Unit Testing with C#

How to Effectively Test Web and Windows Applications

Matt Dixon

Managing Director

Front Range Systems

FRONTRANGESYSTEMS
custom software solutions

About Me

- Started Front Range Systems to focus on helping organizations
- Been developing professionally since 2000
- Used .Net since Beta 2
- When I'm not at my computer, I love being in the mountains or at hockey rinks

Connect

LinkedIn: https://www.linkedin.com/in/mattdixon/

Web: http://frontrangesystems.com/

FRONTRANGESYSTEMS
custom software solutions

Overview

In this discussion you will learn:

- Why Unit Test are valuable
- What makes a good Unit Test
- How to write Unit Tests
- How to test Web Applications
- How to test Windows Applications
- We will touch on TDD, Mocks and Dependency Injection

The Value of Unit Tests

- Ensures decoupled design
- Repeatable
- Can be part of the build
- Confidence to refactor / add features
- Helps new developers get up to speed
- Address critical pieces of the application
- A good way to fix bugs
  - Write a series of failing tests that will pass once the bug is fixed

FRONTRANGESYSTEMS
custom software solutions

Good vs Bad Unit Tests

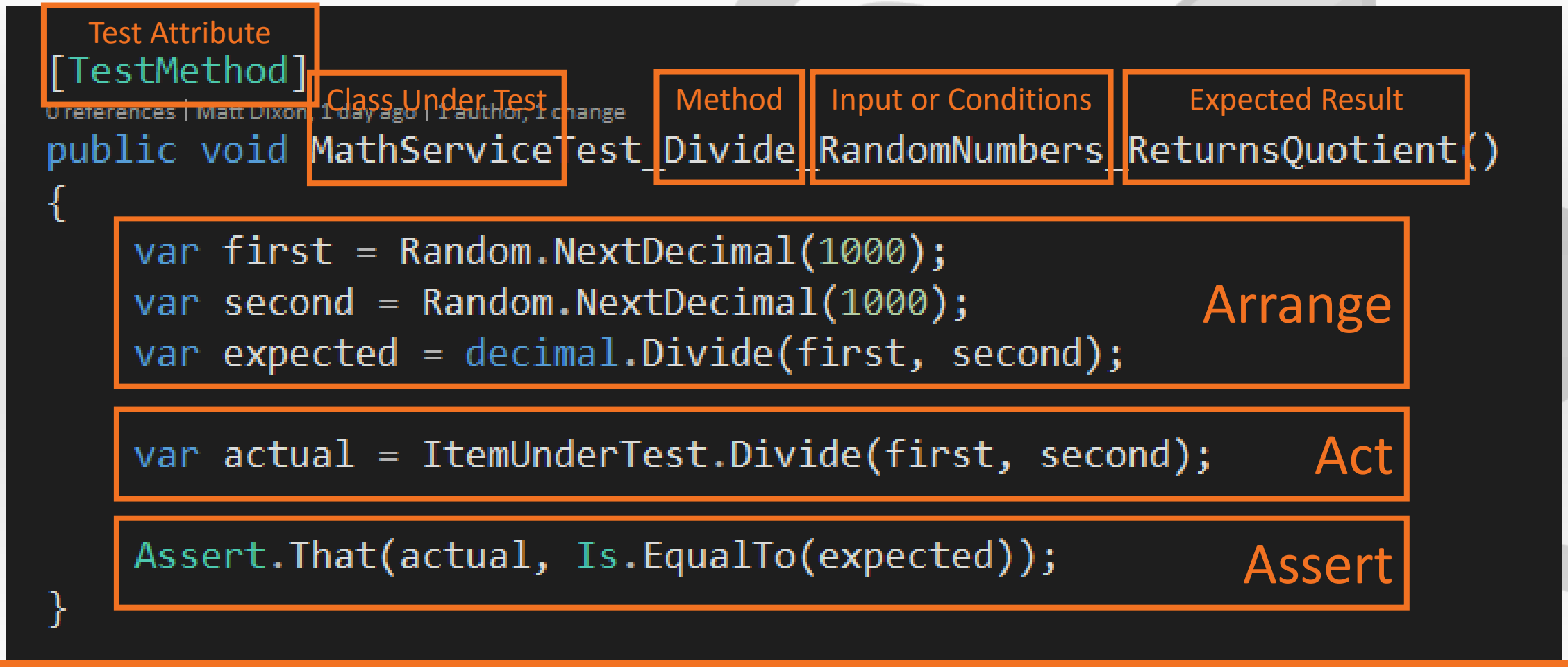
Good

- Only test one thing
- Are fast
- Idempotent
- Triggered by a check in
- Part of the Definition of Done

Bad

- Depends on external resources
  - File system
  - Web services
  - Databases
- Test more than one thing
- Not up to date
- Not run as part of a build

Anatomy of a Unit Test



FRONTRANGESYSTEMS
custom software solutions

Initialization and Cleanup

MS Test and NUnit Equivalents

MS Test		NUnit
TestClass		TestFixture
ClassInitialize		OneTimeSetUp
TestInitialize	Per Test	SetUp
TestMethod		Test
TestCleanup		TearDown
ClassCleanup		OneTimeTearDown

Viewing and Debugging Tests

Demo

Mocks and Dependency Injection

Mocks

- Removes any external dependencies
- Those dependencies can be tested separately
- Moq

Dependency Injection

- Specify mapping between class and interface
- Handles object creation
- Web
  - NinjectWebCommon.cs – line: 65
- WPF
  - AppBootstrapper.cs – line: 25

Testing Web Applications

- MVC
- Web API
- Ninject

Testing Web Applications

Demo

Testing Windows Applications

- MVVM
  - Caliburn Micro
- Test the View Model
- Test the Model
- Test Converters, Behaviors, etc.

Testing Windows Applications

Demo

TDD – Test Driven Development

- Red
  - Write a failing test
- Green
  - Change the code to fix the test
- Refactor
  - How can I make this code better?
  - Simplify
  - Think like an Architect

Resources

- Source Code and Slides
  - https://github.com/mattdixon/unit-testing
- Dependency Injection
  - Ninject – http://www.ninject.org/
- Mocking
  - Moq – https://github.com/moq/moq4
- MVVM
  - Caliburn Micro – http://caliburnmicro.com/
- Resharper
  - https://www.jetbrains.com/resharper/