عنوان: **استفاده از F# در پروژه های WPF**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۴/۱۷ ۸:۳۵
آدرس: www.dotnettips.info
برچسب‌ها: WPF, Programming, F#, FSharpx

در دوره F# این سایت (^) با نحوه کد نویسی و مفاهیم و مزایای این زبان آشنا شده اید. اما دانستن syntax یک زبان برای پیاده سازی یک پروژه کافی نیست و باید با تکنیک‌های مهم دیگر از این زبان آشنا شویم. همان طور که قبلا (فصل اول دوره F#) بیان شد Visual Studio به صورت Visual از پروژه‌های F# پشتیبانی نمی‌کند. یعنی امکان ایجاد یک پروژه WPF یا Windows Application یا حتی پروژه‌های تحت وب برای این زبان همانند زبان C# به صورت Visual در VS.Net تعبیه نشده است. حال چه باید کرد؟ آیا باید در این مواقع این گونه پروژه‌ها را با یک زبان دیگر نظیر C# ایجاد کنیم و از زبان F# در حل برخی مسائل محاسباتی و الگوریتمی استفاده کنیم. این اولین راه حلی است که به نظر می‌رسد. اما در حال حاضر افزونه هایی، توسط سایر تیم‌های برنامه نویسی تهیه شده اند که پیاده سازی و اجرای یک پروژه تحت ویندوز یا وب را به صورت کامل با زبان F# امکان پذیر می‌کنند. در این پست به بررسی یک مثال از پروژه WPF به کمک این افزونه‌ها می‌پردازیم.

نکته : آشنایی با کد نویسی و مفاهیم F# برای درک بهتر مطالب توصیه می‌شود.

معرفی پروژه FSharpX

پروژه FSharpx یک پروژه متن باز است که توسط یک تیم بسیار قوی از برنامه نویسان F# در حال توسعه می‌باشد. این پروژه شامل چندین زیر پروژه و بخش است که هر بخش آن برای یکی از مباحث دات نت در F# تهیه و توسعه داده می‌شود.

این قسمت‌ها عبارتند از :

FSharpx.Core : شامل مجموعه ای کامل از توابع عمومی، پرکاربرد و ساختاری است که برای این زبان توسعه داده شده اند و با تمام زبان‌های دات نت سازگاری دارند؛

FSharpx.Http : استفاده از F# در برنامه نویسی مدل Http؛

FSharpx.TypeProvider : این پروژه خود شامل چندین بخش است که در این جا چند مورد از آن‌ها را عنوان می‌کنم:

FSharpx.TypeProviders.AppSetting : متد خواندن و نوشتن (getter و setter) را برای فایل‌های تنظیمان پروژه (Application Setting File) فراهم می‌کند.

FSharpx.TypeProviders.Vector : برای محاسبات با ساختارهای برداری استفاده می‌شود.

FSharpx.TypeProviders.Machine : برای دسترسی و اعمال تغییرات در رجیستری و فایل‌های سیستمی استفاده می‌شود.

FSharpx.TypeProviders.Xaml : با استفاده از این افزونه می‌توانیم از فایل‌های Xaml، در پروژه‌های F# استفاده کنیم و WPF Designer نرم افزار VS.Net هم برای این زبان قابل استفاده خواهد شد.

FSharpx.TypeProviders.Regex : امکان استفاده از عبارات با قاعده را در این پروژه فراهم می‌کند.

یک مثال از عبارات با قاعده:

```
type PhoneRegex = Regex< @"(?<AreaCode>^\d{3})-(?<PhoneNumber>\d{3}-\d{4}$)">

PhoneRegex.IsMatch "425-123-2345"
|> should equal true

PhoneRegex().Match("425-123-2345").CompleteMatch.Value
|> should equal "425-123-2345"

PhoneRegex().Match("425-123-2345").PhoneNumber.Value
|> should equal "123-2345"
```

شروع پروژه

ایتدا یک پروژه از نوع F# Console Application ایجاد کنید. از قسمت Project Properties (بر روی پروژه کلیک راست کنید و گزینه Properties را انتخاب کنید) نوع پروژه را به Windows Application تغییر دهید(قسمت Out Put Type). اسمبلی‌های زیر را به پروژه ارجاع دهید:

PresentationCore

PresentationFramework

WindowBase

System.Xaml

با استفاده از پنجره Package Manager Console دستور نصب زیر را اجرا کنید(آخرین نسخه این پکیج 1.8.31 و حجم آن کمتر از یک مگابایت است):

```
PM> Install-Package FSharpx.TypeProviders.Xaml
```

حال یک فایل Xaml به پروژه اضافه کنید و کدهای زیر را در آن کپی کنید:

```
<Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
Title="WPF F# Sample By Masoud Pakdel" Height="350" Width="525">
    <Grid Name="MainGrid">
        <StackPanel Name="StackPanel1" Margin="50">
            <Button Name="Button1">Who are you?</Button>
        </StackPanel>
    </Grid>
</Window>
```

کدهای بالا کاملا واضح است و نیاز به توضیح دیده نمی‌شود. اما اگر دقت کنید می‌بینید که این فایل، فایل Code Behind ندارد. برای این کار باید یک فایل جدید از نوع F# Source File ایجاد کنید. بهتر است که فایل جدید شما همنام با همین فایل باشد. پسوند این فایل fs است. حال کدهای زیر را در آن کپی کنید:

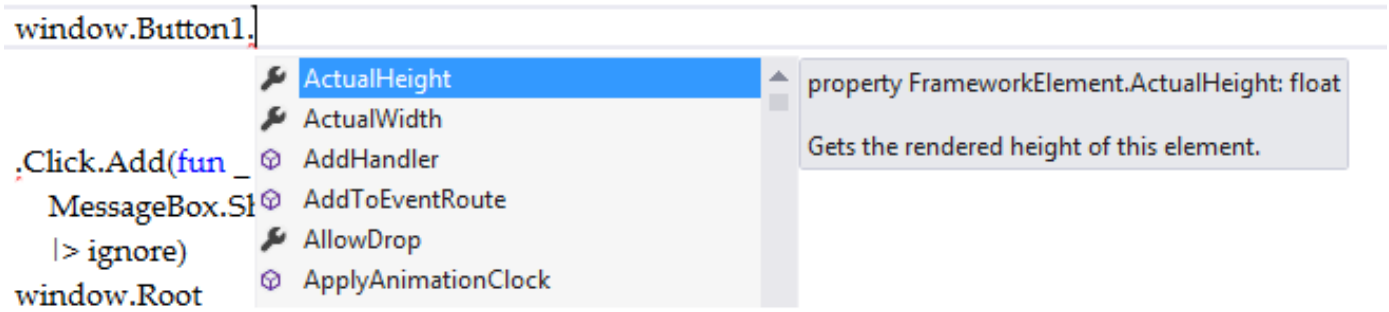
```
open System
open System.Windows
open System.Windows.Controls
open FSharpx

type MainWindow = XAML<"MainWindow.xaml">

let loadWindow() =
    let window = MainWindow()
    window.Button1.Click.Add(fun _ ->
        MessageBox.Show("Masoud Pakdel")
        |> ignore)
    window.Root

[<STAThread>]
(new Application()).Run(loadWindow())
|> ignore
```

نوع XAML استفاده شده که به صورت generic است در فضای نام FSharpx تعبیه شده است و این اجازه را می‌دهد که یک فایل F# بتواند برای مدیریت یک فایل Xaml استفاده شود.برای مثال می‌توانید به اشیا و خواص موجود در فایل Xaml دسترسی داشته باشید. در اینجا دیگر خبری از متد InitializeComponent موجود در سازنده کلاس CodeBehind پروژه‌های #C نیست. این تعاریف و آماده سازی کامپوننت‌ها به صورت توکار در نوع XAML موجود در FSharpx انجام می‌شود.



در تابع loadWindow یک نمونه از کلاس MainWindow ساخته می‌شود و برای button1 آن رویداد کلیک تعریف می‌کنیم. دستورات زیر معادل دستورات شروع برنامه در فایل program پروژه‌های C# است.

```
[<STAThread>]
(new Application()).Run(loadWindow())
|> ignore
```

پروژه را اجرا کنید و بر روی تنهای Button موجود در صفحه، کلیک کنید و پیغام مورد نظر را مشاهده خواهید کرد. به صورت زیر: