Test Framework Overview

Description

Prerequisite

Execute Before (setup) / After (TearDown) every test methods

[Setup]
[Teardown]

```
namespace NUnit.Tests
{
  using System;
  using NUnit.Framework;

  [TestFixture]
  public class SuccessTests
  {
    [SetUp] public void Init()
    { /* ... */ }

    [TearDown] public void Cleanup()
    { /* ... */ }

    [Test] public void Add()
    { /* ... */ }
  }
}
```

Execute once before (Setup) / After (TearDown) any of the fixtures (class)

[Setup Fixture]
[OnTime Setup]
[OnTime TearDown]

```
namespace NUnit.Tests
{
  [SetUpFixture]
  public class MySetUpClass
  {
        [OneTimeSetUp]
        public void RunBeforeAnyTests()
        {
          // ...
        }

        [OneTimeTearDown]
        public void RunAfterAnyTests()
        {
          // ...
        }
  }
}
```

Note: Prior to NUnit 3.0, SetUpFixture used the SetUp and TearDown attributes rather than OneTimeSetUp and OneTimeTearDown. The older attributes *([TestFixtureSetup], [TestFixtureTearDown])* are no longer supported in SetUpFixtures in NUnit 3.0 and later.

NUNIT 3	TestFixture	SetUpFixture
OneTimeSetUp	Supported	Supported
OneTimeTearD	Supported	Supported
TestFixtureSetU	Deprecated	Not Allowed
TestFixtureTear	Deprecated	Not Allowed
SetUp	Supported	Not Allowed
TearDown	Supported	Not Allowed

Test Framework Overview

Description			
Test Annotation	description	[Test, Description(‘Run the Valid user’)]	
	Running Order	[Test, order(1)]	
	Author details	[Test, Author(‘2342342’, ‘emailid’)] [Test, Author =’2342342’)]	
	Ignore test / Ignore until	[Test] [Ignore(‘Issue – 1234’)] [Ignore(‘Issue – 1234’, until = ‘2019.09.31 12:00:00z’’)]	
	Ignore	[Test, Explicit] (Ignore unless explicitly called)	
	Timeout	[Test, Timeout(2000)]	
	Grouping Key, value	[Test, Property(‘Location’,’BLR’)] [Test, Property(‘Severity,’Low’)] - (Grouping Key, value (Selecting Test, reporting)	
	Grouping	[Test] [Category = ‘Regression’]	
	Test Fixture description	[TestFixture, Description(‘Run the Valid user’)]	
	Ignoring Test fixture	[TestFixture, Explicit]	
	Test fixture author details	[TestFixture, Author(‘2342342’, ‘emailid’)] [TestFixture, Author =’2342342’)]	
	Test future level grouping	[TestFixture] [Category = ‘Regression’]	

Test Framework Overview

Description

Parameterization

Test Fixture - parameterization

```
[TestFixtureSource(typeof(MyFixtureData),
"FixtureParms")]
public class ParameterizedTestFixture {
    private string eq1;
    private string eq2;
    private string neq;

public ParameterizedTestFixture(string eq1,
string eq2, string neq){
        this.eq1 = eq1;
        this.eq2 = eq2;
        this.neq = neq;    }

public ParameterizedTestFixture(string eq1,
string eq2) : this(eq1, eq2, null) { }

public ParameterizedTestFixture(int eq1,
int eq2, int neq) {
        this.eq1 = eq1.ToString();
        this.eq2 = eq2.ToString();
        this.neq = neq.ToString(); }
```

```
[Test]
public void TestInequality()   {
Assert.AreNotEqual(eq1, neq);}
}

public class MyFixtureData
{
public static IEnumerable FixtureParms{
 get{
yield return new
TestFixtureData("hello", "hello",
"goodbye");
yield return new TestFixtureData("zip",
"zip");
yield return new TestFixtureData(42,
42, 99);
        }
    }
}
```

Parameterization

Test case parameterization

```
[TestCase(12, 3, 4)]
[TestCase(12, 2, 6)]
[TestCase(12, 4, 3)]
public void DivideTest(int n, int d, int q)
{
    Assert.AreEqual(q, n / d);
}
```

```
[TestCase(12, 3, ExpectedResult=4)]
[TestCase(12, 2, ExpectedResult=6)]
[TestCase(12, 4, ExpectedResult=3)]
public int DivideTest(int n, int d)
{
    return n / d;
}
```

Test Framework Overview nunit

Description | nunit

```
public class MyTestClass
{
    [TestCaseSource(typeof(AnotherClass), "DivideCases")]
    public void DivideTest(int n, int d, int q)
    {
        Assert.AreEqual(q, n / d);
    }
}

class AnotherClass
{
    static object[] DivideCases =
    {
        new object[] { 12, 3, 4 },
        new object[] { 12, 2, 6 },
        new object[] { 12, 4, 3 }
    };
}
```

```
public class MyTestClass
{
    [TestCaseSource(typeof(DivideCases))]
    public void DivideTest(int n, int d, int q)
    {
        Assert.AreEqual(q, n / d);
    }
}

class DivideCases : IEnumerable
{
    public IEnumerator GetEnumerator()
    {
        yield return new object[] { 12, 3, 4 };
        yield return new object[] { 12, 2, 6 };
        yield return new object[] { 12, 4, 3 };
    }
}
```

Test Parameterization

Random

```
[Test]
public void MyTest(
    [Values(1, 2, 3)] int x,
    [Random(-1.0, 1.0, 5)] double d)
{
    ...
}
```

The following test will be executed fifteen times, three times for each value of x, each combined with 5 random doubles from -1.0 to +1.0.

Test Framework Overview

Description			

		Range
		`[Test]`
		`public void MyTest(`
		`    [Values(1, 2, 3)] int x,`
		`    [Range(0.2, 0.6, 0.2)] double d)`
		`{`
		`    ...`
		`}`
			The MyTest method is called nine times, as follows:
			`MyTest(1, 0.2)`
			`MyTest(1, 0.4)`
			`MyTest(1, 0.6)`
			`MyTest(2, 0.2)`
			`MyTest(2, 0.4)`
			`MyTest(2, 0.6)`
			`MyTest(3, 0.2)`
			`MyTest(3, 0.4)`
			`MyTest(3, 0.6)`

		Value
		`[Test]`
		`public void MyTest([Values(1, 2, 3)] int x,`
		`[Values("A", "B")] string s)`
		`{`
		`    ...`
		`}`
			The above test will be executed six times, as follows:
			`MyTest(1, "A")`
			`MyTest(1, "B")`
			`MyTest(2, "A")`
			`MyTest(2, "B")`
			`MyTest(3, "A")`
			`MyTest(3, "B")`

		[Test, Pairwise]
		`[Test, Pairwise]`
		`public void MyTest(`
		`    [Values("a", "b", "c")] string a,`
		`    [Values("+", "-")] string b,`
		`    [Values("x", "y")] string c)`
		`{`
		`    Console.WriteLine("{0} {1} {2}", a, b, c);`
		`}`
			For this test, NUnit currently calls the method six times, producing the following output:
			`a + y`
			`a - x`
			`b - y`
			`b + x`
			`c - x`
			`c + y`

		[Test, Sequential]
		`[Test, Sequential]`
		`public void MyTest(`
		`    [Values(1, 2, 3)] int x,`
		`    [Values("A", "B")] string s)`
		`{`
		`    ...`
		`}`
			MyTest is called three times, as follows:
			`MyTest(1, "A")`
			`MyTest(2, "B")`
			`MyTest(3, null)`

		[Test, combinatorial]
		`[Test, Combinatorial]`
		`public void MyTest(`
		`    [Values(1, 2, 3)] int x,`
		`    [Values("A", "B")] string s)`
		`{`
		`    ...`
		`}`
			MyTest is called six times, as follows:
			`MyTest(1, "A")`
			`MyTest(1, "B")`
			`MyTest(2, "A")`
			`MyTest(2, "B")`
			`MyTest(3, "A")`
			`MyTest(3, "B")`

Asser	Assertion to validate the	**Assert. That (Actual, expected)** `Assert.That(2+2, Is.EqualTo(4));`	Is | Has | Contains | Does | Throws

Test Framework Overview

Description

actual with expected condition.

Assert. AreEqual (Actual, expected)

```
Assert.AreEqual(4, 2+2);
```

Assert.Multiple

```
[Test]
public void ComplexNumberTest()
{
    ComplexNumber result = SomeCalculation();

    Assert.Multiple(() =>
    {
        Assert.AreEqual(5.2, result.RealPart,
"Real part");
        Assert.AreEqual(3.9,
result.ImaginaryPart, "Imaginary part");
    });
}
```

Assert. AreNotEqual (Actual, expected)

Assert. AreNotSame(Actual, expected)

Assert. AreSame(Actual, expected)

```
Assert.That(
(iarray, Is.All.Not.Null);
(iarray, Has.All.GreaterThan(0))
(iarray, Does.Contain(3))
(7, Is.GreaterThan(3));
(42, Is.Positive);
(-5, Is.Negative);
(7, Is.GreaterThanOrEqualTo(3));
(3, Is.LessThan(7));
(42, Is.InRange(1, 100));
(anObject, Is.Null);
(anObject, Is.Not.Null)
(aString, Is.Empty);
(condition, Is.True)
(array, Has.Exactly(5).Items)
(emp.IsSeniorCitizen(),
Throws.Exception);
```

Execution

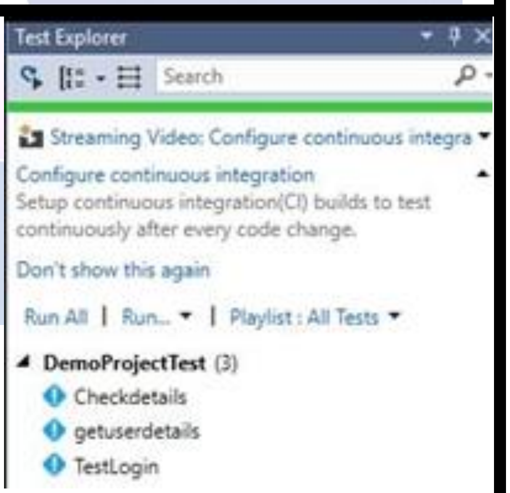
Execute Nunit tests

Visual Studio Test Explorer > Windows > Test Explorer>

Select and run test from List of test case listed in Test explorer panel.

NUNIT3-CONSOLE [inputfiles] [options]

```
nunit3-console.exe path/to/test/assembly.dll
[Options]
➢  --test=NAMES
➢  --testlist=FILE
The name (or path) of a FILE containing a list of
tests to run or explore, one per line.
➢  --timeout=MILLISECONDS
➢  --debug
```



Test Explorer

Search

Streaming Video: Configure continuous integra

Configure continuous integration

Setup continuous integration(CI) builds to test continuously after every code change.

Don't show this again

Run All | Run... | Playlist : All Tests

DemoProjectTest (3)
- Checkdetails
- getuserdetails
- TestLogin

Parallel

Execute Tests in parallel

ParallelScope.self
ParallelScope.children
ParallelScope.fixtures
ParallelScope.all

NonParallelizableAttribute

This Attribute is used to indicate that the test as well as its descendants may not be run in parallel with other tests. Although NonParallelizable] is completely equivalent to [Parallelizable(ParallelScope.None)], we recommend that you use the former for clarity.

Test Framework Overview unit

Description	unit

```
[TestFixture]
[Parallelizable(ParallelScope.All)]
public class MyClassTests {
 [Test]
public void MyParallelTest() {
    }
}
```

For this we can either add the line

```
[assembly:
Parallelizable(ParallelScope.Fixtures)]
```

to the AssemblyInfo.cs file found under Properties in the project directory.
This way we add parallel execution at fixture level for the entire assembly

Value	Meaning	Valid On
ParallelScope.Self	the test itself may be run in parallel with other tests	Classes, Methods
ParallelScope.Children	child tests may be run in parallel with one another	Assembly, Classes
ParallelScope.Fixtures	fixtures may be run in parallel with one another	Assembly, Classes
ParallelScope.All	the test and its descendants may be run in parallel with others at the same level	Classes, Methods

Test Framework Overview nunit

Description		nunit
Listeners		**ITestEventListener**

```
ITestEventListener
[Extension(EngineVersion="3.4")]
public class MyEventListener : ITestEventListener
{
    ...
}
[TypeExtensionPoint(
    Description = "Allows an extension to process progress reports and other events
from the test.")]
public interface ITestEventListener
{
    /// <summary>
    /// Handle a progress report or other event.
    /// </summary>
    /// <param name="report">An XML progress report.</param>
    void OnTestEvent(string report);
}
```

The argument to OnTestEvent is an XML-formatted string, with a different top-level element for each potential event.

Start of run - <start-run...>
End of run - <test-run...>
Start of a test suite - <start-suite...>
End of a test suite - <test-suite...>
Start of a test case - <start-test...>
End of a test case - <test-case...>