Understanding NUnit Tests

Jason Roberts
.NET DEVELOPER

@robertsjason dontcodetired.com

Overview

Why write automated tests?

Regression example

Understanding the NUnit test framework

- NUnit library
- Test runner

Recognizing different testing scenarios

The logical phases of a test

Add a second test

Qualities of good tests

Why Write Automated Tests?

Help to find defects and regressions

Happier:

- End-users
- Business/management
- Developers

Reduce cost of software development

Increase long-term speed of development

Quick to execute

Free to run as often as required

Automated tests give us greater confidence that the software is working as it should

Understanding the NUnit Test Framework

NUnit library

Attributes

Assertions

Extensibility/customization

Test runner

Recognizes attributes

Execute test methods

Report test results

Test Explorer

dotnet test

```
[TestFixture]
[Test]
[Category]

[TestCase]
[Values]
[Sequential]

[SetUp]
[OneTimeSetUp]
```

- Mark a class that contains tests
- Mark a method as a test
- Organize tests into categories

- Data driven test cases
- Data driven test parameters
- How to combine test data

- Run code before each test
- Run code before first test in class

NUnit Assertions Overview

```
// Constraint Model of assertions (newer)
Assert.That(sut.Years, Is.EqualTo(1));
Assert.That(test result, constraint instance);

// Classic Model of assertions (older)
Assert.AreEqual(1, sut.Years);
Assert.NotNull(sut.Years);
Assert.Xyz(…)
```

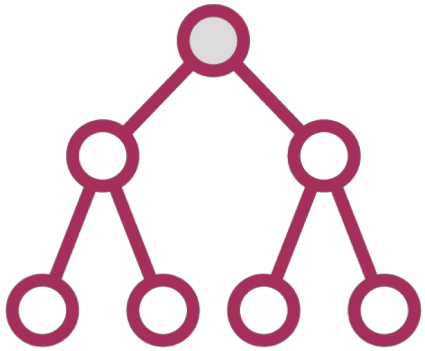
“This Classic Model is still supported but since no new features have been added to it for some time, the constraint-based model must be used in order to have full access to NUnit's capabilities.”

NUnit documentation

https://github.com/nunit/docs/wiki/Assertions

Recognizing Different Testing Scenarios

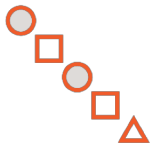
Business logic



Exercise all code branches

Bad data/input

The Logical Arrange, Act, Assert Test Phases



Arrange: set up test object(s), initialize test data, etc.

Act: call method, set property, etc.

Assert: compare returned value/end state with expected

The Arrange, Act, Assert pattern is a guide, occasionally you may not have three explicit phases.

The Logical Arrange, Act, Assert Test Phases

```
[Test]
public void ReturnTermInMonths()
{
    var sut = new LoanTerm(1);

    Assert.That(sut.ToMonths(), Is.EqualTo(12));
}
```

The Logical Arrange, Act, Assert Test Phases

```
[Test]
public void ReturnTermInMonths()
{
    // Arrange
    var sut = new LoanTerm(1);

    // Act
    var numberOfMonths = sut.ToMonths();

    // Assert
    Assert.That(numberOfMonths, Is.EqualTo(12));
}
```

Qualities of Good Tests

Fast

Repeatable

Isolated

Trustworthy

Valuable

Summary

Why write automated tests?
- Happiness, cost, and speed

Catching a regression defect

NUnit library and test runner

NUnit attributes
- [Test] [Category] [TestCase]

NUnit assertions
- Classic Model
- Constraint Model

Recognizing different testing scenarios

Arrange, Act, and Assert logical phases

Qualities of good tests

Up Next:

Asserting on Different Types of Results