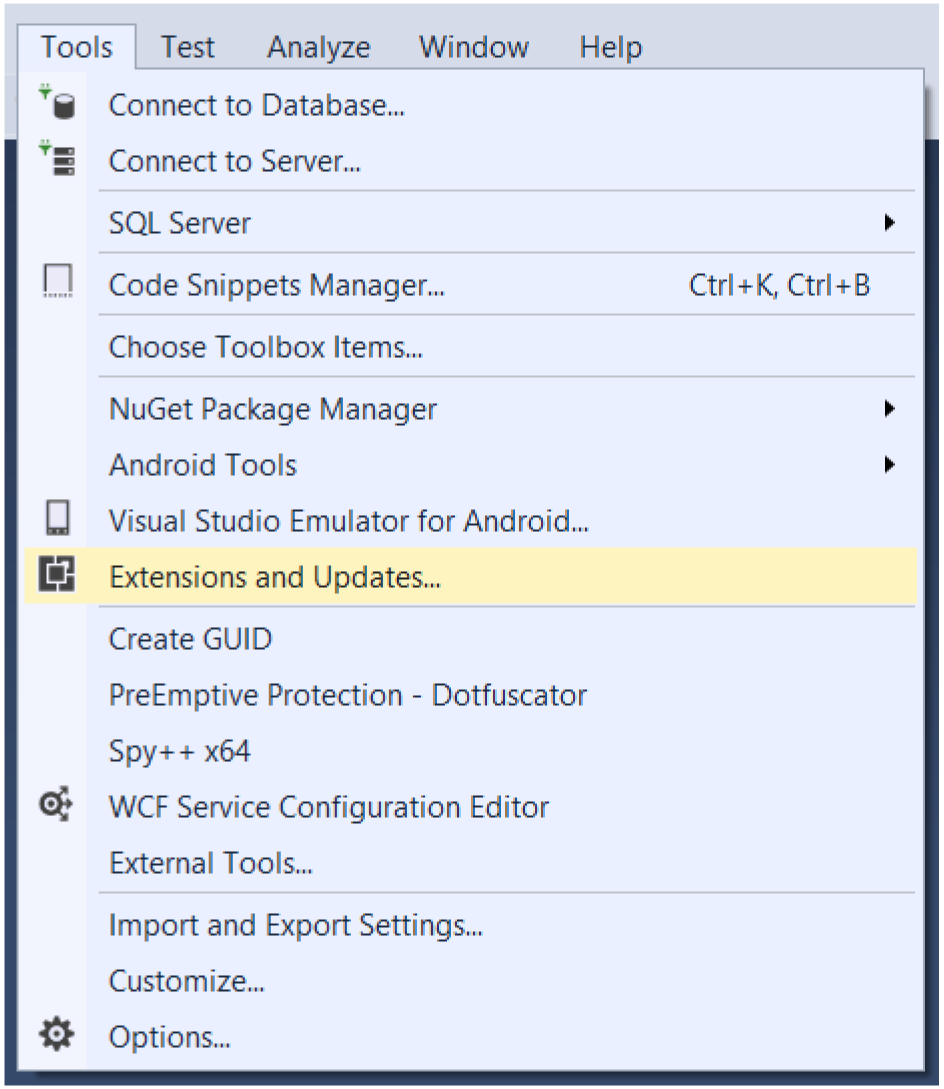
NUnit

ALL CODE IS GULTY ★ UNTIL PROVEN INNOCENT ★ T.D.D

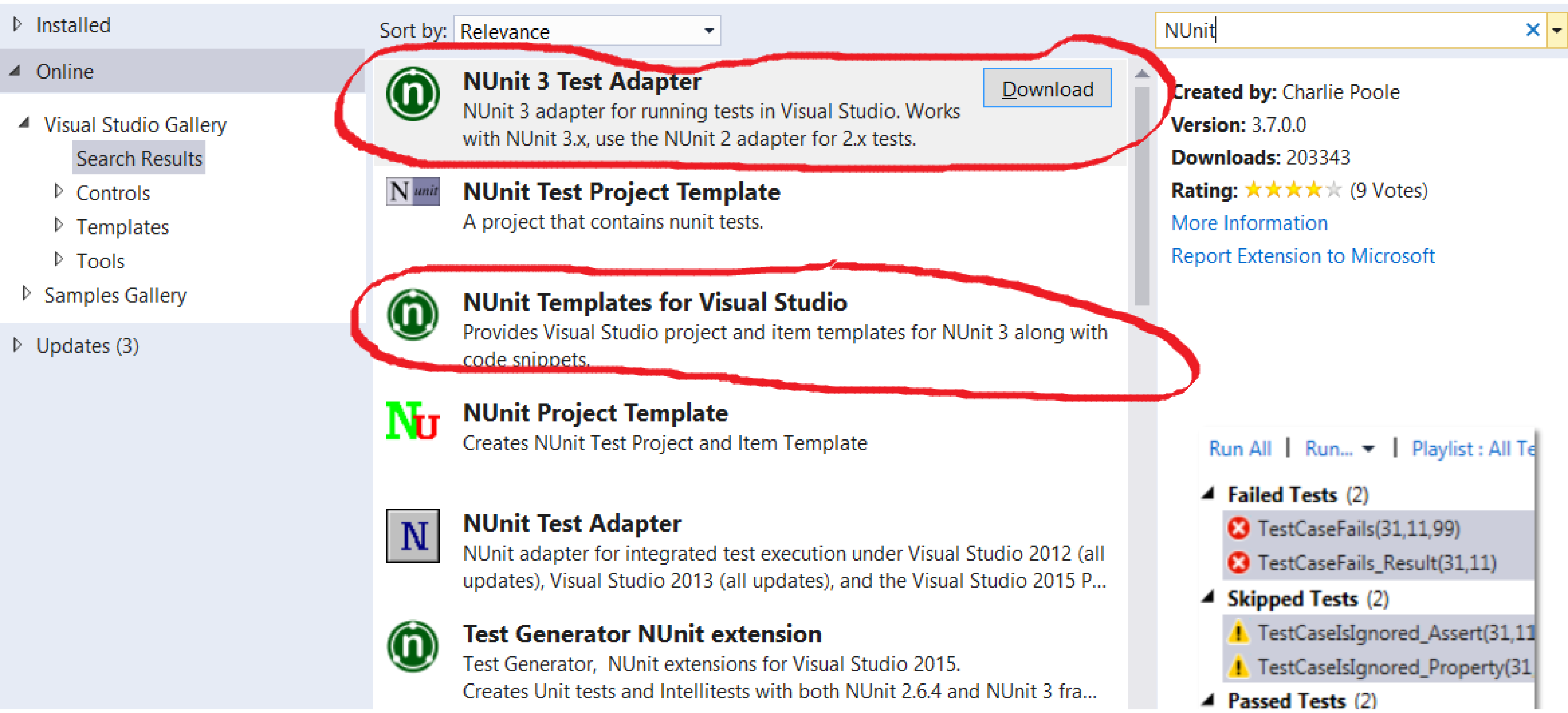
1) NUnit
2) Classic Model vs. Constraint Model
3) Asserts
4) Attributes

В свежих версиях VS и NUnit:



Tools
Test
Analyze
Window
Help
Connect to Database...
Connect to Server...
SQL Server
Code Snippets Manager... Ctrl+K, Ctrl+B
Choose Toolbox Items...
NuGet Package Manager
Android Tools
Visual Studio Emulator for Android...
Extensions and Updates...
Create GUID
PreEmptive Protection - Dotfuscator
Spy++ x64
WCF Service Configuration Editor
External Tools...
Import and Export Settings...
Customize...
Options...

В свежих версиях VS и NUnit:



Installed
Online
Visual Studio Gallery
Search Results
Controls
Templates
Tools
Samples Gallery
Updates (3)

Sort by: Relevance

NUnit

NUnit 3 Test Adapter
NUnit 3 adapter for running tests in Visual Studio. Works with NUnit 3.x, use the NUnit 2 adapter for 2.x tests.

Download

NUnit Test Project Template
A project that contains nunit tests.

NUnit Templates for Visual Studio
Provides Visual Studio project and item templates for NUnit 3 along with code snippets.

NUnit Project Template
Creates NUnit Test Project and Item Template

NUnit Test Adapter
NUnit adapter for integrated test execution under Visual Studio 2012 (all updates), Visual Studio 2013 (all updates), and the Visual Studio 2015 P...

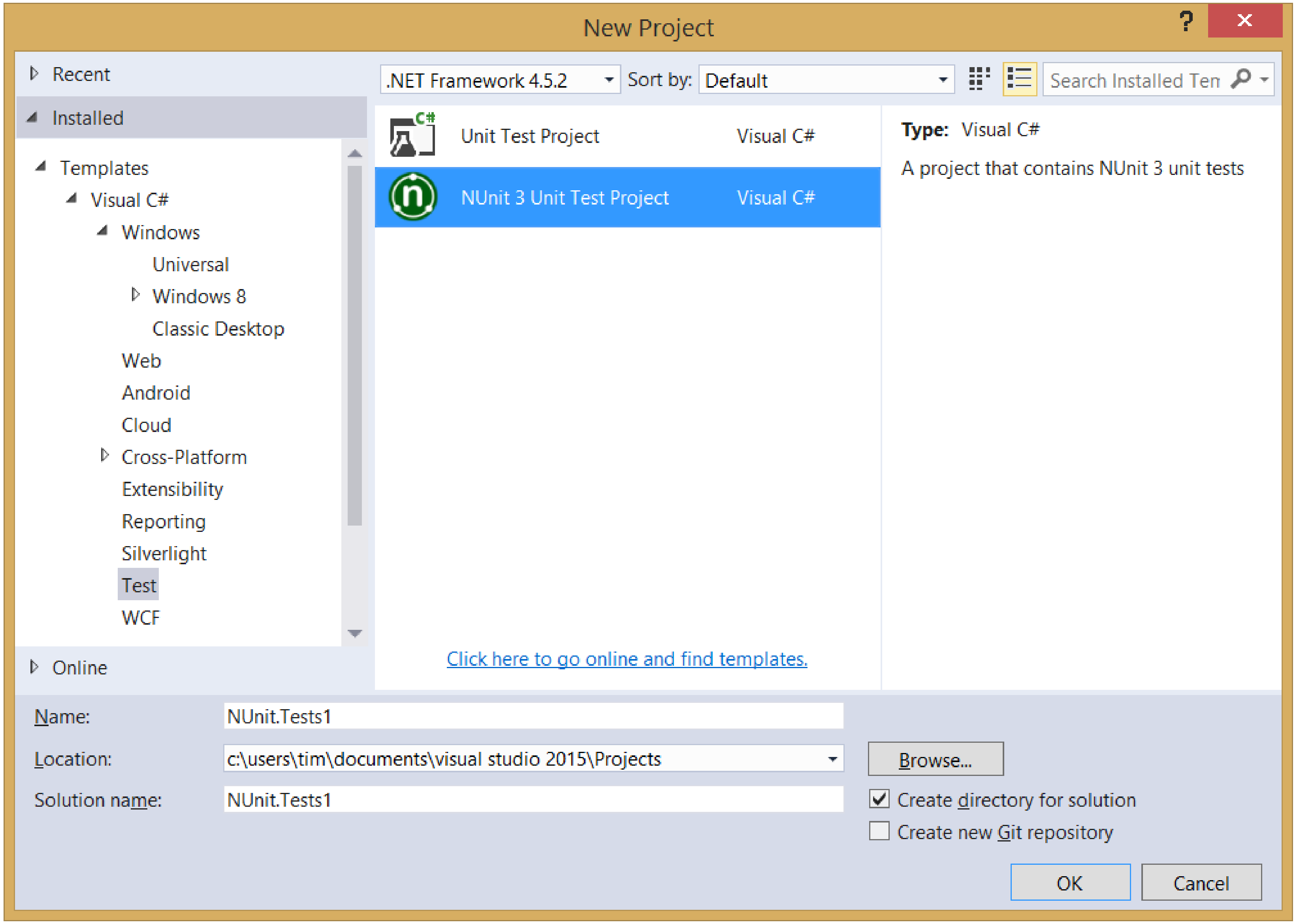
Test Generator NUnit extension
Test Generator, NUnit extensions for Visual Studio 2015.
Creates Unit tests and Intellitests with both NUnit 2.6.4 and NUnit 3 fra...

Created by: Charlie Poole
Version: 3.7.0.0
Downloads: 203343
Rating: ★★★★☆ (9 Votes)
More Information
Report Extension to Microsoft

Run All | Run... | Playlist : All Te

Failed Tests (2)
TestCaseFails(31,11,99)
TestCaseFails_Result(31,11)
Skipped Tests (2)
TestCaseIsIgnored_Assert(31,11
TestCaseIsIgnored_Property(31
Passed Tests (2)

В свежих версиях VS и NUnit:


New Project

Recent

Installed

Templates
Visual C#
Windows
Universal
Windows 8
Classic Desktop
Web
Android
Cloud
Cross-Platform
Extensibility
Reporting
Silverlight
Test
WCF

Online

.NET Framework 4.5.2 · Sort by: Default · Search Installed Ten

Unit Test Project	Visual C#
NUnit 3 Unit Test Project	Visual C#

Click here to go online and find templates.

Type: Visual C#

A project that contains NUnit 3 unit tests

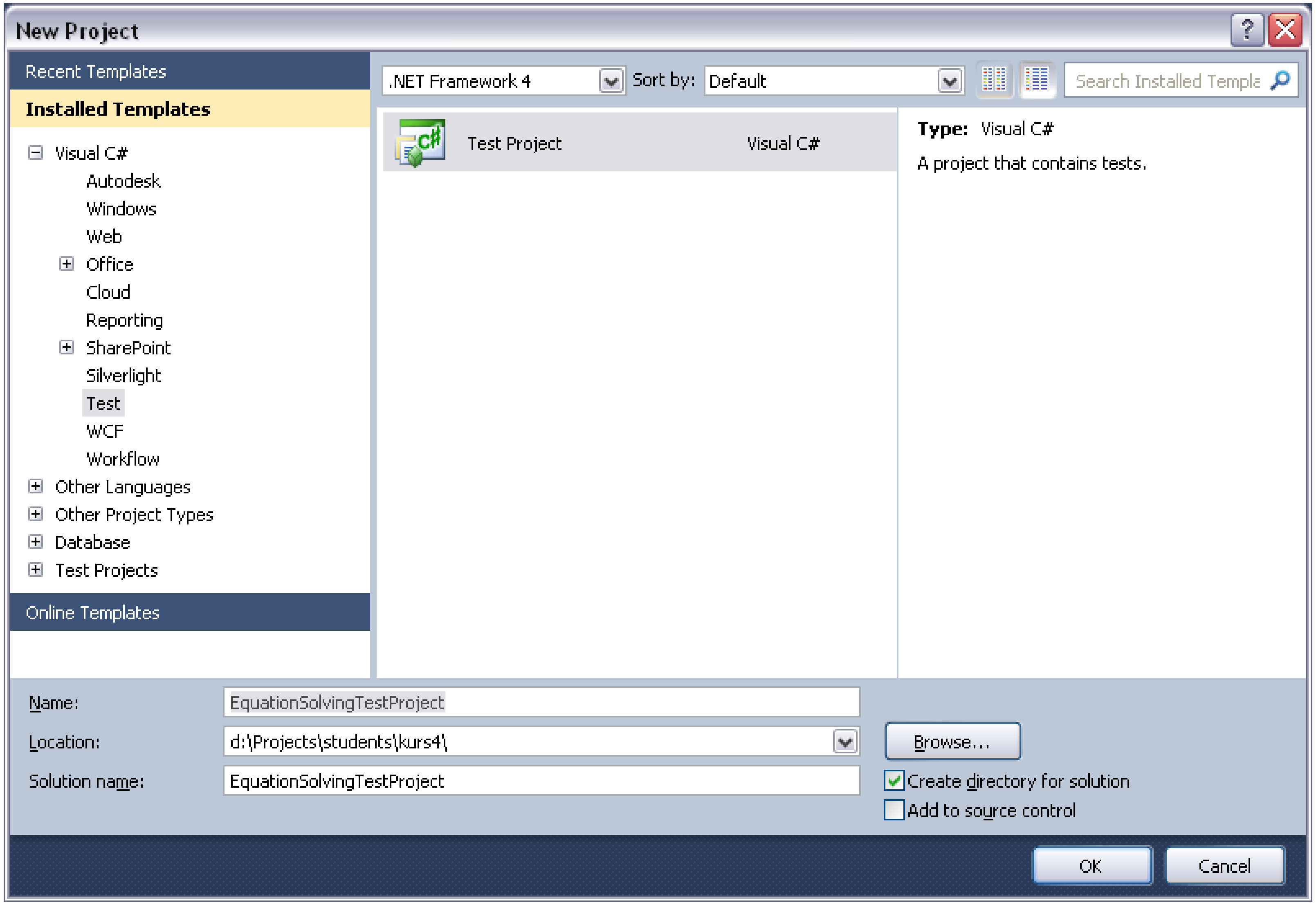
Name:	NUnit.Tests1
Location:	c:\users\tim\documents\visual studio 2015\Projects
Solution name:	NUnit.Tests1

Browse...

[✓] Create directory for solution
[ ] Create new Git repository

OK · Cancel

В старых VisualStudio создаем проект Test Project



New Project

.NET Framework 4 — Sort by: Default

Test Project — Visual C#

Type: Visual C#

A project that contains tests.

Name: EquationSolvingTestProject

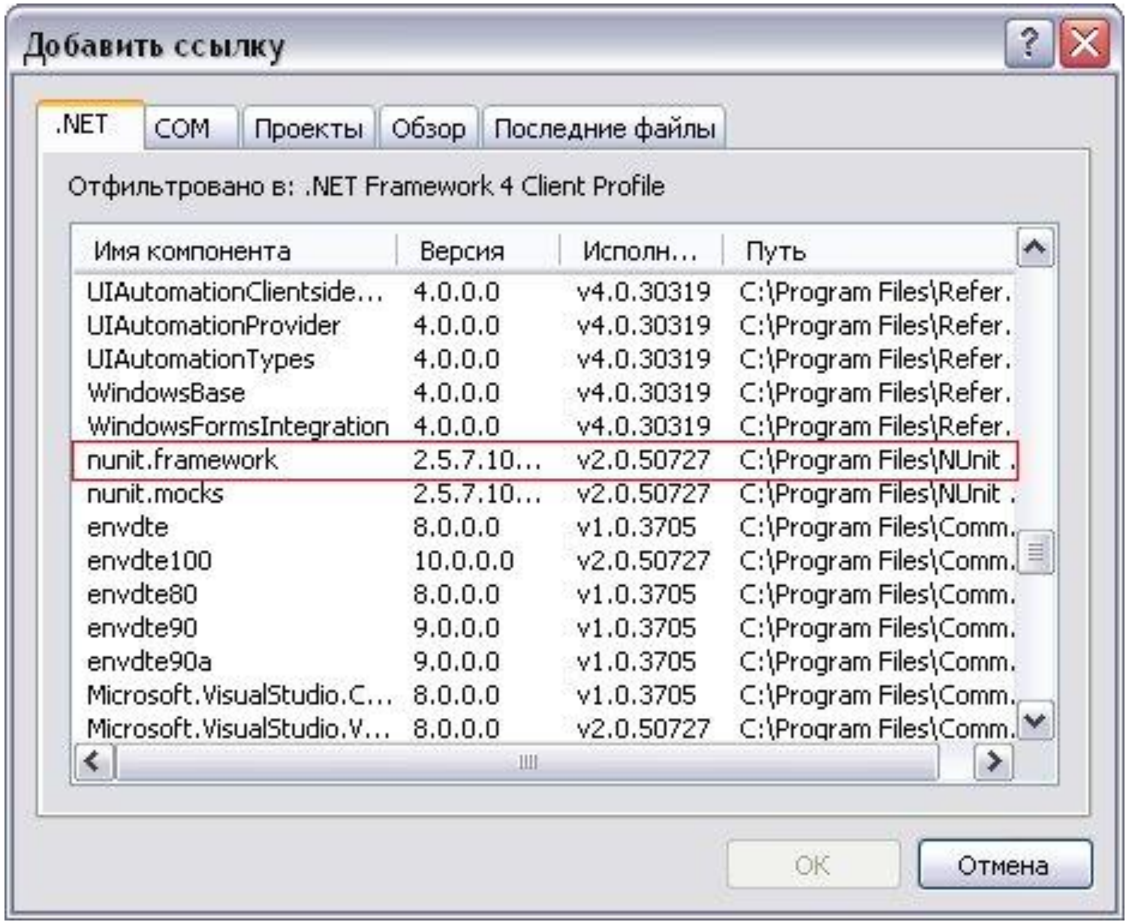
Location: d:\Projects\students\kurs4\

Solution name: EquationSolvingTestProject

[✓] Create directory for solution

[ ] Add to source control

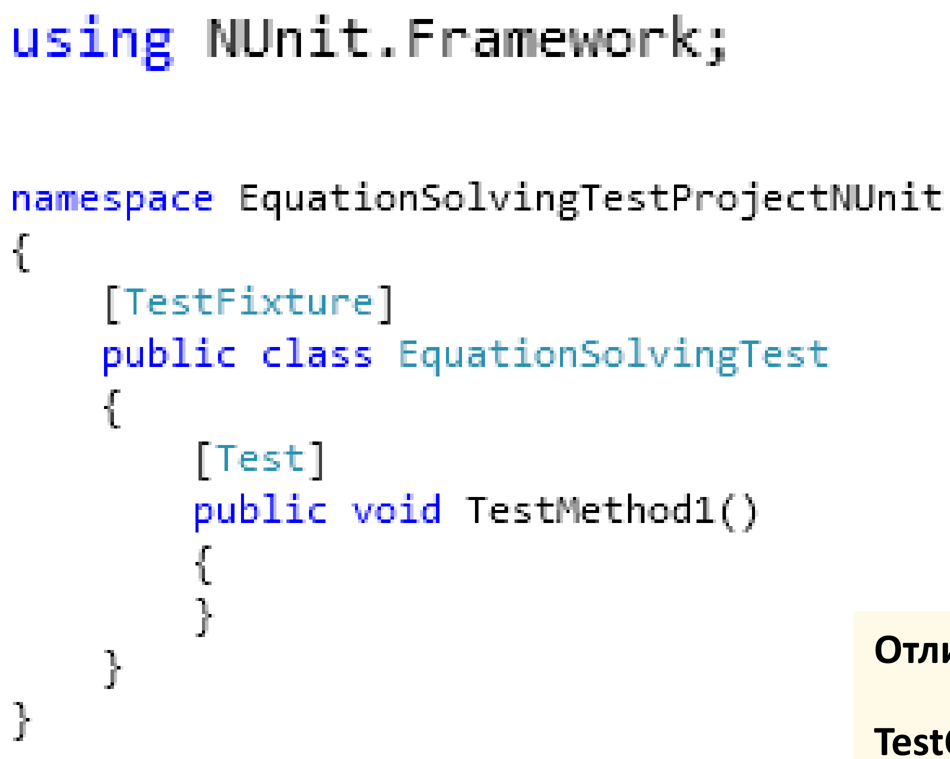
После скачивания и установки NUnit добавляем ссылку на NUnit (Add reference…)



Отфильтровано в: .NET Framework 4 Client Profile

Имя компонента	Версия	Исполн...	Путь
UIAutomationClientside...	4.0.0.0	v4.0.30319	C:\Program Files\Refer.
UIAutomationProvider	4.0.0.0	v4.0.30319	C:\Program Files\Refer.
UIAutomationTypes	4.0.0.0	v4.0.30319	C:\Program Files\Refer.
WindowsBase	4.0.0.0	v4.0.30319	C:\Program Files\Refer.
WindowsFormsIntegration	4.0.0.0	v4.0.30319	C:\Program Files\Refer.
nunit.framework	2.5.7.10...	v2.0.50727	C:\Program Files\NUnit .
nunit.mocks	2.5.7.10...	v2.0.50727	C:\Program Files\NUnit .
envdte	8.0.0.0	v1.0.3705	C:\Program Files\Comm.
envdte100	10.0.0.0	v2.0.50727	C:\Program Files\Comm.
envdte80	8.0.0.0	v1.0.3705	C:\Program Files\Comm.
envdte90	9.0.0.0	v1.0.3705	C:\Program Files\Comm.
envdte90a	9.0.0.0	v1.0.3705	C:\Program Files\Comm.
Microsoft.VisualStudio.C...	8.0.0.0	v1.0.3705	C:\Program Files\Comm.
Microsoft.VisualStudio.V...	8.0.0.0	v2.0.50727	C:\Program Files\Comm.

using NUnit.Framework и меняем атрибуты

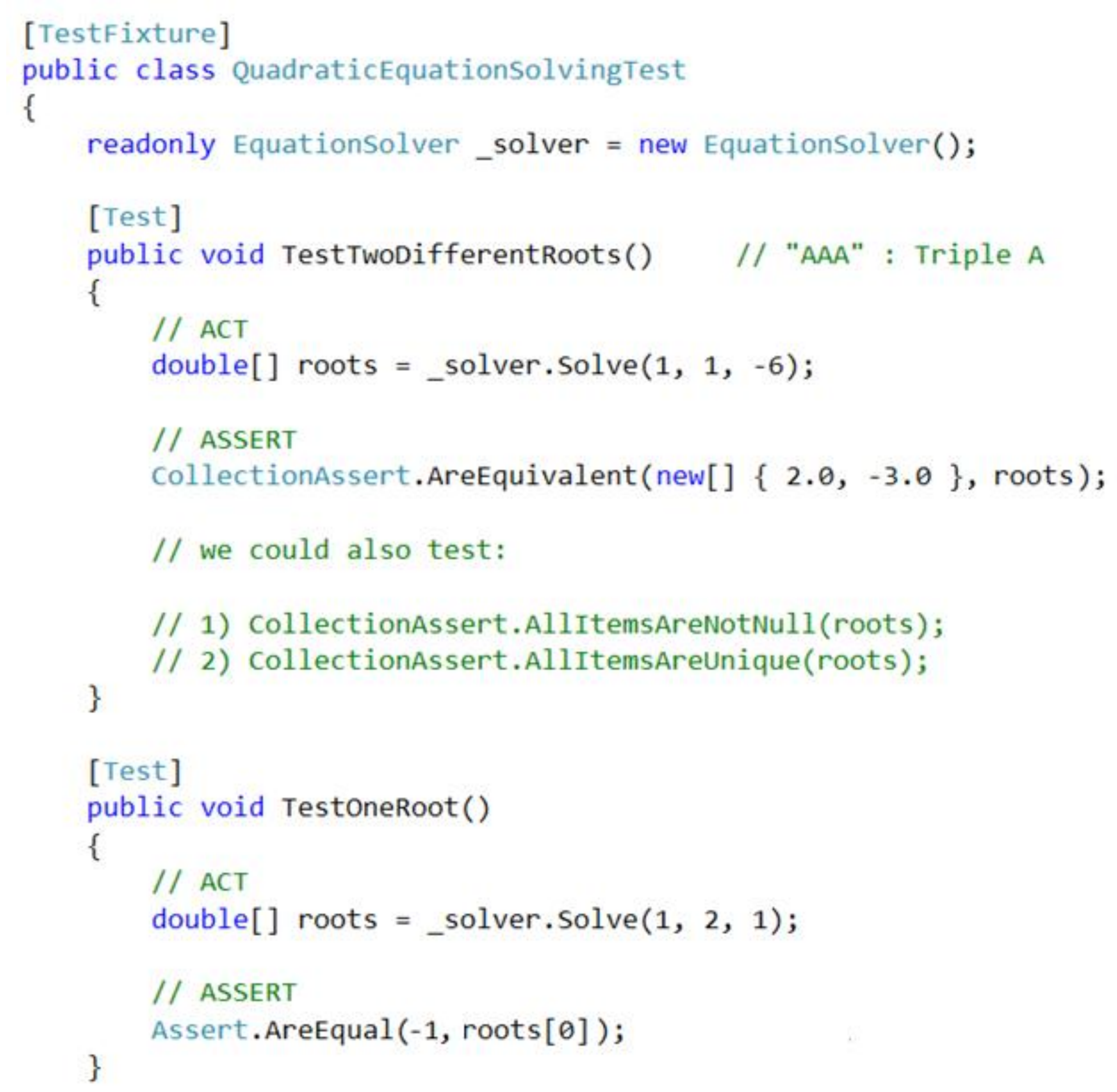


```
using NUnit.Framework;

namespace EquationSolvingTestProjectNUnit
{
    [TestFixture]
    public class EquationSolvingTest
    {
        [Test]
        public void TestMethod1()
        {
        }
    }
}
```

Отличия от MS Test в атрибутах:

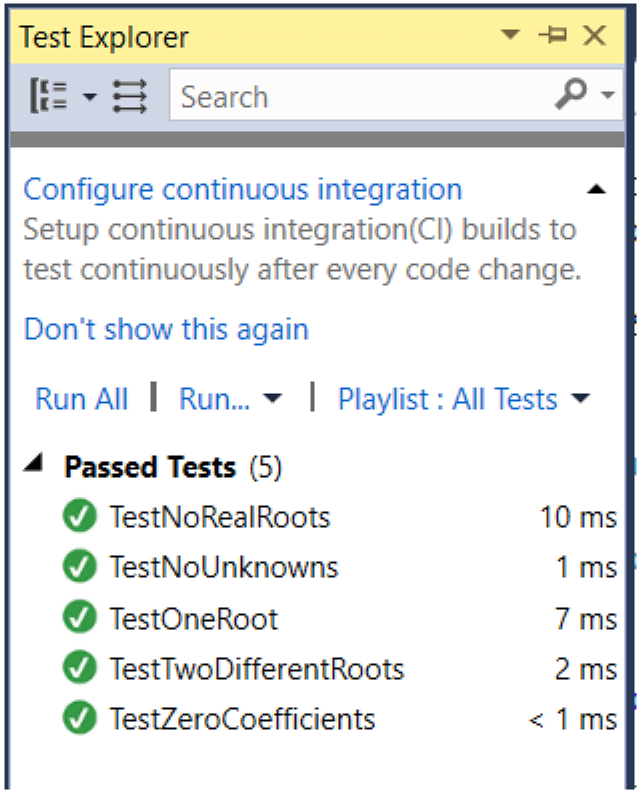
TestClass	**->**	**TestFixture**
TestMethod	**->**	**Test**

Пишем тот же код тестов, что и в лекции1



```
[TestFixture]
public class QuadraticEquationSolvingTest
{
    readonly EquationSolver _solver = new EquationSolver();

    [Test]
    public void TestTwoDifferentRoots()     // "AAA" : Triple A
    {
        // ACT
        double[] roots = _solver.Solve(1, 1, -6);

        // ASSERT
        CollectionAssert.AreEquivalent(new[] { 2.0, -3.0 }, roots);

        // we could also test:

        // 1) CollectionAssert.AllItemsAreNotNull(roots);
        // 2) CollectionAssert.AllItemsAreUnique(roots);
    }

    [Test]
    public void TestOneRoot()
    {
        // ACT
        double[] roots = _solver.Solve(1, 2, 1);

        // ASSERT
        Assert.AreEqual(-1, roots[0] );
    }
```

Запускаем тесты (NUnit adapter)



Test Explorer

Search

Configure continuous integration

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

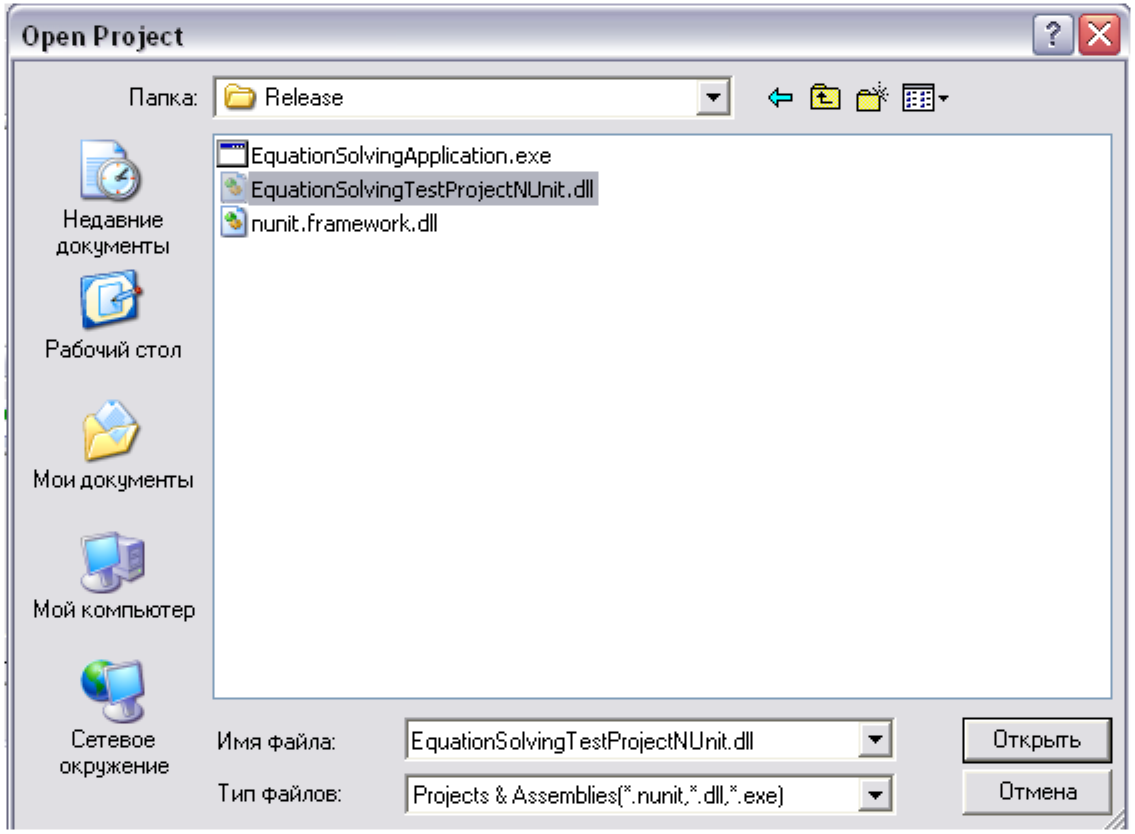
Don't show this again

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

Passed Tests (5)

✓ TestNoRealRoots	10 ms
✓ TestNoUnknowns	1 ms
✓ TestOneRoot	7 ms
✓ TestTwoDifferentRoots	2 ms
✓ TestZeroCoefficients	< 1 ms

В старых версиях запускаем NUnit.exe и открываем скомпилированную DLL проекта тестов

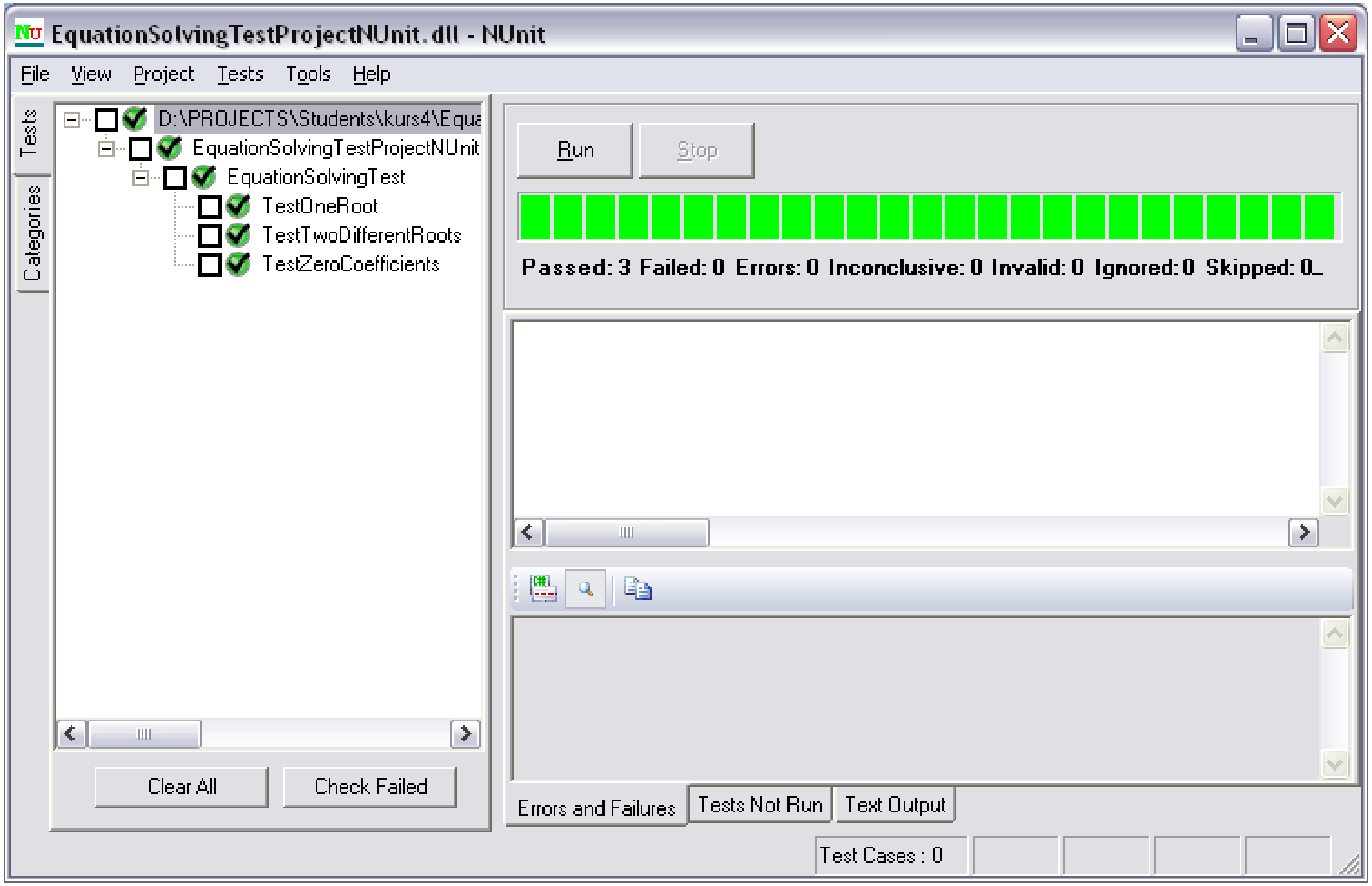


Папка: Release

EquationSolvingApplication.exe
EquationSolvingTestProjectNUnit.dll
nunit.framework.dll

Имя файла: EquationSolvingTestProjectNUnit.dll
Тип файлов: Projects & Assemblies(*.nunit,*.dll,*.exe)

Пример работы NUnit GUI для старых версий

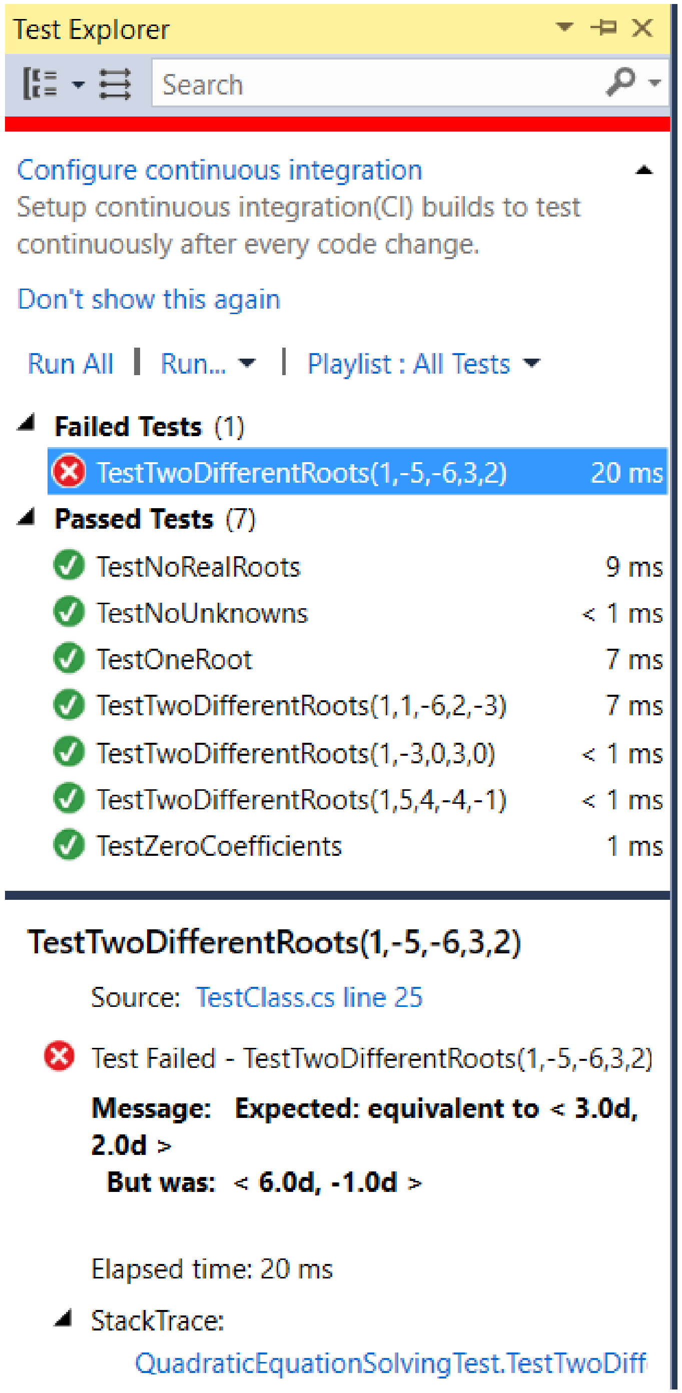


Passed: 3 Failed: 0 Errors: 0 Inconclusive: 0 Invalid: 0 Ignored: 0 Skipped: 0

Атрибут [TestCase]

```
[TestCase(1,  1, -6,  2, -3)]
[TestCase(1,  5,  4, -4, -1)]
[TestCase(1, -3,  0,  3,  0)]
[TestCase(1, -5, -6,  3,  2)]
public void TestTwoDifferentRoots(double a, double b, double c, double r1, double r2)
{
    // ACT
    double[] roots = _solver.Solve(a, b, c);

    // ASSERT
    CollectionAssert.AreEquivalent(new[] { r1, r2 }, roots);

    // we could also test:

    // 1) CollectionAssert.AllItemsAreNotNull(roots);
    // 2) CollectionAssert.AllItemsAreUnique(roots);
}
```

Проверяем



Test Explorer

Search

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

Failed Tests (1)

✗ TestTwoDifferentRoots(1,-5,-6,3,2)	20 ms

Passed Tests (7)

✓ TestNoRealRoots	9 ms
✓ TestNoUnknowns	< 1 ms
✓ TestOneRoot	7 ms
✓ TestTwoDifferentRoots(1,1,-6,2,-3)	7 ms
✓ TestTwoDifferentRoots(1,-3,0,3,0)	< 1 ms
✓ TestTwoDifferentRoots(1,5,4,-4,-1)	< 1 ms
✓ TestZeroCoefficients	1 ms

TestTwoDifferentRoots(1,-5,-6,3,2)

Source: TestClass.cs line 25

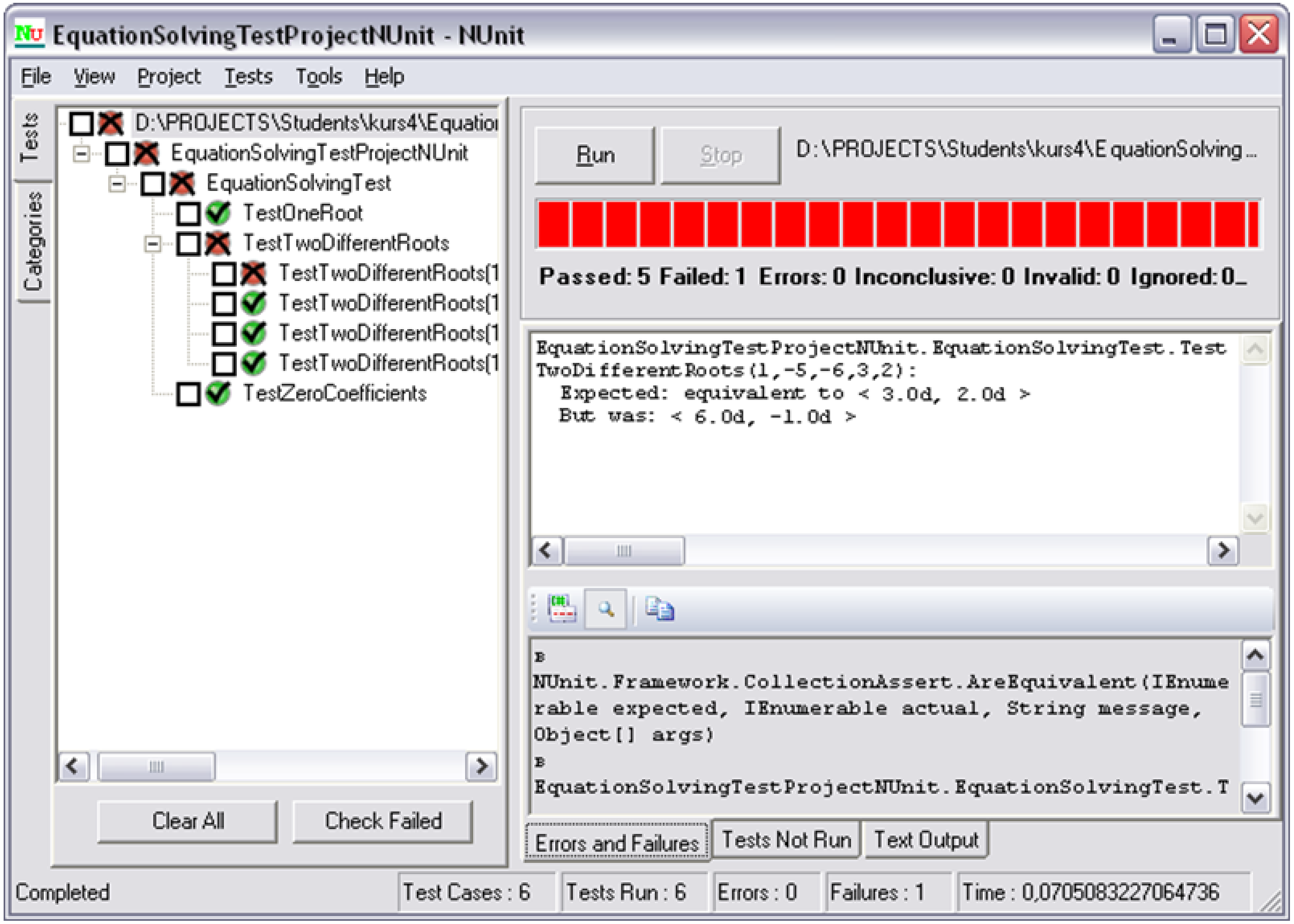
✗ Test Failed - TestTwoDifferentRoots(1,-5,-6,3,2)

```
Message:  Expected: equivalent to < 3.0d, 2.0d >
  But was:  < 6.0d, -1.0d >
```

Elapsed time: 20 ms

StackTrace:
QuadraticEquationSolvingTest.TestTwoDiff

Проверяем в NUnit GUI



Passed: 5 Failed: 1 Errors: 0 Inconclusive: 0 Invalid: 0 Ignored: 0

```
EquationSolvingTestProjectNUnit.EquationSolvingTest.Test
TwoDifferentRoots(1,-5,-6,3,2):
  Expected: equivalent to < 3.0d, 2.0d >
  But was: < 6.0d, -1.0d >
```

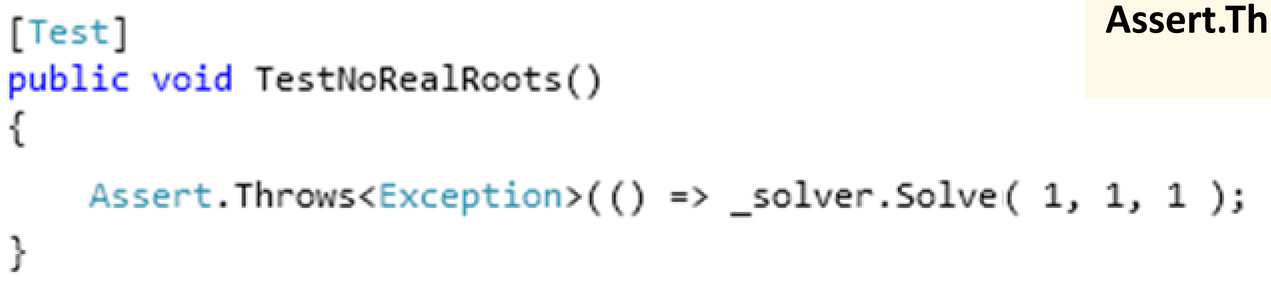
```
в
NUnit.Framework.CollectionAssert.AreEquivalent(IEnume
rable expected, IEnumerable actual, String message,
Object[] args)
в
EquationSolvingTestProjectNUnit.EquationSolvingTest.T
```

Completed | Test Cases : 6 | Tests Run : 6 | Errors : 0 | Failures : 1 | Time : 0,0705083227064736

Проверяем, что бросается исключение

Способ 1

Assert.Throws



```
[Test]
public void TestNoRealRoots()
{
    Assert.Throws<Exception>(() => _solver.Solve( 1, 1, 1 );
}
```

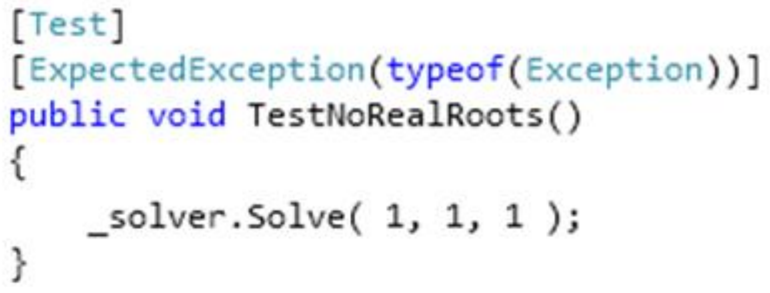
Способ 2 (только в NUnit версии 2.x)

Атрибут ExpectedException

(как и в MS Test)

В NUnit 3 также см. пример:

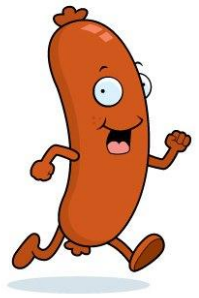
https://github.com/nunit/nunit-csharp-samples/tree/master/ExpectedException Example



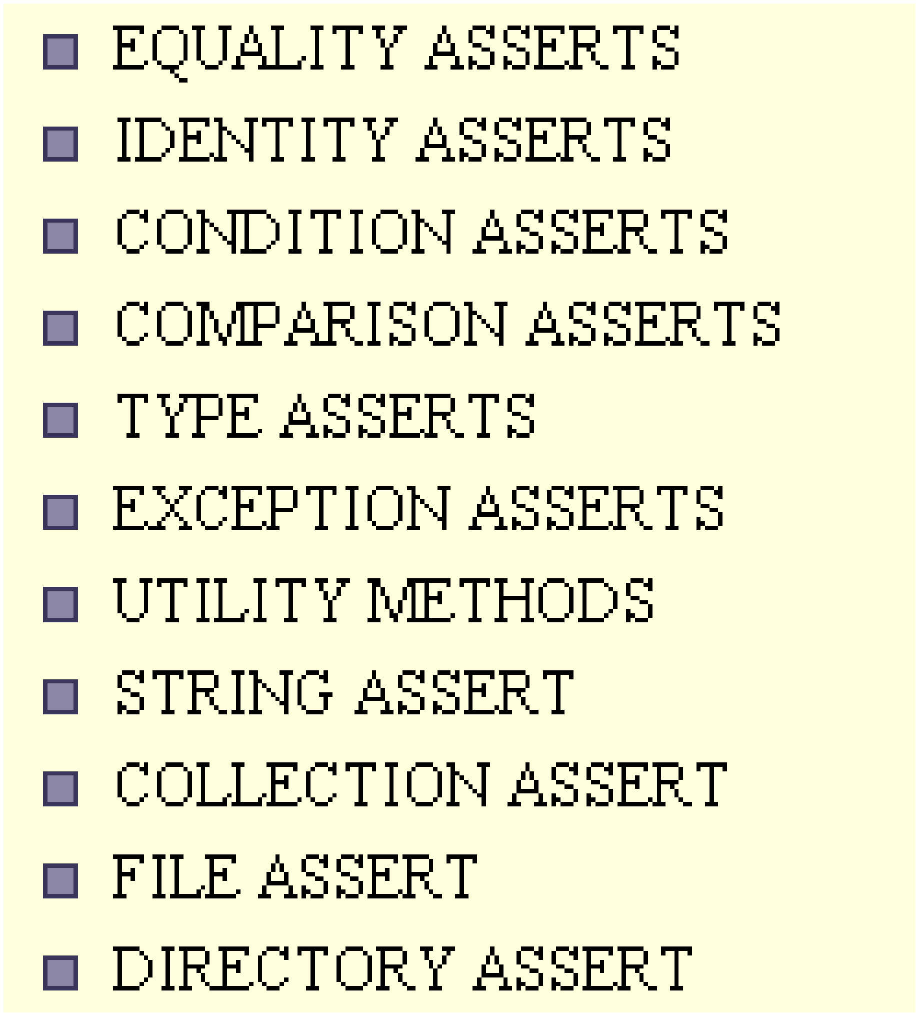
```
[Test]
[ExpectedException(typeof(Exception))]
public void TestNoRealRoots()
{
    _solver.Solve( 1, 1, 1 );
}
```

Это просто сосиска в тесте. Листай дальше

```
[TestFixture]
public class QuadraticEquationSolvingTest
{
    readonly EquationSolver _solver = new EquationSolver();

    [Test]
    public void TestTwoDifferentRoots()     // "AAA" : Triple A
    {
        // ACT
        double[] roots = _solver.Solve(1, 1, -6);

        // ASSERT
        CollectionAssert.AreEquivalent(new[] { 2.0, -3.0 }, roots);

        // we could also test:

        // 1) CollectionAssert.AllItemsAreNotNull(roots);
        // 2) CollectionAssert.AllItemsAreUnique(roots);
    }

    [Test]
    public void TestOneRoot()
    {
        // ACT
        double[] roots = _solver.Solve(1, 2, 1);

        // ASSERT
        Assert.AreEqual(-1, roots[0]);
    }
```



Asserts （NUnit Classic Model）

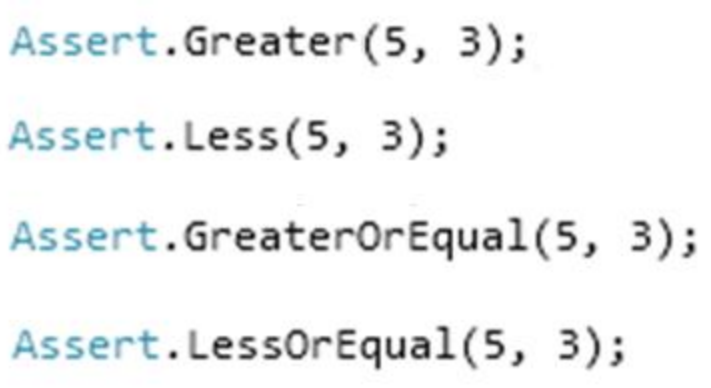


- EQUALITY ASSERTS
- IDENTITY ASSERTS
- CONDITION ASSERTS
- COMPARISON ASSERTS
- TYPE ASSERTS
- EXCEPTION ASSERTS
- UTILITY METHODS
- STRING ASSERT
- COLLECTION ASSERT
- FILE ASSERT
- DIRECTORY ASSERT

Asserts（NUnit Classic Model）

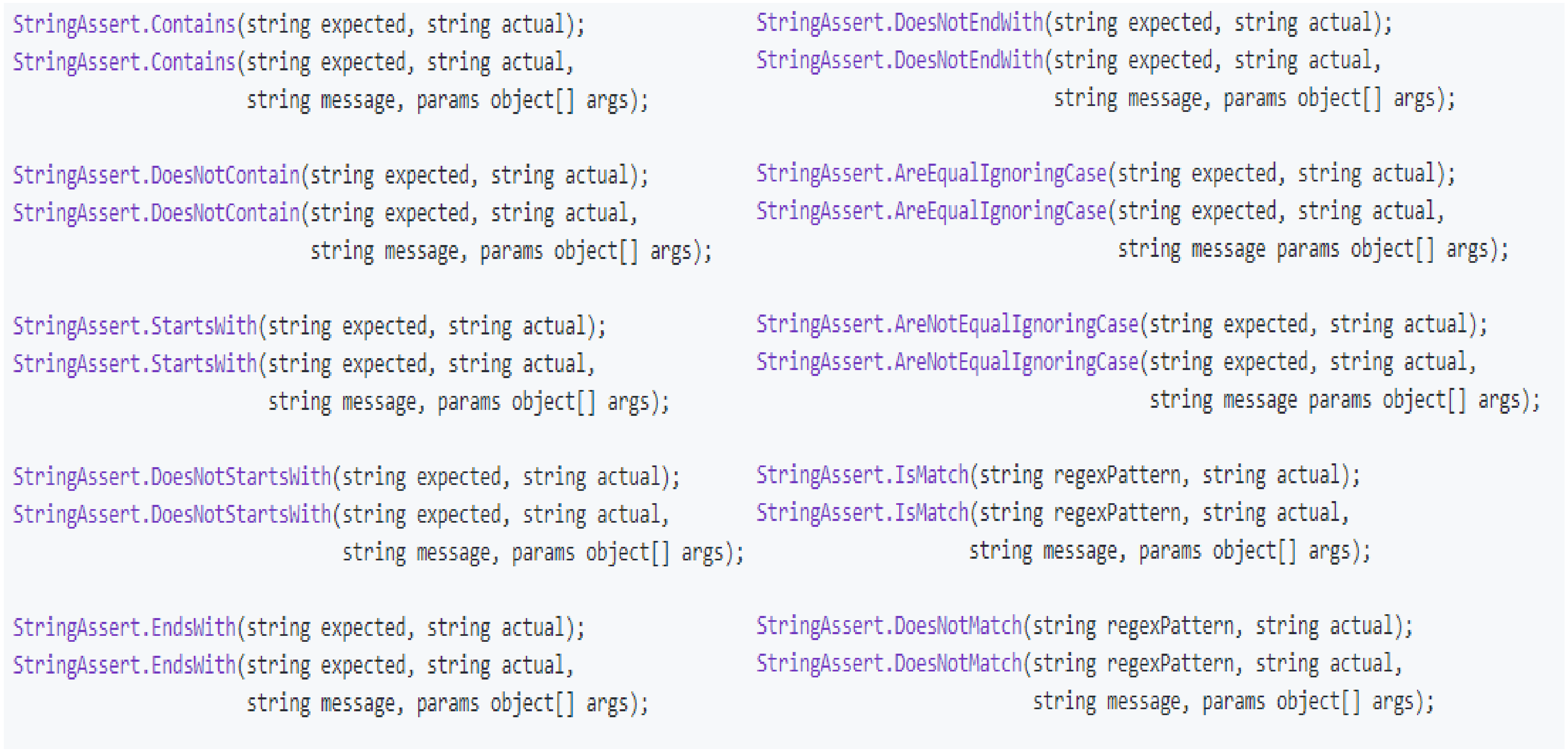
- Assert.True
- Assert.False
- Assert.Null
- Assert.NotNull
- Assert.Zero
- Assert.NotZero
- Assert.IsNaN
- Assert.IsEmpty
- Assert.IsNotEmpty
- Assert.AreEqual
- Assert.AreNotEqual
- Assert.AreSame
- Assert.AreNotSame
- Assert.Contains
- Assert.Greater
- Assert.GreaterOrEqual
- Assert.Less
- Assert.LessOrEqual
- Assert.Positive
- Assert.Negative
- Assert.IsInstanceOf
- Assert.IsNotInstanceOf
- Assert.IsAssignableFrom
- Assert.IsNotAssignableFrom
- Assert.Throws
- Assert.ThrowsAsync
- Assert.DoesNotThrow
- Assert.DoesNotThrowAsync
- Assert.Catch
- Assert.CatchAsync
- Assert.Pass
- Assert.Fail
- Assert.Ignore
- Assert.Inconclusive

Examples of Comparison Assert



```
Assert.Greater(5, 3);

Assert.Less(5, 3);

Assert.GreaterOrEqual(5, 3);

Assert.LessOrEqual(5, 3);
```

String Assert



```
StringAssert.Contains(string expected, string actual);
StringAssert.Contains(string expected, string actual,
                      string message, params object[] args);

StringAssert.DoesNotContain(string expected, string actual);
StringAssert.DoesNotContain(string expected, string actual,
                            string message, params object[] args);

StringAssert.StartsWith(string expected, string actual);
StringAssert.StartsWith(string expected, string actual,
                        string message, params object[] args);

StringAssert.DoesNotStartsWith(string expected, string actual);
StringAssert.DoesNotStartsWith(string expected, string actual,
                               string message, params object[] args);

StringAssert.EndsWith(string expected, string actual);
StringAssert.EndsWith(string expected, string actual,
                      string message, params object[] args);
```

```
StringAssert.DoesNotEndWith(string expected, string actual);
StringAssert.DoesNotEndWith(string expected, string actual,
                            string message, params object[] args);

StringAssert.AreEqualIgnoringCase(string expected, string actual);
StringAssert.AreEqualIgnoringCase(string expected, string actual,
                                  string message params object[] args);

StringAssert.AreNotEqualIgnoringCase(string expected, string actual);
StringAssert.AreNotEqualIgnoringCase(string expected, string actual,
                                     string message params object[] args);

StringAssert.IsMatch(string regexPattern, string actual);
StringAssert.IsMatch(string regexPattern, string actual,
                     string message, params object[] args);

StringAssert.DoesNotMatch(string regexPattern, string actual);
StringAssert.DoesNotMatch(string regexPattern, string actual,
                          string message, params object[] args);
```

File Assert

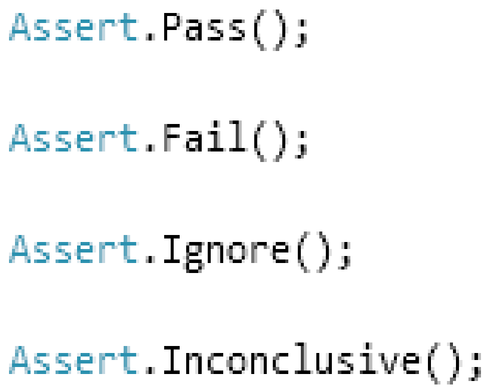
```
FileAssert.AreEqual(Stream expected, Stream actual);
FileAssert.AreEqual(
    Stream expected, Stream actual, string message, params object[] args);

FileAssert.AreEqual(FileInfo expected, FileInfo actual);
FileAssert.AreEqual(
    FileInfo expected, FileInfo actual, string message, params object[] args);

FileAssert.AreEqual(string expected, string actual);
FileAssert.AreEqual(
    string expected, string actual, string message, params object[] args);

FileAssert.AreNotEqual(Stream expected, Stream actual);
FileAssert.AreNotEqual(
    Stream expected, Stream actual, string message, params object[] args);

FileAssert.AreNotEqual(FileInfo expected, FileInfo actual);
FileAssert.AreNotEqual(
    FileInfo expected, FileInfo actual, string message, params object[] args);
```

```
FileAssert.AreNotEqual(string expected, string actual);
FileAssert.AreNotEqual(
    string expected, string actual, string message, params object[] args);

FileAssert.Exists(FileInfo actual);
FileAssert.Exists(
    FileInfo actual, string message, params object[] args);

FileAssert.Exists(string actual);
FileAssert.Exists(
    string actual, string message, params object[] args);

FileAssert.DoesNotExist(FileInfo actual);
FileAssert.DoesNotExist(
    FileInfo actual, string message, params object[] args);

FileAssert.DoesNotExist(string actual);
FileAssert.DoesNotExist(
    string actual, string message, params object[] args);
```

DirectoryAssert

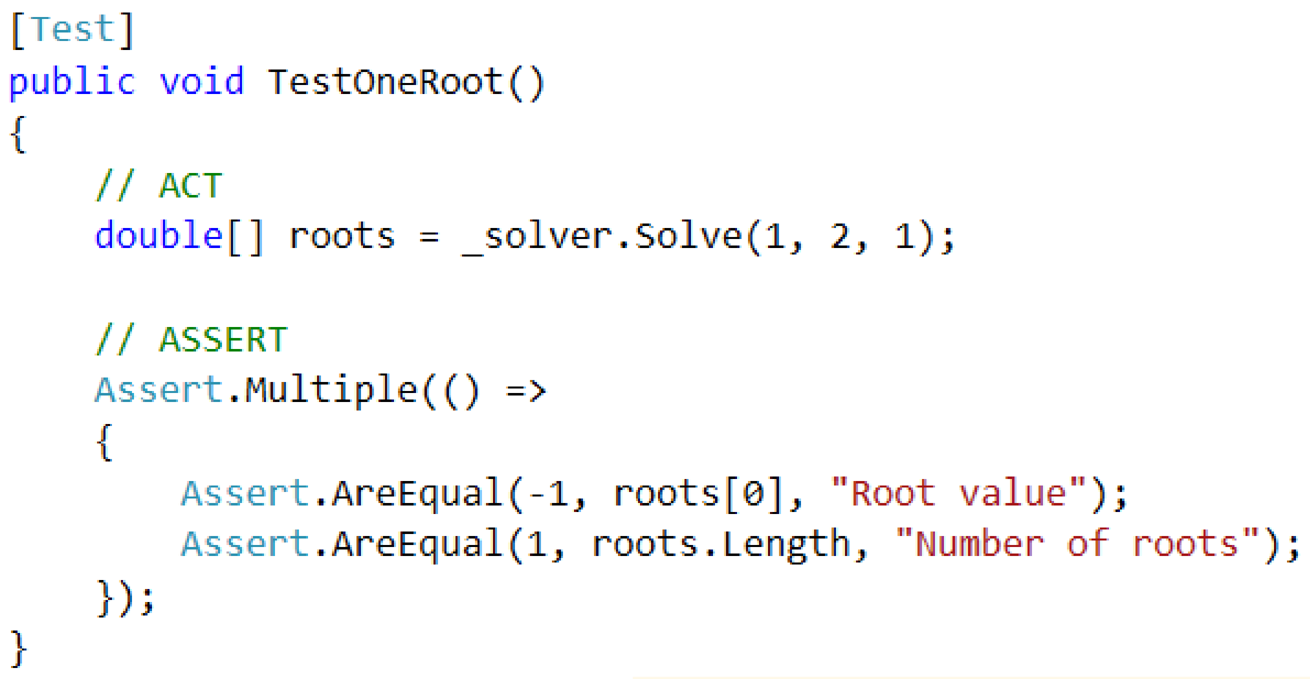
```
DirectoryAssert.AreEqual(DirectoryInfo expected, DirectoryInfo actual);
DirectoryAssert.AreEqual(DirectoryInfo expected, DirectoryInfo actual,
    string message, params object[] args);

DirectoryAssert.AreNotEqual(DirectoryInfo expected, DirectoryInfo actual);
DirectoryAssert.AreNotEqual(DirectoryInfo expected, DirectoryInfo actual,
    string message, params object[] args);

DirectoryAssert.Exists(DirectoryInfo actual);
DirectoryAssert.Exists(DirectoryInfo actual,
    string message, params object[] args);

DirectoryAssert.Exists(string actual);
DirectoryAssert.Exists(string actual,
    string message, params object[] args);

DirectoryAssert.DoesNotExist(DirectoryInfo actual);
DirectoryAssert.DoesNotExist(DirectoryInfo actual,
    string message, params object[] args);

DirectoryAssert.DoesNotExist(string actual);
DirectoryAssert.DoesNotExist(string actual,
    string message, params object[] args);
```

Utility Methods



```
Assert.Pass();

Assert.Fail();

Assert.Ignore();

Assert.Inconclusive();
```

Multiple Asserts



```
[Test]
public void TestOneRoot()
{
    // ACT
    double[] roots = _solver.Solve(1, 2, 1);

    // ASSERT
    Assert.Multiple(() =>
    {
        Assert.AreEqual(-1, roots[0], "Root value");
        Assert.AreEqual(1, roots.Length, "Number of roots");
    });
}
```

Второй ассерт зафейлится.

Можете сами пофиксить production code, чтобы этого не было

Test Context

Each NUnit test runs in an execution context, which includes information about the environment as well as the test itself. The `TestContext` class allows tests to access certain information about the execution context.

Static Properties

CurrentContext
Gets the context of the currently executing test. This context is created separately for each test before it begins execution. See below for properties of the current context.

Out
Gets a TextWriter used for sending output to the current test result.

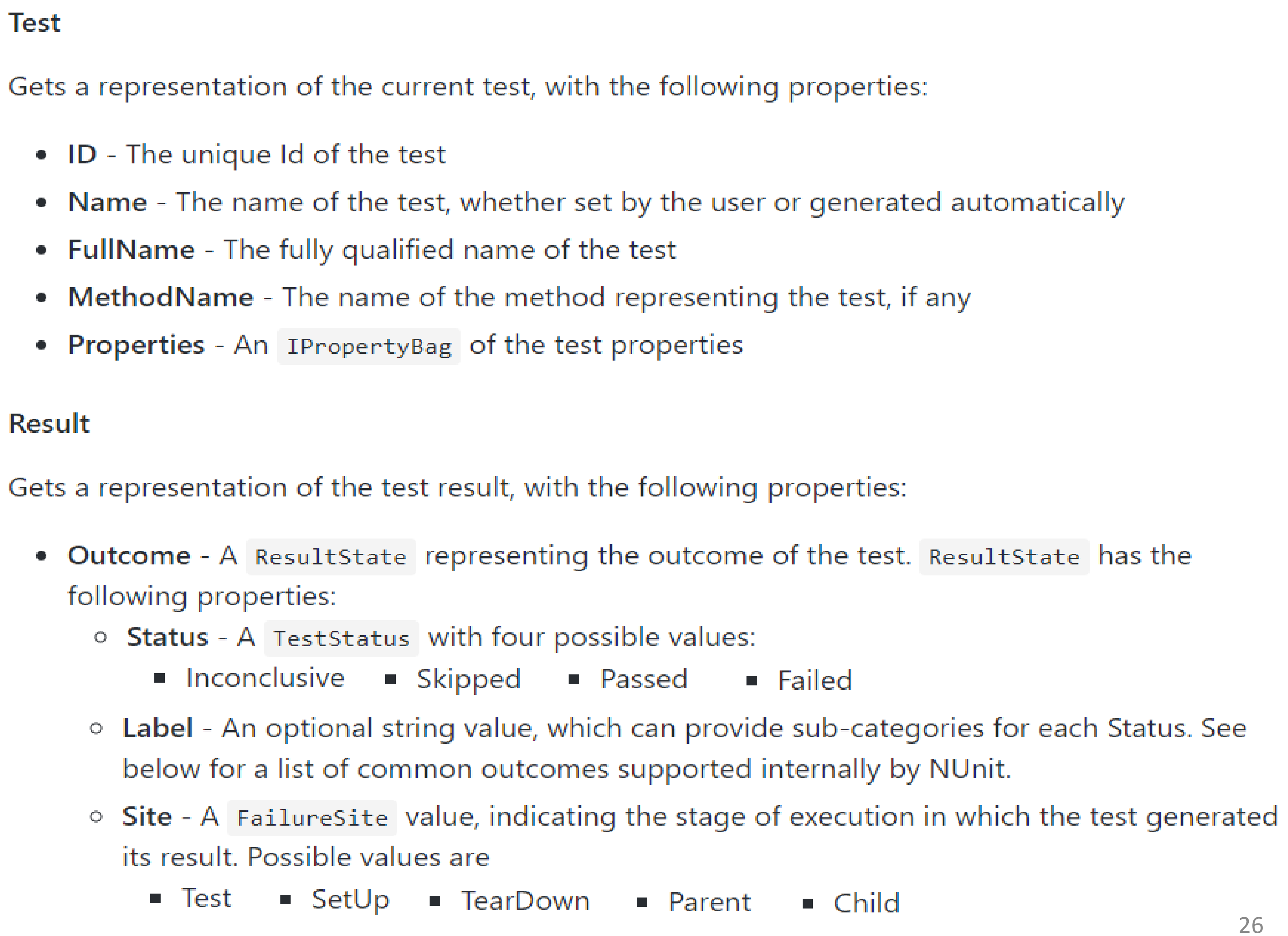
Error
Gets a TextWriter used for sending error output intended for immediate display.

Progress
Gets a TextWriter used for sending normal (non-error) output intended for immediate display.

TestParameters

Test parameters may be supplied to a run in various ways, depending on the runner used. For example, the console runner provides a command-line argument and v3.4 of the NUnit 3 VS Adapter will supports specifying them in a .runsettings file. The static TestParameters property returns an object representing those passed-in parameters.

Properties of the Current Context



Test

Gets a representation of the current test, with the following properties:

- **ID** - The unique Id of the test
- **Name** - The name of the test, whether set by the user or generated automatically
- **FullName** - The fully qualified name of the test
- **MethodName** - The name of the method representing the test, if any
- **Properties** - An `IPropertyBag` of the test properties

Result

Gets a representation of the test result, with the following properties:

- **Outcome** - A `ResultState` representing the outcome of the test. `ResultState` has the following properties:
  - **Status** - A `TestStatus` with four possible values:
    - Inconclusive
    - Skipped
    - Passed
    - Failed
  - **Label** - An optional string value, which can provide sub-categories for each Status. See below for a list of common outcomes supported internally by NUnit.
  - **Site** - A `FailureSite` value, indicating the stage of execution in which the test generated its result. Possible values are
    - Test
    - SetUp
    - TearDown
    - Parent
    - Child

SetUp, TearDown

- SetUpAttribute is now used exclusively for per-test setup.
- TearDownAttribute is now used exclusively for per-test teardown.
- OneTimeSetUpAttribute is used for one-time setup per test-run. If you run *n* tests, this event will only occur once.
- OneTimeTearDownAttribute is used for one-time teardown per test-run. If you run *n* tests, this event will only occur once
- SetUpFixtureAttribute continues to be used as at before, but with changed method attributes.

NUnit Constraint Model

Collection Constraints

Constraint Name
AllItemsConstraint
CollectionContainsConstraint
CollectionEquivalentConstraint
CollectionOrderedConstraint
CollectionSubsetConstraint
CollectionSupersetConstraint
EmptyCollectionConstraint
ExactCountConstraint
NoItemConstraint
SomeItemsConstraint
UniqueItemsConstraint

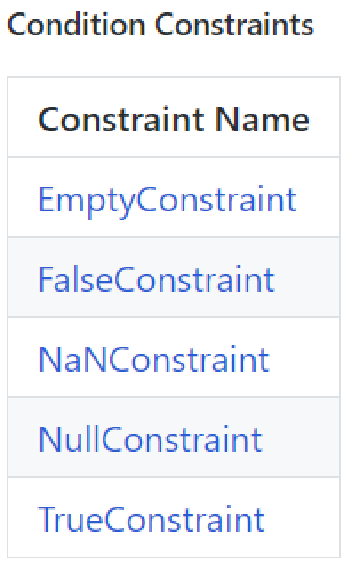
Comparison Constraints

Constraint Name
GreaterThanConstraint
GreaterThanOrEqualConstraint
LessThanConstraint
LessThanOrEqualConstraint
RangeConstraint

Compound Constraints

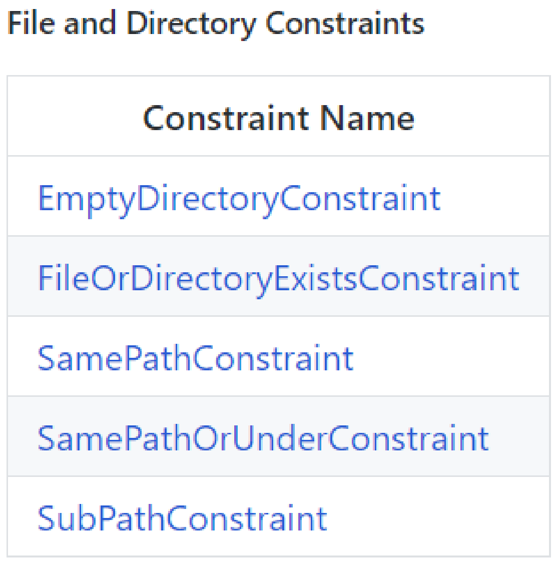
Constraint Name
AndConstraint
NotConstraint
OrConstraint

NUnit Constraint Model



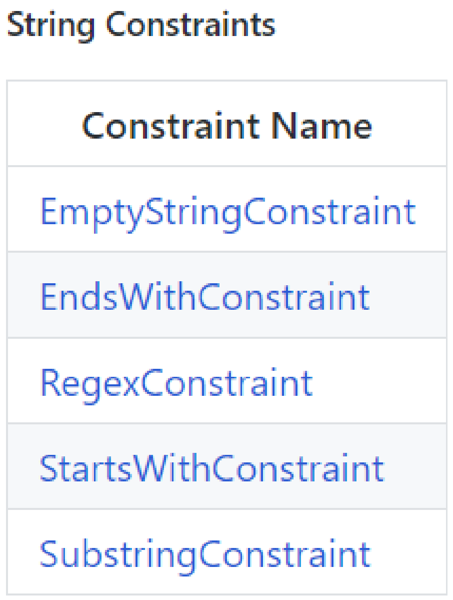
Condition Constraints

Constraint Name
EmptyConstraint
FalseConstraint
NaNConstraint
NullConstraint
TrueConstraint



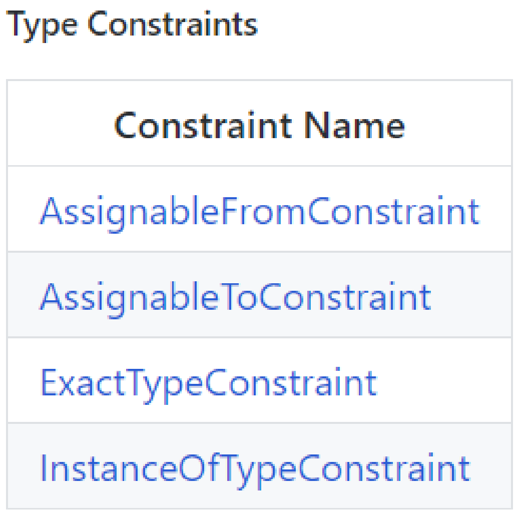
File and Directory Constraints

Constraint Name
EmptyDirectoryConstraint
FileOrDirectoryExistsConstraint
SamePathConstraint
SamePathOrUnderConstraint
SubPathConstraint



String Constraints

Constraint Name
EmptyStringConstraint
EndsWithConstraint
RegexConstraint
StartsWithConstraint
SubstringConstraint



Type Constraints

Constraint Name
AssignableFromConstraint
AssignableToConstraint
ExactTypeConstraint
InstanceOfTypeConstraint

NUnit Constraint Model

Helper class

Constraint Expression

```
[Test]
public void Test2Plus2()
{
    Assert.That(2 + 2, Is.EqualTo(4));
    Assert.That(2 + 2 == 4);
    Assert.That(2 + 2 + 1, Is.Not.EqualTo(4));
    Assert.That(2 + 2 + 1 != 4);
}

[Test]
public void TestSimpleStringConstraints()
{
    Assert.That("Hello", Is.EqualTo("HELLO").IgnoreCase);

    string[] expected = {"hello", "world"};
    string[] actual = {"HELLO", "World"};
    Assert.That(actual, Is.EqualTo(expected).IgnoreCase);
}
```

Классы-хелперы заменяют создание констрейнтов вручную

```
using System;
using NUnit.Framework;
using NUnit.Framework.Constraints;
```

```
[Test]
public void TestSimpleStringConstraints()
{
    Assert.That("Hello", Is.EqualTo("HELLO").IgnoreCase);

    Assert.That("Hello", new EqualConstraint("HELLO").IgnoreCase);
}
```

Collection Constraints

```
[Test]
public void TestCollectionConstraints()
{
    int[] array = { 1, 2, 3, 4, 5 };

    Assert.Multiple(() =>
    {
        Assert.That(array, Is.Unique);
        Assert.That(array, Has.Length.LessThan(10));
        Assert.That(array, Is.Ordered);
        Assert.That(array, Is.All.LessThan(6));
        Assert.That(array, Has.Exactly(1).EqualTo(3));
        Assert.That(array, Has.Exactly(2).GreaterThan(3));
    });
}
```

Another example of collection constraints

```
class Person
{
    public int Id { get; set; }
    public string Lastname { get; set; }
}
```

```
[Test]
public void TestPersonCollection()
{
    Person[] persons =
    {
        new Person {Id = 1, Lastname = "Emerson"},
        new Person {Id = 3, Lastname = "Lake"},
        new Person {Id = 2, Lastname = "Palmer"}
    };

    Assert.That(persons, Is.Ordered.By("Lastname"));
}
```

More examples of constraints

```
[Test]
public void TestNoUnknowns()
{
    Assert.That(() => _solver.Solve(0, 0, 1),
                Throws.InstanceOf<ArgumentException>()
                      .And
                      .Message.Contains("unknowns"));
}
```

```
[Test]
public void TestRange()
{
    Assert.That(4, Is.InRange(1, 10));
}
```

```
[Test]
public void TestMoreStringConstraints()
{
    Assert.That("Donetsk", Does.Contain("net"));
    Assert.That("Donetsk", Does.StartWith("Do"));
    Assert.That("Donetsk", Does.EndWith("sk"));
}
```

ListMapper

```
string[] strings = new string[] { "a", "ab", "abc" };
int[] lengths = new int[] { 1, 2, 3 };

Assert.That(List.Map(strings).Property("Length"),
       Is.EqualTo(lengths));

Assert.That(new ListMapper(strings).Property("Length"),
       Is.EqualTo(lengths));

// Assuming inheritance from AssertionHelper
Expect(Map(strings).Property("Length"), EqualTo(lengths));
```

Собственные Constraints

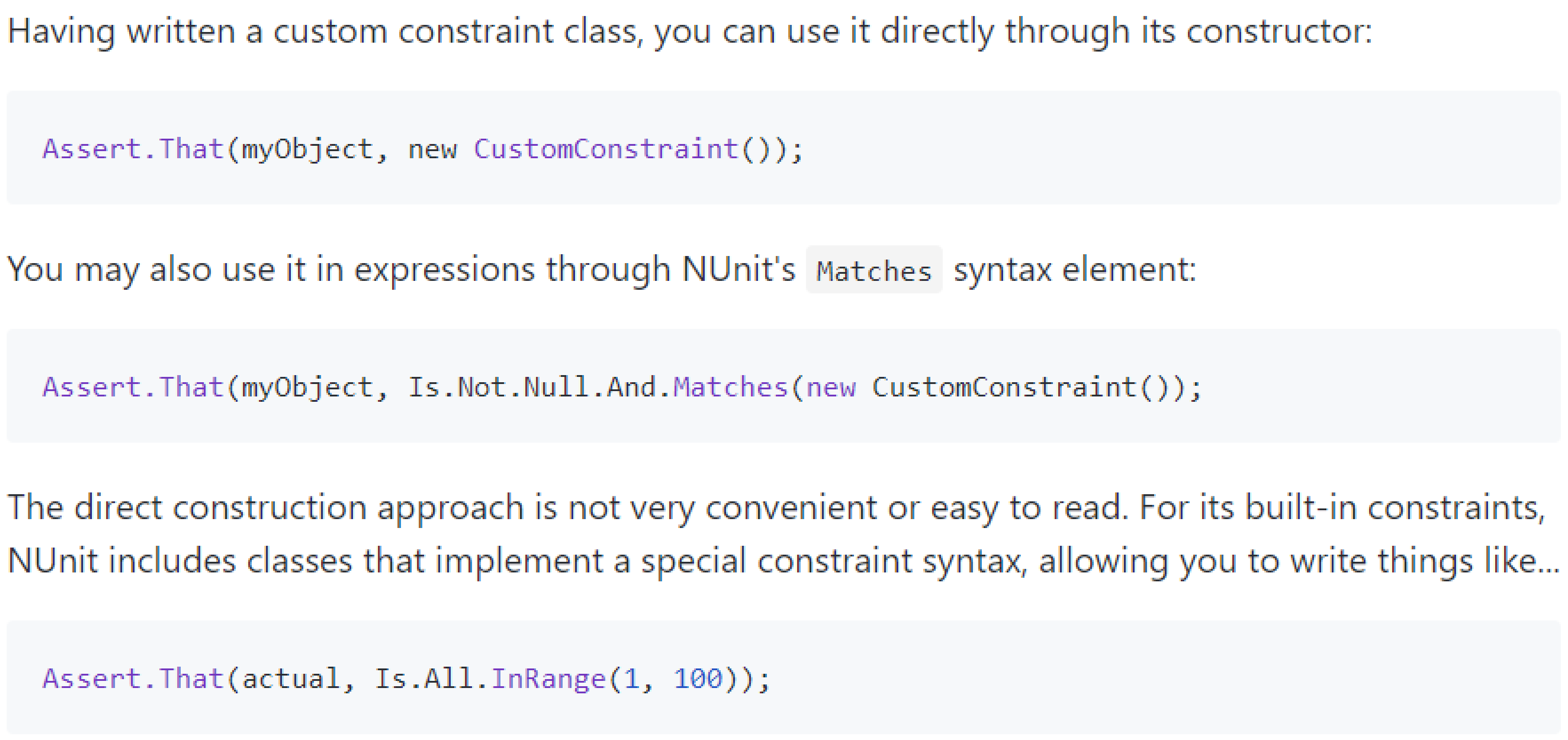
You can implement your own custom constraints by creating a class that inherits from the `Constraint` abstract class, which supports performing a test on an actual value and generating appropriate messages.

`Constraint` Abstract Class

Implementations must override the one abstract method `ApplyTo<TActual>` which evaluates the previously stored expected value (if any) against the method's parameter, the actual value. There are also several virtual methods that may be overridden to change some default behaviors.

```
public abstract class Constraint
{
    protected Constraint(params object[] args) {}
    public abstract ConstraintResult ApplyTo<TActual>(TActual actual);
    ...
    public virtual ConstraintResult ApplyTo<TActual>(ActualValueDelegate<TActual> del) {}
    public virtual ConstraintResult ApplyTo<TActual>(ref TActual actual) {}
    protected virtual object GetTestObject<TActual>(ActualValueDelegate<TActual> del) {}
    public virtual string Description { get; protected set; }
    protected virtual string GetStringRepresentation() {}
}
```

Собственные Constraints



Having written a custom constraint class, you can use it directly through its constructor:

```
Assert.That(myObject, new CustomConstraint());
```

You may also use it in expressions through NUnit's `Matches` syntax element:

```
Assert.That(myObject, Is.Not.Null.And.Matches(new CustomConstraint());
```

The direct construction approach is not very convenient or easy to read. For its built-in constraints, NUnit includes classes that implement a special constraint syntax, allowing you to write things like...

```
Assert.That(actual, Is.All.InRange(1, 100));
```

Атрибуты NUnit

- Apartment Attribute
- Author Attribute
- Category Attribute
- Combinatorial Attribute
- Culture Attribute
- Datapoint Attribute
- DatapointSource Attribute
- Description Attribute
- Explicit Attribute
- Ignore Attribute
- LevelOfParallelism Attribute
- MaxTime Attribute
- NonParallelizable Attribute
- OneTimeSetUp Attribute
- OneTimeTearDown Attribute
- Order Attribute
- Pairwise Attribute
- Parallelizable Attribute
- Platform Attribute
- Property Attribute
- Random Attribute
- Range Attribute
- Repeat Attribute
- RequiresThread Attribute
- Retry Attribute
- Sequential Attribute
- SetCulture Attribute
- SetUICulture Attribute
- SetUp Attribute
- SetUpFixture Attribute
- SingleThreaded Attribute
- TearDown Attribute
- Test Attribute
- TestCase Attribute
- TestCaseSource Attribute
- TestFixture Attribute
- TestFixtureSetup Attribute
- TestFixtureSource Attribute
- TestFixtureTeardown Attribute
- TestOf Attribute
- Theory Attribute
- Timeout Attribute
- Values Attribute
- ValueSource Attribute