عنوان: **آشنایی با آزمایش واحد (unit testing) در دات نت، قسمت 3**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۰/۰۹ ۱۵:۱۹:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Unit testing

آشنایی با NUnit

NUnit یکی از فریم ورک‌های آزمایش واحد سورس باز مخصوص دات نت فریم ورک است. (کلا در دات نت هرجایی دیدید که N ، به ابتدای برنامه‌ای یا کتابخانه‌ای اضافه شده یعنی نمونه منتقل شده از محیط جاوا به دات نت است. برای مثال NHibernate از Hibernate جاوا گرفته شده است و الی آخر)
این برنامه با سی شارپ نوشته شده است اما تمامی زبان‌های دات نتی را پشتیبانی می‌کند (اساسا با زبان نوشته شده کاری ندارد و فایل اسمبلی برنامه را آنالیز می‌کند. بنابراین فرقی نمی‌کند که در اینجا چه زبانی بکار گرفته شده است).

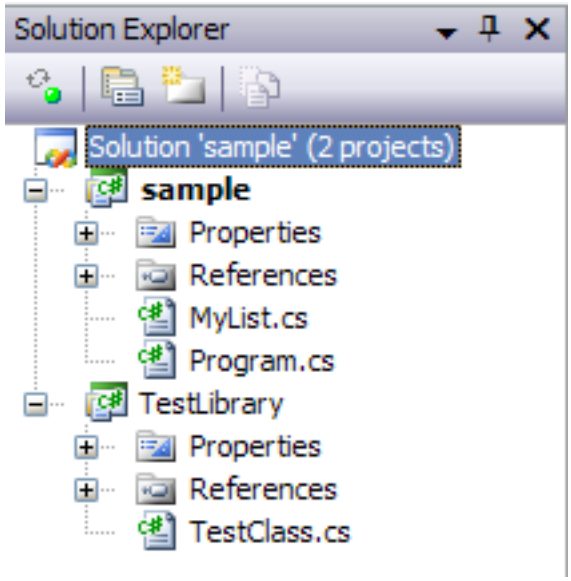
ابتدا NUnit را دریافت نمائید:
http://nunit.org/index.php?p=download

یک برنامه ساده از نوع console را در VS.net آغاز کنید.
کلاس MyList را با محتوای زیر به پروژه اضافه کنید:

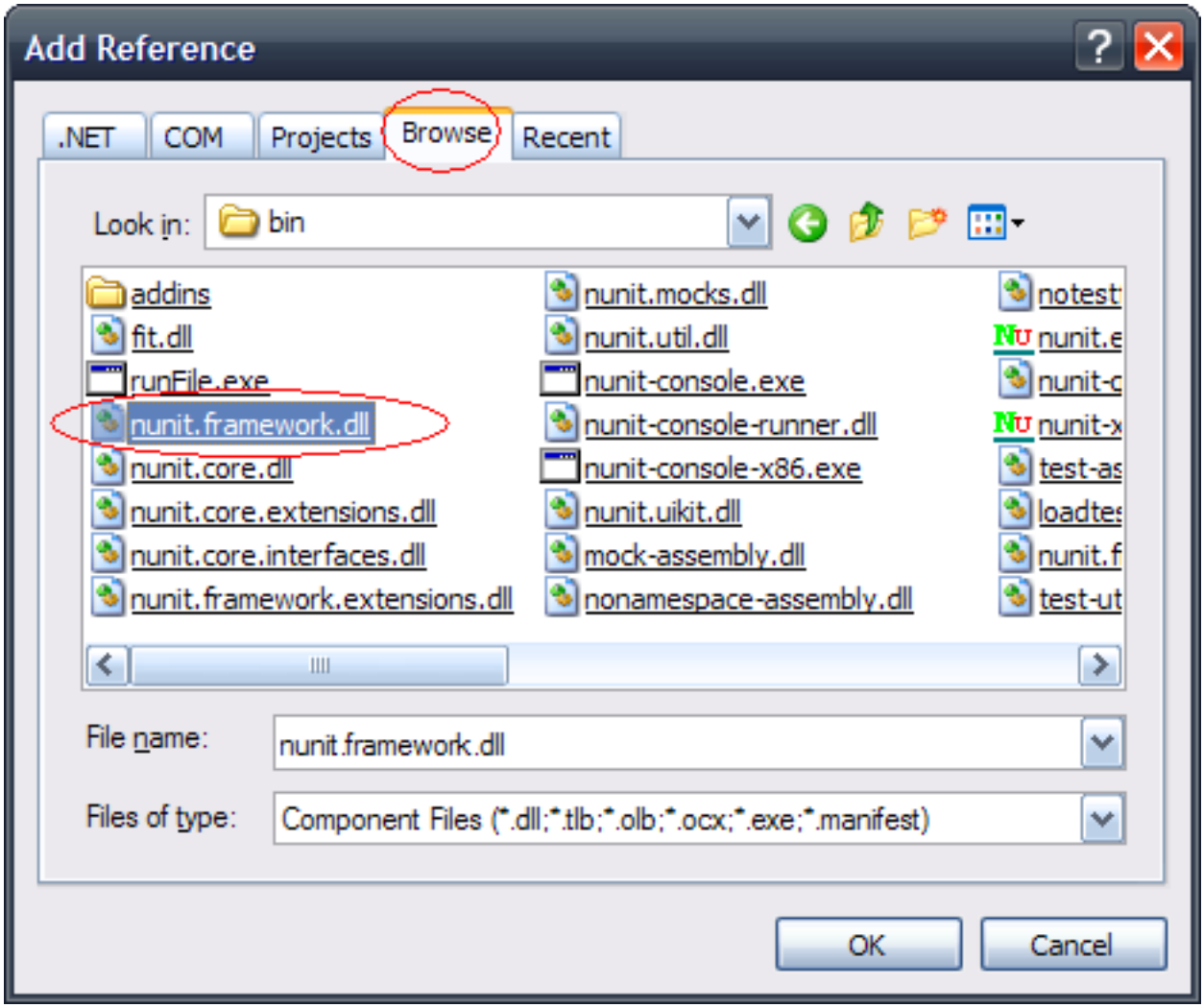
```
using System.Collections.Generic;

namespace sample
{
   public class MyList
   {
       public static List<int> GetListOfIntItems(int numberOfItems)
       {
           List<int> res = new List<int>();

           for (int i = 0; i < numberOfItems; i++)
               res.Add(i);

           return res;
       }
   }

}
```

یکبار پروژه را کامپایل کنید.

اکنون بر روی نام پروژه در قسمت solution explorer کلیک راست کرده و گزینه add->new project را انتخاب کنید. نوع این پروژه را که متدهای آزمایش واحد ما را تشکیل خواهد داد، class library انتخاب کنید. با نام مثلا TestLibrary (شکل زیر).



با توجه به اینکه NUnit ، اسمبلی برنامه (فایل exe یا dll آن‌را) آنالیز می‌کند، بنابراین می‌توان پروژه تست را جدای از پروژه اصلی ایجاد نمود و مورد استفاده قرار داد.
پس از ایجاد پروژه class library ، باید ارجاعی از NUnit framework را به آن اضافه کنیم. به محل نصب NUnit مراجعه کرده (پوشه bin آن) و ارجاعی به فایل nunit.framework.dll را به پروژه اضافه نمائید (شکل زیر).



سپس فضاهای نام مربوطه را به کلاس آزمایش واحد خود اضافه خواهیم کرد:

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

اولین نکته‌ای را که باید در نظر داشت این است که کلاس آزمایش واحد ما باید Public باشد تا در حین آنالیز اسمبلی پروژه توسط NUint، قابل دسترسی و بررسی باشد.
سپس باید ویژگی جدیدی به نام TestFixture را به این کلاس اضافه کرد.

```
[TestFixture]
   public class TestClass
```

این ویژگی به NUnit می‌گوید که در این کلاس به دنبال متدهای آزمایش واحد بگرد. (در NUnit از attribute ها برای توصیف عملکرد یک متد و همچنین دسترسی runtime به آن‌ها استفاده می‌شود)
سپس هر متدی که به عنوان متد آزمایش واحد نوشته می‌شود، باید دارای ویژگی Test باشد تا توسط NUnit بررسی گردد:

```
[Test]
       public void TestGetListOfIntItems()
```

نکته: متد Test ما باید public و از نوع void باشد و همچنین هیچ پارامتری هم نباید داشته باشد.

اکنون برای اینکه بتوانیم متد GetListOfIntItems برنامه خود را در پروژه دیگری تست کنیم، باید ارجاعی را به اسمبلی آن اضافه کنیم. همانند قبل، از منوی project گزینه add reference ، فایل exe برنامه کنسول خود را انتخاب کرده و ارجاعی از آن‌را به پروژه class library اضافه می‌کنیم. بدیهی است امکان اینکه کلاس تست در همان پروژه هم قرار می‌گرفت وجود داشت و صرفا جهت جداسازی آزمایش از برنامه اصلی این‌کار صورت گرفت.
پس از این مقدمات، اکنون متد آزمایش واحد ساده زیر را در نظر بگیرید:

```
[Test]
       public void TestGetListOfIntItems()
       {
           const int count = 5;
           List<int> items = sample.MyList.GetListOfIntItems(count);
           Assert.That(items.Count,Is.EqualTo(5));
       }
```

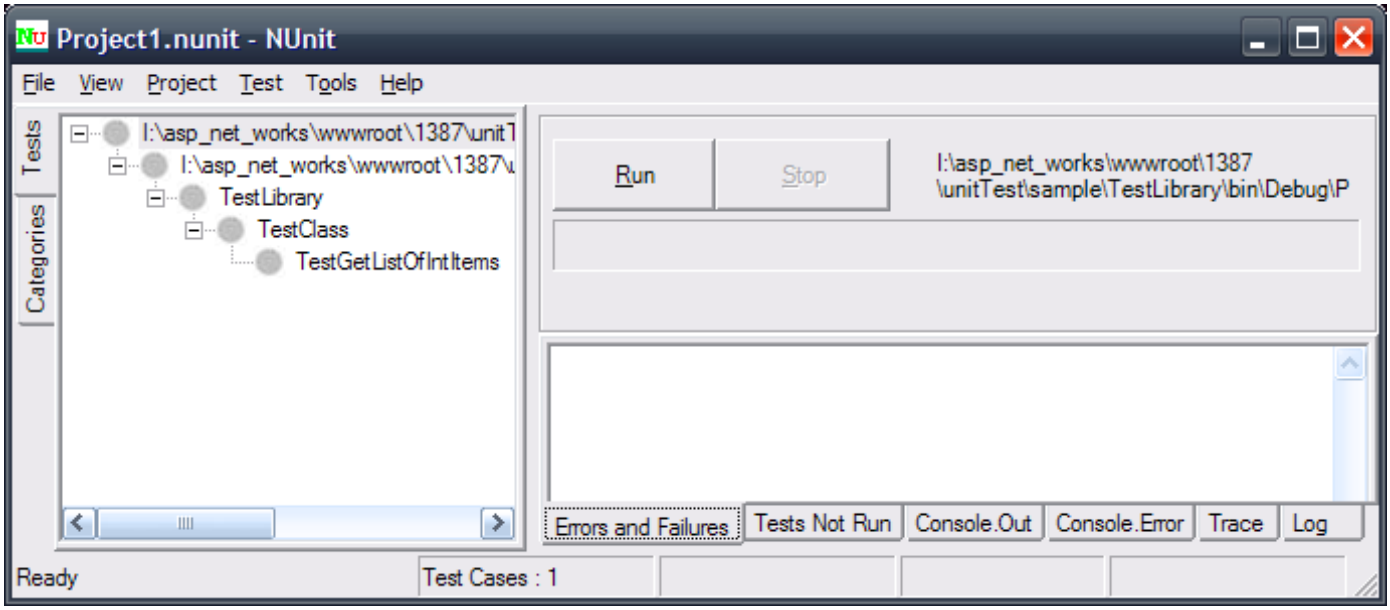
قصد داریم بررسی کنیم آیا متد GetListOfIntItems واقعا همان تعداد آیتمی را که باید برگرداند، بازگشت می‌دهد؟ عدد 5 به آن پاس شده است و در ادامه قصد داریم بررسی کنیم، count شیء حاصل (items در اینجا) آیا واقعا مساوی عدد 5 است؟
اگر آن را (سطر مربوط به Assert را) کلمه به کلمه بخواهیم به فارسی ترجمه کنیم به صورت زیر خواهد بود:
می‌خواهیم اثبات کنیم که count مربوط به شیء items مساوی 5 است.

پس از اضافه کردن متد فوق، پروژه را کامپایل نمائید.

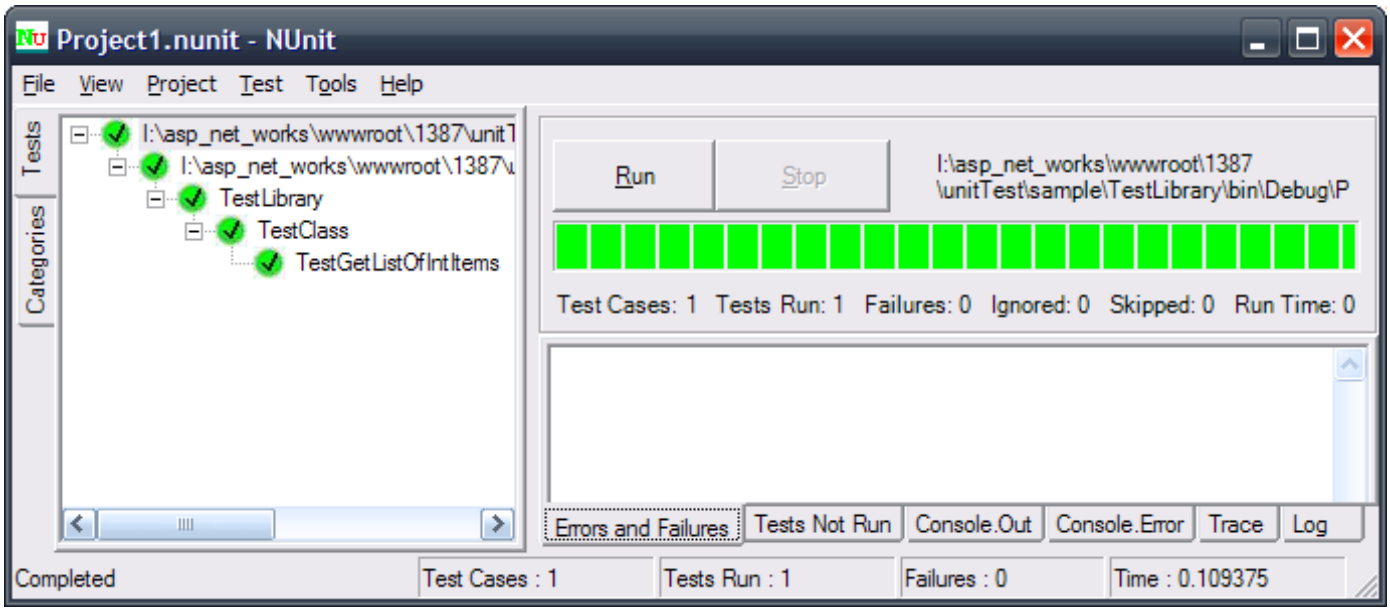
اکنون برنامه nunit.exe را اجرا کنید تا NUnit IDE ظاهر شود (در همان دایرکتوری bin مسیر نصب NUnit قرار دارد).
از منوی File آن یک پروژه جدید را آغاز نموده و آنرا ذخیره کنید.
سپس از منوی project آن، با استفاده از گزینه add assembly ، فایل dll کتابخانه تست خود را اضافه نمائید.
احتمالا پس از انجام این عملیات بلافاصله با خطای زیر مواجه خواهید شد:

```
---------------------------
Assembly Not Loaded
---------------------------
System.ApplicationException : Unable to load TestLibrary because it is not located under
the AppBase
----> System.IO.FileNotFoundException : Could not load file or assembly
'TestLibrary' or one of its dependencies. The system cannot find the file specified.
For further information, use the Exception Details menu item.
```

این خطا به این معنا است که پروژه جدید NUnit باید دقیقا در همان پوشه خروجی پروژه، جایی که فایل dll کتابخانه تست ما تولید شده است، ذخیره گردد. پس از افزودن اسمبلی، نمای برنامه NUnit باید به شکل زیر باشد:



همانطور که ملاحظه می‌کنید، NUnit با استفاده از قابلیت‌های reflection در دات نت، اسمبلی را بارگذاری می‌کند و تمامی کلاس‌هایی که دارای ویژگی TestFixture باشند در آن لیست خواهد شد.
اکنون بر روی دکمه run کلیک کنید تا اولین آزمایش ما انجام شود. (شکل زیر)



رنگ سبز در اینجا به معنای با موفقیت انجام شدن آزمایش است.

ادامه دارد...

نظرات خوانندگان

نویسنده: Anonymous
تاریخ: ۱۳۸۷/۱۱/۲۱ ۰۸:۱۲:۰۰

یک دید کلی نسبت به توابع تستینگ لازم داشتم فوری! که این مطلب به موقع به دادم رسید.
ایوالله.

نویسنده: محمد آزاد
تاریخ: ۱۳۸۸/۰۸/۱۲ ۱۰:۳۳:۴۸

سلام...
آقای نصیری من با
;using NUnit.Framework.SyntaxHelpers
مشکل دارم ...این فضای نام رو ندارم...اما اولیشو دارم..

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۸/۱۲ ۱۲:۵۳:۴۷

سلام
حق با شما است. مطابق مستندات نگارش آخر آن
.NUnit.Framework.SyntaxHelpers namespace no longer exists
.All classes that were in this namespace have been moved to the NUnit.Framework namespace

به این معنا که SyntaxHelpers الان با همان using NUnit.Framework به پروژه شما الحاق می‌شود (به این فضای نام منتقل شده).