

.NET-es alkalmazások verifikálása Visual Studio Test Explorer segítségével

Fekete Krisztián

Motiváció

- ▶ Valóban szükség van-e egység tesztek (unit test) írására?
- ▶ Szerteágazó, elburjánzó kódbázis
- ▶ Különböző esetek lefedése
- ▶ Fejlesztés csapatban (függőség)
- ▶ A GUI-t nem én fejlesztem, de az adatot én adom
- ▶ Bug javítási „overhead” lefedése

Visual Studio Test Explorer

- ▶ Test Driven Development támogatása integrálva a fejlesztőkörnyezettel
- ▶ Felügyelt és natív kód támogatása
- ▶ Microsoft unit testing framework (alapértelmezett) de 3rd party keretrendszerek is használhatóak (pl.: NUnit)
- ▶ Visual Studio Ultimate kell a teljes támogatáshoz
 - ▶ Pl.: tesztesetek futtatása minden fordítás után

Architektúra

Visual Studio Unit
Test Explorer

Command Line
Runner

TeamBuild Unit Test
Activity

Visual Studio Unit Test Platform

MS-Test
Managed

MS-Test
Native

NUnit

xUnit.net

QUnit

MORE!

Tesztelés felépítése

- ▶ **AAA** minta alkalmazása
 - ▶ **Arrange:** azon teszt objektumok inicializálása amelyeket validálni szeretnénk.
 - ▶ **Act:** a teszt metódus meghívása, futtatása.
 - ▶ **Assert:** a kimenet ellenőrzése; azt kaptuk amit vártunk-e?

Egy egyszerű példa

- ▶ Bevásárlókosárba termékeket helyezhetünk
- ▶ Két művelet:
 - ▶ Hozzáadás (Add)
 - ▶ Törlés (Remove)
- ▶ Tesztesetek:
 - ▶ Hozzáadás pozitív egész számmal
 - ▶ Hozzáadás negatív értékkel
 - ▶ Törlés: 0-nál nem lehet kevesebb darab a kosárban

Demó

Piros / Zöld
állapotsáv a
kimenet
alapján

Test Explorer

Search

Run All | Run...

Failed Tests (1)

- ✘ GetDataParsesReturnedJsonCorrectly 269 ms

Passed Tests (21)

- ✔ DefaultCtorCallsStartTimerWhenStartParamet... < 1 ms
- ✔ DefaultCtorCreatesInstanceOfUsgsServiceProxy < 1 ms
- ✔ EarthquakeDataCollectionImplementsINotify... < 1 ms
- ✔ EarthquakeDataPropertyClearsCollectionWhe... < 1 ms
- ✔ EarthquakeDataPropertySetsCollectionWhenGiv... 1 ms
- ✔ GetReturnsValidEarthquakeRecord 249 ms
- ✔ MapCenterPropertyRaisesChangedEventWhe... < 1 ms
- ✔ ProveThatUnitTestAndCodeCoverageAreProc... 132 ms
- ✔ RaisePropertyChangedFiresEventCorrectly 78 ms
- ✔ RefreshMessagePropertyRaisesChangedEvent... < 1 ms
- ✔ RefreshMethodRetrievesFromServiceAndSets... 130 ms

GetDataParsesReturnedJsonCorrectly

Source: no source available

✘ Test Failed - GetDataParsesReturnedJsonCorrectly

Message: Assert.Equal() Failure
Expected: 1/23/2012 2:50:14 PM
Actual: 1/23/2012 3:50:14 PM

Elapsed time: 269 ms

Stack Trace:

<GetDataParsesReturnedJsonCorrectly>d_2.MoveN

Keresés

A sikertelen
tesztesetek
kerülnek
felülre

Tesztelés
időtartama

Tesztelés
adatai

„Teszteld mielőtt írod”

- ▶ Ha a tesztelés mentén akarjuk felépíteni a logikát, lehetőség van arra is, hogy a tesztelő metódusból generáljuk az „éles” kód vázát

```
[TestMethod]
public void TestOrder()
{
    ManagerCart manager = new ManagerCart();

    List<Product> products = new List<Product> { TestProduct };

    manager.Order(products);
}
```

'Shop.ManagerCart' does not contain a definition for 'Order' and no extension method 'Order' accepting a first argument of type 'Shop.ManagerCart'

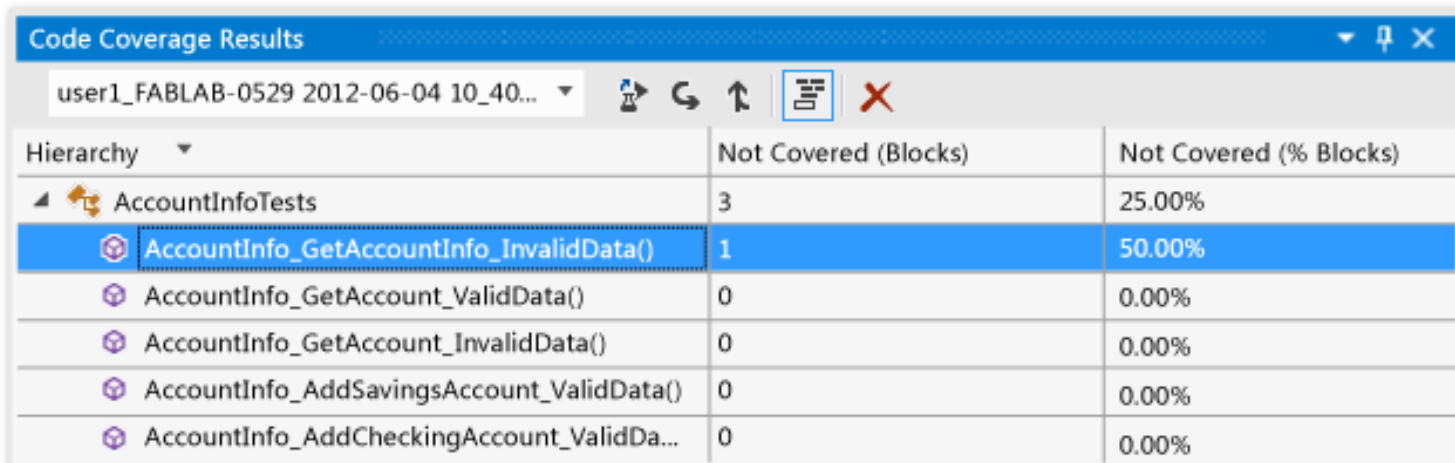
Adatorientált unit test-ek írása

- ▶ Előfordulhat, hogy adatbázisban tárolt adatokon akarjuk a működést tesztelni.
 - ▶ Hogyan olvassuk ki az adatokat?
 - ▶ Ha 1000 sort tárolunk, írjuk meg ezerszer a teszt esetet?
- ▶ Van megoldás: Data Driven Test teszt esetek írásával
 - ▶ Elég egy teszt metódust írunk
 - ▶ Az adatbázis minden egyes sorára lefut
 - ▶ Ha valamelyik sorban hiba történik, a teszt sikertelen lesz

```
[DataSource(
@"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Projects\MyBank\TestData\AccountsTest.accdb",
"AddIntegerHelperData")]
[TestMethod()]
public void AddIntegerHelper_DataDrivenValues_AllShouldPass()
{
    var target = new CheckingAccount();
    int x = Convert.ToInt32(TestContext.DataRow["FirstNumber"]);
    int y = Convert.ToInt32(TestContext.DataRow["SecondNumber"]);
    int expected = Convert.ToInt32(TestContext.DataRow["Sum"]);
    int actual = target.AddIntegerHelper(x, y);
    Assert.AreEqual(expected, actual);
}
```

Mennyire megbízhatóak az eredmények?

- ▶ ~ A tesztesetek során a kódsorok hány %-a kerül futtatásra?
- ▶ ~ Mennyire gondoltunk minden „eshetőségre”?
- ▶ Ezt is monitorozhatjuk, az „Analyze code coverage” funkció segítségével

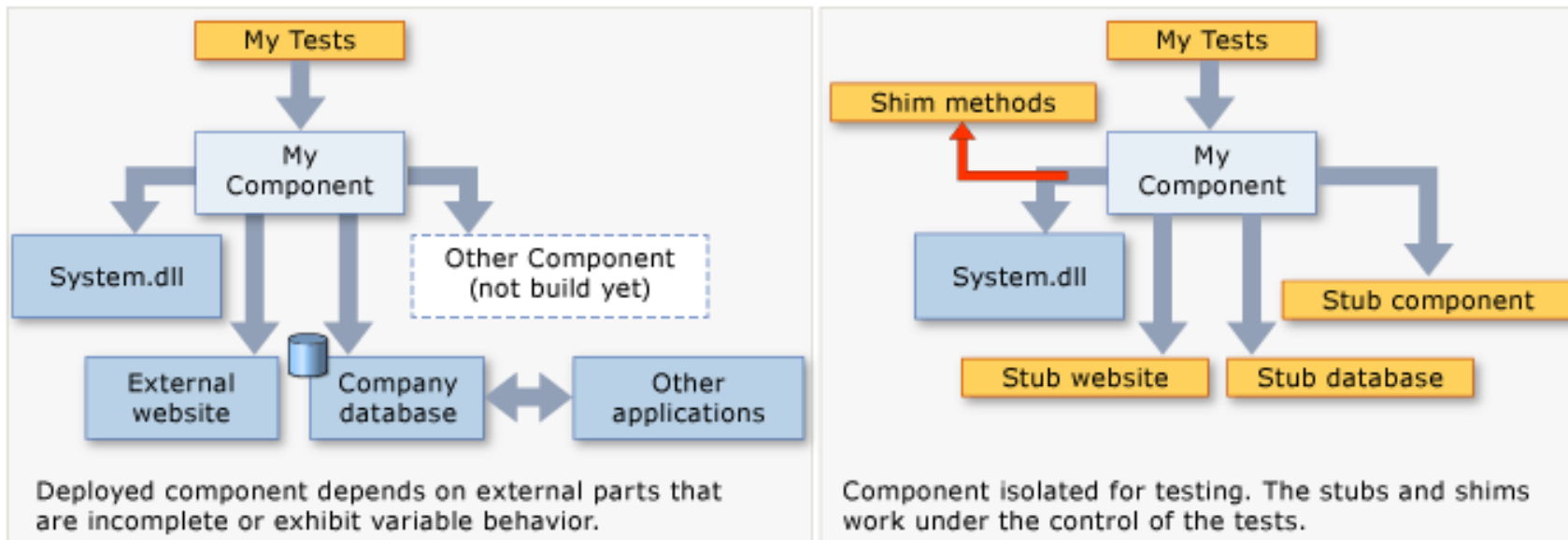


The screenshot shows a window titled "Code Coverage Results" with a toolbar and a table. The table has three columns: "Hierarchy", "Not Covered (Blocks)", and "Not Covered (% Blocks)". The data is as follows:

Hierarchy	Not Covered (Blocks)	Not Covered (% Blocks)
AccountInfoTests	3	25.00%
AccountInfo_GetAccountInfo_InvalidData()	1	50.00%
AccountInfo_GetAccount_ValidData()	0	0.00%
AccountInfo_GetAccount_InvalidData()	0	0.00%
AccountInfo_AddSavingsAccount_ValidData()	0	0.00%
AccountInfo_AddCheckingAccount_ValidDa...	0	0.00%

Komplexebb esetek

- ▶ Mi történik ha olyan összetevők szükségesek a sikeres teszthez, amelyek a teszt(ek) futtatásakor (még) nem állnak rendelkezésre (weboldal, adatbázis, külső - még meg nem épített - komponens).
- ▶ Megoldás: Microsoft Fake szerelvények (assembly) használata



Microsoft Fakes

- ▶ Stub (csonk):
 - ▶ Interfész alapú fejlesztést követel
 - ▶ „Szimulálunk” egy komponenst amely
 - ▶ még nem készült el vagy
 - ▶ nem működőképes a teszt környezetben
 - ▶ „Saját” projekten belül használjuk
- ▶ Shim (alátét):
 - ▶ Általunk nem módosítható komponensek helyettesítése futási időben
 - ▶ Pl.: System.dll
 - ▶ Lassabb mint a Stub

Stub (példa)

```
[TestClass]
class TestStockAnalyzer
{
    [TestMethod]
    public void TestContosoStockPrice()
    {
        // Arrange:

        // Create the fake stockFeed:
        IStockFeed stockFeed =
            new StockAnalysis.Fakes.StubIStockFeed() // Generated by Fakes.
            {
                // Define each method:
                // Name is original name + parameter types:
                GetSharePriceString = (company) => { return 1234; }
            };

        // In the completed application, stockFeed would be a real one:
        var componentUnderTest = new StockAnalyzer(stockFeed);

        // Act:
        int actualValue = componentUnderTest.GetContosoPrice();

        // Assert:
        Assert.AreEqual(1234, actualValue);
    }
    ...
}
```

Shim (példa)

```
[TestClass]
public class TestClass1
{
    [TestMethod]
    public void TestCurrentYear()
    {
        int fixedYear = 2000;

        // Shims can be used only in a ShimsContext:
        using (ShimsContext.Create())
        {
            // Arrange:
            // Shim DateTime.Now to return a fixed date:
            System.Fakes.ShimDateTime.NowGet =
            () =>
            { return new DateTime(fixedYear, 1, 1); };

            // Instantiate the component under test:
            var componentUnderTest = new MyComponent();

            // Act:
            int year = componentUnderTest.GetTheCurrentYear();

            // Assert:
            // This will always be true if the component is working:
            Assert.AreEqual(fixedYear, year);
        }
    }
}
```

Érdekeség, tapasztalatok

- ▶ A „Móricka példa” mindig egyszerű, a való világban minden máshogy megy
- ▶ Az elején lelkesek vagyunk, de ha sok a „macera” hajlamosak vagyunk nem tesztelni
- ▶ Példa:
 - ▶ Weboldalak unit tesztelése (DAL-tól a renderelésig)
 - ▶ Egy SESSION-t hogyan szimulálok?
 - ▶ Egy autentikált SESSION-t hogyan szimulálok?
 - ▶ Külső komponens: Moq használata

Köszönöm a figyelmet!