عنوان: ASP.NET MVC #5
نویسنده: وحید نصیری
تاریخ: ۰۹:۵۳:۰۰ ۱۳۹۱/۰۱/۰۹
آدرس: www.dotnettips.info
گروه‌ها: MVC

بررسی نحوه انتقال اطلاعات از یک کنترلر به Viewهای مرتبط با آن

در ASP.NET Web forms در فایل code behind یک فرم مثلا می‌توان نوشت Label1.Text و سپس مقداری را به آن انتساب داد. اما اینجا به چه ترتیبی می‌توان شبیه به این نوع عملیات را انجام داد؟ با توجه به اینکه در کنترلرها هیچ نوع ارجاع مستقیمی به اشیاء رابط کاربری وجود ندارد و این دو از هم مجزا شده‌اند.
در پاسخ به این سؤال، همان مثال ساده قسمت قبل را ادامه می‌دهیم. یک پروژه جدید خالی ایجاد شده است به همراه HomeController ایی که به آن اضافه کرده‌ایم. همچنین مطابق روشی که ذکر شد، View ایی به نام Index را نیز به آن اضافه کرده‌ایم. سپس برای ارسال اطلاعات از یک کنترلر به View از یکی از روش‌های زیر می‌توان استفاده کرد:

الف) استفاده از اشیاء پویا

ViewBag یک شیء dynamic است که در دات نت 4 امکان تعریف آن میسر شده است. به این معنا که هر نوع خاصیت دلخواهی را می‌توان به این شیء انتساب داد و سپس این اطلاعات در View نیز قابل دسترسی و استخراج خواهد بود. مثلا اگر در اینجا به شیء ViewBag، خاصیت دلخواه Country را اضافه کنیم و سپس مقداری را نیز به آن انتساب دهیم:

```
using System.Web.Mvc;

namespace MvcApplication1.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            ViewBag.Country = "Iran";
            return View();
        }
    }
}
```

این اطلاعات در View مرتبط با اکشنی به نام Index به نحو زیر قابل بازیابی خواهد بود (نحوه اضافه کردن View متناظر با یک اکشن یا متد را هم در قسمت قبل با تصویر مرور کردیم):

```
@{
    ViewBag.Title = "Index";
}
<h2>
    Index</h2>
<p>
    Country : @ViewBag.Country
</p>
```

در این مثال، @ در View engine جاری که Razor نام دارد، به این معنا می‌باشد که این مقداری است که می‌خواهم دریافت کنی (ViewBag.Country) و سپس آن‌را در حین پردازش صفحه نمایش دهی.

ب) انتقال اطلاعات یک شیء کامل و غیر پویا به View

هر پروژه جدید MVC به همراه پوشه‌ای به نام Models است که در آن می‌توان تعاریف اشیاء تجاری برنامه را قرار داد. در پروژه جاری، یک کلاس ساده را به نام Employee به این پوشه اضافه می‌کنیم:

```
namespace MvcApplication1.Models
{
    public class Employee
    {
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public string Email { get; set; }
    }
}
```

اکنون برای نمونه یک وهله از این شیء را در متد Index ایجاد کرده و سپس به view متناظر با آن ارسال می‌کنیم (در قسمت return View کد زیر مشخص است). بدیهی است این وهله سازی در عمل می‌تواند از طریق دسترسی به یک بانک اطلاعاتی یا یک وب سرویس و غیره باشد.

```
using System.Web.Mvc;
using MvcApplication1.Models;

namespace MvcApplication1.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            ViewBag.Country = "Iran";

            var employee = new Employee
            {
                 Email = "name@site.com",
                 FirstName = "Vahid",
                 LastName = "N."
            };

            return View(employee);
        }
    }
}
```

امضاهای متفاوت (overloads) متد کمکی View هم به شرح زیر هستند:

```
ViewResult View(Object model)
ViewResult View(string viewName, Object model)
ViewResult View(string viewName, string masterName, Object model)
```

اکنون برای دسترسی به اطلاعات این شیء employee در View متناظر با این متد، چندین روش وجود دارد:

```
@{
    ViewBag.Title = "Index";
}
<h2>
```

```
    Index</h2>
<div>
    Country: @ViewBag.Country <br />
    FirstName: @Model.FirstName
</div>
```

می‌توان از طریق شیء استاندارد دیگری به نام Model (که این هم یک شیء dynamic است مانند ViewBag قسمت قبل)، به خواص شیء یا مدل ارسالی به View جاری دسترسی پیدا کرد که یک نمونه از آن‌را در اینجا ملاحظه می‌کنید.
روش دوم، بر اساس **تعریف صریح نوع مدل** است به نحو زیر:

```
@model MvcApplication1.Models.Employee
@{
    ViewBag.Title = "Index";
}
<h2>
    Index</h2>
<div>
    Country: @ViewBag.Country
    <br />
    FirstName: @Model.FirstName
</div>
```

در اینجا در مقایسه با قبل، تنها یک سطر به اول فایل View اضافه شده است که در آن نوع شیء Model تعیین می‌گردد (کلمه model هم در اینجا با حروف کوچک شروع شده است). به این ترتیب اینبار اگر سعی کنیم به خواص این شیء دسترسی پیدا کنیم، Intellisense ویژوال استودیو ظاهر می‌شود. به این معنا که شیء Model بکارگرفته شده اینبار دیگر dynamic نیست و دقیقا می‌داند که چه خواصی را باید پیش از اجرای برنامه در اختیار استفاده کننده قرار دهد.
به این روش، روش **Strongly typed view** هم گفته می‌شود؛ چون View دقیقا می‌داند که چون نوعی را باید انتظار داشته باشد؛ تحت نظر کامپایلر قرار گرفته و همچنین Intellisense نیز برای آن مهیا خواهد بود.
به همین جهت این روش Strongly typed view، در بین تمام روش‌های مهیا، به عنوان روش توصیه شده و مرجح مطرح است.
به علاوه استفاده از Strongly typed views یک مزیت دیگر را هم به همراه دارد: فعال شدن یک code generator توکار در VS.NET به نام **scaffolding** . یک مثال ساده:
تا اینجا ما اطلاعات یک کارمند را نمایش دادیم. اگر بخواهیم یک لیست از کارمندها را نمایش دهیم چه باید کرد؟
روش کار با قبل تفاوتی نمی‌کند. اینبار در return View ما، یک شیء لیستی ارائه خواهد شد. در سمت View هم با یک حلقه foreach کار نمایش این اطلاعات صورت خواهد گرفت. راه ساده‌تری هم هست. اجازه دهیم تا خود VS.NET، کدهای مرتبط را برای ما تولید کند.
یک کلاس دیگر به پوشه مدل‌های برنامه اضافه کنید به نام Employees با محتوای زیر:

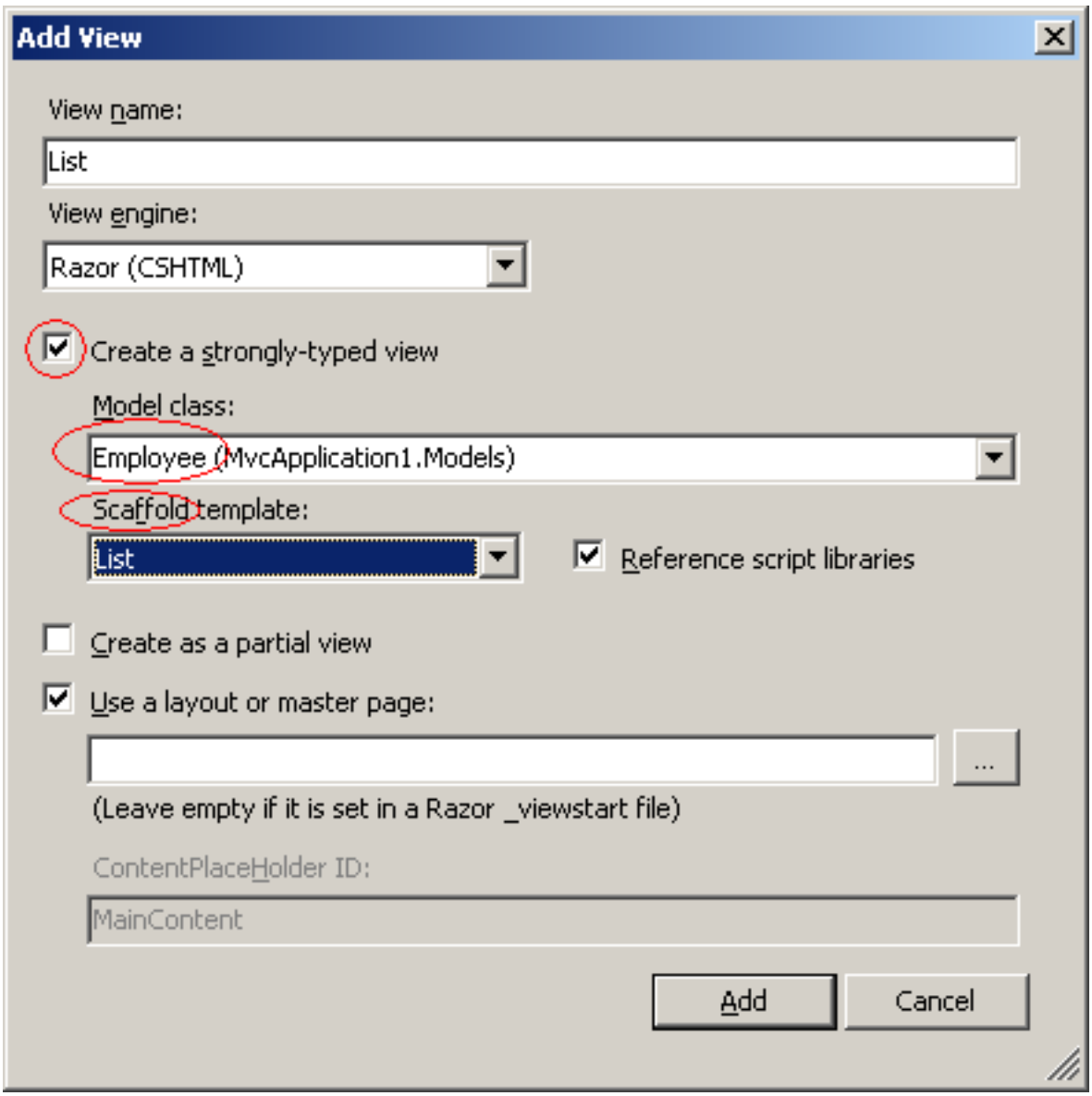
```
using System.Collections.Generic;

namespace MvcApplication1.Models
{
    public class Employees
    {
        public IList<Employee> CreateEmployees()
        {
            return new[]
                {
                    new Employee { Email = "name1@site.com", FirstName = "name1", LastName =
"LastName1" },
                    new Employee { Email = "name2@site.com", FirstName = "name2", LastName =
"LastName2" },
                    new Employee { Email = "name3@site.com", FirstName = "name3", LastName =
"LastName3" }
                };
        }
    }
}
```

سپس متد جدید زیر را به کنترلر Home اضافه کنید.

```
public ActionResult List()
{
    var employeesList = new Employees().CreateEmployees();
    return View(employeesList);
}
```

برای اضافه کردن View متناظر با آن، روی نام متد کلیک راست کرده و گزینه Add view را انتخاب کنید. در صفحه ظاهر شده:



تیک مربوط به Create a strongly typed view را قرار دهید. سپس در قسمت Model class، کلاس Employee را انتخاب کنید (نه Employees جدید را، چون از آن می‌خواهیم به عنوان منبع داده لیست تولیدی استفاده کنیم). اگر این کلاس را مشاهده نمی‌کنید، به

این معنا است که هنوز برنامه را یکبار کامپایل نکرده‌اید تا VS.NET بتواند با اعمال Reflection بر روی اسمبلی برنامه آن‌را پیدا کند. سپس در قسمت Scaffold template گزینه List را انتخاب کنید تا Code generator توکار VS.NET فعال شود. اکنون بر روی دکمه Add کلیک نمائید تا View نهایی تولید شود. برای مشاهده نتیجه نهایی مسیر http://localhost/Home/List باید بررسی گردد.

ج) استفاده از ViewDataDictionary

ViewDataDictionary از نوع IDictionary با کلیدی رشته‌ای و مقداری از نوع object است. توسط آن شیءایی به نام ViewData در ASP.NET MVC به نحو زیر تعریف شده است:

```
public ViewDataDictionary ViewData { get; set; }
```

این روش در نگارش‌های اولیه ASP.NET MVC بیشتر مرسوم بود. برای مثال:

```
using System;
using System.Web.Mvc;

namespace MvcApplication1.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            ViewData["DateTime"] = "<br/>" + DateTime.Now;
            return View();
        }
    }
}
```

و سپس جهت استفاده از این ViewData تعریف شده با کلید دلخواهی به نام DateTime در View متناظر با اکشن Index خواهیم داشت:

```
@{
    ViewBag.Title = "Index";
}
<h2>
    Index</h2>
<div>
    DateTime: @ViewData["DateTime"]
</div>
```

یک نکته امنیتی:

اگر به مقدار انتساب داده شده به شیء ViewDataDictionary دقت کنید، یک تگ br هم به آن اضافه شده است. برنامه را یکبار اجرا کنید. مشاهده خواهید کرد که این تگ به همین نحو نمایش داده می‌شود و نه به صورت یک سطر جدید HTML . چرا؟ چون Razor به صورت پیش فرض اطلاعات را encode شده (فراخوانی متد Html.Encode در پشت صحنه به صورت خودکار) در صفحه نمایش می‌دهد و این مساله از لحاظ امنیتی بسیار عالی است؛ زیرا جلوی بسیاری از حملات cross site scripting یا XSS را خواهد گرفت.

احتمالا الان این سؤال پیش خواهد آمد که اگر «عالمانه» بخواهیم این رفتار نیکوی پیش فرض را غیرفعال کنیم چه باید کرد؟ برای این منظور می‌توان نوشت:

```
@Html.Raw(myString)
```

و یا:

```
<div>@MvcHtmlString.Create("<h1>HTML</h1>")</div>
```

به این ترتیب خروجی Razor دیگر encode شده نخواهد بود.

د) استفاده از TempData

TempData نیز یک dictionary دیگر برای ذخیره سازی اطلاعات است و به نحو زیر در فریم ورک تعریف شده است:

```
public TempDataDictionary TempData { get; set; }
```

TempData در پشت صحنه از سشن‌های ASP.NET جهت ذخیره سازی اطلاعات استفاده می‌کند. بنابراین اطلاعات آن در سایر کنترلرها و View ها نیز در دسترس خواهد بود. البته TempData یک سری تفاوت هم با سشن معمولی ASP.NET دارد:
- بلافاصله پس از خوانده شدن، حذف خواهد شد.
- پس از پایان درخواست از بین خواهد رفت.
هر دو مورد هم به جهت بالابردن کارآیی برنامه‌های ASP.NET MVC و مصرف کمتر حافظه سرور درنظر گرفته شده‌اند.
البته کسانی که برای بار اول هست با ASP.NET مواجه می‌شوند، شاید سؤال بپرسند این مسایل چه اهمیتی دارد؟ پروتکل HTTP، ذاتا یک پروتکل «بدون حالت» است یا Stateless هم به آن گفته می‌شود. به این معنا که پس از ارائه یک صفحه وب توسط سرور، تمام اشیاء مرتبط با آن در سمت سرور تخریب خواهند شد. این مورد متفاوت است با برنامه‌های معمولی دسکتاپ که طول عمر یک شیء معمولی تعریف شده در سطح فرم به صورت یک فیلد، تا زمان باز بودن آن فرم، تعیین می‌گردد و به صورت خودکار از حافظه حذف نمی‌شود. این مساله دقیقا مشکل تمام تازه واردها به دنیای وب است که چرا اشیاء ما نیست و نابود شدند. در اینجا وب سرور قرار است به هزاران درخواست رسیده پاسخ دهد. اگر قرار باشد تمام این اشیاء را در سمت سرور نگهداری کند، خیلی زود با اتمام منابع مواجه می‌گردد. اما واقعیت این است که نیاز است یک سری از اطلاعات را در حافظه نگه داشت. به همین منظور یکی از چندین روش مدیریت حالت در ASP.NET استفاده از سشن‌ها است که در اینجا به نحو بسیار مطلوبی، با سربار حداقل توسط TempData مدیریت شده است.

یک مثال کاربردی در این زمینه:

فرض کنید در متد جاری کنترلر، ابتدا بررسی می‌کنیم که آیا ورودی دریافتی معتبر است یا خیر. در غیراینصورت، کاربر را به یک View دیگر از طریق کنترلری دیگر جهت نمایش خطاها هدایت خواهیم کرد.
همین «هدایت مرورگر به یک View دیگر» یعنی پاک شدن و تخریب اطلاعات کنترلر قبلی به صورت خودکار. بنابراین نیاز است این اطلاعات را در TempData قرار دهیم تا در کنترلری دیگر قابل استفاده باشد:

```
using System;
using System.Web.Mvc;

namespace MvcApplication1.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult InsertData(string name)
        {
            // Check for input errors.
            if (string.IsNullOrWhiteSpace(name))
            {
                TempData["error"] = "name is required.";
                return RedirectToAction("ShowError");
            }
```

```
            // No errors
            // ...
            return View();
        }

        public ActionResult ShowError()
        {
            var error = TempData["error"] as string;
            if (!string.IsNullOrWhiteSpace(error))
            {
                ViewBag.Error = error;
            }
            return View();
        }
    }
}
```

در همان HomeController دو متد جدید به نام‌های InsertData و ShowError اضافه شده‌اند. در متد InsertData ابتدا بررسی می‌شود که آیا نامی وارد شده است یا خیر. اگر خیر توسط متد RedirectToAction، کاربر به اکشن یا متد ShowError هدایت خواهد شد.
برای انتقال اطلاعات خطایی که می‌خواهیم در حین این Redirect نمایش دهیم نیز از TempData استفاده شده است.
بدیهی است برای اجرا این مثال نیاز است دو View جدید برای متدهای InsertData و ShowError ایجاد شوند (کلیک راست روی نام متد و انتخاب گزینه Add view برای اضافه کردن View مرتبط با آن اکشن).
محتوای View مرتبط با متد افزودن اطلاعات فعلا مهم نیست، ولی View نمایش خطاها در ساده‌ترین حالت مثلا می‌تواند به صورت زیر باشد:

```
@{
    ViewBag.Title = "ShowError";
}

<h2>Error</h2>

@ViewBag.Error
```

برای آزمایش برنامه هم مطابق مسیریابی پیش فرض و با توجه به قرار داشتن در کنترلری به نام Home، مسیر http://localhost/Home/InsertData ابتدا باید بررسی شود. چون آرگومانی وارد نشده، بلافاصله صفحه به آدرس http://localhost/Home/ShowError به صورت خودکار هدایت خواهد شد.

نکته‌ای تکمیلی در مورد Strongly typed viewها:

عنوان شد که Strongly typed view روش مرجح بوده و بهتر است از آن استفاده شود، زیرا اطلاعات اشیاء و خواص تعریف شده در یک View تحت نظر کامپایلر قرار می‌گیرند که بسیار عالی است. یعنی اگر در View بنویسم @Model.FirstName1 :FirstName چون FirstName1 وجود خارجی ندارد، برنامه نباید کامپایل شود. یکبار این را بررسی کنید. برنامه بدون مشکل کامپایل می‌شود! اما تنها در زمان اجرا است که صفحه زرد رنگ معروف خطاهای ASP.NET ظاهر می‌شود که چنین خاصیتی وجود ندارد (این حالت پیش فرض است؛ یعنی کامپایل یک View در زمان اجرا). البته این باز هم خیلی بهتر است از ViewBag، چون اگر مثلا ViewBag.Country1 را وارد کنیم، در زمان اجرا تنها چیزی نمایش داده نخواهد شد؛ اما با روش Strongly typed view، حتما خطای Compilation Error به همراه نمایش محل مشکل نهایی، در صفحه ظاهر خواهد شد.
سؤال: آیا می‌شود پیش از اجرای برنامه هم این بررسی را انجام داد؟
پاسخ: بله. باید فایل پروژه را اندکی ویرایش کرده و مقدار MvcBuildViews را که به صورت پیش فرض false هست، true نمود. یا خارج از ویژوال استودیو با یک ادیتور متنی ساده مثلا فایل csproj را گشوده و این تغییر را انجام دهید. یا داخل ویژوال استودیو، بر روی نام پروژه کلیک راست کرده و سپس گزینه Unload Project را انتخاب کنید. مجددا بر روی این پروژه Unload شده کلیک راست نموده و گزینه edit را انتخاب نمائید. در صفحه باز شده، MvcBuildViews را یافته و آن‌را true کنید. سپس پروژه را Reload کنید.

اکنون اگر پروژه را کامپایل کنید، پیغام خطای زیر پیش از اجرای برنامه قابل مشاهده خواهد بود:

```
'MvcApplication1.Models.Employee' does not contain a definition for 'FirstName1'
and no extension method 'FirstName1' accepting a first argument of type
'MvcApplication1.Models.Employee'
could be found (are you missing a using directive or an assembly reference?)
d:\Prog\MvcApplication1\MvcApplication1\Views\Home\Index.cshtml 10 MvcApplication1
```

البته بدیهی است این تغییر، زمان Build پروژه را مقداری افزایش خواهد داد؛ اما امن‌ترین حالت ممکن برای جلوگیری از این نوع خطاهای تایپی است.
یا حداقل بهتر است یکبار پیش از ارائه نهایی برنامه این مورد فعال و بررسی شود.

و یک خبر خوب!

مجوز سورس کد ASP.NET MVC از MS-PL به Apache تغییر کرده و همچنین Razor و یک سری موارد دیگر هم سورس باز شده‌اند.
این تغییرات به این معنا خواهند بود که پروژه از حالت فقط خواندنی MS-PL به حالت متداول یک پروژه سورس باز که شامل دریافت تغییرات و وصله‌ها از جامعه برنامه نویس‌ها است، تغییر کرده است (^ و ^).

نظرات خوانندگان

نویسنده: علی قمشلویی
تاریخ: ۱۳۹۱/۰۱/۰۹ ۱۱:۱۴:۳۷

با سلام و تشکر
با آموزش های فوق العاده شما و قدرت شگفت انگیز MVC فکر نمیکنم دیگه با web form وب بنویسم هر چند برای پروژه های کوچک.

نویسنده: محمد صاحب
تاریخ: ۱۳۹۱/۰۱/۰۹ ۱۱:۲۵:۵۶

با تشکر از شما.

میخواستم بدونم امکان از بین رفتن مقدار TempData تو متد ShowError وجود داره که شما قبل از استفاده برای NULL یا خالی بودن چکش کردید؟

ضمنا درخواستی داشتم من تو وب تازه کارم برا
همین CSS وHTML رو خوب بلد نیستم دنبال منبع آموزشی برای
یادگیریشون هستم.البته میخوام مواردی رو که بیشتر درMVC کاربرد داره رو یاد بگیرم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۱/۰۹ ۱۲:۰۰:۰۲

- خیر. ولی ممکن است یک نفر مستقیما مسیر http://localhost/Home/ShowError را در مرورگر وارد کند که کار غیرمجازی نیست. در این حالت TempData نال خواهد بود چون منشاء آن یعنی http://localhost/Home/InsertData پیشتر فراخوانی نشده است.
- یک سری فریم ورک CSS این روزها خیلی باب است. یک دوره آموزشی هم در این زمینه اینجا هست: A Better CSS: LESS and) (SASS

نویسنده: علیرضا رحیم زاده
تاریخ: ۱۳۹۱/۰۱/۰۹ ۱۹:۱۰:۱۲

سلام
واقعا از آموزش هایی که میگزارید ممنونم.
من از این سری مقالات علاوه بر MVC خیلی نکته های جالب دیگه رو هم دارم یاد میگیرم.
فقط لطفا سطح آموزش هاتون رو همینطور آروم آروم بالا ببرید تا ما هم بتونیم پا به پای آموزشها یاد بگیریم.

نویسنده: Salehi
تاریخ: ۱۳۹۱/۰۱/۰۹ ۱۹:۵۱:۲۰

مورد tempdata خیلی جالب بود. بخصوص که آخر درخواست تخریب میشه. فکر کنم کاربردش بیشتر وقتیه که بین کنترلرهای مختلف باید داده ردو بدل بشه.

نویسنده: Ahmad Moghadas khoo
تاریخ: ۱۳۹۱/۰۱/۱۰ ۰۴:۲۲:۵۵

سلام

سال نو بر شما مبارک
عذر خواهی می کنم امکان ارسال سوالم در زیر مطلب مربوطه نبود.
خواستم سوال کنم در مثالی که در این مطلب http://www.dotnettips.info/2012/02/nh-32.html زده شده چطور می شه خاصیت casecade رو مطابق نیاز تغییر داد.

مثلا وقتی یکی از RelatedCustomers ها رو clear می کنم بدلیل خاصیت "cascade="all,delete-orphan علاوه بر ارجاعات خود Customer هم در هنگام ذخیره سازی پاک میشه.

ممنون میشم راهنماییم کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۱/۱۰ ۰۹:۴۳:۱۳

سلام، این مورد در دو کلاس ManyToManyConventions و NamingConventions قابل تنظیم است. پیش فرض آن‌ها بر این مبنا است که شما حداقل کار ممکن رو بخواهید به صورت دستی انجام بدید. اگر علاقمند بودید این‌ها را تغییر بدید مثلا در کلاس ManyToManyConventions، Cascade.DeleteOrphans را جستجو کنید. یک enum ساده است. این رو تغییر بدید به حالتی که مدنظر است. شبیه به همین مورد در کلاس NamingConventions در دو متد ReferenceConvention و OneToManyConvention وجود دارد.

نویسنده: A. Karimi
تاریخ: ۱۳۹۱/۰۱/۱۰ ۱۲:۲۳:۱۶

تبریک بخاطر شروع مقالات MVC.
فقط در مورد مجوز MS-PL، این مجوز فقط خواندنی نیست و متن باز است! در خصوص MVC قسمت‌هایی از آن متن باز نبود مثل موتور Razor که آنها هم متن باز شدند.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۱/۱۰ ۱۲:۴۵:۲۸

من نگفتم MS-PL متن باز نیست. این مجوز پذیرفته شده OSI است (^) .
متن باز بودن هم به معنای آزادی مطلق نیست. مثلا مجوز GPL به شما می‌گه که من سورس کارت رو هم می‌خوام اگر از کتابخانه من استفاده کردی یا اینکه باید با من به نحوی کنار بیای و هماهنگ کنی.
یا مجوز MIT می‌گه من نمی‌خوام و مهم نیست؛ یک تشکر برای من کافی است.
مجوز MS-PL بیشتر معنای (Shared source) رو از طرف مایکروسافت داره. به عبارتی مجاز هستید در فرم باینری در هر نوع پروژه‌ای از آن استفاده کنید. اما در حالت سورس، فقط جهت مرور و یا یافتن مشکلات امنیتی یا بررسی‌های امنیتی در اختیار عموم قرار گرفته است (مثلا بعضی از دولت‌ها به این مساله حساس هستند و سورس رو جهت بررسی امنیتی نیاز دارند). اما با این حال:
- مجاز هستید سورس رو تغییر بدید و حتی بفروشید اما باز هم تحت مجوز MS-PL
- اگر کاری جدیدی بر مبنای این سورس (نه باینری آن که عنوان شد) تهیه شود، هم باید سورس را ارائه دهید و هم باز هم کل کار باید تحت مجوز MS-PL باشد.

رفتار مایکروسافت با این مجوز خاص خودش، فقط خواندنی است. یعنی پچی رو قبول نمی‌کنه.
به همین جهت این رفتار رو اخیرا اصلاح کردن و به مجوز آپاچی نقل مکان کردند و از حالت shared code فقط خواندنی بیشتر جهت بررسی‌های امنیتی و مرور کد، به یک حالت پویاتر تبدیل شده.

نویسنده: محمود
تاریخ: ۱۳۹۱/۰۳/۲۸ ۱۲:۱۳

سلام.
اگه بخوایم چند تا مدل رو به وسیله یک action به یک view برگردونیم و از روش strongly typed هم استفاده کنیم باید چه کار کنیم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۳/۲۸ ۱۲:۲۰

سلام،
برای اینکار اصطلاحا از ViewModel استفاده می‌کنند. یک کلاس درست کنید مثلا به نام ReportViewModel که در پوشه Models قرار می‌گیره. بعد خواص عمومی این کلاس، شامل مدل‌هایی خواهد بود که مد نظر شما است. به این ترتیب می‌شود چندین و چند مدل رو به View انتقال داد.

نویسنده: TazeKar
تاریخ: ۱۳۹۱/۰۴/۰۱ ۱۱:۵۵

با سلام و تشکر فراوان

شما در مورد TempData فرمودید "اطلاعات آن در سایر کنترلرها و Viewها نیز در دسترس خواهد بود " .پس علت استفاده از کد زیر برای چیست؟

```
ViewBag.Erorr=erorr;
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۰۱ ۱۲:۱۵

در متن توضیح دادم:
البته TempData یک سری تفاوت هم با سشن معمولی ASP.NET دارد:
- **بلافاصله** پس از خوانده شدن، حذف خواهد شد.
و ...

نویسنده: امیر هاشم زاده
تاریخ: ۱۳۹۱/۰۵/۱۴ ۳:۳۴

آیا در روش دوم(تعریف نوع صریح مدل)، مقید هستیم @model با m کوچک تعریف کنیم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۱۴ ۸:۵۰

بله. مدل با M بزرگ به وهله‌ای از شیء مدل اشاره خواهد کرد. مدل با m کوچک جهت تعریف نوع آن شیء بکار می‌رود.

نویسنده: احمد احمدی
تاریخ: ۱۳۹۱/۰۶/۲۴ ۳:۲۷

با سلام و تشکر بخاطر این مقاله‌ی عالی
لطفا کمی در مورد تفاوتهای Html.Raw و MvcHtmlString.Create توضیح بدید .

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۶/۲۴ ۸:۱۹

تفاوت مهمی ندارند. MvcHtmlString.Create وهله‌ای از MvcHtmlString رو ای جاد می‌کنه؛ در حالیکه Html.Raw وهله‌ای از

HtmlString را ایجاد خواهد کرد. اما باید توجه داشت که MvcHtmlString از HtmlString مشتق میشه و بنابراین این دو همانند هم کار خواهند کرد.

نویسنده: امین
تاریخ: ۱۳۹۱/۰۷/۲۲ ۱۸:۱۸

فرق

```
public string FirstName { get; set; }
```

با

```
private string firstName;

       public string FirstName
        {
            get { return firstName; }
            set { firstName = value; }
        }
```

چیه
آخه تو دانشگاه به ما یاد دادند شبیه دومی بنویسیم اما اولی راحتتره حالا کلا با هم چه فرقی دارند؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۲۲ ۲۲:۲۴

فرقی ندارند. مورد اول (Auto-Implemented Properties) از سی شارپ سه به بعد میسر شده و کامپایلر در پشت صحنه حالت دوم را تولید می‌کند.

نویسنده: امیر
تاریخ: ۱۳۹۱/۰۷/۲۷ ۱۵:۳۲

سلام
اگر بخواهیم از داخل کنترلرمون مقادیریمون را در قالب alert تو view نمایش بدیم باید چکار کرد؟
من خودم به این شکل عمل کردم

```
<script language="javascript" type="text/javascript">

        alert(' @ViewData["ErrorMessage"]');
    </script>
```

میخواستم ببینم راه حل بهتری هستش؟
اصلا بهتر نیست از partialview استفاده کنم که یه view برای alert کلا داشته باشم؟
مرسی

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۲۷ ۱۶:۳۲

برای تعریف JavaScript می‌تونید از روش معرفی Section و RenderSection هم استفاده کنید (^).

نویسنده: احمد
تاریخ: ۱۳۹۱/۱۰/۲۱ ۲۰:۵۲

با تشکر از شما

در بالا شما به این کد ViewData["DateTime"] = "
" + DateTime.Now; اشاره کردید و راهکار نمایش آنهم را نشان دادید .(@Html.Raw(myString

حال اگر در view در بخش foreach به جای این @Html.DisplayFor(modelItem => item.FirstName از Html.Raw نمی‌توان استفاده کرد. چه کاری می‌توان انجام داد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۰/۲۱ ۲۱:۲۰

در یک حلقه به این صورت عمل کنید:

```
@foreach (var item in Model)
{
  <div>
     @Html.Raw(@item.Description)
  </div>
}
```

نویسنده: تازه کار
تاریخ: ۱۳۹۱/۱۲/۱۲ ۱۰:۱۶

من میخواستم بدونم این تیکه از کد رو باید نوشت؟

```
public ViewDataDictionary ViewData { get; set; }
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۲/۱۲ ۱۰:۴۰

خیر. کلاس پایه‌ای در ASP.NET MVC وجود دارد به نام (^)WebViewPage. این کلاس حاوی تعاریف اولیه ViewBag، TempData، ViewData و ... Model است؛ جهت استفاده در Viewها. در کنترلرها هم این تعریف در کلاس پایه ControllerBase قرار دارد.

نویسنده: محمد آزاد
تاریخ: ۱۳۹۲/۰۴/۱۰ ۱۵:۳۱

در مورد فعال کردن MvcBuildViews من وقتی این کارو انجام میدم با خطای زیر روبه رو می‌شم.

Error 1 The type or namespace name 'Optimization' does not exist in the namespace 'System.Web' (are you missing an assembly reference?) c:\Windows\Microsoft.NET\Framework\v4.0.30319\Temporary ASP.NET Files\temp\93647db0\578ae027\App_Web_krwrdmvd.0.cs 26

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۱۰ ۱۶:۵۵

در پروژه MVC خود، خط فرمان پاورشل نیوگت را باز کنید و ابتدا دستور

```
PM> Install-Package Microsoft.AspNet.Web.Optimization
```

و بعد دستور زیر را صادر کنید (اگر باز هم کار نکرد، CopyLocal=True را برای اسمبلی System.Web.Optimization در خواص آن انتخاب کنید):

```
PM> Update-Package
```

نویسنده: طاهری
تاریخ: ۱۳۹۲/۰۹/۰۶ ۱۰:۲۵

سلام
Employees رو ایجاد کردم.متد List روهم در کنترلر گذاشتم.ویو رو همونطور که گفتین add کردم(دقیقا کاری که شما انجام دادین)اما وقتی از ویو List اجرا میگیرم تو مرورگر پیغام میده که *The resource cannot be found. من هیچ تغییری تو فایل ویو ایجاد شده ندادم.چیکار کنم*

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۰۶ ۱۰:۵۶

اشاره‌گر رو داخل اکشن متد کنترلر قرار بدید و بعد پروژه رو در دیباگر VS.NET اجرا کنید. مراحل کاری ASP.NET MVC از View شروع نمی‌شود. مراجعه کنید به «تفاوت مهم پردازشی ASP.NET MVC با ASP.NET Web forms» در قسمت دوم این سری.

نویسنده: آروین
تاریخ: ۱۳۹۲/۰۹/۲۱ ۱۲:۵۳

با عرض سلام و خسته نباشید
من یک View دارم شامل چندین PartialView که مقادیر فیلدهای موجود در آنها را از دیتابیس می‌گیرم آیا بهتر است یک ViewModel برای View کلی بسازم و یکبار از دیتابیس تمام مقادیر را بگیرم و از طریق Model آنها را به سایر PartialViewها بفرستم یا اینکه به ازای هر PartialView یک ViewModel بسازم و در Load هرکدام از اینها مقادیر را از دیتابیس بگیرم. در روش اول یک ViewModel دارم که فیلدهای آن زیاد هستند و در PartialViewها از یک سری از این فیلدها استفاده می‌کنم ولی یکبار از دیتابیس می‌گیریم ولی روش دوم فیلدهای هر ViewModel مخصوص همان PartialView هستند ولی ارتباط با دیتابیسم بیشتر میشه. ممنون می‌شم یه توضیحی در مورد سرعت و کارایی هرکدام از روش‌ها بدین و آیا در یک پروژه وب سایت روش اول به صرفه‌تر است یا دوم؟
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۱ ۱۳:۱

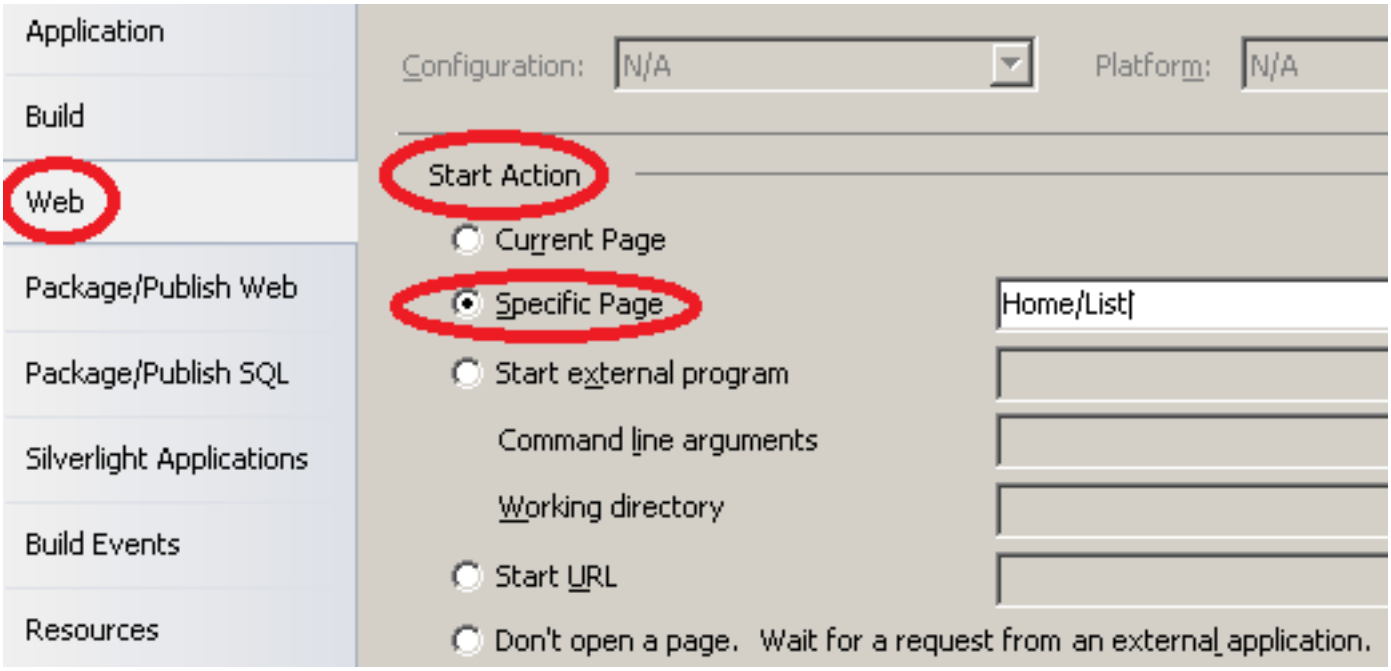
روش اول. اگر اندکی بارگذاری آن طولانی است از روش زیر استفاده کنید:
« به روز رسانی غیرهمزمان قسمتی از صفحه به کمک jQuery در ASP.NET MVC »
+ مباحث کش کردن اطلاعات را هم مدنظر داشته باشید.

نویسنده: nasrin
تاریخ: ۱۳۹۲/۱۱/۰۶ ۲۰:۱۸

سلام. ببخشید بعداز اجرای view index, میخواهم view list رو اجرا کنم. view list اجرا نمیشه. با هر بار اجرا همون view index اجرا میشه. باتشکر.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۰۶ ۲۱:۳۲

- مسیر http://localhost/Home/List را باید دستی در مرورگر وارد کنید.
- همچنین امکان انتخاب action پیش فرض حین دیباگ نیز در VS.NET وجود دارد (در برگه‌ی خواص پروژه):



البته این تنظیم فقط در حالت دیباگ در VS.NET عمل می‌کند. تنظیم اصلی صفحه‌ی پیش فرض، همان default route برنامه است (مسیر پیش فرض صفحه‌ی ابتدایی در سرور).

نویسنده: امیر خلیلی
تاریخ: ۱۳۹۲/۱۱/۱۵ ۲۳:۳۹

آیا برای انتقال اطلاعات در بین دو اکشن روش بهتری نسبت به tempData هم وجود داره ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۶ ۰:۱۰

Redirectها نهایتا به یک درخواست Get ختم می‌شوند. بنابراین هر نوع اطلاعاتی که از طریق کوئری استرینگ‌ها در دسترس قرار گیرند، بلافاصله قابل استفاده خواهند بود.
مثلا برای حالت anonymously typed در پارامتر route values آن (این پارامتر مدل نیست؛ مقادیر route هستند):

```
return RedirectToAction("SomeMethod", new { id = 1 });
```

با این اکشن متد:

```
public ActionResult SomeMethod(int? id)
{
    ...
}
```

در این حالت خاص، برای ارسال یک مدل کامل بهتر است از TempData استفاده کنید.

نویسنده: میثم فغفوری
تاریخ: ۱۳۹۲/۱۱/۱۶ ۱۰:۴۸

یک مشکل کوچک در Razor برام لاینحل مونده ممنون میشم راهنماییم کنین اونم عدم وجود Desing mode برای مثلا مستر پیج یا صفحات دیگرمون هست یعنی الان کاملا ذهنی صفحات رو طراحی میکنم و از اونجایی که موتور Razor خیلی ساده‌تر از Aspx هست نمیخوام بذارمش کنار. برای این مساله یه راه ساده‌تری وجود نداره که دیزاین طراحی پیجامون رو هم داشته باشیم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۶ ۱۱:۳۴

- Page Inspector برای اینکار طراحی شده.
- فایل‌های cshtml در تمام ادیتور بصری موجود قابل گشودن و ویرایش هستند.
- اما ... اکثر ادیتورهای بصری قادر نیستند با بسیاری از فریم ورک‌های جدید CSS کار کنند؛ مانند بوت استرپ. طراحی و کار با آن‌ها عموما بدون ادیتورهای بصری و به کمک استفاده از مرورگرها انجام می‌شود. Razor هم به همین نحو است. صفحه را تغییر داده و Save کنید. بعد مرورگر را Refresh کنید (نیازی به کامپایل مجدد نیست).
- صفحات ASP.NET، یک سری صفحات پویا هستند. نیاز به برقراری اتصالات خاصی بین بانک اطلاعاتی، کوئری استرینگ‌ها، مقادیر Post شده به صفحه و غیره، برای نمایش اطلاعات خاصی است. طراح‌های بصری در یک چنین مواردی کارآمد نیستند و باید چرخه‌ی کامل طول عمر صفحه در مرورگر طی شود.
- خیلی از مسایل توسط طراح‌های بصری قابل پیاده سازی نیستند؛ برای مثال نوشتن یک if و else برای نمایش قسمتی از صفحه به کاربران اعتبارسنجی شده یا نمایش داده‌ها در یک حلقه.

نویسنده: ع طاهری
تاریخ: ۱۳۹۲/۱۲/۰۱ ۱۰:۲۸

سلام؛ برای دادن مقدار پیش فرض به یک property در یک مدل چیکار کنم؟ البته میخواهم این مقدار در ویو هم نمایش داده شود و کاربر اگه خواست تغییرش بدهد. ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۰۱ ۱۰:۴۵

در سی‌شارپ فعلی برای مقدار دهی به خواص خودکار و تعیین مقدار پیش فرض، از مقدار دهی آن‌ها در سازنده کلاس استفاده می‌کنند (یک constructor تعریف کنید و بعد خواص را مقدار دهی کنید). این مورد در سی‌شارپ 6 قرار است ساده‌تر شده و در همان محل تعریف خاصیت، قابل مقدار دهی شود.

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۲/۱۶ ۱۶:۴۲

با عرض سلام
یه مشکل که من موقع ایجاد ویو Scaffolding باهاش روبرو شدم پیام خطای زیر هست:

```
exception has been thrown by the target of an invocation
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۲/۱۶ ۱۶:۵۴

^ و ^

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۲/۱۶ ۱۷:۰

مشکل از وجود خطا در فایل وب کانفیگ بود. محل تگ location اشتباه تعریف شده بود.

نویسنده: عثمان رحیمی
تاریخ: ۱۳۹۴/۰۵/۰۵ ۲۲:۴۱

عنوان شد که Razor به صورت پیش فرض ، پشت پرده اطلاعات رو Encode میکنه ، یعنی Html.Encode رو فراخوانی می‌کند ، اگر همان رشته ای را که در ViewDate قرار داده شده است به متد Html.Encode بدهیم خروجی فرق خواهد کرد(تبدیل کاراکترهای

غیر مجاز)، :

به طور مثال :

```
@Html.Encode("< br/  >hellow word")
```

که خروجی زیر حاصل می‌شود :

```
&lt;br/&gt;hellow word
```

دلیل دو نوع خروجی چیست ؟ در صورتی که اگر مقدار ViewData را نشان بدهیم به صورت زیر خواهد بود :

```
ViewData["mydate"] = "< br/ > hello word";
//output in View :  "< br/ > hello word"
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۵/۰۵ ۲۳:۸

از Html.Encode نباید در Viewهای Razor به صورت مستقیم استفاده کرد. خروجی آن string ساده‌است و IHtmlString نیست. به همین جهت توسط @ یکبار دیگر encode خواهد شد. یعنی نتیجه‌ی آن double-encoded است.