عنوان: ASP.NET MVC #12
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۱/۲۱ ۰۰:۲۰:۰۰
آدرس: www.dotnettips.info
گروه‌ها: MVC

تولید خودکار فرم‌های ورود و نمایش اطلاعات در ASP.NET MVC بر اساس اطلاعات مدل‌ها

در الگوی MVC، قسمت M یا مدل آن یک سری ویژگی‌های خاص خودش را دارد:
شما را وادار نمی‌کند که مدل را به نحو خاصی طراحی کنید. شما را مجبور نمی‌کند که کلاس‌های مدل را برای نمونه همانند کلاس‌های کنترلرها، از کلاس خاصی به ارث ببرید. یا حتی در مورد نحوه‌ی دسترسی به داده‌ها نیز، نظری ندارد. به عبارتی برنامه نویس است که می‌تواند بر اساس امکانات مهیای در کل اکوسیستم دات نت، در این مورد آزادانه تصمیم گیری کند.
بر همین اساس ASP.NET MVC یک سری قرارداد را برای سهولت اعتبار سنجی یا تولید بهتر رابط کاربری بر اساس اطلاعات مدل‌ها، فراهم آورده است. این قراردادها هم چیزی نیستند جز یک سری metadata که نحوه‌ی دربرگیری اطلاعات را در مدل‌ها توضیح می‌دهند. برای دسترسی به آن‌ها پروژه جاری باید ارجاعی را به اسمبلی‌های System.ComponentModel.DataAnnotations.dll و System.Web.Mvc.dll داشته باشد (که VS.NET به صورت خودکار در ابتدای ایجاد پروژه اینکار را انجام می‌دهد).

یک مثال کاربردی

یک پروژه جدید خالی ASP.NET MVC را آغاز کنید. در پوشه مدل‌های آن، مدل اولیه‌ای را با محتوای زیر ایجاد نمائید:

```
using System;

namespace MvcApplication8.Models
{
    public class Employee
    {
        public int Id { set; get; }
        public string Name { set; get; }
        public decimal Salary { set; get; }
        public string Address { set; get; }
        public bool IsMale { set; get; }
        public DateTime AddDate { set; get; }
    }
}
```

سپس یک کنترلر جدید را هم به نام EmployeeController با محتوای زیر به پروژه اضافه نمائید:

```
using System;
using System.Web.Mvc;
using MvcApplication8.Models;

namespace MvcApplication8.Controllers
{
    public class EmployeeController : Controller
    {
        public ActionResult Create()
        {
            var employee = new Employee { AddDate = DateTime.Now };
            return View(employee);
        }
    }
}
```

بر روی متد Create کلیک راست کرده و یک View ساده را برای آن ایجاد نمائید. سپس محتوای این View را به صورت زیر تغییر دهید:

```
@model MvcApplication8.Models.Employee
@{
    ViewBag.Title = "Create";
}

<h2>Create An Employee</h2>

@using (Html.BeginForm(actionName: "Create", controllerName: "Employee"))
{
    @Html.EditorForModel()
    <input type="submit" value="Save" />
}
```

اکنون اگر پروژه را اجرا کرده و مسیر http://localhost/employee/create را وارد نمائید، یک صفحه ورود اطلاعات تولید شده به صورت خودکار را مشاهده خواهید کرد. متد Html.EditorForModel بر اساس اطلاعات خواص عمومی مدل، یک فرم خودکار را تشکیل می‌دهد.
البته فرم تولیدی به این شکل شاید آنچنان مطلوب نباشد، از این جهت که برای مثال Id را هم لحاظ کرده، در صورتیکه قرار است این Id توسط بانک اطلاعاتی انتساب داده شود و نیازی نیست تا کاربر آن‌را وارد نماید. یا مثلا برچسب AddDate نباید به این شکل صرفا بر اساس نام خاصیت متناظر با آن تولید شود و مواردی از این دست. به عبارتی نیاز به سفارشی سازی کار این فرم ساز توکار ASP.NET MVC وجود دارد که ادامه بحث جاری را تشکیل خواهد داد.

سفارشی سازی فرم ساز توکار ASP.NET MVC با کمک Metadata خواص

برای اینکه بتوان نحوه نمایش فرم خودکار تولید شده را سفارشی کرد، می‌توان از یک سری attribute و data annotations توکار دات نت و ASP.NET MVC استفاده کرد و نهایتا این metadata توسط فریم ورک، مورد استفاده قرار خواهند گرفت. برای مثال:

```
using System;
using System.ComponentModel;
using System.ComponentModel.DataAnnotations;
using System.Web.Mvc;

namespace MvcApplication8.Models
{
    public class Employee
    {
        //[ScaffoldColumn(false)]

        [HiddenInput(DisplayValue=false)]
        public int Id { set; get; }

        public string Name { set; get; }

        [DisplayName("Annual Salary ($)")]
        public decimal Salary { set; get; }

        public string Address { set; get; }

        [DisplayName("Is Male?")]
        public bool IsMale { set; get; }

        [DisplayName("Start Date")]
        [DataType(DataType.Date)]
        public DateTime AddDate { set; get; }
    }
}
```

در اینجا به کمک ویژگی HiddenInput از نمایش عمومی خاصیت Id جلوگیری خواهیم کرد یا توسط ویژگی DisplayName، برچسب دلخواه خود را به عناصر فرم تشکیل شده، انتساب خواهیم داد. اگر نیاز باشد تا خاصیتی کلا از رابط کاربری حذف شود می‌توان از ویژگی ScaffoldColumn با مقدار false استفاده کرد. یا توسط DataType، مشخص کرده‌ایم که نوع ورودی فقط قرار است Date باشد و نیازی به قسمت Time آن نداریم.
DataType شامل نوع‌های از پیش تعریف شده دیگری نیز هست. برای مثال اگر نیاز به نمایش TextArea بود از مقدار MultilineText، استفاده کنید:

```
[DataType(DataType.MultilineText)]
```

یا برای نمایش PasswordBox از مقدار Password می‌توان کمک گرفت. اگر نیاز دارید تا آدرس ایمیلی به شکل یک لینک mailto نمایش داده شود از مقدار EmailAddress استفاده کنید. به کمک مقدار Url، متن خروجی به صورت خودکار تبدیل به یک آدرس قابل کلیک خواهد شد.
اکنون اگر پروژه را مجددا کامپایل کنیم و به آدرس ایجاد یک کارمند جدید مراجعه نمائیم، با رابط کاربری بهتری مواجه خواهیم شد.

سفارشی سازی ظاهر فرم ساز توکار ASP.NET MVC

در ادامه اگر بخواهیم ظاهر این فرم را اندکی سفارشی‌تر کنیم، بهتر است به سورس صفحه تولیدی در مرورگر مراجعه کنیم. در اینجا یک سری عناصر HTML محصور شده با div را خواهیم یافت. هر کدام از این‌ها هم با classهای css خاص خود تعریف شده‌اند. بنابراین اگر علاقمند باشیم که رنگ و قلم و غیره این موارد تغییر دهیم، تنها کافی است فایل css برنامه را ویرایش کنیم و نیازی به دستکاری مستقیم کدهای برنامه نیست.

انتساب قالب‌های سفارشی به خواص یک شیء

تا اینجا در مورد نحوه سفارشی سازی رنگ، قلم، برچسب و نوع داده‌های هر کدام از عناصر نهایی نمایش داده شده، توضیحاتی را ملاحظه نمودید.
در فرم تولیدی نهایی، خاصیت bool تعریف شده به صورت خودکار به یک checkbox تبدیل شده است. چقدر خوب می‌شد اگر امکان تبدیل آن مثلا به RadioButton انتخاب مرد یا زن بودن کارمند ثبت شده در سیستم وجود داشت. برای اصلاح یا تغییر این مورد، باز هم می‌توان از متادیتای خواص، جهت تعریف قالبی خاص برای هر کدام از خواص مدل استفاده کرد.
به پوشه Views/Shared مراجعه کرده و یک پوشه جدید به نام EditorTemplates را ایجاد نمائید. بر روی این پوشه کلیک راست کرده و گزینه Add view را انتخاب کنید. در صفحه باز شده، گزینه «Create as a partial view» را انتخاب نمائید و نام آن‌را هم مثلا GenderOptions وارد کنید. همچنین گزینه «Create a strongly typed view» را نیز انتخاب کنید. مقدار Model class را مساوی bool وارد نمائید. فعلا یک hello داخل این صفحه جدید وارد کرده و سپس خاصیت IsMale را به نحو زیر تغییر دهید:

```
[DisplayName("Gender")]
[UIHint("GenderOptions")]
public bool IsMale { set; get; }
```

توسط ویژگی UIHint، می‌توان یک خاصیت را به یک partial view متصل کرد. در اینجا خاصیت IsMale به partial view ایی به نام GenderOptions متصل شده است. اکنون اگر برنامه را کامپایل و اجرا کرده و آدرس ایجاد یک کارمند جدید را ملاحظه کنید، بجای Checkbox باید یک hello نمایش داده شود.
محتویات این Partial view هم نهایتا به شکل زیر خواهند بود:

```
@model bool
<p>@Html.RadioButton("", false, !Model) Female</p>
<p>@Html.RadioButton("", true, Model) Male</p>
```

در اینجا Model که از نوع bool تعریف شده، به خاصیت IsMale اشاره خواهد کرد. دو RadioButton هم برای انتخاب بین حالت زن و مرد تعریف شده‌اند.

یا یک مثال جالب دیگر در این زمینه می‌تواند تبدیل enum به یک Dropdownlist باشد. در این حالت partial view ما شکل زیر را خواهد یافت:

```
@model Enum
@Html.DropDownListFor(m => m, Enum.GetValues(Model.GetType())
         .Cast<Enum>()
         .Select(m => {
                   string enumVal = Enum.GetName(Model.GetType(), m);
                   return new SelectListItem() {
                             Selected = (Model.ToString() == enumVal),
                             Text = enumVal,
                             Value = enumVal
                  };
}))
```

و برای استفاده از آن، از ویژگی زیر می‌توان کمک گرفت (مزین کردن خاصیتی از نوع یک enum دلخواه، جهت تبدیل خودکار آن به یک دراپ داون لیست):

```
[UIHint("Enum")]
```

سایر متدهای کمکی تولید و نمایش خودکار اطلاعات از روی اطلاعات مدل‌های برنامه

متدهای دیگری نیز در رده‌ی Templated helpers قرار می‌گیرند. اگر از متد Html.EditorFor استفاده کنیم، از تمام این اطلاعات متادیتای تعریف شده نیز استفاده خواهد کرد. همانطور که در قسمت قبل (قسمت 11) نیز توضیح داده شد، صفحه استاندارد Add view در VS.NET به همراه یک سری قالب تولید فرم‌های Create و Edit هم هست که دقیقا کد نهایی تولیدی را بر اساس همین متد تولید می‌کند.
استفاده از Html.EditorFor انعطاف پذیری بیشتری را به همراه دارد. برای مثال اگر یک طراح وب، طرح ویژه‌ای را در مورد ظاهر فرم‌های سایت به شما ارائه دهد، بهتر است از این روش استفاده کنید. اما خروجی نهایی Html.EditorForModel به کمک تعدادی متادیتا و اندکی دستکاری CSS، از دیدگاه یک برنامه نویس بی نقص است!
به علاوه، متد Html.DisplayForModel نیز مهیا است. بجای اینکه کار تولید رابط کاربری اطلاعات نمایش جزئیات یک شیء را انجام دهید، اجازه دهید تا متد Html.DisplayForModel اینکار را انجام دهد. سفارشی سازی آن نیز همانند قبل است و بر اساس متادیتای خواص انجام می‌شود. در این حالت، مسیر پیش فرض جستجوی قالب‌های UIHint آن، Views/Shared/DisplayTemplates می‌باشد. همچنین Html.DisplayFor نیز جهت کار با یک خاصیت مدل تدارک دیده شده است. البته باید درنظر داشت که استفاده از پوشه Views/Shared اجباری نیست. برای مثال اگر از پوشه Views/Home/DisplayTemplates استفاده کنیم، قالب‌های سفارشی تهیه شده تنها جهت Viewهای کنترلر home قابل استفاده خواهند بود.
یکی دیگر از ویژگی‌هایی که جهت سفارشی سازی نحوه نمایش خودکار اطلاعات می‌تواند مورد استفاده قرار گیرد، DisplayFormat است. برای مثال اگر مقدار خاصیت در حال نمایش نال بود، می‌توان مقدار دیگری را نمایش داد:

```
[DisplayFormat(NullDisplayText = "-")]
```

یا اگر علاقمند بودیم که فرمت اطلاعات در حال نمایش را تغییر دهیم، به نحو زیر می‌توان عمل کرد:

```
[DisplayFormat(DataFormatString  = "{0:n}")]
```

مقدار DataFormatString در پشت صحنه در متد string.Format مورد استفاده قرار می‌گیرد.
و اگر بخواهیم که این ویژگی در حالت تولید فرم ویرایش نیز درنظر گرفته شود، می‌توان خاصیت ApplyFormatInEditMode را نیز مقدار دهی کرد:

```
[DisplayFormat(DataFormatString  = "{0:n}", ApplyFormatInEditMode = true)]
```

بازنویسی قالب‌های پیش فرض تولید فرم یا نمایش اطلاعات خودکار ASP.NET MVC

یکی دیگر از قراردادهای بکارگرفته شده در حین استفاده از قالب‌های سفارشی، استفاده از نام اشیاء می‌باشد. مثلا در پوشه Views/Shared/DisplayTemplates، اگر یک Partial view به نام String.cshtml وجود داشته باشد، از این پس نحوه رندر کلیه خواص رشته‌ای تمام مدل‌ها، بر اساس محتوای فایل String.cshtml مشخص می‌شود؛ به همین ترتیب در مورد datetime و سایر انواع مهیا.
برای مثال اگر خواستید تمام تاریخ‌های میلادی دریافتی از بانک اطلاعاتی را شمسی نمایش دهید، فقط کافی است یک فایل datetime.cshtml سفارشی را تولید کنید که Model آن تاریخ میلادی دریافتی است و نهایتا کار این Partial view، رندر تاریخ تبدیل شده به همراه تگ‌های سفارشی مورد نظر می‌باشد. در این حالت نیازی به ذکر ویژگی UIHint نیز نخواهد بود و همه چیز خودکار است.
به همین ترتیب اگر نام مدل ما Employee باشد و فایل Partial view ایی به نام Employee.cshtml در پوشه Views/Shared/DisplayTemplates قرار گیرد، متد Html.DisplayForModel به صورت پیش فرض از محتوای این فایل جهت رندر اطلاعات نمایش جزئیات شیء Employee استفاده خواهد کرد.
داخل Partial viewهای سفارشی تعریف شده به کمک خاصیت ViewData.TemplateInfo.FormattedModelValue مقدار نهایی فرمت شده قابل استفاده را فراهم می‌کند. این مورد هم از این جهت حائز اهمیت است که نیازی نباشد تا ویژگی DisplayFormat را به صورت دستی پردازش کنیم. همچنین اطلاعات ViewData.ModelMetadata نیز دراینجا قابل دسترسی هستند.

سؤال: Partial View چیست؟

همانطور که از نام Partial view برمی‌آید، هدف آن رندر کردن قسمتی از صفحه است به همراه استفاده مجدد از کدهای تولید رابط کاربری در چندین و چند View؛ چیزی شبیه به User controls در ASP.NET Web forms البته با این تفاوت که Page life cycle و Code behind و سایر موارد مشابه آن در اینجا حذف شده‌اند. همچنین از Partial viewها برای به روز رسانی قسمتی از صفحه حین فراخوانی‌های Ajaxایی نیز استفاده می‌شود. مهم‌ترین کاربرد Partial views علاوه بر استفاده مجدد از کدها، خلوت کردن Viewهای شلوغ است جهت ساده‌تر سازی نگهداری آن‌ها در طول زمان (یک نوع Refactoring فایل‌های View محسوب می‌شوند).
پسوند این فایل‌ها نیز بسته به موتور View مورد استفاده تعیین می‌شود. برای مثال حین استفاده از Razor، پسوند Partial views همان cshtml یا vbhtml می‌باشد. یا اگر از web forms view engine استفاده شود، پسوند آن‌ها ascx است (همانند User

controls در وب فرم‌ها).
البته چون در حالت استفاده از موتور Razor، پسوند View و Partial viewها یکی است، مرسوم شده است که نام Partial viewها را با یک underline شروع کنیم تا بتوان بین این دو تمایز قائل شد.
اگر این فایل‌ها را در پوشه Views/Shared تعریف کنیم، در تمام Viewها قابل استفاده خواهند بود. اما اگر مثلا در پوشه Views/Home آن‌هارا قرار دهیم، تنها در Viewهای متعلق به کنترلر Home، قابل بکارگیری می‌باشند.
Partial views را نیز می‌توان strongly typed تعریف کرد و به این ترتیب با مشخص سازی دقیق نوع model آن، علاوه بر بهره‌مندی از Intellisense خودکار، رندر آن‌را نیز تحت کنترل کامپایلر قرار داد.
مقدار Model در یک View بر اساس اطلاعات مدلی که به آن ارسال شده است تعیین می‌گردد. اما در یک Partial view که جزئی از یک View را نهایتا تشکیل خواهد داد، بر اساس مقدار ارسالی از طریق View معین می‌گردد.

یک مثال

در ادامه قصد داریم کد حلقه نمایش لیستی از عناصر تولید شده توسط VS.NET را به یک Partial view منتقل و Refactor کنیم.
ابتدا یک منبع داده فرضی زیر را در نظر بگیرید:

```
using System;
using System.Collections.Generic;

namespace MvcApplication8.Models
{
    public class Employees
    {
        public IList<Employee> CreateEmployees()
        {
            return new[]
            {
                new Employee { Id = 1, AddDate = DateTime.Now.AddYears(-3), Name = "Emp-01", Salary =
3000},
                new Employee { Id = 2, AddDate = DateTime.Now.AddYears(-2), Name = "Emp-02", Salary =
2000},
                new Employee { Id = 3, AddDate = DateTime.Now.AddYears(-1), Name = "Emp-03", Salary =
1000}
            };
        }
    }
}
```

سپس از آن در یک کنترلر برای بازگشت لیستی از کارکنان استفاده خواهیم کرد:

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

View متناظر با این متد را هم با کلیک راست بر روی متد، انتخاب گزینه Add view و سپس ایجاد یک strongly typed view از نوع کلاس Employee، ایجاد خواهیم کرد.
در ادامه قصد داریم بدنه حلقه زیر را refactor کنیم و آن‌را به یک Parial view منتقل نمائیم تا View ما اندکی خلوت‌تر و مفهوم‌تر شود:

```
@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.Name)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Salary)
```

```
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Address)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.IsMale)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.AddDate)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.Id }) |
            @Html.ActionLink("Details", "Details", new { id=item.Id }) |
            @Html.ActionLink("Delete", "Delete", new { id=item.Id })
        </td>
    </tr>
}
```

سپس بر روی پوشه Views/Employee کلیک راست کرده و گزینه Add|View را انتخاب کنید. در اینجا نام _EmployeeItem را وارد کرده و همچنین گزینه Create as a partial view و create a strongly typed view را نیز انتخاب کنید. نوع مدل هم Employee خواهد بود. به این ترتیب فایل زیر تشکیل خواهد شد:

```
\Views\Employee\_EmployeeItem.cshtml
```

ابتدای نام فایل‌را با underline شروع کرده‌ایم تا بتوان بین Viewها و Partial views تفاوت قائل شد. همچنین این Partial view چون داخل پوشه Employee تعریف شده، فقط در Viewهای کنترلر Employee در دسترس خواهد بود.
در ادامه کل بدنه حلقه فوق را cut کرده و در این فایل جدید paste نمائید. مرحله اول refactoring یک view به همین نحو آغاز می‌شود. البته در این حالت قادر به استفاده از Partial view نخواهیم بود چون اطلاعاتی که به این فایل ارسال می‌گردد و مدلی که در دسترس آن است از نوع Employee است و نه لیستی از کارمندان. به همین جهت باید item را با Model جایگزین کرد:

```
@model MvcApplication8.Models.Employee

    <tr>
        <td>
            @Html.DisplayFor(x => x.Name)
        </td>
        <td>
            @Html.DisplayFor(x => x.Salary)
        </td>
        <td>
            @Html.DisplayFor(x => x.Address)
        </td>
        <td>
            @Html.DisplayFor(x => x.IsMale)
        </td>
        <td>
            @Html.DisplayFor(x => x.AddDate)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id = Model.Id }) |
            @Html.ActionLink("Details", "Details", new { id = Model.Id }) |
            @Html.ActionLink("Delete", "Delete", new { id = Model.Id })
        </td>
    </tr>
```

سپس برای استفاده از این Partial view در صفحه نمایش لیست کارمندان خواهیم داشت:

```
@foreach (var item in Model) {
    @Html.Partial("_EmployeeItem", item)
```

```
}
```

متد Html.Partial، اطلاعات یک Partial view را پردازش و تبدیل به یک رشته کرده و در اختیار Razor قرار می‌دهد تا در صفحه نمایش داده شود. پارامتر اول آن نام Partial view مورد نظر است (نیازی به ذکر پسوند فایل نیست) و پارامتر دوم، اطلاعاتی است که به آن ارسال خواهد شد.
متد دیگری هم وجود دارد به نام Html.RenderPartial. کار این متد نوشتن مستقیم در Response است، برخلاف Html.Partial که فقط یک رشته را بر می‌گرداند.

نمایش اطلاعات از کنترلرهای مختلف در یک صفحه

Html.Partial بر اساس اطلاعات مدل ارسالی از یک کنترلر، کار رندر قسمتی از آن‌را در یک View خاص عهده دار خواهد شد. اما اگر بخواهیم مثلا در یک صفحه یک قسمت را به نمایش آخرین اخبار و یک قسمت را به نمایش آخرین وضعیت آب و هوا اختصاص دهیم، از روش دیگری به نام RenderAction می‌توان کمک گرفت. در اینجا هم دو متد Html.Action و Html.RenderAction وجود دارند. اولی یک رشته را بر می‌گرداند و دومی اطلاعات را مستقیما در Response درج می‌کند.

یک مثال:
کنترلر جدیدی را به نام MenuController به پروژه اضافه کنید:

```
using System.Web.Mvc;

namespace MvcApplication8.Controllers
{
    public class MenuController : Controller
    {
        [ChildActionOnly]
        public ActionResult ShowMenu(string options)
        {
            return PartialView(viewName: "_ShowMenu", model: options);
        }
    }
}
```

سپس بر روی نام متد کلیک راست کرده و گزینه Add view را انتخاب کنید. در اینجا قصد داریم یک partial view که نامش با underline شروع می‌شود را اضافه کنیم. مثلا با محتوای زیر (با توجه به اینکه مدل ارسالی از نوع رشته‌ای است):

```
@model string

<ul>
    <li>
        @Model
    </li>
</ul>
```

حین فراخوانی متد Html.Action، یک متد در یک کنترلر فراخوانی خواهد شد (که شامل ارائه درخواست و طی سیکل کامل پردازشی آن کنترلر نیز خواهد بود). سپس آن متد با بازگشت دادن یک PartialView، اطلاعات پردازش شده یک partial view را به فراخوان بازگشت می‌دهد. اگر نامی ذکر نشود، همان نام متد در نظر گرفته خواهد شد. البته از آنجائیکه در این مثال در ابتدای نام Partial view یک underline قرار دادیم، نیاز خواهد بود تا این نام صریحا ذکر گردد (چون دیگر هم نام متد یا ActionName آن نیست). ویژگی ChildActionOnly سبب می‌شود تا این متد ویژه تنها از طریق فراخوانی Html.Action در دسترس باشد.
برای استفاده از آن هم در Viewایی دیگر خواهیم داشت:

```
@Html.Action(actionName: "ShowMenu", controllerName: "Menu",
                    routeValues: new { options = "some data..." })
```

در اینجا هم پارامتر ارسالی به کمک anonymously typed objects مشخص و مقدار دهی شده است.

سؤال مهم: چه تفاوتی بین RenderPartial و RenderAction وجود دارد؟ به نظر هر دو یک کار را انجام می‌دهند، هر دو مقداری HTML را پس از پرداش به صفحه تزریق می‌کنند.

پاسخ: اگر View والد، دارای کلیه اطلاعات لازم جهت نمایش اطلاعات Partial view است، از RenderPartial استفاده کنید. به این ترتیب برخلاف حالت RenderAction درخواست جدیدی به ASP.NET Pipeline صادر نشده و کارآیی نهایی بهتر خواهد بود. صرفا یک الحاق ساده به صفحه انجام خواهد شد.

اما اگر برای رندر کردن این قسمت از صفحه که قرار است اضافه شود، نیاز به دریافت اطلاعات دیگری خارج از اطلاعات مهیا می‌باشد، از روش RenderAction استفاده کنید. برای مثال اگر در صفحه جاری قرار است لیست پروژه‌ها نمایش داده شود و در کنار صفحه مثلا منوی خاصی باید قرار گیرد، اطلاعات این منو در View جاری فراهم نیست (و همچنین مرتبط به آن هم نیست). بنابراین از روش RenderAction برای حل این مساله می‌توان کمک گرفت.

به صورت خلاصه برای نمایش اطلاعات تکراری در صفحات مختلف سایت در حالتیکه این اطلاعات از قسمت‌های دیگر صفحه ایزوله است (مثلا نمایش چند ویجت مختلف در صفحه)، روش RenderAction ارجحیت دارد.

یک نکته

فراخوانی متدهای RenderAction و RenderPartial در حین کار با Razor باید به شکل فراخوانی یک متد داخل {} باشند:

```
@{ Html.RenderAction("About"); }
And not @Html.RenderAction("About")
```

علت این است که @ به تنهایی به معنای نوشتن در Response است. متد RenderAction هم خروجی ندارد و مستقیما در Response اطلاعات خودش را درج می‌کند. بنابراین این دو با هم همخوانی ندارند و باید به شکل یک متد معمولی با آن رفتار کرد.

اگر حجم اطلاعاتی که قرار است در صفحه درج شود بالا است، متدهای RenderAction و RenderPartial نسبت به Html.Action و Html.Partial کارآیی بهتری دارند؛ چون یک مرحله تبدیل کل اطلاعات به رشته و سپس درج نتیجه در Response، در آن‌ها حذف شده است.

نظرات خوانندگان

نویسنده: Mojtaba
تاریخ: ۱۳۹۱/۰۱/۲۸ ۰۰:۰۱:۵۴

سلام
ضمن تشکر فراوان از زحماتتان
اگر ممکنه استفاده از DisplayTemplates برای نمایش عمومی فیلد datetime و datetime? رو در قالب یک مثال توضیح دهید
چندین مرتبه روشی رو که فرمودید انجام داد ولی فقط یک پروژه بود که دقیق عمل میکرد آیا نکته خاصی داره
و همچنین بازهم اگر ممکنه لطف کنید چگونگی یک validator رو برای ورود تاریخ شمسی در MVC توضیح دهید
مجددا بسیار سپاسگزارم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۱/۲۸ ۰۱:۱۳:۳۲

- برای نمایش خودکار تاریخ فارسی می‌تونید از این فایل استفاده کنید (تست شده) : (^)
- برای حالت nullable هم همین فایل کافی است. یعنی در اصل بهتر است DisplayTemplate برای حالت nullable نوشته شود که برای هر دو حالت مورد استفاده قرار خواهد گرفت.

- در مورد validator هم می‌تونید یک attribute سفارشی تهیه کنید. در قسمت 13 راه کلی انجام کار رو توضیح دادم. برای تاریخ شمسی بحث مفصلی است. یک کلاس قبلا در این مورد تهیه کردم : (^) البته این فقط یک ایده برای شروع است که چه فرمت‌هایی می‌تونه وارد بشه و قابل قبول باشه.

نویسنده: Mojtaba
تاریخ: ۱۳۹۱/۰۱/۲۸ ۰۲:۳۹:۰۸

نکته : تفاوت این دو خط منجر به عدم توانایی برنامه برای استفاده ازDisplayTemplates میشود

```
@item.LastLogOutDate) String.Format("{0:g}", m => item.LastLogOutDate).Html.DisplayFor(
```

از لطف شما نیز در پاسخ فنی و سریعتان نیز سپاسگزارم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۱/۲۸ ۱۰:۰۸:۵۳

- اگر قرار است فایل datetime.cshtml فرمت نهایی را اعمال کند، چرا جایی دیگری می‌خواهید آن‌را فرمت کنید که نوع و نحوه نمایش آن کلا عوض شود؟
- اگر علاقمندید از String.Format استفاده کنید، اصلا نیازی به Html.DisplayFor نبوده. برای نمونه در مثال قسمت 12 جاری، فقط کافی است که Model.AddDate را به تابع String.Format ارسال کنید تا g:0 مورد نظر شما اعمال شود. خروجی نهایی هم یک رشته است و دیگر همانند یک DateTime پردازش نمی‌شود.
- در قسمت 13 در مورد ویژگی به نام DisplayFormat توضیح داده شد. توسط آن هم می‌شود DataFormatString را اعمال کرد. در این حالت برای دستیابی به اطلاعات نهایی فرمت شده در یک display template سفارشی باید از ViewData.TemplateInfo.FormattedModelValue استفاده کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۱/۲۸ ۱۰:۱۴:۵۱

به صورت خلاصه DisplayTemplates فقط به متد Html.DisplayFor اعمال می‌شوند و نه خارج از آن. زمانیکه مستقلا از String.Format استفاده می‌کنید، یعنی خودتان قصد دارید اطلاعات را پردازش و فرمت کنید.

نویسنده: Mojtaba
تاریخ: ۰۲:۴۴:۵۸ ۱۳۹۱/۰۱/۲۹

سلام آقای نصیری خسته نباشید!
اگر بخواهم ازیک jQuery datepicker تاریخ شمسی، مثل آنچه که معرفی کرده اید در EditorTemplates استفاده کنم، چه راه حلی رو پیشنهاد می‌کنید .
ممنونم از لطفتون
ضمنا محبت کنید برای فایل خلاصه‌ی وبلاگ از آخرین تاریخ تهیه به عنوان اسم استفاده کنید تا تاریخ آپدیت آن مشخص شود.

نویسنده: وحید نصیری
تاریخ: ۰۹:۴۶:۲۲ ۱۳۹۱/۰۲/۰۱

می‌تونید داخل templated helper ایی که تعریف می‌کنید یک Html.TextBox اضافه کنید که پارامتر html attributes آن مقدار دهی شده است. چون برای نمونه این datePicker مثلا به TextBox ایی با class خاص اعمال می‌شود. مثلا: "new { @class = "datepicker

نویسنده: Milad Mohseni
تاریخ: ۱۱:۵۷:۰۰ ۱۳۹۱/۰۲/۱۱

با سلام خدمت جناب نصیری
در خصوص این مبحث یک سوالی برایم پیش آمد:
همانطور که فرمودید در ساخت View های مختلف میتوان از Helper های مختلف استفاده کرد. مثلا Html.DropDownListFor یا BeginForm و غیره ...
نهایتاً تمام این ها سمت سرور اجرا شده و نتیجه HTML برمیگرداند. حال به نظر شما مشکلی دارد پس از آنکه با استفاده از Helper های مختلف View نهایی تولید شد و دیگر قصد تغییر آن را نداشتیم، نتیجهٔ HTML شده را جهت استفاده نهایی استفاده کنیم، یعنی در Vew نهایی که بر روی Host آپلود می شود، کد های HTML تولید شده در مرحله قبل را قرار دهیم.
با این کار دیگر لازم نیست سمت سرور Viewها ساخته شوند و همه چیز آماده است؛ که میتواند باعث افزایش سرعت شود. آیا درست متوجه شدم؟
البته قسمت هایی مثل ActionLink ها که با توجه به موقعیت جاری ساخته میشوند به حالت قبل در View نوشته شوند.

نویسنده: وحید نصیری
تاریخ: ۱۲:۰۷:۴۶ ۱۳۹۱/۰۲/۱۱

- Viewها در ASP.NET MVC پس از اولین بار فراخوانی، کامپایل می‌شوند. البته می‌شود با دستکاری فایل پروژه، آن‌ها را در زمان build هم کامپایل کرد. در قسمت‌های قبل توضیح دادم. بنابراین سربار این مساله بسیار کم است.
- شاید برای مواردی مانند تکست باکس، لینک و امثال آن، آنچنان تفاوتی در این بین نباشد که از اصل آن‌ها استفاده شود، یا یک متد کمکی. اما مثلا در مورد رندر کردن یک گرید پویا بر اساس پارامترهای انتخابی یک گزارش چطور؟ هدف اصلی، بیشتر این نوع موارد است.
- مباحث caching اطلاعات را هم در قسمت‌های بعدی به آن اشاره کردم. این مورد سبب می‌شود تا اصلا کدی در سمت سرور اجرا نشود. البته در مورد بایدها و نبایدهای آن هم بحث شده در همان مطلب.

نویسنده: Milad Mohseni
تاریخ: ۱۶:۱۷:۳۰ ۱۳۹۱/۰۲/۱۱

بسیار متشکرم.
کاملاً فهمیدم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۲/۱۱ ۱۶:۲۵:۵۱

خواهش می‌کنم. یک مورد دیگر هم هست:
- متدی مانند Html.DisplayFor بر اساس خواص مدل‌های برنامه، به صورت Strongly typed تعریف می‌شود (به کمک همین lambda expressions ایی که مشاهده می‌کنید). اگر مدلی تغییر کند، این View دیگر اجرا نخواهد شد و خطای کامپایل رو دریافت می‌کنید. اما در حالت نوشتن مستقیم و معمولی مثلا یک برچسب یا یک تکست باکس و موارد مشابه، این compile time checking رو از دست خواهید داد.

نویسنده: محمد شهریاری
تاریخ: ۱۳۹۱/۰۳/۳۱ ۱۱:۳۲

هنگام استفاده از PartialView در صورتی که بخواهیم به فرض از یک Enume استفاده کنیم با توجه به توضیحاتی که در بالا امده است و انرا هنگام نمایش به DropDownList تبدیل کنیم بایستی فقط از EditorForModel استفاده کنیم ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۳/۳۱ ۱۱:۳۸

«فقط» ؟ خیر.
همان قطعه کد Html.DropDownListFor یاد شده را مستقیما در یک View هم می‌تونید استفاده کنید. اینجا فقط یک کپسوله سازی جهت استفاده مجدد بدون تکرار کدها صورت گرفته است.

نویسنده: محسن
تاریخ: ۱۳۹۱/۰۴/۲۰ ۱۱:۵۳

سلام آقای نصیری
با تشکر از توضیحاتتون
من کد زیر رو که خودتون واسه شمسی کردن تاریخ گذاشته بودین، به نام datetime.cshtml و صورت PartialView داخل پوشه DisplayTemplates قرار دادم. اما همچنان DateTimeها رو به فرمت میلادی نمایش میده. میشه بگین اشکال کارم کجاست ؟

```
@using System.Globalization
@model Nullable<DateTime>

@helper ShamsiDateTime(DateTime info, string separator = "/", bool includeHourMinute = true)
{
    int ym = info.Year;
    int mm = info.Month;
    int dm = info.Day;
    var sss = new PersianCalendar();
    int ys = sss.GetYear(new DateTime(ym, mm, dm, new GregorianCalendar()));
    int ms = sss.GetMonth(new DateTime(ym, mm, dm, new GregorianCalendar()));
    int ds = sss.GetDayOfMonth(new DateTime(ym, mm, dm, new GregorianCalendar()));
    if (includeHourMinute)
    {
        @(ys + separator + ms.ToString("00") + separator + ds.ToString("00") + " " + info.Hour + ":" +
info.Minute)
    }
    else
    {
        @(ys + separator + ms.ToString("00") + separator + ds.ToString("00"))
    }
}

@if (@Model.HasValue)
{
  @ShamsiDateTime(@Model.Value , separator: "/", includeHourMinute: false)
}
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۲۰ ۱۲:۱

در نظرات بالاتر جواب دادم:
«به صورت خلاصه DisplayTemplates فقط به متد Html.DisplayFor اعمال می‌شوند و نه خارج از آن. زمانیکه مستقلا از String.Format استفاده می‌کنید، یعنی خودتان قصد دارید اطلاعات را پردازش و فرمت کنید.»

نویسنده: محسن
تاریخ: ۱۳۹۱/۰۵/۰۱ ۱۴:۲۵

سلام آقای نصیری خسته نباشید
یه سوال از خدمتتون داشتم اونم اینکه اگر توی پروژه جایی نیاز باشه که ما اطلاعات یک فیلد رو از دیتابیس بخونیم و توی یک DropDownList نمایش بدیم، وقتی که میخوایم Value این DropDownList رو سمت کنترلر بفرستیم باید چیکار کنیم؟ منظورم اینه که فرض کنید در جدولی قرار است Username کاربران به عنوان فیلدی ذخیره شود و نام تمامی کاربران را در DropDownList نمایش داده و برای هر کدام Username را به عنوان Value به DropDownList بایند میکنیم. حال مطابق با متد Srongly Type View برای متد Create در کنترلر یک View ایجاد میکنیم. همون طور که میدونید Razor به صورت پیش فرض برای فیلد Username یک EditorFor قرار میده. در صورتی که ما میخوایم یک DropDownList به کاربر نشون بدیم که به راحتی بتونه کاربر مورد نظرش رو انتخاب کنه. حالا چجوری میشه این Username که Value این DropDownList هست رو در موقع کلیک بر روی دکمه ذخیره به سمت کنترلر فرستاد؟ در واقع من نمیدونم اصلا میشه Value رو از FormCollection گرفت یا نه ؟ امیدوارم منظورمو خوب بیان کرده باشم
و یه سوال دیگه اینکه در موقع ویرایش چجوری میشه Vaule ای که در جدول Insert شده رو به این DropDownList بایند کرد جوری که از بین کل مقدارهای بایند شده این Value خاص انتخاب شده باشد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۰۱ ۱۶:۴۲

- یک سری پیشنیاز طراحی رو باید بدونید مانند « کار با کلیدهای اصلی و خارجی در EF Code first ». در اینجا با نحوه تعریف صحیح کلید خارجی به صورت یک خاصیت عددی آشنا خواهید شد. این مورد همان چیزی است که باید از یک drop down list دریافت شود. فقط primary key یک رکورد مهم است نه تمام خواص و فیلدهای مرتبط با آن.
- مرحله بعد ارسال اطلاعات به View هست به این ترتیب:

```
public ActionResult Index()
{
    var query = db.Users.Select(c => new SelectListItem
                                          {
                                              Value = c.UserId,
                                              Text = c.UserName,
                                              Selected = c.UserId.Equals(3)
                                          });
    var model = new MyFormViewModel
                    {
                        List = query.ToList()
                    };
    return View(model);
}
```

در اینجا بر اساس اطلاعات بانک اطلاعاتی SelectListItemها تشکیل شده و به ViewModel فرم جاری ارسال می‌شود (به علاوه باید با ViewModel کار کنید نه مدل جداول بانک اطلاعاتی).

```
public class MyFormViewModel
{
    public int UserId { get; set; }
    public IList<SelectListItem> List { get; set; }
}
```

در مورد نگاشت‌ها هم مباحث auto-mapper در سایت هست.

همچنین خاصیت Selected هم در اینجا بر اساس یک شرط تعیین شده.
- قسمت View برنامه هم به این شکل خواهد بود:

```
@model MyFormViewModel

@Html.DropDownListFor(m => m.UserId, Model.List, "--Select One--")
```

نویسنده: امیر
تاریخ: ۱۳۹۱/۰۷/۱۹ ۱۸:۱۲

سلام
در قطعه کد

```
@model bool
<p>@Html.RadioButton("", false, !Model) Female</p>
<p>@Html.RadioButton("", true, Model) Male</p>
```

فرق model با با Model چیه؟
بعدش اینکه برای فهم قطعه کد زیر باید با چه مفاهیمی آشنا باشیم (پر کردن DropDown)

```
@model Enum
@Html.DropDownListFor(m => m, Enum.GetValues(Model.GetType())
         .Cast<Enum>()
         .Select(m => {
                   string enumVal = Enum.GetName(Model.GetType(), m);
                   return new SelectListItem() {
                             Selected = (Model.ToString() == enumVal),
                             Text = enumVal,
                             Value = enumVal
                  };
}))
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۱۹ ۲۱:۲۵

- فرق model و Model : (^ و ^)
- پیشنیازها:

Enum

LINQ

نویسنده: امیر
تاریخ: ۱۳۹۱/۰۷/۲۲ ۱۲:۶

سلام
شرمنده من همچنان کاملا متوجه نشدم فرق model با Model چیه؟
چرا تو view بالای صفحه یه @mode از یکی مدلهای خودمون وهله سازی میکنیم اما پایینتر وقتی خواستیم از اون مدل وهله سازی شده استفاده کنیم مینویسم Model

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۲۲ ۱۲:۳۶

به زبان ساده: اینها رو یک سری syntax درنظر بگیرید.
مطابق میل طراح Razor، زمانیکه model با m کوچک مطرح می‌شود، یعنی قرار است جنس مدل مورد استفاده صریحا ذکر شود.
زمانیکه از Model با M بزرگ استفاده می‌شود، یعنی با یک وهله از مدلی با جنس مشخص شده سر و کار داریم.

نویسنده: سعید یزدانی
تاریخ: ۱۳۹۱/۱۱/۱۷ ۳:۰

سلام آقای نصیری
در قسمتی که از radio استفاده کردید وقتی که model رو از نوع bool قرار میدم error میده

```
The model item passed into the dictionary is null
```

وقتی create فراخوانی میشه و بعدش partial رو فراخوانی میکنه . مدل partial view نال هست
اما وقتی که partial view رو بدون مدل set کردم مشکلی پیش نیومد و به فیلدم bind شد
میشه لطف کنید بگید اشکال کارم کجاست

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۱۷ ۸:۵۹

- کلیه مثال‌های این سری از اینجا قابل دریافت هستند.
+ مراجعه کنید به مطالب زیر برای توضیحات بیشتر و تکمیلی:

RadioButtonList در ASP.NET MVC

CheckBoxList در ASP.NET MVC

نویسنده: سعید یزدانی
تاریخ: ۱۳۹۱/۱۱/۱۷ ۱۷:۱۰

قسمتی که enum رو به dropdown تغییر دادیم رو خوب متوجه نشدم اگه لینکی هست لطف کنید قرار بدید

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۱۷ ۱۷:۱۹

به پیشنیازهای Enum در سایت مراجعه کنید.

نویسنده: میثم خوشقدم
تاریخ: ۱۳۹۲/۰۲/۳۰ ۱۳:۳۷

سلام؛ زمانی که در پروژه از [UIHint("GenderOptions")] استفاده می‌کنم به خوبی خروجی می‌گیرم اما زمانی که که Model من در DataModel و یک پروژه مجزا و در یک Dll دیگر است کار نمی‌کند.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۳۰ ۱۴:۱۴

مشکلی مشاهده نشد. در پروژه ذیل، کلاس‌های مدل قسمت جاری به یک اسمبلی جداگانه منتقل شدند و باز هم پروژه بدون مشکل کار می‌کند:

MVC-12-02.zip

نویسنده: میثم خوشقدم
تاریخ: ۱۳۹۲/۰۲/۳۱ ۱۵:۲۰

سلام

ممنون از پاسختون و همچنین پروژه ای که Attach کردید.

پس از مقایسه و پیگیری متوجه شدم اگر GenderOptions در فولدر Shared\EditorTemplates باشد کار می‌کند اما من این UiHint و صرفا برای یک ویو خاص می‌خوام. از این رو اگز این UidHint و در هر مسیری به غیر از مسیر فوق قرار بدم شناسایی نمیشه. این مسئله مخصوصا به این شکل است و یا من درجایی اشتباه کردم.

یک سوال دیگه که برام پیش اومده این است که ویو من اتوماتیک و با استفاده از متد @Hml.Editorformodels

ساخته میشه اما در متد Post مدلی به ویو پاس ندادم و صرفا return view() زدم و در متد Get کنترل پارامتری از نوع مدل مورد نظر گرفتم. حالا سوال من این است که درسته که در ابتدای ویو با @model myProject.MyModel به صورت Strongly type تعریف کردم اما در صدا زدن ویو مدلی را ارسال نکردم اما ویو من از روی strongly type ساخته میشه ! و این ساخته شدن مشخص نیست به چه شکله چرا که متد سازنده کلاس (Constructor) را هم صدا نمی‌زند!

باز هم ممنون از پاسختون.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۳۱ ۱۶:۴۷

- در متن توضیح دادم: « البته باید درنظر داشت که استفاده از پوشه Views/Shared *اجباری نیست* . برای مثال اگر از پوشه DisplayTemplates/ *Home* /Views استفاده کنیم، قالب‌های سفارشی تهیه شده *تنها* جهت Viewهای کنترلر *home* قابل استفاده خواهند بود.»
در متن عنوان شده DisplayTemplates، شما این رو مثلا به EditorTemplates تغییر بدید. اصول یکی است.
- مراجعه کنید به سورس ASP.NET MVC و قسمت‌های مرتبط رو مطالعه کنید؛ جهت آشنایی بیشتر با سازوکار درونی آن‌ها.

نویسنده: ahmadb7
تاریخ: ۱۳۹۲/۰۳/۱۱ ۲۰:۱۸

```
public enum MyGrade { A = 20, B =15, C =10, }
        [UIHint("Enum")]
        public MyGrade Grade { set; get; }
```

در فولدر EditorTemplates ،
partial view مانند مثال شما ایجادکردم اما زمانی که یک view ایجاد میکنم
Dropdownlist تولید نمی‌شود

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۱۱ ۲۲:۱۶

سورس کامل این سری از اینجا و یا سورس قسمت 12 از اینجا قابل دریافت است. نام فایل‌ها، مسیرها و محتوای آن‌ها رو با کار خودتون مقایسه کنید.

نویسنده: لیلا
تاریخ: ۱۳۹۲/۰۴/۲۹ ۹:۳۹

با سلام و خسته نباشید
فایل این سری(12) حاوی مثال Dropdownlist نمی‌باشد و متاسفانه Dropdownlist تولید نمی‌شود . از روش دیگری توانستم این کار را انجام دهم از اینجا ولی روش شما قابل فهم‌تر است لطف بفرمایید راهنمایی کنید.
با تشکر فراوان

نویسنده: سامان
تاریخ: ۱۳۹۲/۰۴/۲۹ ۱۴:۲۳

با سلام و تشکر
چیزی که الان یاد گرفتم اینه که می‌توان متادیتا را در یک کلاس جدا کرد و سپس آنرا به مدل اعمال کرد، آیا این امکان وجود داره که در runtime به مدل مورد نظر کلاس metadata اضافه شود؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۲۹ ۱۸:۵۹

- ویژگی‌ها یا Attributes در دات نت، استاتیک متادیتا هستند؛ مانند تعداد پارامترها، نام متدها و امثال آن که به صورت کامپایل شده در فایل باینری نهایی قرار می‌گیرند و نهایتا از طریق Reflection قابل دسترسی خواهند بود. تغییر آن‌ها یا افزودن آن‌ها عموما از طریق دستکاری کدهای IL میسر است یا از روش‌های IL Code weaving مباحث AOP یا روش‌هایی مانند Reflection.Emit و همانند آن.
- یک استثناء در اینجا وجود دارد و آن هم متد TypeDescriptor.AddAttributes است که در زمان اجرا کار می‌کند. استفاده از آن هم فقط زمانی جواب خواهد داد که فریم ورک پایه از متد TypeDescriptor.GetAttributes برای یافتن ویژگی‌ها استفاده کرده باشد.

نویسنده: حمیدی
تاریخ: ۱۳۹۲/۰۶/۰۱ ۱:۱۱

سلام؛ من می‌خوام به layout توی پروژه چندتا model پاس بدم اما نمیتونم. از راههایی هم سعی کردم اما ارور میده. البته فکر میکنم ارورش به خاطره تداخل با مدلیه که به view پاس دادم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۶/۰۱ ۸:۳۲

- از Html.RenderAction استفاده کنید.
+ و یا همچنین layout، مدل محتوای خودش را به ارث می‌برد. یعنی مدلی که در View تنظیم می‌شود، همان مدلی است که layout به آن دسترسی خواهد داشت. به همین جهت مثلا می‌تونید توسط ViewBag، عنوان صفحه را که در layout تعریف شده، مقدار دهی کنید.
اگر می‌خواهید Strongly typed کار کنید، روش Html.RenderAction یک راه حل است و روش دوم به صورت زیر است:
یک کلاس پایه abstract تعریف کنید:

```
public abstract class BaseViewModel
{
    public string Name { get; set; }
}
```

بعد تمام مدل‌ها یا ViewModelهایی که قرار است در برنامه شما به Viewها ارسال شوند، باید از این کلاس پایه ارث بری کنند.
مثلا:

```
public class HomeViewModel : BaseViewModel
{
   public int Data1 { set; get;}
   // ...
}
```

در این حالت و با رعایت این شرط، می‌تونید در فایل layout، نوع مدل را بجای حالت dynamic فعلی، تبدیل کنید به نوع کلاس پایه‌ایی که ذکر شد:

```
@model BaseViewModel
```

```
<!DOCTYPE html>
<html>
    <head>
        <meta name="viewport" content="width=device-width" />
        <title>Test</title>
    </head>
    <body>
        <header>
            Hello @Model.Name
        </header>
        <div>
            @this.RenderBody()
        </div>
    </body>
</html>
```

الان در layout، نوع کلاس پایه، به عنوان نوع مدل اصلی تعریف شده. بنابراین در این فایل layout مشترک بین تمام Viewها، خواص قرار گرفته شده در کلاس پایه‌ای که توسط ViewModelها به Viewها ارسال می‌شوند، به صورت strongly typed قابل دسترسی خواهند بود.

نویسنده: حمیدی
تاریخ: ۱۳۹۲/۰۶/۰۱ ۹:۲۶

این که گفتید کاملا انجام شد و ممنون؛ اما من میخوام از foreach استفاده کنم. چطور یه لیست پاس بدم که مثلا 10 مطلب آخر و به صورت desc نشون بده..

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۶/۰۱ ۹:۴۳

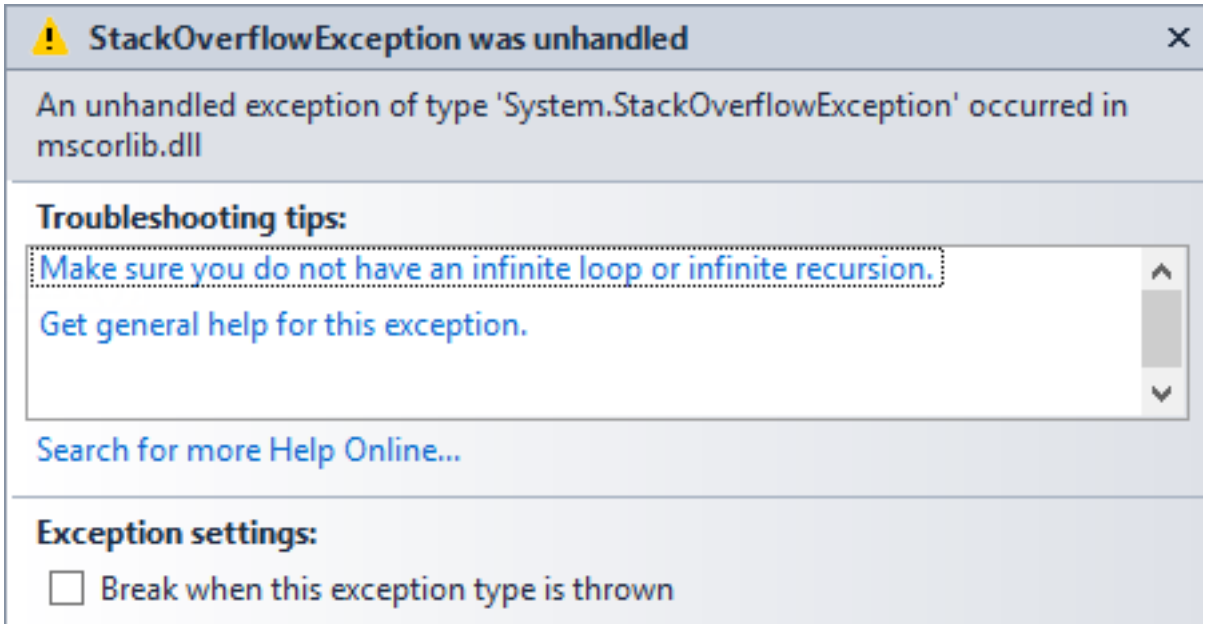
- ابتدا نیاز است با مفهوم ViewModel آشنا باشید.
- اگر لیستی قرار است در تمام صفحات نمایش داده شود و محل آن هم باید در layout باشد، یعنی باید این لیست در هر بار نمایش و یا تولید هر View (تمام Viewهای سایت)، تولید شود. بنابراین در BaseViewModelایی که عنوان شد، تعریف خاصیت این لیست را قرار دهید و در layout از آن استفاده کنید.

```
// در پایه مدل‌ها
public abstract class BaseViewModel
{
    public IList<Post> Posts { get; set; } // خاصیت عمومی که قرار است در فایل مستر قابل دسترسی باشد
}

// در اکشن متد
 return View(model: new HomeViewModel { Posts = .... });

// در اینجا ویومدل ارسالی از پایه مدل‌ها مشتق می‌شود
public class HomeViewModel : BaseViewModel
```

- و اگر نمی‌خواهید به ازای هر Viewایی که در سایت تولید می‌شود یکبار این لیست را مقدار دهی کنید، بهتر است از روش Html.RenderAction استفاده کنید (بحث «نمایش اطلاعات از کنترلرهای مختلف در یک صفحه» در مطلب فوق). مباحث Caching را هم برای لیست‌های عمومی فراموش نکنید.

نویسنده: رضا منصوری
تاریخ: ۱۳۹۲/۰۸/۱۷ ۸:۵۲

با تشکر از مطلب خوبتون
امکان استفاده از RenderAction در Shared/_Layout وجود نداره ؟ وقتی اینکارو انجام میدم با ارور StackOverflowException برخورد میکنم تکه کد اجرای RenderAction انقدر تکرار میشه تا به این ارور برخورد مبکنه!



میخوام لیست گروهبندی مطالبمو تو _Layout.cshtml (نمایش در تمام صفحات مثل MasterPage در WebForm) نشون بدم چه راهی رو پیشنهاد میدید ؟ خیلی ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۱۷ ۹:۳۵

- مشکلی از این لحاظ نیست که نشود از RenderAction در فایل layout استفاده کرد.
- منطق تهیه گروه بندی مطالب شما احتمالا یک حلقه بی‌نهایت دارد یا یک الگوریتم بازگشتی بی پایان است.
- ممکن است در این حالت از return PartialView استفاده نکردید و return View بوده . در این حالت View بازگشتی ارجاعی را به فایل layout خواهد داشت. یعنی به صورت تو در تو فایل layout اجرا و تکرار می‌شود.
- یا در اینجا بهتر است در ابتدای فایل بنویسید Layout = null تا زمان رندر شدن در فایل layout دوباره سبب ارجاع بی‌نهایتی به فایل Layout نشود.

نویسنده: رضا منصوری
تاریخ: ۱۳۹۲/۰۸/۱۷ ۱۵:۲۲

حدس شما درست بود از return PartialView استفاده نکرده بودم بازم ممنون

نویسنده: م رمضانی
تاریخ: ۱۳۹۲/۱۱/۱۱ ۸:۴۶

سلام؛ اگر بخواهیم textboxئی که در view برای فیلدی با مورد مصرفی currency ولی data type: decimal (در لایه مدل) نمایش داده میشه از فرمت سه رقم سه رقم جدا کننده پیروی کنه، به چه صورتی باید عمل بشه؟
و یه سوال دیگه، این فرمت گذاری برای textboxئی که دریافت کننده تاریخه ولی فیلدش در لایه Model از نوع int تعریف شده به چه صورتیه (یعنی بین روز و تاریخ و ماه تو حالت نمایش جدا کننده ای وجود داشته باشه)؟
متشکرم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۱ ۹:۴۹

- دومین DisplayFormat ایی که در متن هست، دقیقا کار افزودن سه رقم جدا کننده را انجام می‌دهد.

- از UIHint به همراه یک قالب سفارشی استفاده کنید. مثال مفهومی آن هم در متن هست و هم در قسمت نظرات. یا می‌شود از یک ViewModel استفاده کرد به همراه یک فیلد محاسباتی. یا حتی امکان استفاده از متدهای الحاقی جهت فرمت کردن خروجی نیز در Viewها میسر است.

نویسنده: م رمضانی
تاریخ: ۱۳۹۲/۱۱/۱۱ ۱۱:۱۲

برای پیاده سازی سه رقم جداکننده به این شیوه عمل کردم:

در Model

```
[DisplayFormat(DataFormatString = "{0:n}", ApplyFormatInEditMode = true)]
public decimal cost { get; set; }
```

در View

```
@model MyMVCProject.Models.MyModel
.
.
.
.

@Html.EditorFor(m => m.cost)
@*@Html.TextBoxFor(m => m.cost)*@
```

ولی متاسفانه جوابی نگرفتم! (برای طراحی view از bootstrap استفاده میکنم)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۱۱ ۱۲:۱۳

- DisplayFormat برای کارهای نمایشی سمت سرور است. مثلا در یک حلقه Razor سمت سرور، لیستی را نمایش می‌دهید؛ متد Html.DisplayFor از آن استفاده خواهد کرد.

- اگر قرار است در حین ورود اطلاعات کاربر، به صورت خودکار سه رقم جداکننده اضافه شود، ارتباطی به MVC سمت سرور نداشته و راه حل آن مثلا استفاده از افزونه‌های جی‌کوئری است که سمت کاربر ورودی را تحت نظر قرار داده و آن‌را فرمت می‌کنند. مثلا

نویسنده: میثم فغفوری
تاریخ: ۱۳۹۲/۱۱/۱۸ ۱۹:۳

سلام؛ جوری که من متوجه شدم در Razor وقتی تعدادی ماژول داریم (مثل آخرین اخبار و دسته بندی مطالب و ...) باید Controller و view مخصوص به خودش رو ایجاد کنیم و توی Layout برنامه با متدهای Html.RenderAction و Html.Action اون هارو توی Sectionهای دلخواه فراخوانی کنیم تا توی همه پیج‌ها نمایش داده بشن درسته؟ ممنون.

نویسنده: ح مرداد
تاریخ: ۱۳۹۲/۱۱/۲۳ ۳:۱

سلام،
بله، طبق علم بنده، توی هر View فقط با یک مدل میشه کار کرد (یک indstance از یک کلاس رو میشه بهش پاس داد) ؛ بنابراین برای نمایش اطلاعات غیر مرتبطی که نمیشه توی یک کلاس گنجوند باید از چند partial view مجزا استفاده کنیم که هر ویو یک بخش رو نمایش بده و بعد به کمک دستوراتی که خودتون عرض کردین اینا رو توی یک ویوی مجزا نمایش میدیم.
مثلا می‌خوایم موضوعات ، آخرین اخبار و آمار سایت رو توی یک صفحه داشته باشیم؛ حال آنکه اینا هر کدوم یک دیتای مجزان مسلما باید یک partial view برای موضوعات ، یک partial view برای آخرین اخبار و یک partial view برای آمار سایت ایجاد کنیم و در پایان اینا رو توی مستر پیج (Layout) بوسیله دستورات RenderAction یا Action در کنار هم نمایش میدیم.

توضیح تکمیلی:
توی وب فرم ما یک عدد مستر پیج و چند عدد یوزر کنترل داشتیم و یوزر کنترل هامونو درگ می‌کردیم توی مسترپیج ، توی MVC همین کارو می‌کنیم ، فقط به جای مسترپیج Layout داریم و به جای یوزر کنترل Partial View داریم..... همین!
به جای درگ کردن یوزرکنترل توی مستر پیج هم RenderAction و Action داریم...
{ کلا MVC اومده و همه کارها رو کدی کرده و به جای درگ کردن کنترل توی صفحه و .... باید کد بنویسیم:) }
البته اولش ممکنه برای آدم سخت باشه ولی کم کم که بهش عادت کنی و خروجی html صفحاتتو ببینی که چقدر تمیز و خوشگل ! شده ، ازش خوشت میاد :)))

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۱/۲۳ ۹:۳۷

در یک View امکان استفاده از چند مدل هم هست. اگر بخواهید dynamic کار کنید می‌شود از ViewBag استفاده کرد اگر بخواهید strongly typed کار کنید می‌شود از ViewModelها استفاده کرد: « نحوه استفاده از ViewModel در ASP.NET MVC »

نویسنده: م رمضانی
تاریخ: ۱۳۹۲/۱۱/۲۵ ۱۲:۳

ممنون؛ به سایتی که معرفی کردید رفتم و هر دو فایل jquery.price_format.2.0.js و jquery.price_format.2.0.min.js رو به پروژه اضافه کردم. تو view خطوط زیر رو نوشتم:

```
<script src="../../Scripts/jquery-1.9.1.min.js" type="text/javascript"></script>
<script src="../../Scripts/jquery.price_format.2.0.js" type="text/javascript"></script>
```

ولی باز هم نتیجه نگرفتم. میشه لطف کنید راهنمائیم کنید که چطور از اون مثال استفاده کنم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۲۵ ۱۲:۴۹

- نحوه استفاده از افزونه Firebug برای دیباگ برنامه‌های ASP.NET مبتنی بر jQuery
- محل تعریف اسکریپت‌ها باید در فایل Layout باشد و بهتر است از bundling موجود استفاده کنید.
+ مسیردهی صحیح فایل‌ها در ASP.NET MVC با توجه به مسایل مسیریابی آن باید توسط Url.Content صورت گیرد:

```
src='@Url.Content("~/Contents/Scripts/jQuery.js")'
```

البته Razor نگارش 2 به بعد با روش زیر هم سازگار است:

```
src='~/Contents/Scripts/jQuery.js'
```