NUnit Framework

NUnit je open-source testovací framework pro všechny .NET jazyky. Patří do rodiny xUnit nástrojů (založených na Smalltalku) a je klíčovým nástrojem pro psaní jednotkových testů a testů řízených daty.

1. Základní architektura NUnit

Na rozdíl od xUnit.net, NUnit přistupuje k životnímu cyklu testu odlišně:

- **Trvání instance:** Pro všechny testy v rámci jedné třídy se standardně vytváří **jedna instance** třídy. To umožňuje sdílet stav mezi testy (což je ale nutné dělat opatrně).
- **Atributy:** NUnit silně spoléhá na atributy (metadatové značky), kterými se definují třídy s testy a jejich chování.

—

2. Klíčové atributy a syntaxe

Aby NUnit rozpoznal kód jako testovací, používá specifické značky:

- **[TestFixture]:** Označuje třídu, která obsahuje testy.
- **[Test]:** Označuje konkrétní metodu jako testovací případ.
- **[SetUp]:** Metoda označená tímto atributem se spustí **před každým** jednotlivým testem (např. pro inicializaci proměnných).
- **[TearDown]:** Spustí se **po každém** testu (např. pro úklid prostředků).

Příklad základního testu:

```
[TestFixture]
public class CalculatorTests {
    private Calculator _calc;

    [SetUp]
    public void Init() {
        _calc = new Calculator();
    }

    [Test]
    public void Add_SimpleValues_ReturnsSum() {
        int result = _calc.Add(5, 10);
        Assert.That(result, Is.EqualTo(15));
    }
}
```

—

3. Model Assertací (Ověřování)

NUnit nabízí dva styly zápisu ověřovacích podmínek:

Klasický model

Starší styl, který je stručný, ale méně čitelný.

```
Assert.AreEqual(15, result);
```

Constraint Model (Fluent)

Moderní a doporučovaný styl, který se čte jako anglická věta. Používá metodu `Assert.That()`.

```
Assert.That(result, Is.EqualTo(15));
Assert.That(list, Has.Exactly(3).Items);
Assert.That(name, Does.StartWith("J").And.EndWith("n"));
```

—

4. Parametrizované testy: [TestCase]

NUnit exceluje v testování různých vstupů pomocí jednoho testu.

```
[TestCase(1, 2, 3)]
[TestCase(-1, 1, 0)]
[TestCase(10, 20, 30)]
public void Add_MultiData_ReturnsCorrectSum(int a, int b, int expected) {
    Assert.That(_calc.Add(a, b), Is.EqualTo(expected));
}
```

—

5. Srovnání s xUnit

Vlastnost	**NUnit**	**xUnit**
Značení třídy	`[TestFixture]`	Není potřeba
Značení testu	`[Test]`	`[Fact]`
Příprava dat	`[SetUp]`	Konstruktor třídy
Izolace	Sdílená instance třídy	Nová instance pro každý test

—

6. Pokročilé funkce

- **[Ignore]:** Dočasně zakáže spuštění testu s uvedením důvodu.
- **[MaxTime]:** Test selže, pokud trvá déle než stanovený limit v milisekundách.
- **[Category]:** Umožňuje třídit testy (např. „LongRunning“, „Integration“) a spouštět je odděleně.

Související články:

- Unit Testing a kvalita kódu
- xUnit.net Framework
- Mockování v .NET (Moq, NSubstitute)

Tagy: programming dot-net testing nunit unit-test csharp