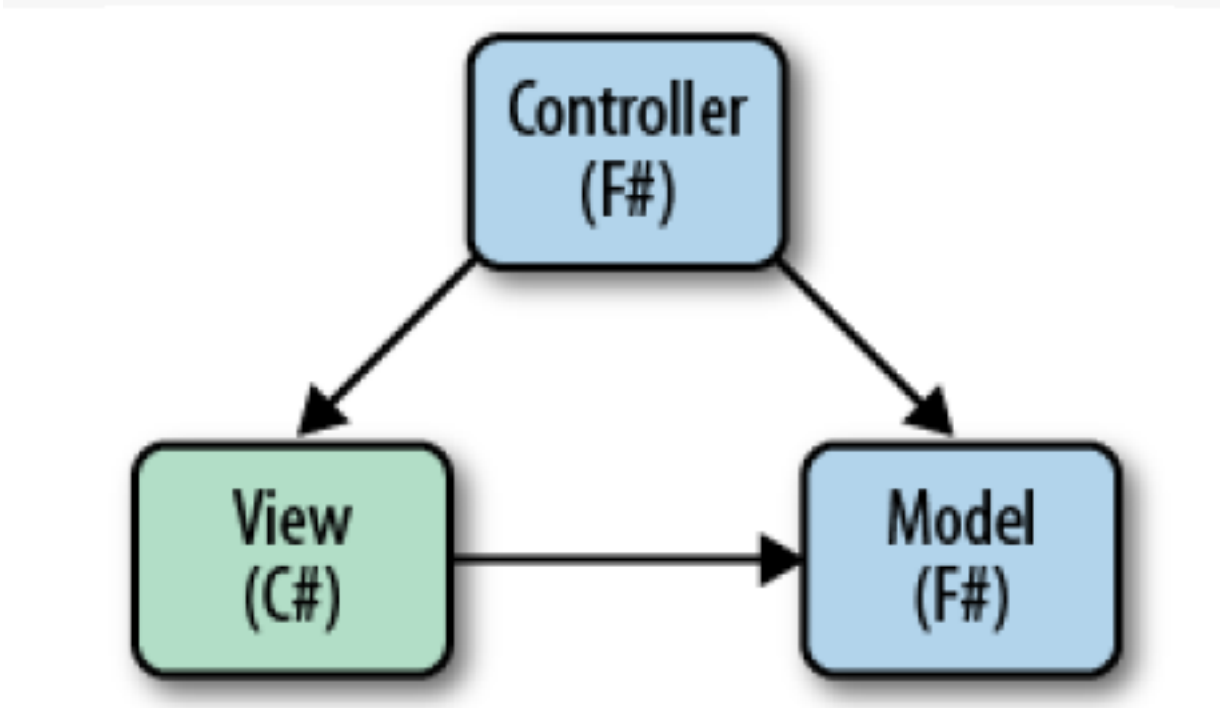
عنوان: **استفاده از #F در پروژه های MVC4**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۲/۱۱ ۹:۵
آدرس: www.dotnettips.info
گروه‌ها: #MVC, F

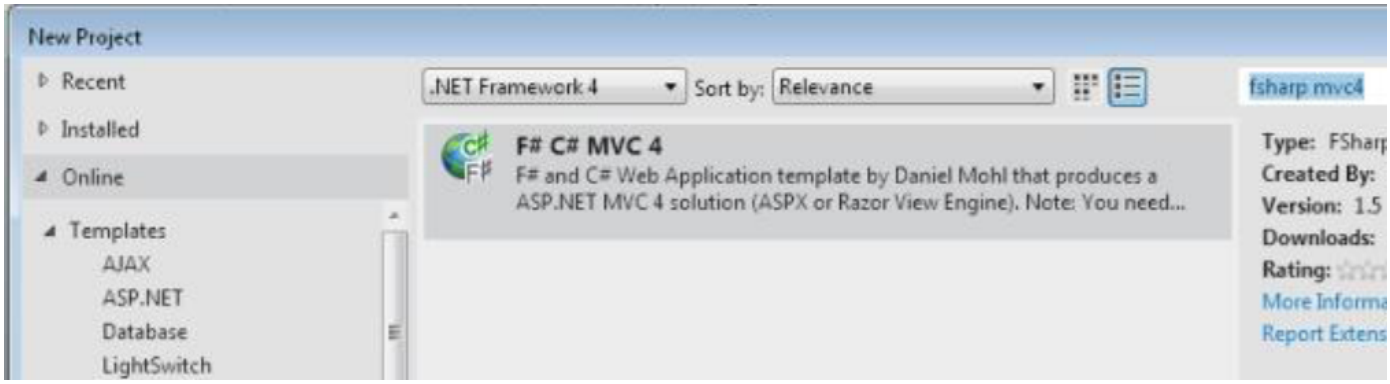
در این پست با روش پیاده سازی پروژه‌های WPF با استفاده از #F آشنا شدید. قصد دارم در طی چند پست روش پیاده سازی پروژه‌های Asp.Net MVC 4 با استفاده از #F را شرح دهم.
»اگر با #F آشنایی ندارید می‌توانید از اینجا شروع کنید.
به صورت کلی برای استفاده گسترده از #F در پروژه‌های وب نیاز به یک سری templateهای آماده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



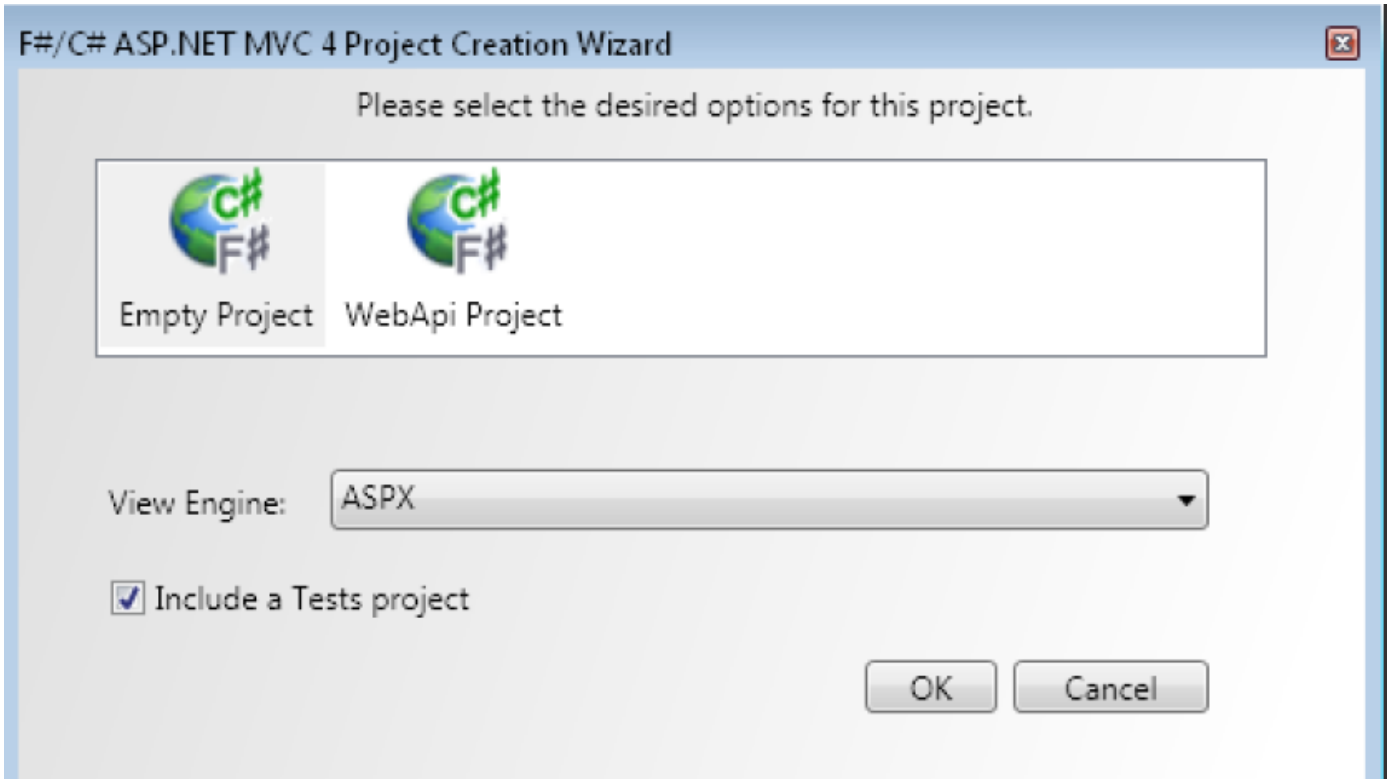
واضح است که با توجه به تصویر بالا کنترلرها و البته مدل‌های app و هر آنچه که سمت سرور به آن نیاز است باید با استفاده از #F پیاده سازی شوند. اما هنگامی که کنترلرها با استفاده از #F نوشته شوند سیستم مسیر یابی نیز تحت تاثیر قرار خواهد گرفت. علاوه بر آن باید فکری برای بخش Bundling و همچنین فیلترهاو... نمود. 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

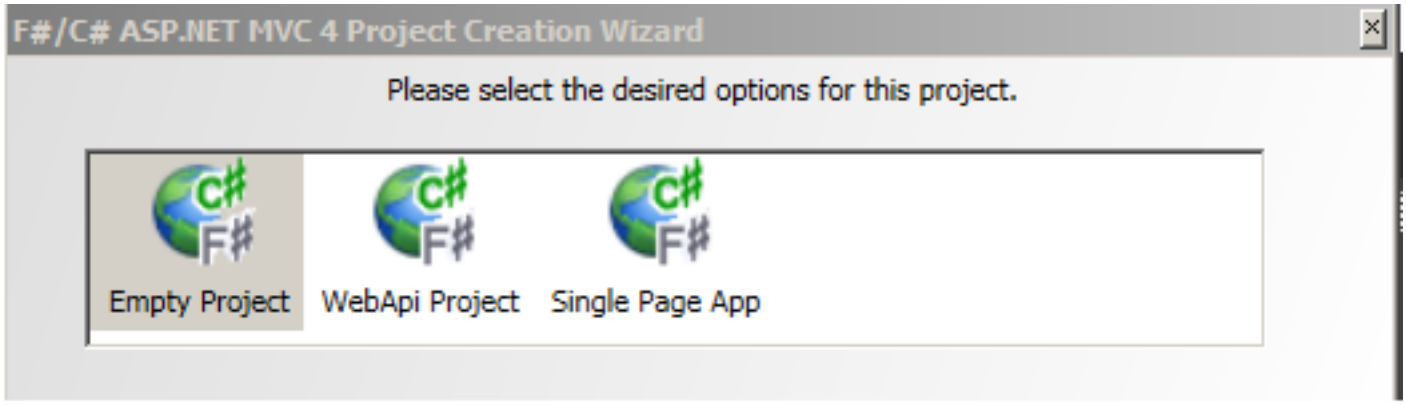
ابتدا در VS.Net یک پروژه جدید ایجاد نمایید. از بخش Online Template گزینه FSharp MVC 4 را جستجو کنید.



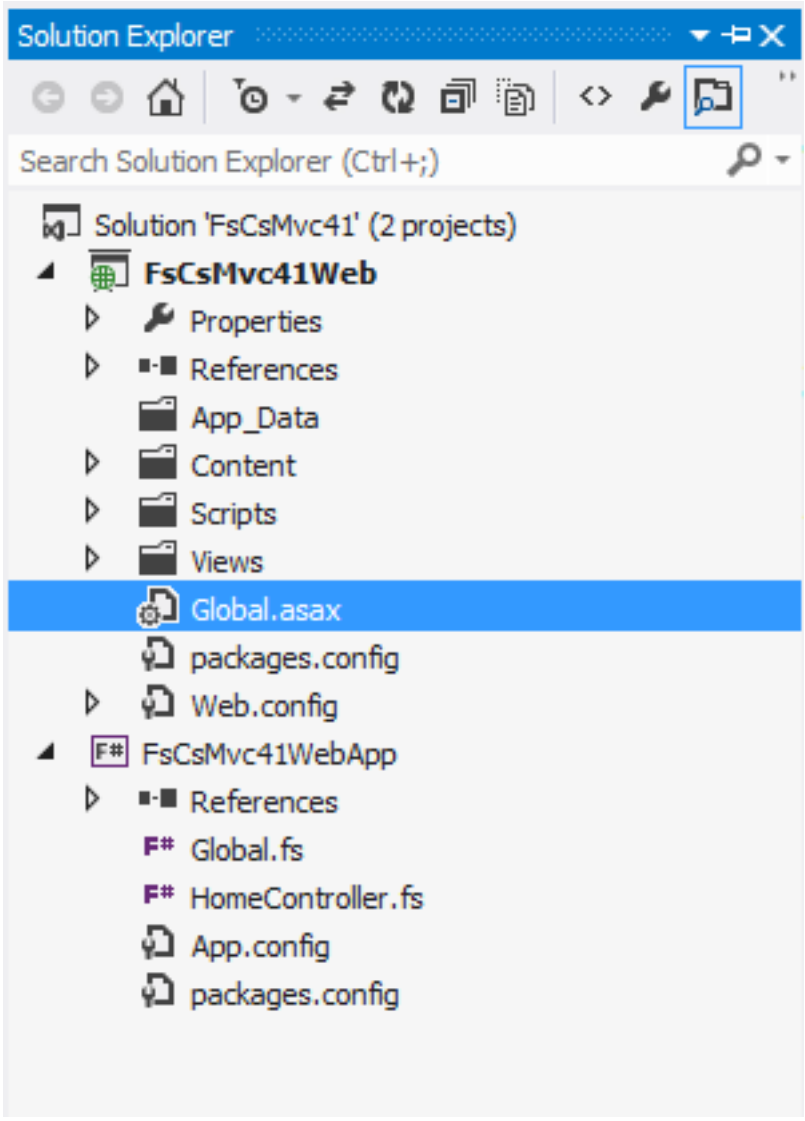
بعد از انتخاب نام پروژه و کلیک بر روی Ok (و البته دانلود حدود ده MB اطلاعات) صفحه زیر نمایان می‌شود. در این قسمت تنظیمات مربوط به انتخاب View Engine و نوع قالب پروژه را وجود دارد. در صورتی که قصد استفاده از Web Api را دارید گزینه Web Api Project را انتخاب کنید در غیر این صورت گزینه Empty Project.



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.