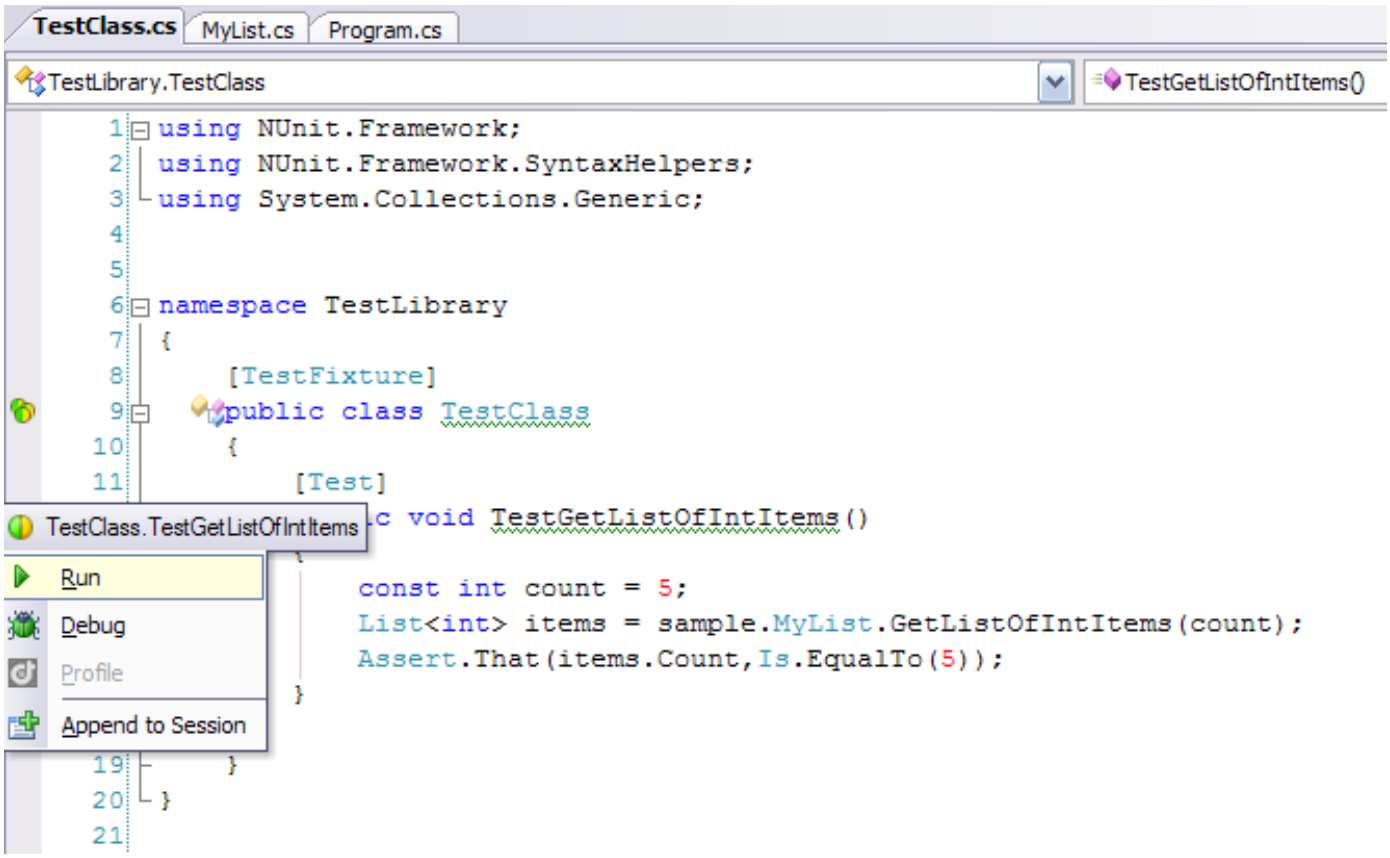
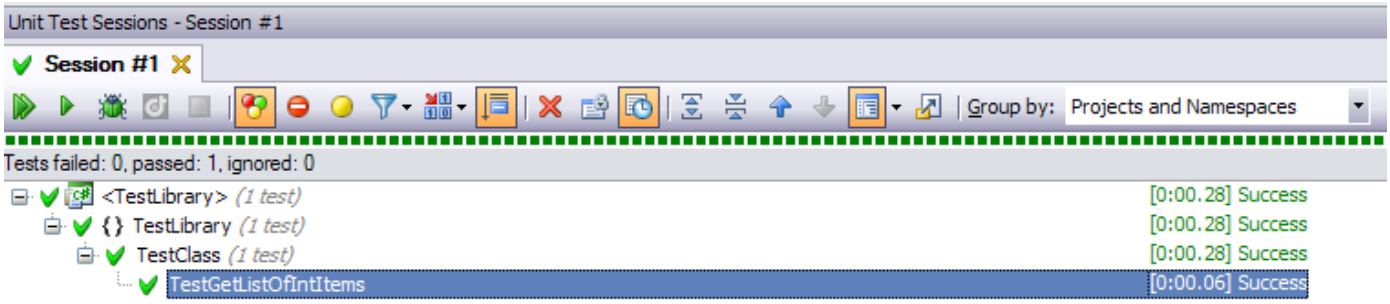
عنوان: **آشنایی با آزمایش واحد (unit testing) در دات نت، قسمت 4**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۱۳ ۰۰:۲۳:۲۸
آدرس: www.dotnettips.info
برچسب‌ها: Unit testing

ادامه آشنایی با NUnit

اگر قسمت سوم را دنبال کرده باشید احتمالا از تعداد مراحلی که باید در خارج از IDE صورت گیرد گلایه خواهید کرد (کامپایل کن، اجرا کن، اتچ کن، باز کن، ذخیره کن، اجرا کن و ...). خوشبختانه افزونه ReSharper این مراحل را بسیار ساده و مجتمع کرده است. این افزونه به صورت خودکار متدهای آزمایش واحد یک پروژه را تشخیص داده و آنها را با آیکون‌هایی (Gutter icons) متمایز مشخص می‌سازد (شکل زیر). پس از کلیک بر روی آن‌ها ، امکان اجرای آزمایش یا حتی دیباگ کردن سطر به سطر یک متد آزمایش واحد درون IDE ویژوال استودیو وجود خواهد داشت.



برای نمونه پس از اجرای آزمایش واحد قسمت قبل، نتیجه حاصل مانند شکل زیر خواهد بود:



راه دیگر، استفاده از افزونه TestDriven.NET است که نحوه استفاده از آن‌را اینجا می‌توانید ملاحظه نمائید. به منوی جهنده کلیک راست بر روی یک صفحه، گزینه run tests را اضافه می‌کند و نتیجه حاصل را در پنجره output ویژوال استودیو نمایش می‌دهد.

ساختار کلی یک کلاس آزمایش واحد مبتنی بر NUnit framework :

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

namespace TestLibrary
{
   [TestFixture]
   public class Test2
   {
       [SetUp]
       public void MyInit()
       {
           //کدی که در این قسمت قرار می‌گیرد پیش از اجرای هر متد تستی اجرا خواهد شد
       }

       [TearDown]
       public void MyClean()
       {
           //کدی که در این قسمت قرار می‌گیرد پس از اجرای هر متد تستی اجرا خواهد شد
       }

       [TestFixtureSetUp]
       public void MyTestFixtureSetUp()
       {
           // کدی که در اینجا قرار می‌گیرد در ابتدای بررسی آزمایش واحد و فقط یکبار اجرا می‌شود
       }

       [TestFixtureTearDown]
       public void MyTestFixtureTearDown()
       {
           // کدهای این قسمت در پایان کار یک کلاس آزمایش واحد اجرا خواهند شد
       }

       [Test]
       public void Test1()
       {
           //بدنه آزمایش واحد در اینجا قرار می‌گیرد
           Assert.That(2, Is.EqualTo(2));
       }
   }
}
```

شبیه به روال‌های رخداد گردان load و close یک فرم، یک کلاس آزمایش واحد NUnit نیز دارای ویژگی‌های TestFixtureSetUp و TestFixtureTearDown است که در ابتدا و انتهای آزمایش واحد اجرا خواهند شد (برای درک بهتر موضوع و دنبال کردن نحوه‌ی اجرای این روال‌ها، داخل این توابع break point قرار دهید و با استفاده از ReSharper ، آزمایش را در حالت دیباگ آغاز کنید)، یا SetUp و TearDown که در زمان آغاز و پایان بررسی هر متد آزمایش واحدی فراخوانی می‌شوند.
همانطور که در قسمت قبل نیز ذکر شد، به امضاهای متدها و کلاس فوق دقت نمائید (عمومی ، void و بدون آرگومان ورودی).

بهتر است از ویژگی‌های SetUp و TearDown با دقت استفاده نمود. عموما هدف از این روال‌ها ایجاد یک شیء و تخریب و پاک سازی آن است. حال اینکه این روال‌ها قبل و پس از اجرای هر متد آزمایش واحدی فراخوانی می‌شوند. بنابراین به این موضوع دقت داشته باشید.

همچنین توصیه می‌شود که کلاس‌های آزمایش واحد را در اسمبلی دیگری مجزا از پروژه اصلی پیاده سازی کنید (برای مثال یک پروژه جدید از نوع class library)، زیرا این موارد مرتبط با بررسی کیفیت کدهای شما هستند که موضوع جداگانه‌ای نسبت به پروژه اصلی محسوب می‌گردد (نحوه پیاده سازی آن‌را در قسمت قبل ملاحظه نمودید). همچنین در یک پروژه تیمی این جدا سازی، مدیریت آزمایشات را ساده‌تر می‌سازد و بعلاوه سبب حجیم شدن بی‌مورد اسمبلی‌های اصلی محصول شما نیز نمی‌گردند.

ادامه دارد...

نظرات خوانندگان

نویسنده: افشار محبی
تاریخ: ۱۳۸۸/۰۹/۲۳ ۱۰:۱۴:۲۹

ای کاش می‌شد در TestFixtureSetup به دیتابیس متصل شد. حتی اگر اجرای تست کند شود مهم نیست چون بررسی درستی بعضی عملیات مرتبط با دیتابیس خیلی مهم‌تر از زمان اجرای تست است.

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۹/۲۳ ۱۲:۱۴:۳۶

- در NHibernate برای این نوع تست‌ها تا جایی که دیدم از دیتابیس SQLite تشکیل شده در حافظه استفاده می‌کنند. به این صورت مزایای سرعت و همچنین حذف خودکار داده‌ها پس از پایان کار برقرار است.
- ضمناآزمایش واحدی که از مرزهای برنامه خارج شود دیگر آزمایش واحد نام ندارد به همین جهت mocking frameworks برای این نوع کارها ایجاد شده است. (برای کار با دیتابیس، کار با smtp server ، کار با فایل سیستم و مواردی از این دست)

نویسنده: افشار محبی
تاریخ: ۱۳۸۸/۰۹/۲۳ ۱۴:۴۶:۵۶

آره، mocking framework و ابزارهای تست دیتابیس کارهای جالب و قشنگی می‌کنند. من هم فهمیدم آن چیزی که بهش نیاز دارم همان integration test است نه unit test.
در NUnit همه کاری می‌شود انجام داد حتی اتصال به دیتابیس (البته اسمش می‌شود integration test) و راهش هم اضافه کردن app.config یا web.config به همان پروژه class library مخصوص تست است. راه این کار هم در خیلی جاها گفته شده ولی اگر مثل من در خواندن و استفاده از app.config در برنامه دچار مشکل شدید به لینک زیر مراجعه کنید:
http://david.givoni.com/blog/?p=4