عنوان: **Test Driven Development #3**
نویسنده: شهروز جعفری
تاریخ: ۱۳۹۲/۰۳/۱۱ ۲۱:۳۵
آدرس: www.dotnettips.info
برچسب‌ها: Unit testing, TDD

در پست قبلی با نوشتن یک تست ساده، با مفهوم TDD بیشتر آشنا شدیم .در این پست قصد بر این است که به وسیله Mvc.Net شروع به نوشتن تست‌های جدی‌تر کرده و از مزایای آن بهره ببریم .
برای شروع یک پروژه Mvc.Net ساخته و Nunit را در آن نصب می‌کنیم.
مدل زیر را در پوشه مدل‌ها می‌سازیم:

```
public class Idea
    {
        public static List<Idea> Ideas = new List<Idea>
            {
                new Idea{Content="سایتی که در ایده به اشتراک گذاشته شود",Title = "سایت ایده ها"},
                new Idea{Content="عینک گوگل را فارسی کنم",Title = "عینک گوگل"},
            };
        public string Content { get;  set;}
        public string Title { get; set; }
    }
```

```
[TestFixture]
    public class IdeaTest
    {
        [Test]
        public void ShouldDisplayListOfIdea()
        {
            var viewResult = new IdeaController().Index() as ViewResult;
            Assert.AreEqual(Idea.Ideas, viewResult.Model)
             Assert.IsNotNull(viewResult.Model);


        }
 }
```

کد بالا شامل مقایسه مقدار خروجی Action با لیستی از مدل Idea و همچنین اطمینان از خالی نبودن مدل ارسالی به view می‌باشد.در خط اول یک وهله از Controller می سازیم و Action مورد نظر را به شی از جنس ViewResult تبدیل(Cast) می‌کنیم پس از آن به وسیله viewResult.Model به مدلی که به سمت view پاس داده می‌شود دسترسی خواهیم داشت.اکنون اگر تست را اجرا کنیم با خطای کامپایل مواجه می‌شویم.حال Controller و Action مورد نظر را به صورتی که تست ما پاس شود پیاده سازی می‌کنیم.

```
public class IdeaController : Controller
    {

        public ActionResult Index()
        {
            return View(Idea.Ideas);
        }
      }
```

کد بالا مقدار Ideas را به view برمیگرداند.
در این دروره ما به تست کردن ویوها نخواهیم پرداخت.
تست بعدی تست ساده ای است که فقط میخواهیم از از وجود داشتن یک Action و نام view بازگشتی اطمینان حاصل کنیم.

```
[Test]
        public void ShouldLoadCreateIdeaView()
        {
            var viewResult = new IdeaController().Create() as ViewResult;
            Assert.AreEqual(string.Empty, viewResult.ViewName);
        }
```

در کد بالا مثل تست قبل، یک وهله از Controller می سازیم و سپس نام view بازگشتی را با string.Empty مقایسه میکنیم به این معنی که view خروجی Action ما نباید نامی داشته باشد و براساس قرار دادها باید هم نام اکشن باشد.
حال نوبت به پیاده سازی اکشن رسید.:

```
public ActionResult Create()
        {
            return View();
        }
```

در تست بعدی میخواهیم عملیات اضافه شدن یک Idea را به لیست بررسی کنیم:

```
[Test]
        public void ShouldAddIdeaItem()
        {
            var idea = new Idea { Title = "شبکه اجتماعی ", Content = " شبکه اجتماعی سینمایی" };
            var redirectToRouteResult = new IdeaController().Create(idea) as RedirectToRouteResult;
            Assert.Contains(idea, Idea.Ideas);
            Assert.AreEqual("Index",redirectToRouteResult.RouteValues["action"]);
        }
```

تست بالا نیز مانند دو تست قبل است با این تفاوت که مخواهیم ریدارکت شدن به یک Action خاص را نیز تست کنیم.برای همین مقدار خروجی را به RedirectToRouteResult تبدیل می‌کنیم.در ادامه یک Idea جدید ساخته و به لست اضافه میکنیم سپس از وجود داشتن آن در لیست Ideas اطمینان حاصل می‌کنیم.در خط آخر نیز نام Action که انتظار داریم بعد از اضافه شدن یک Idea ,کاربر به آن هدایت شود را ست می‌کنیم.
پیاده سازی Action به شکل زیر است:

```
public ActionResult Create(Idea idea)
        {
            Idea.Ideas.Add(idea);
            return RedirectToAction("Index");
        }
```

در این پست شما با مدل تست نویسی برایMvc.Net آشنا شدید.در مطلب بعدی شما با تست حذف و اصلاح Ideas آشنا خواهید شد.

نظرات خوانندگان

نویسنده: دنیس ریچی
تاریخ: ۱۳۹۲/۰۳/۱۸ ۱۵:۵۹

بسیار عالی بود. لطفن ادامه بدید. اگه میشه در قسمتهای بعدی راجع به TDD کار کردن برای جاوا اسکریپت در ویوها و QUnit هم توضیح بدید

نویسنده: s.t
تاریخ: ۱۳۹۲/۰۵/۰۹ ۱۷:۵۲

بسیار عالی،
منتظر ادامه‌ی این مبحث هستیم