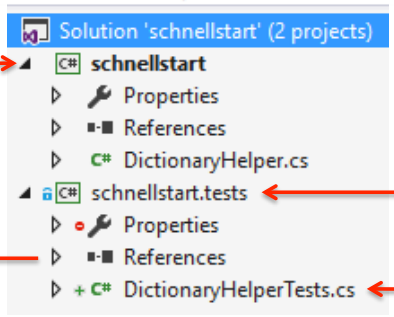
Schnellstart: Unit Tests mit NUnit

Grundlagen



1. Projekt hinzufügen
(Class Library / Klassenbibliothek)
Testprojekt pro Projekt

2. Referenz hinzufügen

3. NuGet Package "NUnit" hinzufügen

4. Optional: NuGet Package "NUnitTestAdapter" hinzufügen

5. Klasse hinzufügen
Testklasse pro Klasse

```
using NUnit.Framework;

namespace schnellstart.tests
{
    [TestFixture]
    public class DictionaryHelperTests
    {
        [Test]
        public void Integrationstest() {

            var sut = new DictionaryHelper();

            var result = sut.ToDictionary("a=1;b=2;c=3");

            Assert.That(result.Count, Is.EqualTo(3));
        }
    }
}
```

NUnit Namespace

Entspricht dem Projektnamen

"Klasse enthält Tests"

Klassenname + "Tests"

"Dies ist eine Testmethode"

Name des Tests

"System under Test"

1. Arrange

2. Act

3. Assert

http://nunit.org

Tests ausführen

ReSharper:

Ausführen: Im Menü unter *ReSharper - Unit Tests - Run Unit Tests*
Anzeigen: Im Menü unter *ReSharper - Windows - Unit Tests*

JetBrains Rider:

Ausführen: Im Menü unter *Tests - Run Unit Tests*
Anzeigen: Im Menü unter *View - Tool Windows - Unit Tests*

Visual Studio:

Ausführen: Im Menü unter *Test - Run - All Tests*
Anzeigen: Im Menü unter *Test - Windows - Test Explorer*
Voraussetzung: Visual Studio erkennt die NUnit Tests nur, wenn der *NUnitTestAdapter* mit NuGet installiert wurde.

Assert Beispiele

Vergleiche

```
Assert.That(x, Is.EqualTo(y));
Assert.That(x, Is.SameAs(y));
Assert.That(x, Is.InRange(1, 10));
Assert.That(x, Is.GreaterThan(1));
Assert.That(x, Is.LessThan(10));
```

Null Prüfung

```
Assert.That(x, Is.Null);
```

Negieren

```
Assert.That(x, Is.Not.Null);
```

Aufzählungen (Array, List, IEnumerale, etc.)

```
Assert.That(x, Is.Empty);
Assert.That(x, Is.EquivalenTo(y));
Assert.That(x, Is.Ordered);
Assert.That(x, Is.Unique);
Assert.That(x, Does.Contain(5);
```

Boolesche Ausdrücke

```
Assert.That(x, Is.True);
Assert.That(x, Is.False);
Assert.That(x, Is.EqualTo(1).Or.EqualTo(7));
Assert.That(x, Is.Ordered.And.Unique);
```

Strings

```
Assert.That(s, Does.StartWith("abc"));
Assert.That(s, Does.Match("^[a-zA-Z]"));
```

Dateien

```
FileAssert.AreEqual("expected.txt", "a.txt");
```

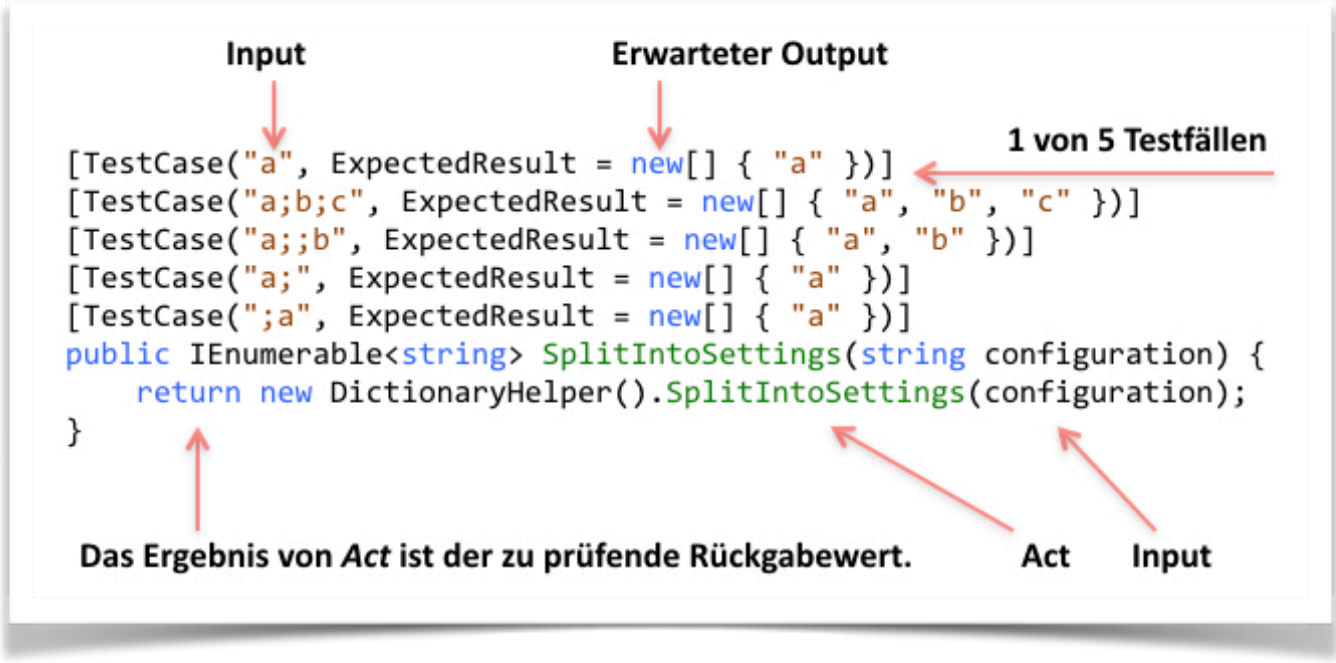
float, double, decimal

```
Assert.That(d, Is.EqualTo(1.42).Within(0.001));
```

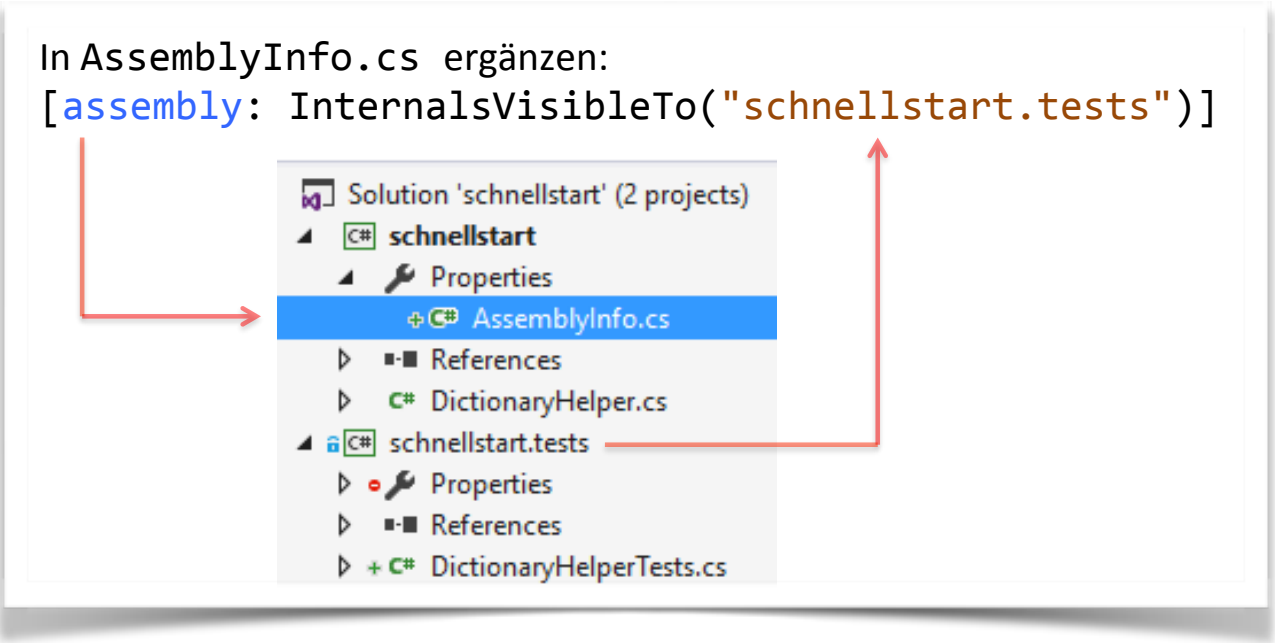
Exceptions

```
Assert.Throws<ArgumentOutOfRangeException>(() => sut.ToDictionary(""));
Assert.Catch<Exception>(() => sut.ToDictionary(""));
Assert.That(() => sut.ToDictionary(""), Throws.InstanceOf<Exception>());
Assert.That(() => sut.ToDictionary("a=1"), Throws.Nothing);
```

Datengetriebene Tests



Interna testen (Whitebox)



```
public class DictionaryHelper
{
    ...
    internal IEnumerable<string> SplitIntoSettings(string configuration) {
        return configuration.Split(new [] {";"},
            StringSplitOptions.RemoveEmptyEntries);
    }

    ...
}

[TestFixture]
public class DictionaryHelperTests
{
    [Test]
    public void Empty_setting_is_ignored() {
        var sut = new DictionaryHelper();
        var result = sut.SplitIntoSettings("a;;b");
        Assert.That(result, Is.EqualTo(new[] {"a", "b"}));
    }
}
```

`internal` **statt** `private`**, um im Test erreichbar zu sein (Whitebox Test)**

Interne Methode testen

Dateien

Unterverzeichnis für die Testdaten

Eigenschaft „Copy to Output directory“ setzen

```
public class FileTests
{
    [SetUp]
    public void Setup() {
        Directory.SetCurrentDirectory(TestContext.CurrentContext.TestDirectory);
    }

    [Test]
    public void Read_from_File_test() {
        var filename = Path.Combine("testdata", "datafile.txt");
        var lines = File.ReadAllLines(filename);
        Assert.That(lines, Is.EqualTo(new[]{"A", "B", "C"}));
    }
}
```

Aktuelles Directory ist andernfalls „WorkDirectory“. Dann werden die Dateien nicht gefunden.

Pfade mit „Path.Combine“ zusammensetzen, wegen der Unterschiede von Windows/Mac/Linux.

Attribute

`[Explicit]`	Test wird nur ausgeführt, wenn er explizit gestartet wird. Nützlich für GUI oder Ressourcentests.
`[Apartment(ApartmentState.STA)]`	Test wird im *Single Thread Apartment* ausgeführt. Erforderlich für WPF bzw. WinForms UI Tests.
`[Categorie("Kat. 1")]`	Ordnet einen Test einer Kategorie zu. Tests können nach Kategorien angeordnet werden. Kategorien können im Testlauf ausgeschlossen werden. Nützlich bei *Continuous Integration*.
`[Description("Eine Beschreibung")]`	Eine Beschreibung des Tests.