

# UNIT TESTING IN VISUAL STUDIO 2010



UTVS2010 | 2 Days

---

## INTRODUCTION

This two-day, instructor-led course provides students with the knowledge and skills to effectively use Visual Studio 2010 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.

## AUDIENCE

This course is intended for current software development professionals, including architects, developers, and white-box testers who are involved with building .NET applications. The student will learn and receive hands-on experience using Visual Studio to design, write, and run high-quality .NET unit tests and learn the practice of Test-Driven Development.

## PREREQUISITES



Before attending this course, students should have:

- Experience creating, maintaining, and debugging software
- Experience with the Visual C# language
- Experience with Visual Studio 2005, 2008, or (preferably) 2010
- Familiarity with their organization's development lifecycle

## AT COURSE COMPLETION

This course teaches students how to effectively design, write, and run high-quality unit tests using Visual Studio 2010. At course completion, attendees will have had exposure to these topics:

- 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 and expected exceptions
- Test class inheritance
- Testing private methods
- Visual Studio test projects
- Visual Studio test-related windows and tools
- How basic and standard unit tests differ
- How to unit test without access to source code
- How and when to use categories and lists
- How and when to use ordered tests
- Running tests and managing test results
- The importance of Test-Driven Development
- Implementing TDD in Visual Studio 2010
- 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
- Behavior-Driven Development principles
- How to use data-driven unit tests
- Why and how to calculate code coverage
- Why and how to use test impact analysis
- Team Foundation Server support for testing
- Why and how to use continuous integration
- Using dummies, fakes, stubs, and mocks
- Why and how to use Rhino Mocks (3<sup>rd</sup> party)
- Why and how to use Microsoft Pex
- Why and how to use Microsoft Moles

## **COURSE OUTLINE**

### **MODULE 1: UNIT TESTING IN .NET**

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

#### Lessons

- The role of the developer
- Unit tests explained
- .NET unit testing frameworks
- MSTest.exe
- Writing unit tests

#### Lab

- Setup the learning environment
- Create and run a unit test
- Create and run an integration test
- Migrate NUnit tests to MSTest (optional)

### **MODULE 2: UNIT TESTING IN VISUAL STUDIO 2010**

This module introduces Visual Studio test projects, the testing windows, and the functionality for effectively writing and running unit tests and managing the test results.

#### Lessons

- Testing support in Visual Studio 2010
- Test projects
- Testing windows
- Unit testing in Visual Studio 2010
- Managing a large number of tests
- Test categories vs. test lists
- Running tests
- Managing test results

#### Lab

- Review existing codebase and tests
- Execute tests in various ways
- Manage tests using test categories
- Execute tests in sequence using an ordered test
- Manage tests using test lists
- Managing test results

## **MODULE 3: TEST-DRIVEN DEVELOPMENT**

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

### Lessons

- What is TDD
- TDD benefits
- Common objections to TDD
- Implementing TDD in Visual Studio 2010
- Refactoring
- Legacy Code

### Lab

- The Bowling Kata
- TDD and Refactoring

## **MODULE 4: WRITING GOOD UNIT TESTS**

Just knowing how to write unit tests and being disciplined in TDD is not enough. This module will demonstrate several other practices for ensuring that you write high-quality unit tests.

### Lessons

- Know your code
- Path testing
- The sad path
- Right BICEP
- Testing for expected exceptions
- Maintaining high-quality test code
- Behavior-Driven Development (BDD)
- BDD naming conventions
- Organizing test libraries (assemblies)

### Lab

- Handle bad input values
- Refactor unit tests
- Add a sad path test
- The challenge!

## **MODULE 5: LEVERAGING VISUAL STUDIO**

This module examines additional unit testing features found in Visual Studio, including code coverage, data-driven unit tests, and test impact analysis. In addition, many unit-testing specific features in Team Foundation Server and Team Foundation Build will be part of the discussion.

### Lessons

- Code coverage
- Using code coverage as a metric
- Data-driven unit tests
- Test impact analysis
- Team Foundation Server
- Testing check-in policy
- Team Foundation Build
- Build verification tests (BVTs)
- Gated check-ins
- Continuous Integration (CI)

### Lab

- Use a data-driven unit test
- Calculate code coverage
- Use test impact analysis

Note: some of the topics and hands-on exercises assume Visual Studio 2010 Premium or Ultimate edition as well as Team Foundation Server.

## **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 (i.e.ASP.NET, SharePoint, etc.)

### Lessons

- What are doubles
- Dummies, stubs, fakes, and mocks
- Mocking Frameworks
- Rhino Mocks
- Microsoft Pex
- Microsoft Moles

### Lab

- Explore Rhino Mocks
- Explore Microsoft Pex (optional)
- Explore Microsoft Moles (optional)

### **Course Designer**

This course was designed by Richard Hundhausen of Accentient, Inc. Richard is a Visual Studio ALM MVP and Microsoft Regional Director, as well as an experienced developer and trainer.

For more information, visit [www.accentient.com](http://www.accentient.com)