عنوان: **Test Driven Development #2**
نویسنده: شهروز جعفری
تاریخ: ۱۳۹۲/۰۳/۰۹ ۲۰:۲۰
آدرس: www.dotnettips.info
برچسب‌ها: Unit testing, TDD

در مطلب قبل شما با TDD آشنا شدید اکنون بهتر است با یک مثال نشان دهم منظور از Test Driven Development چیست.
برای شروع کافی است یک پروژه کنسول ساخته و Nunit را از طریق کنسول Nuget نصب کنید.

PM> Install-Package NUnit

معمولا برای کلاس‌های تست یک پروژه جدا در نظر گرفته می‌شود، ولی برای شروع می‌توانید از همان پروژه اصلی استفاده کنید.
پس از نصب شدن Nunit می توانیم شروع به ساختن کلاس‌های تست کنیم:

```
[TestFixture]
    public class HelloWorldTest
    {
    }
```

همانطور که ملاحظه می‌کنید کلاس ما با Attribute به نام TestFixture مزین شده است که خاص فریمورک Nunit است، در صورتی که از فریمورک دیگری برای تست استفاده می‌کنید باید تنظیمات مربوط به آن را انجام دهید.متدهای تست ما نیز با Attribute به نام Test مزین می‌شوند.

```
[Test]
   public void ShouldSayHelloWorld()
  {
  }
```

همانطور که دقت کردید متد ما به صورتی نام گذاری شده است که مشخص کننده کاری باشد که قرار است انجام دهد.این یکی دیگر از مزایای تست نویسی است که یک داکیومنت تقریباً کامل در طول تولید نرم افزار ایجاد میشود.همچنین متد تست باید غیر استاتیک با خروجی void باشد .متدهای تست بهتر است فقط یک موضوع را تست کنند، به طور مثال نباید هم اضافه شدن یک رکورد و هم ریدایرکت شدن به صفحه ای خاص را تست کرد .
حالا وقت آن است که قبل از نوشتن کد اول تستش را بنویسیم.

```
[Test]
        public void ShouldSayHelloWorld()
        {
            const string result = "Hello World";
            Assert.AreEqual(result, HelloWorld.SayHello());
        }
```

کلاس Assert شامل توابعی بسیار قدرتمند است که مارا در اجرای تست بهتر کمک میکند.شامل متد هایی مانند .

AreEqual

AreNotEqual

AreNotSame

AssertDoublesAreEqual

Contains

DoesNotThrow

Equals

Fail

Greater

GreaterOrEqual

Ignore

IsEmpty

IsInstanceOf

IsNaN

IsNotNull

True

...

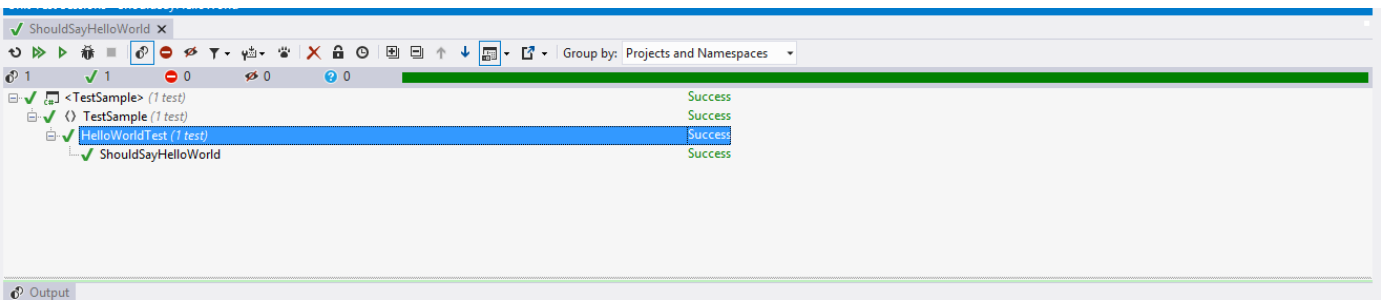
است.

هر کدام از متدهای بالا کاربرد خاصی را دارند که به طور جداگانه به آن می‌پردازیم.

به علت وجود نداشتن کلاس HelloWorld در زمان کامپایل با خطا مواجه می‌شویم.سپس کلاس مربوطه را ساخته و متد SayHello طوری پیاده سازی می‌کنیم که تست ما را پاس کند..(برنامه resharper برای اجرای متدهای تست بسیار کار آمد است)

```
public class HelloWorld
    {
        public static string SayHello()
        {
            return "Hello World";
        }
    }
```

حال دوباره تست را اجرا کرده و می‌بینید که تست ما پاس شد.



نیازی به مرحله ریفکتورینگ نیست زیرا کلاس ما به اندازه کافی ساده است.
برای مقایسه بین Nunit و ابزار توکار ویژوال استودیو می توانید به این سوال نگاهی بیاندازید.
در مطلب بعدی با استفاده از تست پذیری Mvc.net شروع به نوشتن تست هایی جدی‌تر خواهیم کرد.

نظرات خوانندگان

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۳/۱۰ ۲۲:۵۶

با تشکر. روش دوم بدون استفاده از ری شارپر:

در VS 2012 بعد از نصب NUnit Test adaptor ، میشه از Visual Studio 2012 Test Runner مستقیما برای کار با NUnit استفاده کرد.