NUnitForVS Internals

Contents

[Introduction 3](#_Toc221696592)

[NUnitForVS Architecture 4](#_Toc221696593)

[Test Elements 5](#_Toc221696594)

[NUnit Test Adapter 6](#_Toc221696595)

[NUnit Handling 8](#_Toc221696596)

[Testing NUnitForVS with Unit Tests 9](#_Toc221696597)

# Introduction

Professional, Developer, and Tester Editions of Visual Studio are capable of running unit tests. Unfortunately they do not support NUnit tests natively. Converting NUnit tests to Visual Studio unit tests takes resources and is sometimes not even possible. That's why NUnit for Visual Studio (NUnitForVS) was created. It allows Visual Studio to see and run NUnit tests without changing any NUnit test code.



Figure NUnitForVS in Visual Studio and MStest

NUnitForVS add-in for Visual Studio Team System will allow you to run NUnit tests just like Visual Studio Unit tests integrated in Visual Studio. It makes Visual Studio to see NUnit test as MStest. That means everything you can do with Visual Studio Unit tests you can also do with NUnit tests. This includes publishing the results or code coverage in TFS.

# NUnitForVS Architecture



Figure Architecture NUnitForVS

The NUnitForVS add-in interacts with: Visual Studio Team System, MStest, NUnit, and your NUnit tests. As shown in Figure 2 Architecture NUnitForVS, NUnitForVS is therefore divided into four distinct components:

1. NUnit Test Adapter; Responsible for the interactions with MStest
2. NUnitForVS VSUI; Extends the User Interface in Visual Studio System for NUnit Tests
3. NUnit Handling; Handling and execution of the NUnit Tests
4. NUnitForVS Common; Common Items, which are re-used in most other components

# Test Elements



Figure Test Elements mapping on a NUnit TestFixture

In the MStest Framework there is the notion of a Test Element. Each individual Test Element within the MStest framework information about a test method. Since the NUnit equivalence of a test method is a test, Test Elements within the NUnitForVS adapter represent individual tests as can be seen in Figure 3 Test Elements mapping on a NUnit Test.

Figure Differences between MStest and NUnit test execution order

While the Test Element map directly on a NUnit test, its execution does not as can be seen in Figure 4 Differences between MStest and NUnit test execution order. Because legacy NUnit tests can have dependencies on this order of execution the NUnitForVS changes this order.

# NUnit Test Adapter



Figure Interaction between MStest and NUnit Test Adapter

NUnitForVS provides a custom NUnit implementation of the Test Element. Within the MStest framework these Test Elements are instantiated by the Test Host as can be seen in Figure 5 Interaction between MStest and NUnit Test Adapter. The Test elements are then handed over to the Agent Object. This Agent Object is the internal implementation of the MStest Test runner. Depending on the executing Test Element type a different MStest Adapter used. For our NUnit Test Element type the NUnit Test Adapter is created by the Agent Object.

The Agent Object creates an Agent Execution for each of the Test Elements and puts the Agent Executions into an Execution List. Each Agent Execution processes a Test Element with use of the Test Adapter and provides the Result Sink to contain the test execution result.

Figure NUnit Test Adapter Execution Order Shuffle

As written before the execution within the MStest framework is different than the execution within the NUnit framework. Therefore an execution order shuffle takes place within the NUnit Test Adapter as can be seen in Figure 6 NUnit Test Adapter Execution Order Shuffle.

When the first Test Element of a class is provided to the NUnit Test Adapter, the NUnit Test Adapter processes all Test Elements for that class and caches the result. The trailing Test Elements of a class are then returned their cached result.

# NUnit Handling



Figure Handling of the NUnit Test

To process the Test Elements, the NUnit Test Adapter creates an application domain for each test class. Within this application domain a NUnit Runner is created which creates NUnit Executors for each Test Element. These NUnit Executors are used to execute the individual NUnit test methods.

# Testing NUnitForVS with Unit Tests



Figure Vocabulary within unit tests NUnitForVS

As with all software projects it is wise to test components with unit tests. The Unit Tests which test NUnitForVS are implemented in MStest. However since NUnitForVS by itself includes references to two Unit Tests Frameworks, the unit tests may appear a bit daunting. To clarify the unit tests performed on NUnitForVS the following namespace aliases are used as shown in Figure 8 Vocabulary within unit tests NUnitForVS:

1. MSTestSide; used for MStest and visual studio components
2. NUnitSide; used for NUnit components
3. Target; Unit tests implemented in NUnit which are used as a target to test NUnitForVS
4. The Default namespace; The unit tests which test NUnitForVS