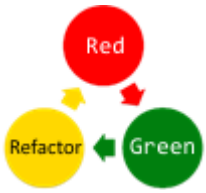
# Unit Testing in Visual Studio 2015




UTVS2015 | 2 Days

This two-day, instructor-led course provides students with the knowledge and skills to effectively use Visual Studio 2015 to design, write, and run high-quality .NET unit tests. The course focuses on the applicable features and capabilities of Visual Studio as it relates to unit testing and Test-Driven Development. This course also introduces other popular unit testing tools and techniques, and demonstrates how they integrate with Visual Studio and your development lifecycle.

## Course Objectives

At course completion, attendees will have had exposure to …

- ✓ Why unit tests are critical to software quality
- ✓ How unit tests and integration tests differ
- ✓ Popular unit testing frameworks
- ✓ The anatomy of a unit test
- ✓ The 3A pattern (Arrange, Act, Assert)
- ✓ Assertions
- ✓ Expected exceptions
- ✓ Test class inheritance
- ✓ Testing support by Visual Studio edition
- ✓ Visual Studio test projects
- ✓ Test Explorer and other related tools
- ✓ How basic and standard unit tests differ
- ✓ How and when to use traits and playlists
- ✓ How and when to use ordered tests
- ✓ Running tests and managing test results
- ✓ The purpose of Test-Driven Development
- ✓ Practicing TDD within Visual Studio
- ✓ How to effectively refactor within TDD
- ✓ Why and how to refactor legacy code
- ✓ Happy path vs. sad path testing
- ✓ Testing boundary conditions
- ✓ Organizing tests and test assemblies
- ✓ Test naming conventions (e.g. BDD)
- ✓ How to use data-driven unit tests
- ✓ Why and how to calculate code coverage
- ✓ How to use code coverage as a metric
- ✓ Using dummies, fakes, stubs, and mocks
- ✓ Why and how to use Microsoft Fakes
- ✓ Why and how to use Rhino Mocks
- ✓ Why and how to use Microsoft IntelliTest

## Who Should Attend

This course is intended for current software development professionals who are involved with building high-quality .NET applications. Students will use Visual Studio while learning how to design, write, and run C# unit tests. They will also learn many relevant practices and techniques, such as TDD, refactoring, and how to test difficult code using doubles.

## Prerequisites

Before attending this course, a student should have experience or familiarity with:

- ✓ The Visual C# language
- ✓ Visual Studio 2012, 2013, or 2015
- ✓ Writing, debugging, and maintaining code
- ✓ Application Lifecycle Management basics
- ✓ Building a high-quality software product
- ✓ Their organization's development lifecycle

# Unit Testing in Visual Studio 2015

UTVS2015 | 2 Days

## Modules

**Module 1: Unit Testing in .NET**
This module introduces the concepts of unit testing and how it is supported by various unit testing frameworks and tools for .NET, including MSTest and NUnit. The anatomy of a unit test is also detailed in this module.

- ✓ The role of the developer
- ✓ Unit tests explained
- ✓ .NET unit testing frameworks
- ✓ MSTest, NUnit, xUnit.net, and others
- ✓ The anatomy of a unit test
- ✓ Writing your first unit test

**Module 2: Unit Testing in Visual Studio**
This module introduces Visual Studio test projects, Test Explorer and other testing windows, and the functionality for effectively writing and running unit tests and managing test results.

- ✓ Testing support in Visual Studio
- ✓ Test projects
- ✓ Test Explorer and other windows
- ✓ Unit testing in Visual Studio
- ✓ Running tests
- ✓ Managing test results
- ✓ Managing a large number of tests

**Module 3: Test-Driven Development (TDD)**
This module introduces Test Driven Development (TDD) and the business case for why you should practice it. Refactoring as well as a discussion of how to work with legacy code are also part of this module.

- ✓ TDD overview and benefits
- ✓ Practicing TDD within Visual Studio
- ✓ Refactoring
- ✓ Using CodeLens to support TDD and refactoring
- ✓ Working with legacy code

**Module 4: Writing Good Unit Tests**
Just knowing how to write unit tests and being disciplined in TDD is not enough. This module introduces several other practices for ensuring that you write high-quality unit tests that cover more than just the happy path.

- ✓ Know your code
- ✓ Path testing (i.e. sad path)
- ✓ Right BICEP
- ✓ Testing for expected exceptions
- ✓ Maintaining high-quality test code
- ✓ Unit test naming conventions (e.g. BDD)
- ✓ Organizing unit tests

**Module 5: Advanced Unit Testing in Visual Studio**
This module examines additional unit testing features found in Visual Studio, including code coverage, data-driven unit tests, and continuous testing tools.

- ✓ Code coverage
- ✓ Using code coverage as a metric
- ✓ Data-driven unit tests
- ✓ Continuous testing in Visual Studio
- ✓ Concurrent testing using Ncrunch

**Module 6: Testing Difficult Code**
This module introduces some tools and techniques for testing difficult code, such as code that can't be tested without being hosted in another environment (e.g. ASP.NET, SharePoint, etc.)

- ✓ The need to isolate code under test
- ✓ Doubles (dummies, stubs, fakes, and mocks)
- ✓ Microsoft Fakes framework (stubs and shims)
- ✓ Mocking frameworks (Rhino Mocks)
- ✓ Profiling slow running unit tests
- ✓ Using IntelliTest with legacy code

## Course Designer

This course was designed by Richard Hundhausen, a Visual Studio ALM MVP, Microsoft Regional Director, Professional Scrum Trainer, co-creator of the Scaled Professional Scrum framework, and an experienced software developer and trainer. To see other developer courses, visit www.accentient.com.

Accentient