xUnit.net Framework

xUnit.net je open-source testovací nástroj pro .NET aplikace (včetně .NET Core, 5/6/7/8+). Je navržen tak, aby podporoval moderní vývojové postupy, jako je TDD, a vynucoval čistý design testovacího kódu.

1. Klíčové rozdíly oproti jiným frameworkům

xUnit přichází s architekturou, která se zbavuje historických nánosů a zbytečných atributů:

- **Izolace testů:** Pro každý jednotlivý test (metodu) se vytvoří **zcela nová instance** testovací třídy. To zaručuje, že testy jsou na sobě nezávislé a nesdílejí stav.
- **Žádné [SetUp] a [TearDown]:** Namísto speciálních atributů se používá standardní **konstruktor** (pro přípravu) a rozhraní **IDisposable** (pro úklid).
- **Žádné [TestFixture]:** Třída obsahující testy nemusí mít žádný speciální atribut, stačí, když jsou testovací metody veřejné.

—

2. Základní stavební kameny: Fact a Theory

V xUnit rozlišujeme dva základní typy testovacích metod:

[Fact]

Představuje invariantní test – tedy takový, který je vždy pravdivý a nevyžaduje žádné parametry.

```
[Fact]
public void OvereniVypoctu_Vzdy_VraciSpravnyVysledek() {
    // Arrange
    var calc = new Calculator();
    // Act
    var result = calc.Add(10, 20);
    // Assert
    Assert.Equal(30, result);
}
```

[Theory]

Představuje test, který je pravdivý pro konkrétní sadu dat. Umožňuje spustit stejnou testovací logiku s různými vstupy.

```
[Theory]
```

```
[InlineData(1, 2, 3)]
[InlineData(-1, 1, 0)]
[InlineData(0, 0, 0)]
public void Sctiani_RuznaData_VraciOcekavanySoucet(int a, int b, int
ocekavany) {
    var calc = new Calculator();
    var vysledek = calc.Add(a, b);
    Assert.Equal(ocekavany, vysledek);
}
```

—

3. Práce s Assertací

Třída `Assert` v xUnit poskytuje širokou škálu metod pro ověření výsledků:

Metoda	**Účel**
`Assert.Equal(exp, act)`	Ověří rovnost hodnot.
`Assert.True(bool)`	Ověří, zda je výraz pravdivý.
`Assert.NotNull(obj)`	Ověří, že objekt není null.
`Assert.Throws<T>(...)`	Ověří, že kód vyvolá konkrétní výjimku typu T.
`Assert.Contains(val, coll)`	Ověří, že kolekce nebo řetězec obsahuje danou hodnotu.

—

4. Sdílení kontextu (Fixtures)

Protože xUnit vytváří pro každý test novou instanci třídy, pro sdílení „drahých“ prostředků (např. databázové spojení) používáme tzv. **Class Fixtures**:

```
1. Vytvoříte třídu (např. `DbFixture`), která se postará o inicializaci.
2. Testovací třída implementuje rozhraní `IClassFixture<DbFixture>`.
3. xUnit vytvoří instanci `DbFixture` pouze jednou pro celou testovací třídu
a předá ji do konstruktoru.
```

—

5. Integrace a spouštění

Testy xUnit lze spouštět mnoha způsoby:

- **IDE:** Integrovaný Test Explorer ve Visual Studiu nebo Rideru.
- **CLI:** Pomocí příkazu `dotnet test`.
- **CI/CD:** Automatizované spouštění v rámci GitHub Actions, Azure DevOps nebo Jenkins.

Související články:

- Unit Testing a kvalita kódu
- Mockování v .NET (Moq, NSubstitute)
- Principy SOLID

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