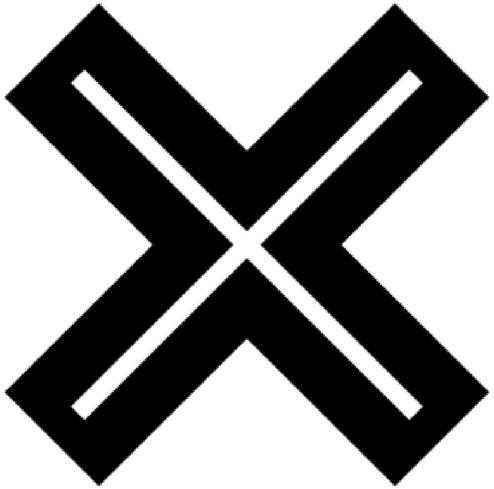
Docs: https://docs.nunit.org/articles/nunit/intro.html

XUnit

xUnit.Net is an open-source testing framework based on the .NET framework. ‘x’ stands for the programming language, e.g., JUnit, NUnit, etc.
The creators of NUnit created xUnit as they wanted to build a better framework rather than adding incremental features to the NUnit framework.

There is a big difference between xUnit and NUnit/MSTest. Xunit creates a new instance of the test class for very EACH of the test methods. Whereas Nunit/MSTest creates a new instance of the test class and then runs all of the test methods from the same instance. So if you use xUnit, you could be sure that your test methods are completely isolated.

Docs: https://xunit.net/#documentation



Side by Side Comparison

DESCRIPTION	NUNIT	MSTEST	XUNIT
Marks a test method/individual test	[Test]	[TestMethod]	[Fact]
Indicates that a class has a group of unit tests	[TestFixture]	[TestClass]	N.A
Contains the initialization code, which is triggered before every test case	[SetUp]	[TestInitialize]	Constructor
Contains the cleanup code, which is triggered after every test case	[TearDown]	[TestCleanup]	IDisposable.Dispose
Contains method that is triggered once before test cases start	[OneTimeSetUp]	[ClassInitialize]	IClassFixture<T>
Contains method that is triggered once before test cases end	[OneTimeTearDown]	[ClassCleanup]	IClassFixture<T>
Contains per-collection fixture setup and teardown	N.A	N.A	ICollectionFixture<T>
Ignores a test case	[Ignore("reason")]	[Ignore]	[Fact(Skip="reason")]
Categorize test cases or classes	[Category()]	[TestCategory("")]	[Trait("Category", "")]
Identifies a method that needs to be called before executing any test in test class/test fixture	[TestFixtureSetup]	[ClassInitialize]	N.A
Identifies a method that needs to be called after executing any test in test class/test fixture	[TestFixtureTearDown]	[ClassCleanUp]	N.A
Identifies a method that needs to be called before the execution of any tests in Test Assembly	N.A	[AssemblyInitialize]	N.A
Identifies a method that needs to be called after execution of tests in Test Assembly	N.A	[AssemblyCleanUp]	N.A

4

MOQ, NSUBSTITUTE

.NET Mocking Libraries

About Mocking

What is mocking?

Prologue: If you look up the noun mock in the dictionary you will find that one of the definitions of the word is something made as an imitation.
Mocking is primarily used in unit testing. An object under test may have dependencies on other (complex) objects. To isolate the behavior of the object you want to replace the other objects by mocks that simulate the behavior of the real objects.
In short, mocking is creating objects that simulate the behavior of real objects.

Test doubles?

The term “test double” was coined by Gerard Meszaros in the book xUnit Test Patterns.
A test double is an object that can stand in for a real object in a test, similar to how a stunt double stands in for an actor in a movie. These are sometimes all commonly referred to as “mocks”, but it's important to distinguish between the different types of test doubles since they all have different uses.
Types: Dummy, Stub, Spy, Mock, Fake

.NET Mocking Frameworks

Constrained vs Unconstrained Frameworks?

Constrained Frameworks

Class Inheritance On-Demand - Moq, NSubstitute, RhinoMocks, and FakeItEasy.
In almost all cases, these frameworks use the Castle DynamicProxy.
All of these are free, open-source.

Unconstrained Frameworks

Make Your Own Man-In-The-Middle Attack - Telerik JustMock, Typemock Isolator, and Microsoft Fakes.
These are all (to the best of my knowledge) closed-source, commercial (licensed) software.

Moq

Moq (pronounced "Mock-you" or just "Mock") is the only mocking library for .NET developed from scratch to take full advantage of .NET Linq expression trees and lambda expressions, which makes it the most productive, type-safe and refactoring-friendly mocking library available.

Docs: https://github.com/Moq/moq4/wiki/Quickstart

NSubstitute

NSubstitute is designed as a friendly substitute for .NET mocking libraries.
It is an attempt to satisfy our craving for a mocking library with a succinct syntax that helps us keep the focus on the intention of our tests, rather than on the configuration of our test doubles.

Docs: https://nsubstitute.github.io/help/getting-started/

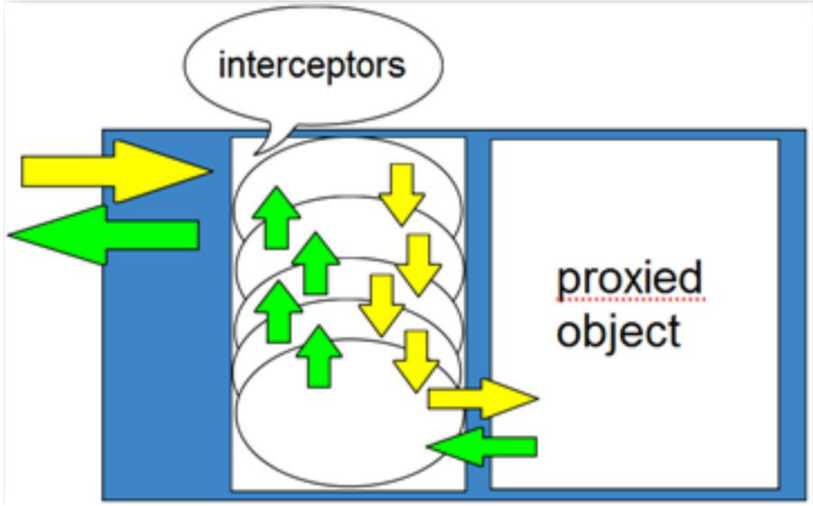
My Mock Demo

Using CreateInterfaceProxyWithoutTarget from Castle Core project (DynamicProxy more precisely).

Castle DynamicProxy is a library for generating lightweight .NET proxies on the fly at runtime. Proxy objects allow calls to members of an object to be intercepted without modifying the code of the class. Both classes and interfaces can be proxied, however only virtual members can be intercepted.

Docs: https://github.com/castleproject/Core/blob/master/docs/dynamicproxy.md

Why use proxies?
Proxy objects can assist in building a flexible application architecture because it allows functionality to be transparently added to code without modifying it. For example, a class could be proxied to add logging or security checking without making the code aware this functionality has been added.



5

AUTOFIXTURE

Improving the Arrange Part

AutoFixture

AutoFixture is an open source library for .NET designed to minimize the 'Arrange' phase of your unit tests in order to maximize maintainability.
Its primary goal is to allow developers to focus on what is being tested rather than how to setup the test scenario, by making it easier to create object graphs containing test data.

Docs: https://github.com/AutoFixture/AutoFixture/wiki/Cheat-Sheet

6

FLUENTASSERTIONS

Improving the Assert Part

FluentAssertions

Write assertions that keep you and your fellow developers out of the debugger hell. A very extensive set of extension methods that allow you to more naturally specify the expected outcome of a TDD or BDD-style unit tests.

Docs: https://fluentassertions.com/introduction


endava