

## آشنایی با نحوه معرفی تعاریف طرحبندی سایت به کمک Razor

ممکن است یک سری از اصطلاحات را در قسمت‌های قبل مانند master page در لابلای توضیحات ارائه شده، مشاهده کرده باشید. این نوع مفاهیم برای برنامه نویسی‌های ASP.NET Web forms آشنا است (و اگر با Web forms view engine در ASP.NET MVC کار کنید، دقیقاً یکی است؛ البته با این تفاوت که فایل code behind آن‌ها حذف شده است). به همین جهت در این قسمت برای تکمیل بحث، مروری خواهیم داشت بر نحوه‌ی معرفی جدید آن‌ها توسط Razor. در یک پروژه جدید ASP.NET MVC و در پوشه Views\Shared\\_Layout.cshtml آن، فایل Layout آن، مفهوم master page را دارد. در این نوع فایل‌ها، زیر ساخت مشترک تمام صفحات سایت قرار می‌گیرند:

```
<!DOCTYPE html>
<html>
<head>
  <title>@ViewBag.Title</title>
  <link href="@Url.Content("~/Content/Site.css")" rel="stylesheet" type="text/css" />
  <script src="@Url.Content("~/Scripts/jquery-1.5.1.min.js")" type="text/javascript"></script>
</head>
<body>
  @RenderBody()
</body>
</html>
```

اگر دقت کرده باشید، در هیچ‌کدام از فایل‌های View ای که تا این قسمت به پروژه‌های مختلف اضافه کردیم، تگ‌هایی مانند body، title و امثال آن وجود نداشتند. در ASP.NET مرسوم است کلیه اطلاعات تکراری صفحات مختلف سایت را مانند تگ‌های یاد شده به همراه منویی که باید در تمام صفحات قرار گیرد یا footer مشترک بین تمام صفحات سایت، به یک فایل اصلی به نام master page که در اینجا layout نام گرفته، Refactor کنند. به این ترتیب حجم کدها و markup تکراری که باید در تمام View‌های سایت قرار گیرند به حداقل خواهد رسید. برای مثال محل قرار گیری تعاریف Content-Type تمام صفحات و همچنین favicon سایت، بهتر است در فایل layout باشد و نه در تک تک View‌های تعریف شده:

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="shortcut icon" href="@Url.Content("~/favicon.ico")" type="image/x-icon" />
```

در کدهای فوق یک نمونه پیش فرض فایل layout را مشاهده می‌کنید. در اینجا توسط متد RenderBody، محتوای رندر شده یک View درخواستی، به داخل تگ body تزریق خواهد شد. تا اینجا در تمام مثال‌های قبلی این سری، فایل layout در View‌های اضافه شده معرفی نشد. اما اگر برنامه را اجرا کنیم باز هم به نظر می‌رسد که فایل layout اعمال شده است. علت این است که در صورت عدم تعریف صریح layout در یک View، این تعریف از فایل Views\\_ViewStart.cshtml دریافت می‌گردد:

```
@{
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

فایل `ViewStart`، محل تعریف کدهای تکراری است که باید پیش از اجرای هر `View` مقدار دهی یا اجرا شوند. برای مثال در اینجا می‌شود بر اساس نوع مرورگر، `layout` خاصی را به تمام `View`ها اعمال کرد. مثلاً یک `layout` ویژه برای مرورگرهای موبایل‌ها و `layout` دیگری برای مرورگرهای معمولی. امکان دسترسی به متغیرهای تعریف شده در یک `View` در فایل `ViewStart` از طریق `ViewContext.ViewData` میسر است.

ضمن اینکه باید در نظر داشت که می‌توان فایل `ViewStart` را در زیر پوشه‌های پوشه اصلی `View` نیز قرار داد. مثلاً اگر فایل `ViewStart` ای در پوشه `Views/Home` قرار گرفت، این فایل محتوای `ViewStart` اصلی قرار گرفته در ریشه پوشه `Views` را بازنویسی خواهد کرد.

برای معرفی صریح فایل `layout`، تنها کافی است مسیر کامل فایل `layout` را در یک `View` مشخص کنیم:

```
@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<h2>Index</h2>
```

اهمیت این مساله هم در اینجا است که یک سایت می‌تواند چندین `layout` یا `master page` داشته باشد. برای نمونه یک `layout` برای صفحات ورود اطلاعات؛ یک `layout` خاص هم مثلاً برای صفحات گزارش گیری نهایی سایت. همانطور که پیشتر نیز ذکر شد، در ASP.NET حرف ~ به معنای ریشه سایت است که در اینجا ابتدای محل جستجوی فایل `layout` را مشخص می‌کند.

به این ترتیب زمانیکه یک کنترلر، `View` خاصی را فراخوانی می‌کند، کار از فایل `Views\Shared\_Layout.cshtml` شروع خواهد شد. سپس `View` درخواستی پردازش شده و محتوای نهایی آن، جایی که متد `RenderBody` قرار دارد، تزریق خواهد شد. همچنین مقدار `ViewBag.Title` ای که در فایل `View` تعریف شده، در فایل `layout` جهت رندر مقدار تگ `title` استفاده می‌شود (انتقال یک متغیر از `View` به یک فایل `master page`؛ کلاس `layout`، مدل `View` ای را که قرار است رندر کند به ارث می‌برد).

### یک نکته:

در نگارش سوم ASP.NET MVC امکان بکارگیری حرف ~ به صورت مستقیم در حین تعریف یک فایل `js` یا `css` وجود ندارد و حتماً باید از متد سمت سرور `Url.Content` کمک گرفت. در نگارش چهارم ASP.NET MVC، این محدودیت برطرف شده و دقیقاً همانند متغیر `Layout` ای که در بالا مشاهده می‌کنید، می‌توان بدون نیاز به متد `Url.Content`، مستقیماً از حرف ~ کمک گرفت و به صورت خودکار پردازش خواهد شد.

### تزریق نواحی ویژه یک View در فایل layout

توسط متد `RenderBody`، کل محتوای `View` درخواستی در موقعیت تعریف شده آن در فایل `Layout`، رندر می‌شود. این ویژگی به نحو یکسانی به تمام `View`ها اعمال می‌شود. اما اگر نیاز باشد تا `view` بتواند محتوای `markup` قسمت ویژه‌ای از `master page` را مقدار دهی کند، می‌توان از مفهومی به نام `Sections` استفاده کرد:

```
<!DOCTYPE html>
<html>
<head>
    <title>@ViewBag.Title</title>
    <link href="@Url.Content("~/Content/Site.css")" rel="stylesheet" type="text/css" />
    <script src="@Url.Content("~/Scripts/jquery-1.5.1.min.js")" type="text/javascript"></script>
</head>
<body>
```

```

<div id="menu">
    @if (IsSectionDefined("Menu"))
    {
        RenderSection("Menu", required: false);
    }
    else
    {
        <span>This is the default ...!</span>
    }
</div>
<div id="body">
    @RenderBody()
</div>
</body>
</html>

```

در اینجا ابتدا بررسی می‌شود که آیا قسمتی به نام Menu در View جاری که باید رندر شود وجود دارد یا خیر. اگر بله، توسط متد `RenderSection`، آن قسمت نمایش داده خواهد شد. در غیراینصورت، محتوای پیش فرضی را در صفحه قرار می‌دهد. البته اگر از متد `RenderSection` با آرگومان `required: false` استفاده شود، در صورتیکه View جاری حاوی قسمتی به نام مثلا `menu` نباشد، تنها چیزی نمایش داده نخواهد شد. اگر این آرگومان را حذف کنیم، یک استثنای عدم یافت شدن ناحیه یا قسمت مورد نظر صادر می‌گردد.

نحوه‌ی تعریف یک Section در View‌های برنامه به شکل زیر است:

```

@{
    ViewBag.Title = "Index";
    //Layout = null;
    Layout = "~/Views/Shared/_Layout.cshtml";
}
<h2>
    Index</h2>
@section Menu{
    <ul>
        <li>item 1</li>
        <li>item 2</li>
    </ul>
}

```

برای مثال فرض کنید که یکی از View‌های شما نیاز به دو فایل اضافی جاوا اسکریپت برای اجرای صحیح خود دارد. می‌توان تعاریف الحاق این دو فایل را در قسمت `header` فایل `layout` قرار داد تا در تمام View‌ها به صورت خودکار لحاظ شوند. یا اینکه یک section سفارشی را به نحو زیر در آن View خاص تعریف می‌کنیم:

```

@section JavaScript
{
    <script type="text/javascript" src="@Url.Content("~/Scripts/SomeScript.js")" />;
    <script type="text/javascript" src="@Url.Content("~/Scripts/AnotherScript.js")" />;
}

```

سپس کافی است جهت تزریق این کدها به `header` تعریف شده در `master page` مورد نظر، یک سطر زیر را اضافه کرد:

```
@RenderSection("JavaScript", required: false)
```

به این ترتیب، اگر view ایی حاوی تعریف قسمت `JavaScript` نبود، به صورت خودکار شامل تعاریف الحاق اسکریپت‌های یاد شده نیز نخواهد گردید. در نتیجه دارای حجمی کمتر و سرعت بارگذاری بالاتری نیز خواهد بود.

## مدیریت بهتر فایل‌ها و پوشه‌های یک برنامه ASP.NET MVC به کمک Areas

به کمک قابلیت به نام Areas می‌توان یک برنامه بزرگ را به چندین قسمت کوچکتر تقسیم کرد. هر کدام از این نواحی، دارای تعاریف مسیریابی، کنترلرها و Viewهای خاص خودشان هستند. به این ترتیب دیگر به یک برنامه‌ی از کنترل خارج شده ASP.NET MVC که دارای یک پوشه Views به همراه صدها زیر پوشه است، نخواهیم رسید و کنترل این نوع برنامه‌های بزرگ ساده‌تر خواهد شد.

برای مثال یک برنامه بزرگ را در نظر بگیرید که به کمک قابلیت Areas، به نواحی ویژه‌ای مانند گزارشگیری، قسمت ویژه مدیریتی، قسمت کاربران، ناحیه بلاگ سایت، ناحیه انجمن سایت و غیره، تقسیم شده است. به علاوه هر کدام از این نواحی نیز هنوز می‌توانند از اطلاعات ناحیه اصلی برنامه مانند master page آن استفاده کنند. البته باید در نظر داشت که فایل viewStart به پوشه جاری و زیر پوشه‌های آن اعمال می‌شود. اگر نیاز باشد تا اطلاعات این فایل به کل برنامه اعمال شود، فقط کافی است آن را به یک سطح بالاتر، یعنی ریشه سایت منتقل کرد.

### نحوه افزودن نواحی جدید

افزودن یک Area جدید هم بسیار ساده است. بر روی نام پروژه در VS.NET کلیک راست کرده و سپس گزینه Add|Area را انتخاب کنید. سپس در صفحه باز شده، نام دلخواهی را وارد نمائید. مثلاً نام Reporting را وارد نمائید تا ناحیه گزارشگیری برنامه از قسمت‌های دیگر آن مستقل شود. پس از افزودن یک Area جدید، به صورت خودکار پوشه جدیدی به نام Areas به ریشه سایت اضافه می‌شود. سپس داخل آن، پوشه‌ی دیگری به نام Reporting اضافه خواهد شد. پوشه reporting اضافه شده هم دارای پوشه‌های Model، Views و Controllers خاص خود می‌باشد. اکنون که پوشه Areas به ریشه سایت اضافه شده است، با کلیک راست بر روی این پوشه نیز گزینه‌ی Add|Area در دسترس می‌باشد. برای نمونه یک ناحیه جدید دیگر را به نام Admin به سایت اضافه کنید تا بتوان امکانات مدیریتی سایت را از سایر قسمت‌های آن مستقل کرد.

### نحوه معرفی تعاریف مسیریابی نواحی تعریف شده

پس از اینکه کار با Areas را آغاز کردیم، نیاز است تا با نحوه‌ی مسیریابی آن‌ها نیز آشنا شویم. برای این منظور فایل Global.asax.cs قرار گرفته در ریشه اصلی برنامه را باز کنید. در متد Application\_Start، متدی به نام AreaRegistration.RegisterAllAreas، کار ثبت و معرفی تمام نواحی ثبت شده را به فریم ورک، به عهده دارد. کاری که در پشت صحنه انجام خواهد شد این است که به کمک Reflection تمام کلاس‌های مشتق شده از کلاس پایه AreaRegistration به صورت خودکار یافت شده و پردازش خواهند شد. این کلاس‌ها هم به صورت پیش فرض به نام SomeNameAreaRegistration.cs در ریشه اصلی هر Area توسط VS.NET تولید می‌شوند. برای نمونه فایل ReportingAreaRegistration.cs تولید شده، حاوی اطلاعات زیر است:

```
using System.Web.Mvc;

namespace MvcApplication11.Areas.Reporting
{
    public class ReportingAreaRegistration : AreaRegistration
    {
        public override string AreaName
        {
            get
            {
                return "Reporting";
            }
        }

        public override void RegisterArea(AreaRegistrationContext context)
        {
            context.MapRoute(
                "Reporting_default",
                "Reporting/{controller}/{action}/{id}",
                new { action = "Index", id = UrlParameter.Optional }
            );
        }
    }
}
```

```
}
```

توسط `AreaName`، یک نام منحصر بفرد در اختیار فریم ورک قرار خواهد گرفت. همچنین از این نام برای ایجاد پیوند بین نواحی مختلف نیز استفاده می‌شود.

سپس در قسمت `RegisterArea`، یک مسیریابی ویژه خاص ناحیه جاری مشخص گردیده است. برای مثال تمام آدرس‌های ناحیه گزارشگیری سایت باید با `http://localhost/reporting` آغاز شوند تا مورد پردازش قرار گیرند. سایر مباحث آن هم مانند قبل است. برای مثال در اینجا نام اکشن متد پیش فرض، `index` تعریف شده و همچنین ذکر قسمت `id` نیز اختیاری است. همانطور که ملاحظه می‌کنید، تعاریف مسیریابی و اطلاعات پیش فرض آن منطقی هستند و آنچنان نیازی به دستکاری و تغییر ندارند. البته اگر دقت کرده باشید مقدار نام `controller` پیش فرض، مشخص نشده است. بنابراین بد نیست که مثلاً نام `Home` یا هر نام مورد نظر دیگری را به عنوان نام کنترلر پیش فرض در اینجا اضافه کرد.

### تعاریف کنترلرهای هم نام در نواحی مختلف

در ادامه مثال جاری که دو ناحیه `Admin` و `Reporting` به آن اضافه شده، به پوشه‌های `Controllers` هر کدام، یک کنترلر جدید را به نام `HomeController` اضافه کنید. همچنین این `HomeController` را در ناحیه اصلی و ریشه سایت نیز اضافه نمایید. سپس برای متد پیش فرض `Index` هر کدام هم یک `View` جدید را با کلیک راست بر روی نام متد و انتخاب گزینه `Add view`، اضافه کنید. اکنون برنامه را به همین نحو اجرا نمایید. اجرای برنامه با خطای زیر متوقف خواهد شد:

```
Multiple types were found that match the controller named 'Home'. This can happen if the route that services this request ('{controller}/{action}/{id}') does not specify namespaces to search for a controller that matches the request. If this is the case, register this route by calling an overload of the 'MapRoute' method that takes a 'namespaces' parameter.
```

```
The request for 'Home' has found the following matching controllers:
MvcApplication11.Areas.Admin.Controllers.HomeController
MvcApplication11.Controllers.HomeController
```

فوق العاده خطای کاملی است و راه حل را هم ارائه داده است! برای اینکه مشکل ابهام یافتن `HomeController` برطرف شود، باید این جستجو را به فضاهای نام هر قسمت از نواحی برنامه محدود کرد (چون به صورت پیش فرض فضای نامی برای آن مشخص نشده، کل ناحیه ریشه سایت و زیر مجموعه‌های آن را جستجو خواهد کرد). به همین جهت فایل `Global.asax.cs` را گشوده و متد `RegisterRoutes` آن را مثلاً به نحو زیر اصلاح نمایید:

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

    routes.MapRoute(
        "Default", // Route name
        "{controller}/{action}/{id}", // URL with parameters
        new { controller = "Home", action = "Index", id = UrlParameter.Optional } // Parameter defaults
        , namespaces: new[] { "MvcApplication11.Controllers" }
    );
}
```

آرگومان چهارم معرفی شده، آرایه‌ای از نام‌های فضاهای نام مورد نظر را جهت یافتن کنترلرهایی که باید توسط این مسیریابی یافت شوند، تعریف می‌کند.

اکنون اگر مجدداً برنامه را اجرا کنیم، بدون مشکل `View` متناظر با متد `Index` کنترلر `Home` نمایش داده خواهد شد. البته این مشکل با نواحی ویژه و غیر اصلی سایت وجود ندارد؛ چون جستجوی پیش فرض کنترلرها بر اساس ناحیه است.

در ادامه مسیر `http://localhost/Admin/Home` را نیز در مرورگر وارد کنید. سپس بر روی صفحه در مرورگر کلیک راست کرده و سورت صفحه را بررسی کنید. مشاهده خواهید کرد که master page یا فایل layout ایی به آن اعمال نشده است. علت را هم در ابتدای بحث Areas مطالعه کردید. فایل `Views\_ViewStart.cshtml` در سطحی که قرار دارد به ناحیه Admin اعمال نمی‌شود. آن را به ریشه سایت منتقل کنید تا layout اصلی سایت نیز به این قسمت اعمال گردد. البته بدیهی است که هر ناحیه می‌تواند layout خاص خودش را داشته باشد یا حتی می‌توان با مقدار دهی خاصیت Layout نیز در هر view، فایل master page ویژه‌ای را انتخاب و معرفی کرد.

### نحوه ایجاد پیوند بین نواحی مختلف سایت

زمانیکه پیوندی را به شکل زیر تعریف می‌کنیم:

```
@Html.ActionLink(linkText: "Home", actionName: "Index", controllerName: "Home")
```

یعنی ایجاد لینکی در ناحیه جاری. برای اینکه پیوند تعریف شده به ناحیه‌ای خارج از ناحیه جاری اشاره کند باید نام Area را صریحاً ذکر کرد:

```
@Html.ActionLink(linkText: "Home", actionName: "Index", controllerName: "Home",  
routeValues: new { Area = "Admin" }, htmlAttributes: null)
```

همین نکته را باید حین کار با متد `RedirectToAction` نیز در نظر داشت:

```
public ActionResult Index()  
{  
    return RedirectToAction("Index", "Home", new { Area = "Admin" });  
}
```

## نظرات خوانندگان

نویسنده: Mohsen

تاریخ: ۱۴:۴۴:۱۳ ۱۳۹۱/۰۱/۲۶

با سلام و عرض خسته نباشید بابت این دوره ی خوب آموزشی شما مهندس نصیری:  
در قسمت "تزریق نواحی ویژه یک View در فایل layout" در صورتی که در ابتدای دستور :  
;(RenderSection("Menu", required: false  
از @ استفاده نکنیم با خطای زیر مواجه می شویم:

The following sections have been defined but have not been rendered for the layout page  
("~/Views/Shared/\_Layout.cshtml": "Menu" ضمن اینکه دستور در داخل بلاک است!(!)

نویسنده: وحید نصیری

تاریخ: ۱۶:۰۶:۲۶ ۱۳۹۱/۰۱/۲۶

بله. درسته. علت هم این است که خروجی RenderSection نهایتاً یک رشته است که باید توسط @ در Response درج شود.

نویسنده: Foad abdollahi

تاریخ: ۱۸:۳۶:۴۱ ۱۳۹۱/۰۲/۱۰

سلام میشه لطف کنید در مورد تعریف یک area به صورت sub domain و همچنین ایجاد ساب دامین داینامیک مطلب به اشتراک بگذارید من چند موردی تو سایت های خارجی دیدم ولی اکثراً با جواب نمی داد یا فقط رو 2 mvc بود

نویسنده: وحید نصیری

تاریخ: ۱۹:۴۱:۴۲ ۱۳۹۱/۰۲/۱۰

خلاصه سؤال در حالت کلی: قصد داریم آدرس متداول site.com/admin/report/list/1 را به admin.site.com/report/list/1 نگاشت کنیم. آیا امکان پذیر است؟  
این سؤال مستقیماً به Areas مرتبط نمی شود. اصل سؤال این است: «آیا Routing و مسیریابی، از ساب دومین ها پشتیبانی می کند؟» پاسخ: به صورت توکار خیر. اما با استفاده از [IIS7 URL Rewrite module](#) می تونید اینکار رو انجام بدید. یا اینکه باید کد نویسی کنید: (^)

نویسنده: Foad abdollahi

تاریخ: ۲۲:۲۰:۵۳ ۱۳۹۱/۰۲/۱۰

ممنون که جواب دادید  
ولی من این 2 تا پست را دیدم که ادعا دارن که با route می شه این کار رو هندل کرد  
<http://admsteck.blogspot.com/2010/04/aspnet-mvc-domain-routing-and-areas.html>  
<http://blog.maartenballiauw.be/post/2009/05/20/ASPNET-MVC-Domain-Routing.aspx>  
ولی من نتونستم درست پیاده سازی کنم ولی از بعضی کامنت ها اینطور معلوم بود که جواب گرفتن

شما نظرتون چیه؟

من در مورد url rewrite تو asp form مطلب دیدم تو mvc نه دیدم و نه میدونم چه فرقی در نحوه کارش داره

نویسنده: وحید نصیری

تاریخ: ۲۳:۴۰:۴۸ ۱۳۹۱/۰۲/۱۰

این مورد به IIS7 مربوط است البته: (^)

نویسنده: میثم

تاریخ: ۱۰:۱۹ ۱۳۹۱/۰۴/۲۶

سلام و خسته نباشید

من فایل ViewStart.cshtml رو به ریشه منتقل کردم اما خطای زیر رو دریافت کردم ممنون میشم اگه راهنمایی بفرمایید.

Unable to cast object of type 'ASP.\_Page\_ViewStart\_cshtml' to type 'System.Web.WebPages.StartPage'.

و سوال دیگر این که امکان استفاده از master page های تو در تو شبیه به ASP.NET Web Forms وجود دارد؟

نویسنده: وحید نصیری

تاریخ: ۹:۴۵ ۱۳۹۱/۰۴/۲۷

- فایل ViewStart فقط باید در پوشه View قرار گیرد و نه جای دیگری. مسایل در نگارش های مختلف اندکی فرق کردن.  
- بله. [امکانش هست](#).

نویسنده: بهروز

تاریخ: ۹:۵۴ ۱۳۹۱/۱۲/۲۶

سلام

من نیاز دارم که به صفحه مستر خودم یک مدل پاس بدهم و قسمتهایی از مستریج با آن مدل پر شود  
چطور می توانم این کار را انجام دهم؟ (یعنی می خواهم به ازای همه اکشن متد هایی که کاربر فراخوانی می کند یک اکشن متد دیگه  
هم که مربوط به صفحه layout\_ هست هم فراخوانی شده و مقداری را به layout\_ ارسال کند)

نویسنده: وحید نصیری

تاریخ: ۹:۵۹ ۱۳۹۱/۱۲/۲۶

از [Html.RenderAction](#) استفاده کنید.

نویسنده: debugger

تاریخ: ۹:۲۹ ۱۳۹۲/۰۳/۰۷

با سلام: آیا این امکان وجود دارد که هر Area به صورت یک Assembly مجزا تبدیل گردد ؟  
آیا Area روشی برای ماژولار بودن برنامه های MVC است ؟

نویسنده: وحید نصیری

تاریخ: ۹:۴۰ ۱۳۹۲/۰۳/۰۷

امکان قرار دادن فایل های View در یک اسمبلی جداگانه وجود دارد:

« [توزیع پروژه های ASP.NET MVC بدون ارائه فایل های View آن](#) »

نویسنده: صابر فتح الهی

تاریخ: ۱۱:۴۷ ۱۳۹۲/۰۳/۰۷

البته شما می تونین فایل های View را به صورت یک Resource در اسمبلی قرار بدین اما باید خودتون یک انجین سفارشی برای  
بازیابی آنها طراحی کنید.

نویسنده: محمد میرزایی



تاریخ: ۱۶:۲۷ ۱۳۹۲/۰۵/۲۹

سلام،

من به لینک توی مستر پیچ قرار دادم که به یک ویو داخل Area میبره و این لینک درست کار می‌کنه

```
<li>@Html.ActionLink("Admin", "Index", "Home", routeValues: new { Area = "Administration" },
htmlAttributes: null)</li>
```

اما مشکل اینه که بعد این که روی این لینک کلیک کردم و به ویو مورد نظر در اون Area رفتم حالا وقتی روی بقیه لینک‌ها کلیک می‌کنم تا به ویوهای مورد نظر که بیرون از اون Area هست درست کار نمی‌کنه یعنی همچنان داخل اون Area دنبال اون ویو می‌گرده

نویسنده: وحید نصیری  
تاریخ: ۱۷:۳۲ ۱۳۹۲/۰۵/۲۹

- بله. در انتهای مطلب هم ذکر شده «زمانیکه پیوندی را به شکل زیر تعریف می‌کنیم (بدون ذکر نام ناحیه)، یعنی ایجاد لینکی در ناحیه جاری.»  
- برای ایجاد یک لینک از داخل یک ناحیه به خارج از آن حتما باید Area مورد نظر ذکر شود و اگر داخل Area خاصی نیست، این Area را باید با string.Empty مقدار دهی کرد:

```
@Html.ActionLink("Back", "Index", "Home", new { area = "" }, null)
```

و اگر نمی‌خواهید درگیر این مسایل شوید، بهتر است از [T4MVC](#) استفاده کنید:

```
// برای لینک دادن معمولی
@Html.ActionLink("Back", MVC.Home.Index())
// برای لینک دادن به یک ناحیه
@Html.ActionLink("Some Link", MVC.Admin.SomeController.SomeAction())
```

نویسنده: رضا منصوری  
تاریخ: ۱۳:۲۴ ۱۳۹۲/۰۸/۲۰

با تشکر از مطلب عالیتون در استفاده از @RenderBody در Layout\_ (مستر پیچم) وقتی @RenderBody را در جای مد نظر خودم استفاده می‌کنم لینک‌های Post در صفحات کار نمی‌کنه مثلا در کنترلر لوگین وقتی روی دکمه‌ی Submit کلیک می‌کنم به Action مورد نظر نمی‌رود و چنین Url تشکیل می‌شود

📍 [localhost:9764/Account/Login?\\_RequestVerificationToken=nBQV3rE8qXQ6Z\\_Qchu-Ns6gdj5xo3kAxmRgACFZMmQGcKQHgASI](http://localhost:9764/Account/Login?_RequestVerificationToken=nBQV3rE8qXQ6Z_Qchu-Ns6gdj5xo3kAxmRgACFZMmQGcKQHgASI) ☆

ولی وقتی @RenderBody رو ابتدای Layout\_ (مستر پیچم) می‌زارم بعد از کلیک وارد اکشن مربوطه میشه و درست کار می‌کنه!

جایی که اکشن‌ها درست کار می‌کنه (بعد از تگ &lt;body&gt;)

```

</head>
<body>
    @RenderBody()
    <form id="frmmain" runat="server">
        <!-- BEGIN wrapper -->
        <div id="wrapper">

```

جایی که درست کار نمیکنه ( وسط ویو و جای مورد نظر ما ). به نظر @RenderBody باید جای خاصی باشه ؟

با تشکر از راهنماییتون

نویسنده: وحید نصیری  
تاریخ: ۱۳:۵۰ ۱۳۹۲/۰۸/۲۰

- از @RenderBody در فایل layout باید استفاده شود و نه در یک فایل View.  
- runat server چرا؟ این MVC هست نه وب فرم.  
- در MVC می شود به ازای یک صفحه چندین فرم داشت که هر کدام به اکشن متد و کنترلر خاصی اشاره می کنند. فرق دارد با وب فرم های تک فرمه.  
- در MVC نباید یک فرم را به صورت کلی در فایل layout قرار داد. باید در جای مناسب در View مرتبط قرار گیرد آن هم با ذکر [Html.BeginForm](#) تا [action](#) فرم به صورت خودکار به اکشن متد متناظر در کنترلر خاصی هدایت شود. آن هم قسمت کوچکی از صفحه و نه post کل صفحه به سرور؛ برخلاف وب فرم ها (MVC بهینه تر عمل می کند از این لحاظ و حجم کمتری را به سرور ارسال می کند). اگر فرمی action نداشته باشد، الزامی ندارد که حتما به اکشن متد مدنظر شما هدایت شود. در این حالت محتویات آن به آدرس جاری صفحه ارسال می شوند (یعنی اکشن متد پیش فرض تعریف شده در مسیریابی سایت).  
- زمانی اجزای فرم در Url به صورت کوئری استرینگ نمایان می شوند که متد ارسال اطلاعات Get باشد و نه [FormMethod.Post](#) (اگر نحوه ارسال اطلاعات صریحا ذکر نشود، از حالت [Get](#) استفاده می شود).

نویسنده: رضا منصوری  
تاریخ: ۱۶:۳۸ ۱۳۹۲/۰۸/۲۰

از پاسخ کامل شما سپاسگزارم مشکل از همان مورد Runat=Server بود. بسیار ممنون.

نویسنده: شقایق  
تاریخ: ۱۰:۴۴ ۱۳۹۲/۰۹/۰۶

با سلام من از همین روش استفاده کردم و روی لوکال هیچ مشکلی ندارد و به خوبی کار میکند اما وقتی پروژه را روی هاست آپلود می کنم مثلا اگه در Area به این صورت تعریف کرده باشم

```

@using (Html.BeginForm("GotoIndex1", "Home", new { area="Admin"}))
{
    <input type="submit" value="Send" />
}

```

در زمان اجرا بر روی هاست به این شکل می شود

```

<form method="post" action="">
<input type="submit" value="Send">
</form>

```

انگار که اکشن را مقدار دهی نکردم .

فایل AdminAreaRegistration.cs به صورت زیر می باشد:

```
public override void RegisterArea(AreaRegistrationContext context)
{
    context.MapRoute(
        "Admin_default11",
        "Admin/{controller}/{action}/{id}/{*Extparam}",
        new { action = "Index", controller = "Home", id = UrlParameter.Optional }
    );
}
```

و فایل RouteConfig به صورت زیر است

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

باز هم تاکید می کنم که روی لوکال بدون هیچ مشکلی اجرا می شود. اما روی سرور که می رود انگار action فرم مقدار دهی نشده.

نویسنده: وحید نصیری  
تاریخ: ۱۱:۳۲ ۱۳۹۲/۰۹/۰۶

در نگارش های قبلی MVC (یعنی هاست شما آنچنان به روز نیست) اگر در Route تعریف شده دو پارامتر اختیاری کنار هم قرار می گرفتند، سبب می شدند تا URL تولید شده برای آن ها خالی باشد ( ^ ). بنابراین در Admin\_default11 تعریف شده که دو پارامتر اختیاری کنار هم دارد، باید برای آن ها یک مقدار اولیه در نظر بگیرید.

نویسنده: منصور  
تاریخ: ۱۰:۱۸ ۱۳۹۲/۱۲/۰۷

سلام؛ زمانی که ما در سایتمون یک قسمت داریم مربوط به بخش آخرین اخبار و یک بخش داریم مربوط به لینک دوستان و... یا بطور کلی شاید بار ها...  
وقتی که ما متد RenderBody() رو در MasterPage تعریف میکنیم برای تعریف سایدبار باید در هر View اونو تعریف کنیم(همون آخرین اخبار) ؟ یا اینکه در همون فایل Layout تعریف میشن؟

نویسنده: وحید نصیری  
تاریخ: ۱۰:۳۴ ۱۳۹۲/۱۲/۰۷

از [Html.RenderAction](#) در layout استفاده کنید.

نویسنده: منصور  
تاریخ: ۱۱:۵۵ ۱۳۹۲/۱۲/۰۷

بسیار ممنونم که پاسخ دادید. یعنی اینکه پیام در یک پارشال ویوو بخش مربوطه رو قرار بدم و بعدش از [Html.RenderAction](#) در layout استفاده کنم؟

نویسنده: وحید نصیری  
تاریخ: ۱۲:۰۶ ۱۳۹۲/۱۲/۰۷

بله. در فایل layout به هر تعداد ChildAction که نیاز باشد، قابل تعریف و رندر هستند.  
در اینجا برای کش کردن و کاهش بار سیستم می توان یک ChildAction خاص را طراحی کرد که Partial View آن متشکل از چند

Html.RenderPartial باشد. زمانیکه OutputCache روی آن قرار داده می‌شود، تمام زیر مجموعه‌ها با هم و یکباره کش خواهند شد.

برای مثال اگر قسمت سمت راست صفحه از 5 ویجت تشکیل می‌شود، نیازی نیست 5 بار از Html.RenderAction در فایل Layout استفاده کنید. یک اکشن متد کلی طراحی کنید که توسط ViewModel ایی مشخص، دیتای View متشکل از چند Partial View خودش را که از چند Html.RenderPartial استفاده می‌کند، تامین کند. بعد در فایل Layout فقط همین تک اکشن متد OutputCache دار را توسط Html.RenderAction رندر کنید.

نویسنده: م سلیمانی  
تاریخ: ۱۳۹۳/۱۱/۰۸ ۱:۳۳

با سلام

چطور میشه به Area رو به عنوان یه پروژه جدا تو پترن‌های چند لایه ایجاد کرد.  
مثلا تو NopCommerce این کار انجام شده. Admin Area به عنوان یه پروژه جدا تعریف شده!

نویسنده: وحید نصیری  
تاریخ: ۱۳۹۳/۱۱/۰۸ ۱:۴۴

« [توزیع پروژه‌های ASP.NET MVC بدون ارائه فایل‌های View آن](#) »