عنوان: **آموزش F#**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۳/۲۴ ۲۳:۴۵
آدرس: www.dotnettips.info
گروه‌ها: #Programming, F

در نظر سنجی که قبلا توسط دوستان درباره میزان آشنایی و استفاده از زبان‌های مختلف برنامه نویسی در تولید پروژه‌های نرم افزاری انجام شده بود (^) تعداد رای زبان F# سه رای بود(یعنی کمتر از یک درصد). یکی از دلایلی که F# کمتر از سایر زبان‌ها مورد توجه است (البته تا این زمان) نبود منبع یا کتاب فارسی در زمینه یادگیری و هم چنین عدم شناخت از امکانات و قدرت این زبان است. در نتیجه تصمیم گرفتم در طی دو یا چند دوره به آموزش برنامه نویسی این زبان بپردازم. دوره اول که از قسمت دوره‌ها (^)در این سایت در دسترس عموم قرار دارد سطوح مقدماتی و متوسط را پوشش می‌دهد (سرفصل‌های این دوره در قسمت آموزش F# ذکر شده است). به دلیل حجم گسترده مطالب امکان ارایه تمام مفاهیم و روش‌ها در طی یک دوره امکان پذیر نبود در نتیجه تصمیم بر آن شد که با توجه به اولویت‌های آموزشی این مطالب طبقه بندی شوند و طی دو یا چند دوره به دوستان عزیز ارائه شوند.
دوره ای که هم اکنون در دسترس است صرفا جهت آشنایی دوستان با نوع کدنویسی و مفاهیم برنامه نویسی این زبان تهیه شده است اما دوره پیشرفته این زبان که بعدا در طی چند فصل، آموزش داده خواهد شد دارای سرفصل‌های زیر خواهد بود:
استفاده از F# در پروژه‌های تولید شده با زبان C# و در محیط Visual Studio.Net

استفاده از EntityFramework در زبان F#
تولید و توسعه پروژهای Windows Application با زبان F#

تولید و توسعه پروژهای WPF با زبان F#
تولید و توسعه پروژه‌های تحت Silverlight با زبان F#
و...

موفق باشید.

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۲۵ ۱۰:۴۰

- عالی است. مطلبی رو اینجا دیدم در مورد « ۱۰ رفتار که باید هنگام استخدام یک برنامه نویس دنبالشان بگردید ». خصوصا این چند مورد ویژه که در مورد شما بیشتر صادق است؛ برخلاف جماعت صرفا کارمند :

۱- کنجکاوی

۵- یادگیری سریع

۶- مهارت‌های خودآموزی

۷- علاقه

نویسنده: مسعود م.پاکدل
تاریخ: ۱۳۹۲/۰۳/۲۵ ۱۱:۱

ممنونم جناب نصیری.

تشکر و قدردانی اصلی برای شماست به خاطر فراهم نمودن محیطی عالی برای ارائه مطالب.

نویسنده: اردلان شاه قلی
تاریخ: ۱۳۹۲/۰۹/۲۴ ۹:۳۲

سلام . لطفا آموزش #F را ادامه دهید. با تشکر

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:

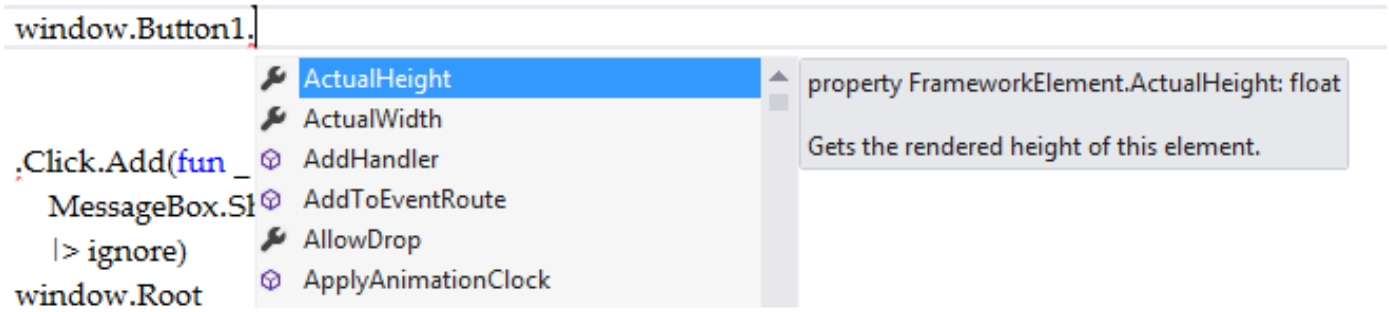
```
open System
open System.Windows
open System.Windows.Controls
open FSharpx

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
```

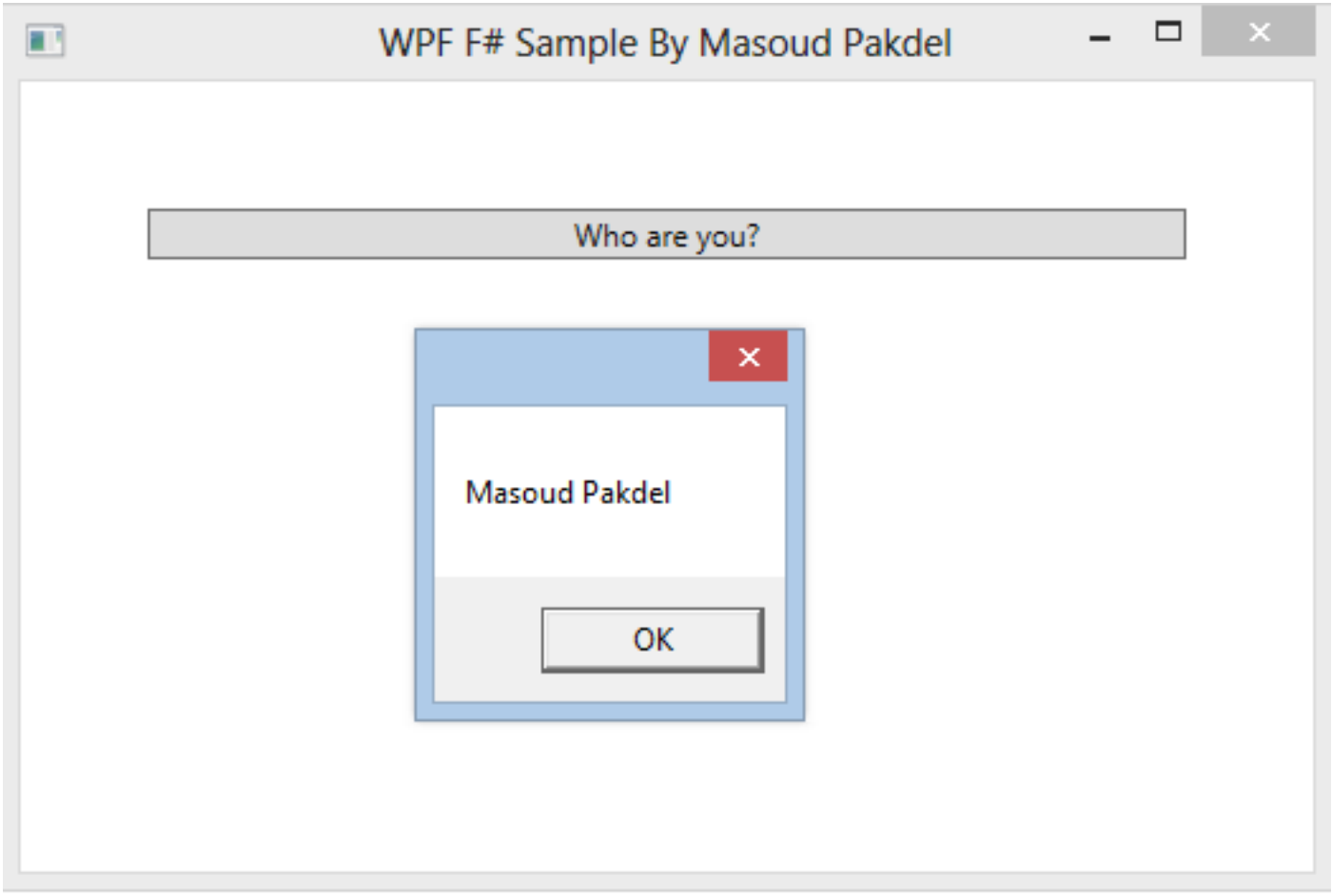
نوع XAML استفاده شده که به صورت generic است در فضای نام FSharpx تعبیه شده است و این اجازه را می‌دهد که یک فایل F# بتواند برای مدیریت یک فایل Xaml استفاده شود.برای مثال می‌توانید به اشیا و خواص موجود در فایل Xaml دسترسی داشته باشید. در اینجا دیگر خبری از متد InitializeComponent موجود در سازنده کلاس CodeBehind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

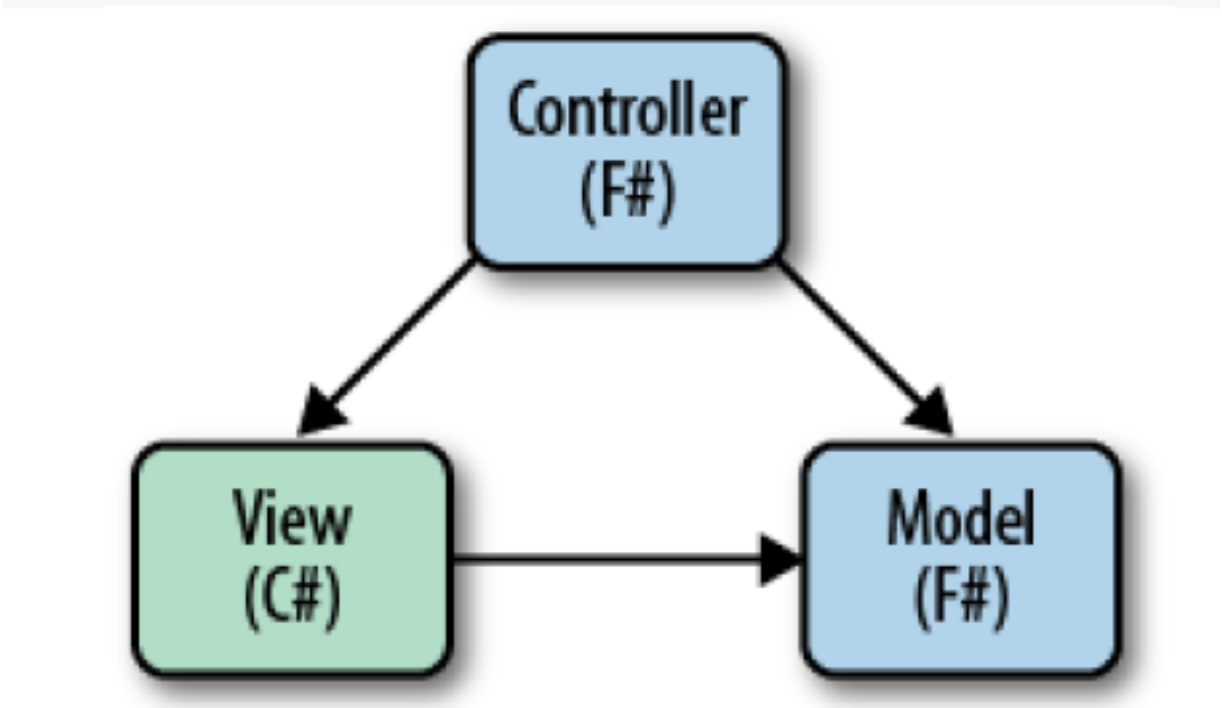
```
[<STAThread>]
(new Application()).Run(loadWindow())
|> ignore
```

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:



عنوان: **استفاده از #F در پروژه های MVC4**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۲/۱۱ ۹:۵
آدرس: www.dotnettips.info
گروه‌ها: #MVC, F

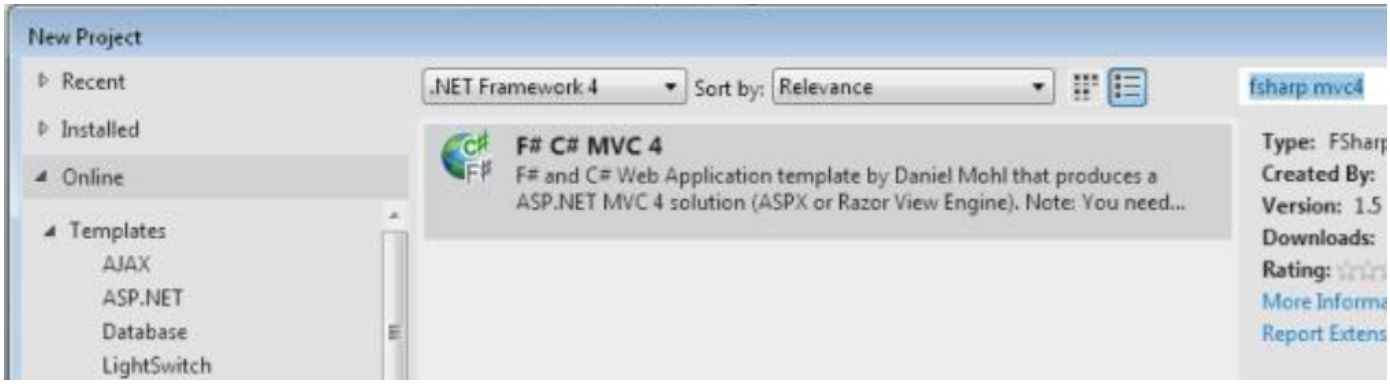
در این پست با روش پیاده سازی پروژه‌های WPF با استفاده از #F آشنا شدید. قصد دارم در طی چند پست روش پیاده سازی پروژه‌های Asp.Net MVC 4 با استفاده از #F را شرح دهم.
»اگر با #F آشنایی ندارید می‌توانید از اینجا شروع کنید.
به صورت کلی برای استفاده گسترده از #F در پروژه‌های وب نیاز به یک سری templateهای آماده داریم در غیر این صورت کار کمی سخت خواهد شد. به تصویر زیر دقت نمایید:



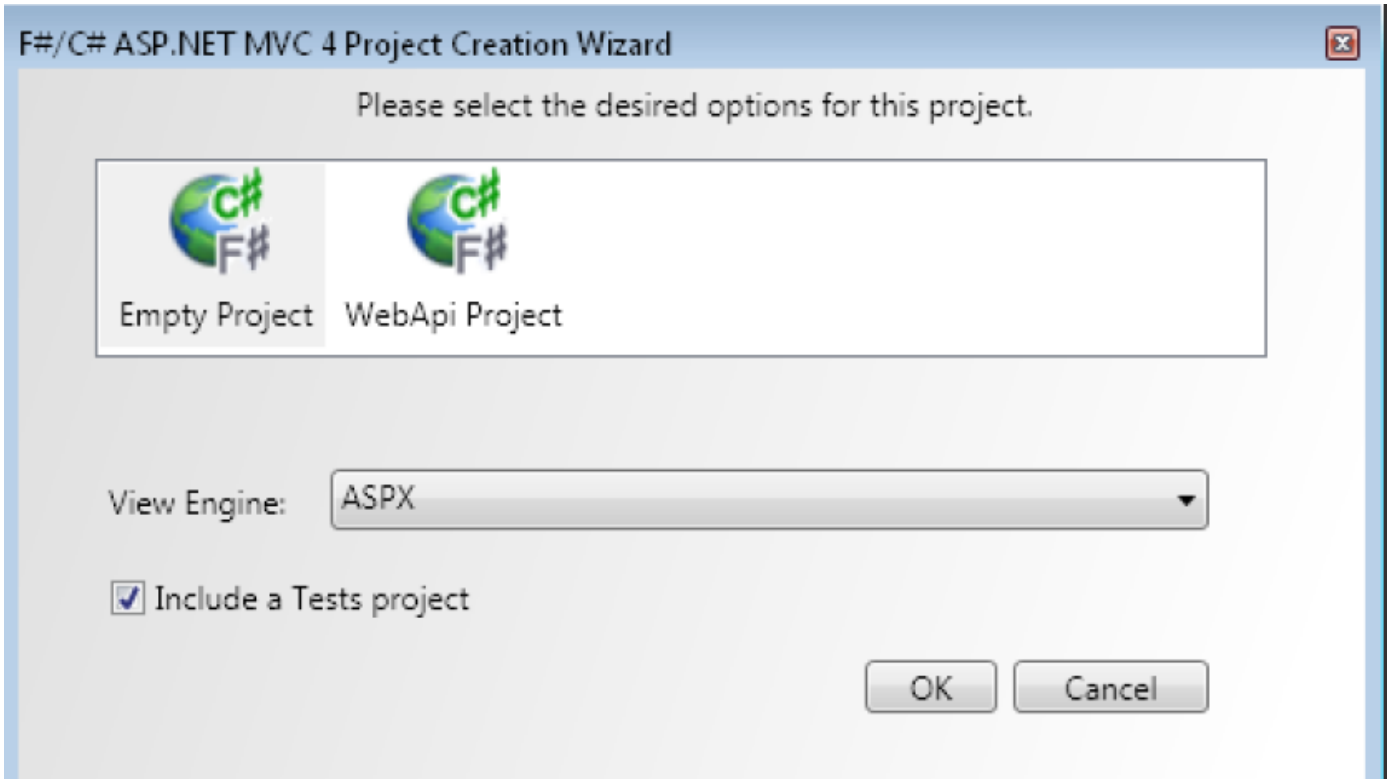
واضح است که با توجه به تصویر بالا کنترلرها و البته مدل‌های app و هر آنچه که سمت سرور به آن نیاز است باید با استفاده از #F پیاده سازی شوند. اما هنگامی که کنترلرها با استفاده از #F نوشته شوند سیستم مسیر یابی نیز تحت تاثیر قرار خواهد گرفت. علاوه بر آن باید فکری برای بخش Bundling و همچنین فیلترهاو... نمود. در نتیجه با توجه به template پروژه مورد نظر بر خلاف حالت پیش فرض #C آن که در قالب یک پروژه ارائه می‌شود در این جا حداقل به دو پروژه نیاز داریم. خوشبختانه همانند پروژه FSharpX که برای WPF مناسب است برای MVC نیز template آماده موجود است که در ادامه با آن آشنا خواهیم شد.

شروع به کار

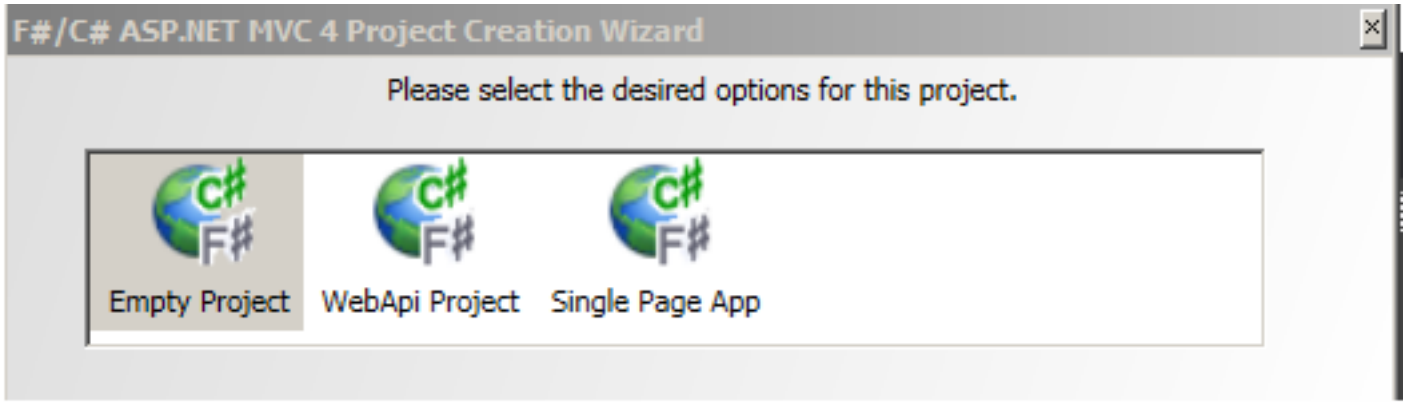
ابتدا در VS.Net یک پروژه جدید ایجاد نمایید. از بخش Online Template گزینه FSharp MVC 4 را جستجو کنید.



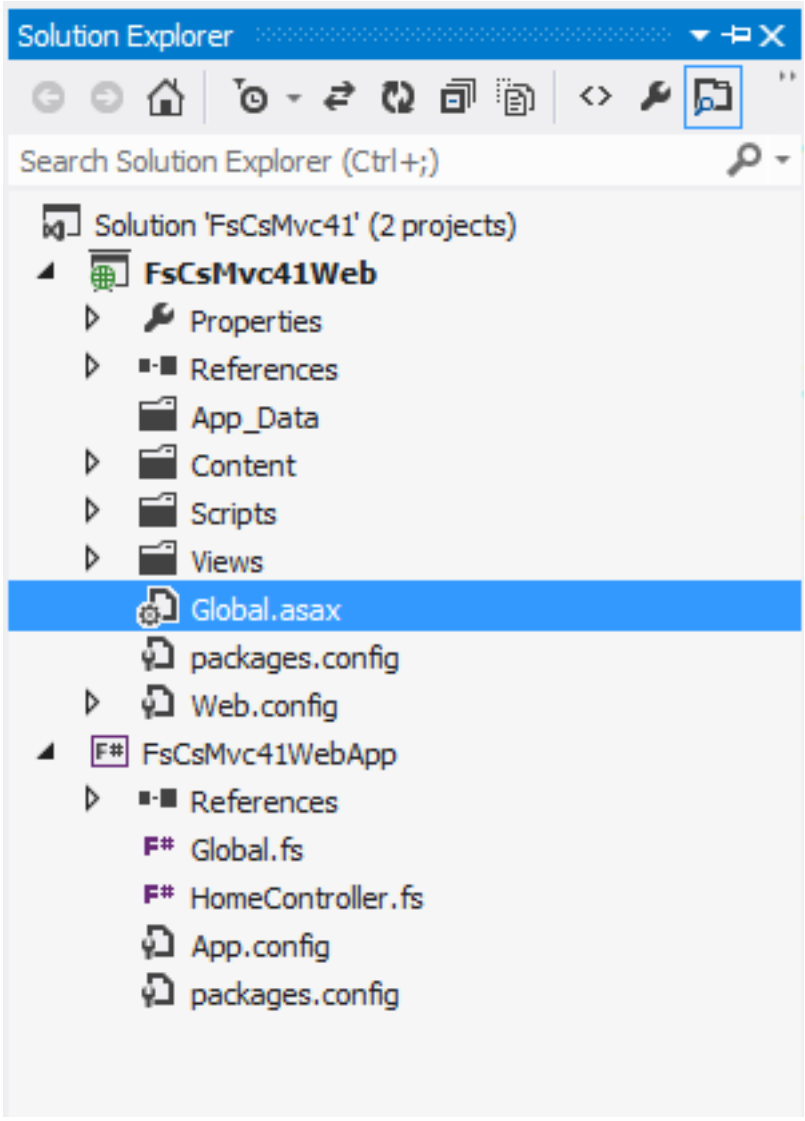
بعد از انتخاب نام پروژه و کلیک بر روی Ok (و البته دانلود حدود ده MB اطلاعات) صفحه زیر نمایان می‌شود. در این قسمت تنظیمات مربوط به انتخاب View Engine و نوع قالب پروژه را وجود دارد. در صورتی که قصد استفاده از Web Api را دارید گزینه Web Api Project را انتخاب کنید در غیر این صورت گزینه Empty Project.



البته از Visual Studio 2012 به بعد این بخش به صورت زیر خواهد بود که قسمت Single Page App به آن اضافه شده است:



بعد از کلیک بر روی Ok یک پروژه بر اساس Template مورد نظر ساخته می‌شود. همانند تصویر زیر:



بررسی تغییرات

در یک نگاه به راحتی می‌توان تغییرات زیر را در پروژه Web تشخیص داد:

»پوشه Controller وجود ندارد؛

»پوشه مدل وجود ندارد؛

»فایل Global.asax دیگر فایلی به نام Global.asax.cs را همراه با خود ندارد.

دلیل اصلی عدم وجود موارد بالا این است که تمام این موارد باید به صورت #F پیاده سازی شوند در نتیجه به پروژه #F ساخته شده منتقل شده اند. فایل Global.asax را باز نمایید. سورس زیر قابل مشاهده است:

```
<%@ Application Inherits="FsWeb.Global" Language="C#" %> <script Language="C#" RunAt="server">



 // Defines the Application_Start method and calls Start in // System.Web.HttpApplication from which Global inherits.

protected void Application_Start(Object sender, EventArgs e)
{
 base.Start();
}
 </script>
```

تمامی کاری که تکه کد بالا انجام می‌دهد فراخواهی متد Start در فایل Global متناظر در پروژه #F است. اگر به قسمت referenceهای پروژه Web دقت کنید خواهید که به پروژه #F متناظر رفرنس دارد.
حال به بررسی پروژه #F ساخته شده خواهیم پرداخت. در این پروژه یک فایل Global.fs وجود دارد که سورس آن به صورت زیر است:

```
namespace FsWeb

open System
open System.Web
open System.Web.Mvc
open System.Web.Routing

type Route = { controller : string
               action : string
               id : UrlParameter }

type Global() =
    inherit System.Web.HttpApplication()

    static member RegisterRoutes(routes:RouteCollection) =
        routes.IgnoreRoute("{resource}.axd/{*pathInfo}")
        routes.MapRoute("Default",
                        "{controller}/{action}/{id}",
                        { controller = "Home"; action = "Index"
                          id = UrlParameter.Optional } )

    member this.Start() =
        AreaRegistration.RegisterAllAreas()
        Global.RegisterRoutes(RouteTable.Routes)
```

در پروژه‌های MVC بر اساس زبان #C، کلاسی به نام RouteConfig وجود دارد که در فایل Global.asax.cs فراخوانی می‌شود:

```
RouteConfig.RegisterRoutes(RouteTable.Routes);
```

و کدهای مورد نظر جهت تعیین الگوی مسیر یابی کنترلرها و درخواست‌ها در کلاس RouteConfig نیز به صورت زیر می‌باشد:

```
public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
            );
        }
    }
```

اما در اینجا بر خلاف C#، تعریف و اجرای سیستم مسیر یابی در یک فایل انجام می‌شود. در بخش اول ابتدا یک type تعریف می‌شود که معادل با تعیین Routing در زبان C# است. علاوه بر آن متد RegisterRoutes جهت تعیین و انتساب Route Type تعریف شده به Route Collection مورد نظر نیز در این فایل می‌باشد.

```
//تعریف الگوی مسیر یابی
type Route = { controller : string
               action : string
               id : UrlParameter }

type Global() =
    inherit System.Web.HttpApplication()

    static member RegisterRoutes(routes:RouteCollection) =
       //فراخوانی  و انتساب الگوی مسیر یابی به  مسیرهای تعریف شده
       routes.IgnoreRoute("{resource}.axd/{*pathInfo}")
        routes.MapRoute("Default",
                        "{controller}/{action}/{id}",
                        { controller = "Home"; action = "Index"
                          id = UrlParameter.Optional } )
```

در نهایت نیز تابع RegisterRoute در تابع Start فراخوانی خواهد شد.

```
Global.RegisterRoutes(RouteTable.Routes)
```

نتیجه کلی تا اینجا:
در این پست با Template پروژه‌های F# MVC 4 اشنا شدیم و از طرفی مشخص شد که برای پیاده سازی این گونه پروژه‌ها حداقل نیاز به دو پروژه داریم. یک پروژه که از نوع C# است ولی در آن فقط View ها و فایل جاوااسکریپتی و البته Css وجود دارد. از طرف دیگر کنترلرها و مدل‌ها و هر چیز دیگر که مربوط به سمت سرور است در قالب یک پروژه F# پیاده سازی می‌شود.
در پست بعدی با روش تعریف و توسعه کنترلرها و مدل‌ها آشنا خواهیم شد.

عنوان: **توسعه کنترلر و مدل در F# MVC4**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۲/۱۷ ۲۳:۵
آدرس: www.dotnettips.info
گروه‌ها: #MVC, F

در پست قبلی با F# MVC4 Template آشنا شدید. در این پست به توسعه کنترلر و مدل در قالب مثال خواهم پرداخت. برای شروع ابتدا یک پروژه همانند مثال ذکر شده در پست قبلی ایجاد کنید. در پروژه C# ساخته شده که صرفا برای مدیریت Viewها است یک View جدید به صورت زیر ایجاد نمایید:

```
@model IEnumerable<FsWeb.Models.Book>
<!DOCTYPE html>
<html>
<head>
    <title>@ViewBag.Title</title>
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <link rel="stylesheet"
        href="http://code.jquery.com/mobile/1.0.1/jquery.mobile-1.0.1.min.css" />
</head>

<body>
    <div data-role="page" data-theme="a" id="booksPage">
        <div data-role="header">
            <h1>Guitars</h1>
        </div>
        <div data-role="content">
        <ul data-role="listview" data-filter="true" data-inset="true">
            @foreach(var x in Model) {
                <li><a href="#">@x.Name</a></li>
            }
        </ul>
        </div>
    </div>
    <script src="http://code.jquery.com/jquery-1.6.4.min.js">
    </script>
    <script src="http://code.jquery.com/mobile/1.0.1/jquery.mobile-1.0.1.min.js">
    </script>

    <script>
        $(document).delegate("#bookPage", 'pageshow', function (event) {
            $("div:jqmData(role='content') > ul").listview('refresh');
        });
    </script>
</body>
</html>
```

از آنجا که هدف از این پست آشنایی با بخش F# پروژه‌های وب است در نتیجه نیازی به توضیح کدهای بالا دیده نمی‌شود.
برای ساخت کنترلر جدید، در پروژه F# ساخته شده یک Source File ایجاد نمایید و کدهای زیر را در آن کپی نمایید:

```
namespace FsWeb.Controllers

open System.Web.Mvc
open FsWeb.Models

[<HandleError>]
type BooksController() =
    inherit Controller()
    member this.Index () =
        seq { yield  Book(Name = "My F# Book")
              yield Book(Name = "My C# Book") }
        |> this.View
```

در کدهای بالا ابتدا یک کنترلر به نام BookController ایجاد کردیم که از کلاس Controller ارث برده است(با استفاده از inherit). سپس یک تابع به نام Index داریم(به عنوان Action مورد نظر در کنترلر) که آرایه ای از کتاب‌ها به عنوان پارامتر به تابع View می‌فرستد.(توسط اپراتور Pipe - |>). در نهایت دستور this.View معادل فراخوانی اکشن ()View در پروژه‌های C# است که View متناظر با اکشن را فراخوانی می‌کند. همان طور که ملاحظه می‌نمایید بسیار شبیه به پیاده سازی C# است.
اما نکته ای که در مثال بالا وجود دارد این است که دو نمونه از نوع Book را برای ساخت seq وهله سازی می‌کند. در نتیجه باید

Book Type را به عنوان مدل تعریف کنیم. به صورت زیر:

```
namespace FsWeb.Models
 type Book = { Id : Guid; Name : string }
```

البته در F# 3.0 امکانی فراهم شده است به نام Auto-Properties که شبیه تعریف خواص در C# است. در نتیجه می‌توان تعریف بالا را به صورت زیر نیز بازنویسی کرد:

```
namespace FsWeb.Models
type Book() = member val Name = "" with get, set
```

Attributeها مدل

اگر همچون پروژه‌های C# قصد دارید با استفاده از Attributeها مدل خود را اعتبارسنجی نمایید می‌توانید به صورت زیر اقدام نمایید:

```
open System.ComponentModel.DataAnnotations
 type Book() = [<Required>] member val Name = "" with get, set
```

هم چنین می‌توان Attributeهای مورد نظر برای مدل EntityFramework را نیز اعمال نمود(نظیر Key):

```
namespace FsWeb.Models
open System
open System.ComponentModel.DataAnnotations
type Book() = [<Key>] member val Id = Guid.NewGuid() with get, set
[<Required>] member val Name = "" with get, set
```

نکته: دستور open معادل با using در C# است. در پست بعدی برای تکمیل مثال جاری، روش طراحی Repository با استفاده از EntityFramework بررسی خواهد شد.

نظرات خوانندگان

نویسنده: vahid
تاریخ: ۱۳۹۲/۱۲/۱۹ ۱۲:۴۳

دلیل استفاده F# در MVC چیست ؟ چه مزایایی برای ما دارد ؟ ممنون

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۲/۱۹ ۱۴:۴۶

در این پست درباره مزایای F# بحث شد. البته در اینجا نیز مزایای F# برای پیاده سازی پروژه‌های وب از زبان طراحان آن بیان شده است.

عنوان: **تعامل با پایگاه داده با استفاده از EntityFramework در پروژه های F# MVC 4**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۲/۱۸ ۰:۵
آدرس: www.dotnettips.info
گروه‌ها: #Entity framework, MVC, F

در پست‌های قبلی (^ و ^) با template و ساخت کنترلر و مدل در پروژه‌های F# MVC آشنا شدید. در این پست به طراحی Repository با استفاده از EntityFramework خواهم پرداخت. در ادامه مثال قبل، برای تامین داده‌های مورد نیاز کنترلرها و نمایش آن‌ها در View نیاز به تعامل با پایگاه داده وجود دارد. در نتیجه با استفاده از الگوی Repository، داده‌های مورد نظر را تامین خواهیم کرد. به صورت پیش فرض با نصب Template جاری (F# MVC4) تمامی اسمبلی‌های مورد نیاز برای استفاده از در EF در پروژه‌های F# نیز نصب می‌شود.

پیاده سازی DbContext مورد نیاز

برای ساخت DbContext می‌توان به صورت زیر عمل نمود:

```
namespace FsWeb.Repositories

open System.Data.Entity
open FsWeb.Models

type FsMvcAppEntities() =
    inherit DbContext("FsMvcAppExample")

    do Database.SetInitializer(new CreateDatabaseIfNotExists<FsMvcAppEntities>())

    [<DefaultValue()>] val mutable books: IDbSet<Guitar>
    member x.Books with get() = x.books and set v = x.books <- v
```

همان طور که ملاحظه می‌کنید با ارث بری از کلاس DbContext و پاس دادن ConnectionString یا نام آن در فایل app.config، به راحتی FsMVCAppEntities ساخته می‌شود که معادل DbContext پروژه مورد نظر است. با استفاده از دستور do متد SetInitializer برای عملیات migration فراخوانی می‌شود. در پایان نیز یک DbSet به نام Books ایجاد کردیم. فقط از نظر syntax با حالت C# آن تفاوت دارد اما روش پیاده سازی مشابه است.

اگر syntax زبان F# برایتان نامفهوم است می‌توانید از این دوره کمک بگیرید.

پیاده سازی کلاس BookRepository

ابتدا به کدهای زیر دقت کنید:

```
namespace FsWeb.Repositories

type BooksRepository() =
    member x.GetAll () =
        use context = new FsMvcAppEntities()
        query { for g in context.Books do
                select g }
        |> Seq.toList
```

در کد بالا ابتدا تابعی به نام GetAll داریم. در این تابع یک نمونه از DbContext پروژه وهله سازی می‌شود. نکته مهم این است به جای شناسه let از شناسه use استفاده کردم. شناسه use دقیقا معال دستور using(){} در C# است. بعد از اتمام عملیات شی مورد نظر Dispose خواهد شد.
در بخش بعدی یک کوئری از DbSet مورد نظر گرفته می‌شود. این روش Query گرفتن در F# 3.0 مطرح شده است. در نتیجه در نسخه‌های قبلی آن (F# 2.0) اجرای این کوئری باعث خطا می‌شود. اگر قصد دارید با استفاده از F# 2.0 کوئری‌های خود را ایجاد نماید باید به طریق زیر عمل نمایید:
ابتدا از طریق nuget اقدام به نصب package ذیل نمایید:

```
FSPowerPack.Linq.Community
```

سپس در ابتدا Source File خود، فضای نام Microsoft.FSharp.Linq.Query را باز(استفاده از دستور open) کنید. سپس می‌توانید با اندکی تغییر در کوئری قبلی خود، آن را در F# 2.0 اجرا نمایید.

```
query <@ seq { for g in context.Books -> g } @> |> Seq.toList
```

حال باید Repository طراحی شده را در کنترلر مورد نظر فراخوانی کرد. اما اگر کمی سلیقه به خرج دهیم به راحتی می‌توان با استفاده از تزریق وابستگی ، BookRepository را در اختیار کنترلر قرار داد. همانند کد ذیل:

```
[<HandleError>]
type BooksController(repository : BooksRepository) =
    inherit Controller()
    new() = new BooksController(BooksRepository())
    member this.Index () =
        repository.GetAll()
        |> this.View
```

در کدهای بالا ابتدا وابستگی به BookRepository در سازنده BookController تعیین شد. سپس با استفاده از سازنده پیش فرض، یک وهله از وابستگی مورد نظر ایجاد و در اختیار سازنده کنترلر قرار گرفت(همانند استفاده از کلمه this در سازنده کلاس‌های C#). با فراخوانی تابع GetAll داده‌های مورد نظر از database تامین خواهد شد.

نکته : تنظیمات مروط به ConnectionString را فراموش نکنید:

```
<add name="FsMvcAppExample"
     connectionString="YOUR CONNECTION STRING"
     providerName="System.Data.SqlClient" />
```

موفق باشید.

نظرات خوانندگان

نویسنده: Ara
تاریخ: ۱۳۹۲/۱۲/۲۰ ۱:۴

سلام
با تشکر از زحمات شما

به نظر من استفاده از F# در کنار C# به عنوان Library برای حل مسائل خاص خیلی میتونه مفید باشه

اگه ممکنه در مورد صورت مسئله و راه حل‌های ارائه شده با F# بیشتر بنویسید تا بیشتر مورد استفاده قرار بگیره ،و خیلی کاربردی‌تر موضوع رو ببینیم

ممنون