عنوان: **آشنایی با Should Library**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۸/۱۵ ۱۲:۱۵
آدرس: www.dotnettips.info
برچسب‌ها: Unit testing, TDD, Should Library

نوشتن Assert در کدهای تست، وابستگی مستقیم به انتخاب کتابخانه تست دارد. برای مثال:

:NUnit

```
using NUnit.Framework;
using NUnit.Framework.SyntaxHelpers;

namespace TestLibrary
{
   [TestFixture]
   public class MyTest
   {
       [Test]
       public void Test1()
       {
           var expectedValue = 2;
           Assert.That(expectedValue , Is.EqualTo(2));
       }
   }
}
```

: Microsoft UnitTesting

```
using Microsoft.VisualStudio.TestTools.UnitTesting ;

namespace TestLibrary
{
   [TesClass]
   public class MyTest
   {
       [TestMethod]
       public void Test1()
       {
           var expectedValue = 2;
           Assert.AreEqual (expectedValue , 2);
       }
   }
}
```

کدهای Assert نوشته شده در مثال بالا با توجه به فریم ورک مورد استفاده متفاوت است. در حالی که کتابخانه Should، مجموعه ای از Extension Method هاست برای قسمت Assert در UnitTestهای نوشته شده. با استفاده از این کتابخانه دیگر نیازی به نوشتن Assert به سبک و سیاق فعلی نیست. کدهای Assert بسیار **خواناتر و قابل درک** خواهند بود و از طرفی وابستگی به سایر کتابخانه‌های تست از بین خواهد رفت.
نکته: مورد استفاده این کتابخانه فقط در قسمت Assert کدهای تست است و استفاده از سایر کتابخانه‌های جانبی الزامی است.
این کتابخانه به دو صورت مورد استفاده قرار می‌گیرد:
»Standard که باید از Should.dll استفاده نمایید؛
»Fluent که باید از Should.Fluent.dll استفاده نمایید؛(پیاده سازی همان فریم ورک Should به صورت Static Reflection)

نصب کتابخانه Should با استفاده از nuget (آخرین نسخه آن در حال حاضر 1.1.20 است ) :

```
Install-Package Should
```

نصب کتابخانه Should.Fluent با استفاده از nuget(آخرین نسخه آن در حال حاضر 1.1.19 است):

```
Install-Package ShouldFluent
```

در ابتدا همان مثال قبلی را با این کتابخانه بررسی خواهیم کرد:

```
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace TestLibrary
{
   [TesClass]
   public class MyTest
   {
       [TestMethod]
       public void Test1()
       {
           var expectedValue = 2;
           expectedValue.Should().Equal( 2 );
       }
   }
}
```

در نگاه اول چیز خاصی به چشم نمی‌خورد، اما اگر از این پس قصد داشته باشیم کدهای تست خود را تحت فریم ورک NUnit پیاده سازی کنیم در قسمت Assert کدهای خود هیچ گونه خطایی را مشاهده نخواهیم کرد.

مثال:

```
[TestMethod]
public void AccountConstructorTest()
{
     const int expectedBalance = 1000;
     Account bankAccount = new Account();

     // Assert.IsNotNull(bankAccount, "Account was null.");
    // Assert.AreEqual(expectedBalance, bankAccount.AccountBalance, "Account balance not mathcing");



    bankAccount.ShouldNotBeNull("Account was null");
    bankAccount.AccountBalance.ShouldEqual(expectedBalance, "Account balance not matching");
}
```

در مثال بالا ابتدا با استفاده از Ms UnitTesting دو Assert نوشته شده است سپس در خطوط بعدی همان دو شرط با استفاده از کتابخانه Should نوشتم. در ذیل چند مثال از استفاده این کتابخانه (البته نوع Fluent آن) در هنگام کار با رشته ها، آبجکت ها، boolean و Collectionها را بررسی خواهیم کرد:

**Should.Fluent#**

```
public void Should_fluent_assertions()
{
    object obj = null;
    obj.Should().Be.Null();

    obj = new object();
    obj.Should().Be.OfType(typeof(object));
    obj.Should().Equal(obj);
    obj.Should().Not.Be.Null();
    obj.Should().Not.Be.SameAs(new object());
    obj.Should().Not.Be.OfType<string>();
    obj.Should().Not.Equal("foo");

    obj = "x";
    obj.Should().Not.Be.InRange("y", "z");
    obj.Should().Be.InRange("a", "z");
    obj.Should().Be.SameAs("x");

    "This String".Should().Contain("This");
    "This String".Should().Not.Be.Empty();
    "This String".Should().Not.Contain("foobar");

    false.Should().Be.False();
    true.Should().Be.True();
```

```
    var list = new List<object>();
    list.Should().Count.Zero();
    list.Should().Not.Contain.Item(new object());

    var item = new object();
    list.Add(item);
    list.Should().Not.Be.Empty();
    list.Should().Contain.Item(item);
};
```

#مثال‌های استفاده از متغیرهای DateTime و Guid

```
public void Should_fluent_assertions()
{
    var id = new Guid();
    id.Should().Be.Empty();

    id = Guid.NewGuid();
    id.Should().Not.Be.Empty();

    var date = DateTime.Now;
    date1.Should().Be.Today();

    var str = "";
    str.Should().Be.NullOrEmpty();

    var one = "1";
    one.Should().Be.ConvertableTo<int>();

    var idString = Guid.NewGuid().ToString();
    idString.Should().Be.ConvertableTo<Guid>();
}
```