


اکثر برنامه نویسان با مباحث Unit Testing آشنایی دارند و بعضی برنامه نویسان هم، از این مباحث در پروژه‌های خود استفاده می‌کنند. ساختار الگوهای MVC و MVVM به گونه ای است که به راحتی می‌توان برای این گونه پروژه‌ها Unit Test بنویسیم. در پروژه‌های MVC به دلیل عدم وابستگی بین View و Controller به طور مستقیم، امکان نوشتن Unit Test برای Controller امکان پذیر است و از طرفی در الگوی MVVM به دلیل منطق وجود ViewModel می‌توان برای اینگونه پروژه‌ها نیز Unit Test نوشت. اما ساختار سایر پروژه‌ها به گونه ای است که نوشتن Unit Test برای آن‌ها مشکل و در بعضی مواقع غیر ممکن می‌شود. برای مثال در پروژه‌های Desktop نظیر Windows Application و حتی وب به صورت Asp.Net Web Forms به دلیل وابستگی مستقیم کنترل‌های UI به منطق اجرای برنامه، طراحی و نوشتن Unit Test بسیار مشکل و در برخی موارد بیهوده است. در VS.Net ابزاری وجود دارد به نام Coded UI Test که برای تست این گونه پروژه‌ها طراحی شده است و همان طور که از نامش پیداست صرفاً برای تست کنترل‌های UI و رویدادهای کنترل‌ها و تست درستی برنامه با توجه به داده‌های ورودی به کار می‌رود. یکی از مزیت‌های اصلی آن تسریع عملیات تست در حجم بالا است و زمان ایجاد unit test را به حداقل می‌رساند. مزیت دوم آن امکان ایجاد unit test برای پروژه‌های که در مراحل پایانی تولید هستند ولی هنوز اطمینانی به عملکرد صحیح برنامه در حالات مختلف نیست. در این پست قصد دارم روش استفاده از این گونه پروژه‌های تست را با ذکر یک مثال بررسی کنیم و در پست‌های بعدی به بررسی امکانات دیگر خواهیم پرداخت.

نکته : فقط در Vs.Net با نسخه‌های Ultimate و Premium می‌توانید از Code UI Test استفاده کنید که البته به دلیل اینکه در ایران پیدا کردن نسخه‌های دیگر Vs.Net به غیر از Ultimate سخت‌تر است به طور قطع این محدودیت برای برنامه نویسان ما وجود نخواهد داشت. برای اینکه از نسخه Vs.Net خود اطمینان حاصل کنید از منوی Help گزینه About Microsoft Visual Studio رو انتخاب کنید. پنجره ای به شکل زیر مشاهده خواهید کرد که در آن مشخصات کامل Vs.Net ذکر شده است.

About Microsoft Visual Studio



## Visual Studio™

Licensed to:  
M.F

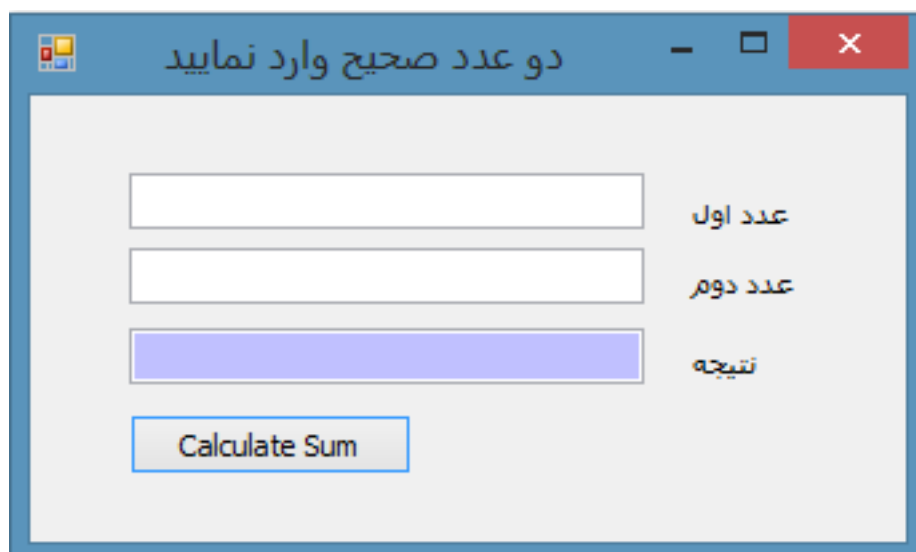
Microsoft Visual Studio Ultimate 2012  
Version 11.0.50727.1 RTMREL  
© 2012 Microsoft Corporation.  
All rights reserved.

Microsoft .NET Framework  
Version 4.5.50709  
© 2012 Microsoft Corporation.  
All rights reserved.

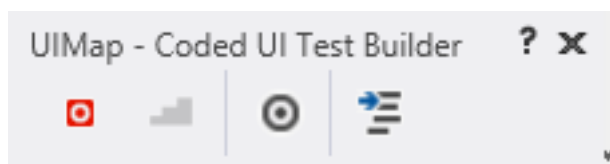
Installed products:

Architecture and Modeling Tools	04940-004-0039002-02413
LightSwitch for Visual Studio 2012	04940-004-0039002-02413
Office Developer Tools	04940-004-0039002-02413
Team Explorer for Visual Studio 2012	04940-004-0039002-02413
Visual Basic 2012	04940-004-0039002-02413
Visual C# 2012	04940-004-0039002-02413
Visual C++ 2012	04940-004-0039002-02413
Visual F# 2012	04940-004-0039002-02413

در این مرحله قصد داریم برای فرم زیر Unit Test طراحی کنیم. پروژه به صورت زیر است:



کاملاً واضح است که در این فرم دو عدد به عنوان ورودی دریافت می‌شود و بعد از کلیک بر روی CalculateSum نتیجه در textbox سوم نمایش داده می‌شود. برای تست عملکرد صحیح فرم بالا ابتدا به Solution مورد نظر از منوی test Project یک Coded UI Test Project اضافه می‌کنیم. به دلیل اینکه این قبلاً در این Solution پروژه تست از نوع Coded UI Test نبود بلافاصله یک پنجره نمایش داده می‌شود. مطمئن شوید گزینه اول انتخاب شده و بعد بر روی Ok کلیک کنید. (گزینه اول به معنی است که قصد داریم عملیات مورد نظر بر روی UI را رکورد کنیم و گزینه دوم به معنی است که قصد داریم از عملیات رکورد شده قبلی استفاده کنیم). یک کلاس به نام CodeUITest1 به همراه یک متد تست به نام CodedUITestMethod1 ساخته می‌شود. اولین چیزی که جلب توجه می‌کند این است که این کلاس به جای TestClassAttribute دارای نشان CodeUITestAttribute است. در گوشه سمت راست خود یک پنجره کوچک به نام UI Map Test Builder مانند شکل زیر خواهید دید.



دکمه قرمز رنگ به نام Record Button است و عملیات تست را رکورد خواهد کرد. دکمه دایره ای به رنگ مشکی برای تعیین Assertion به کار می‌رود. و در نهایت گزینه آخر کدهای مورد نظر مراحل قبل را به صورت خودکار تولید خواهد کرد.

## #روش کار

روش کار به این صورت است که ابتدا شما مراحل تست خود را شبیه سازی خواهید کرد و بعد از آن Test Builder مراحل تست شما را به صورت کامل به صورت کدهای قابل فهم تولید خواهد کرد. (دقیقاً شبیه به ایجاد UnitTest به روش Arrange/Act/Assert است با این تفاوت که این مراحل توسط UI Map رکورد شده و نیازی به کد نویسی ندارد). در پایان باید یک Data Driven Coded UI Test طراحی کنید تا بتوانید از این مراحل رکورد استفاده نمایید.

## #چگونگی شبیه سازی :

پروژه را اجرا نمایید. زمانی که فرم مورد نظر ظاهر شد بر روی گزینه Record در TestBuilder کلیک کنید. عملیات ذخیره سازی شروع شده است. در نتیجه به فرم مربوطه رفته و در Textbox اول مقدار 10 و در textbox دوم مقدار 5 را وارد نمایید. با کلیک بر روی دکمه CalculateSum مقدار 15 نمایش داده خواهد شد. از برنامه خارج شوید و بعد بر روی گزینه Generate Code در TestBuilder کلیک کنید با از کلیدهای ترکیبی Alt + G استفاده نمایید. (اگر در این مرحله، از برنامه خارج نشده باشید با خطا مواجه خواهید شد.) در پنجره نمایش داده شده یک نام به متد اختصاص دهید. عملیات تولید کد شروع خواهد شد. بعد کدی مشابه زیر را در متد مربوطه مشاهده خواهید کرد.

```
[TestMethod]
public void CodedUITestMethod1()
{
    this.UIMap.CalculateSum();
    this.UIMap.txtSecondValueMustBe10();
}
```

بخشی از سورس کد تولید شده برای متد CalculateSum به شکل زیر است:

```
public void CodedUITestMethod1
(
    {
        #region Variable Declarations
        WinEdit uITxtFirstNumberEdit =
this.UIMap.Window.UITxtFirstNumberEdit;
        WinEdit uITxtSecondNumberEdit =
this.UIMap.Window.UITxtSecondNumberEdit;
        WinButton uICalculateSumButton =
this.UIMap.Window.UICalculateSumButton;
        #endregion

        // Type '10' in 'txtFirstNumber' text box
        uITxtFirstNumberEdit.Text = this.CalculateSumParams.UITxtFirstNumberEditText;

        // Type '{Tab}' in 'txtFirstNumber' text box
        Keyboard.SendKeys(uITxtFirstNumberEdit,
this.CalculateSumParams.UITxtFirstNumberEditSendKeys, ModifierKeys.None);

        // Type '10' in 'txtSecondNumber' text box
        uITxtSecondNumberEdit.Text = this.CalculateSumParams.UITxtSecondNumberEditText;

        // Click 'Calculate Sum' button
        Mouse.Click(uICalculateSumButton, new Point(83, 12));

        // Type '10' in 'txtFirstNumber' text box
        uITxtFirstNumberEdit.Text = this.CalculateSumParams.UITxtFirstNumberEditText1;

        // Type '{Tab}' in 'txtFirstNumber' text box
        Keyboard.SendKeys(uITxtFirstNumberEdit,
this.CalculateSumParams.UITxtFirstNumberEditSendKeys1, ModifierKeys.None);

        // Type '10' in 'txtSecondNumber' text box
        uITxtSecondNumberEdit.Text = this.CalculateSumParams.UITxtSecondNumberEditText1;

        // Type '{Tab}' in 'txtSecondNumber' text box
        Keyboard.SendKeys(uITxtSecondNumberEdit,
this.CalculateSumParams.UITxtSecondNumberEditSendKeys, ModifierKeys.None);

        // Click 'Calculate Sum' button
        Mouse.Click(uICalculateSumButton, new Point(49, 11));

        // Type '10' in 'txtFirstNumber' text box
        uITxtFirstNumberEdit.Text = this.CalculateSumParams.UITxtFirstNumberEditText2;

        // Type '{Tab}' in 'txtFirstNumber' text box
        Keyboard.SendKeys(uITxtFirstNumberEdit,
this.CalculateSumParams.UITxtFirstNumberEditSendKeys2, ModifierKeys.None);

        // Type '5' in 'txtSecondNumber' text box
        uITxtSecondNumberEdit.Text = this.CalculateSumParams.UITxtSecondNumberEditText2;

        // Type '{Tab}' in 'txtSecondNumber' text box
        Keyboard.SendKeys(uITxtSecondNumberEdit,
```

```

this.CalculateSumParams.UITxtSecondNumberEditSendKeys1, ModifierKeys.None);

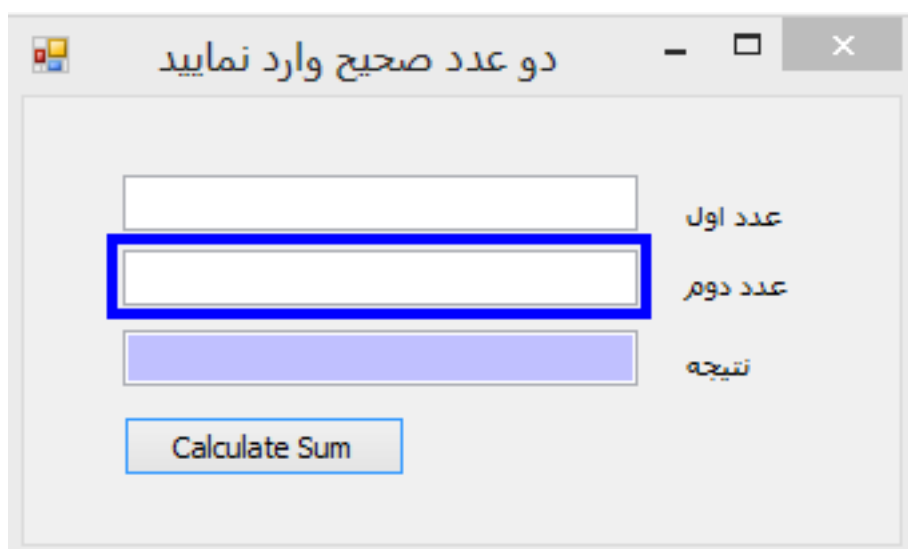
    // Click 'Calculate Sum' button
    Mouse.Click(uiCalculateSumButton, new Point(74, 16));
}

```

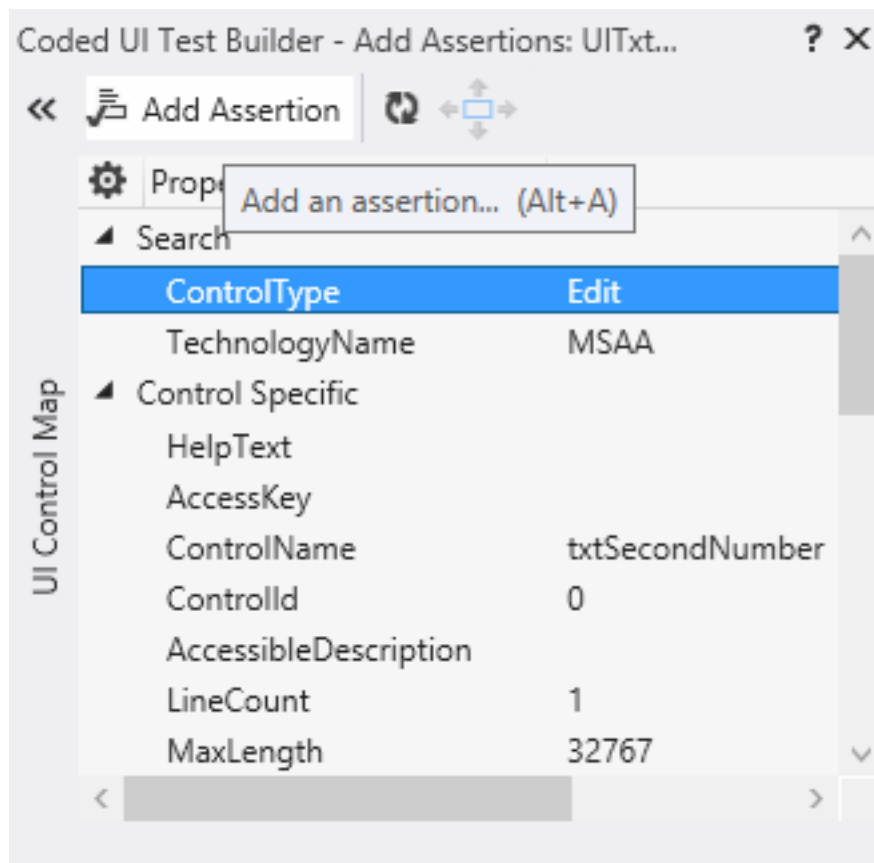
همان طور که می بینید تمام مراحل تست شما رکورد شده است و به صورت کد قابل فهم بالا ایجاد شده است.

#### چگونگی ایجاد Assertion

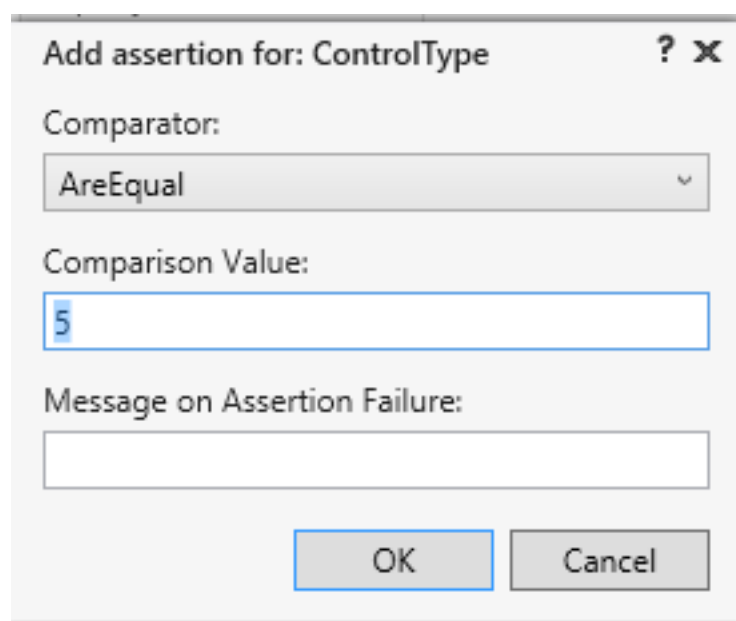
اگر به کد متد تست CodedUITestMethod1 در بالا دقت کنید یک متد به صورت `this.UIMap.txtSecondValueMustBe10` فراخوانی شده است. این در واقع یک Assertion است که در هنگام عملیات رکورد ایجاد کردم و به این معنی است که مقدار `TextBox` دوم حتما باید 10 باشد. حال روش تولید Assertion ها را بررسی خواهیم کرد. بعد از شروع شدن مرحله رکورد اگر قصد دارید برای یک کنترل خاص `Assert` بنویسید، دکمه `assertion` (به رنگ مشکی و به صورت دایره است) را بر روی کنترل مورد نظر `drag&drop` کنید. یک `border` آبی برای کنترل مورد نظر ایجاد خواهد شد:



به محض اتمام عملیات `drag&drop` منوی زیر ظاهر خواهد شد:



از گزینه Add Assertion استفاده کنید و برای کنترل مورد نظر یک assert بنویسید. در شکل زیر یک assert برای textbox دوم نوشتیم به صورتی که مقدار آن باید با 5 برابر باشد.



از گزینه آخر برای نمایش پیام مورد نظر خودتون در هنگامی که assert با شکست مواجه می‌شود استفاده کنید. کد تولید شده زیر برای عملیات assert بالا است:

```
public void txtSecondValueMustBe10()
{
    #region Variable Declarations
    WinEdit uITxtSecondNumberEdit =
this.UIدو عدد صحیح وارد نمایدthis.UIدو عدد صحیح وارد نمایدWindow.UITxtSecondNumberWindow.UITxtSecondNumberEdit;
    #endregion

    // Verify that the 'ControlType' property of 'txtSecondNumber' text box equals '10'
    Assert.AreEqual(this.txtSecondValueMustBe10ExpectedValues.UITxtSecondNumberEditControlType,
uITxtSecondNumberEdit.ControlType.ToString());
}
```

مرحله اول انجام شد. برای تست این مراحل باید یک Data DrivenTest بسازید که در پست بعدی به صورت کامل شرح داده خواهد شد.