عنوان: **TwitterBootstrapMVC**
نویسنده: رضا رضایی
تاریخ: ۱۳۹۲/۰۹/۱۲ ۲۳:۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, ASP.Net MVC 4, Mvc helper, Twitter Bootstrap, MVC 4, HTML helpers

TwitterBootstrapMVC یا به اختصار BMVC یک کتابخانه از Helper های مفید برای ساده سازی استفاده از Twitter Bootstrap در MVC می‌باشد .

در این کتابخانه امکانات مختلف Bootstrap از طریق Helper های نوشته شده برای MVC براحتی قابل استفاده می‌باشد و فرایند کد نویسی را ساده‌تر و در عین حال خواناتر می‌کند ، Helper های موجود در این کتابخانه به صورت زنجیره ای (fluent syntax) نوشته شده که استفاده از آن را سهولت می‌بخشد .

برای استفاده از آن در mvc 4 کافی است بعد از پیکر بندی Bootstrap (راهنمایی) به کتابخانه witterBootstrapMvc T رفرنسی ایجاد کنید و با استفاده از این راهنما نحوه استفاده را فرا گیرید . همچنین می‌توانید آن را از طریق NuGet بارگذاری نمایید .

نسخه‌ی MVC4 آن‌را در اینجا برای شما نیز آپلود نمودم در زیر نمونه ای از استفاده از آن را می‌بینید

```
@Html.Bootstrap().LabelFor(x => x.UserName)
@Html.Bootstrap().TextBoxFor(m => m.UserName)
@Html.Bootstrap().PasswordFor(m => m.Password)
@Html.Bootstrap().FileFor(m => m.File)
@Html.Bootstrap().CheckBoxFor(m => m.IsActivated)
@Html.Bootstrap().RadioButtonFor(m => m.Gender, "male")
@Html.Bootstrap().DropDownListFor(m => m.State, Model.UsaStates)
@Html.Bootstrap().ListBoxFor(m => m.State, Model.UsaStates)
@Html.Bootstrap().TextAreaFor(m => m.Description)
```

ایجاد یک فرم

```
@using (Html.Bootstrap().Begin(new Form().Type(FormType.Inline)))
{
    @Html.Bootstrap().TextBoxFor(m => m.Email).Placeholder("Email")
    @Html.Bootstrap().PasswordFor(m => m.Password).Placeholder("Password")
    @Html.Bootstrap().CheckBoxFor(m => m.RememberMe).Label()
    @Html.Bootstrap().SubmitButton().Text("Sign in")
}
```

یک فرم Modal

```
@Html.Bootstrap().Button().Text("Show Modal").IconAppend(Icons.camera) .TriggerModal("MyModal")


@using(var modal = Html.Bootstrap().Begin(new Modal()   .Id("MyModal")    .HtmlAttributes(new { @class
= "custom-class" })    .Fade() )){
    using(modal.BeginHeader())
    {
        <h2>Some header</h2>
    }
    using(modal.BeginBody())
    {
        <p>Some body<p>
    }
    using(modal.BeginFooter())
    {
        <p>Footer here.<p>
        @Html.Bootstrap().Button().Text("Close")
    }
}
```

نظرات خوانندگان

نویسنده: Hamid NCH
تاریخ: ۱۳۹۲/۰۹/۱۳ ۱۲:۱۸

این کتابخانه محدودیت استفاده نداره؟منظورم اینه کافیه این کتابخونه رو به رفرنس اد کنیم و استفاده کنیم!
ممنون.

نویسنده: افشار محبی
تاریخ: ۱۳۹۲/۰۹/۱۳ ۱۲:۵۷

تا حالا فرصت نشده که به طور اساسی با Twitter Bootstrap درگیر بشوم. اما وجود این تعداد راهنما و ابزار جانبی آدم را امیدوار می‌کند که بتواند یک خروجی خیلی خوب از آن بگیرد.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۱۳ ۱۲:۵۸

- داره (مجانی نیست). البته فایل‌هایی که ایشون پیوست کردند به نظر محدودیت ندارند.
- ضمنا باز هم هستند یک سری Wrapper برای بوت استرپ که می‌توانند مورد استفاده قرار گیرند:
TwitterBootstrapMvc (نسخه سورس باز مطلب جاری است)
Mvc Bootstrap Html Helper Extensions
Bootstrap Helpers (معرفی در اینجا)
Twitter Bootstrap Controls for ASP.NET

نویسنده: رضا رضایی
تاریخ: ۱۳۹۲/۰۹/۱۳ ۲۳:۱۰

با سلام
متاسفانه این کتابخانه از نسخه 2 به بعد به صورت غیر رایگان عرضه شده است ، هر چند نسخه آپلود شده آخرین نسخه آن برای MVC 4 می‌باشد .
این کتابخانه برا اساس امکانات Bootstrap 3 نوشته شده است و هدفش تسهیل استفاده از Bootstrap 3 در MVC می باشد و بسیار کاربردیست .
دارای محدودیت خاصی نمی‌باشد و بسیاری از امکانات پیچیده Bootstrap را ساده نموده است

نویسنده: رضا رضایی
تاریخ: ۱۳۹۲/۰۹/۱۳ ۲۳:۱۴

twitter Bootstrap کتابخانه جامعی از CSSها و JScriptها برای طراحی صفحات وب می‌باشد ک امروزه بسیار مقبول واقع شده است و ظاهری ساده و در عین حال کارامد را برای وب سایت شما با پشتیبانی از انواع مختلف از نمایشگرها به ارمغان می‌آورد .

نویسنده: هومن
تاریخ: ۱۳۹۲/۰۹/۱۵ ۱۱:۲۸

سلام
مطلب بسیار مفیدی بود ممنون.
مشکلی که من دارم اینه که اولا از طریق نیوگت قابل نصب نیست یعنی طبق سایت خودشون اسم بکیجشو که مینویسم چیزی پیدا نمیکنه...
منم dll هارو از سایت خودشون دانلود کردم و دستی add refrence کردم... در webconfig قسمت viewها هم دو تگ مربوط را اضاف کردم.

اما در ریزور تا اینجا Bootstrap . Html () را میشناسد اما بعد از پرانتز intelli-sense آن را نمیشناسد. مشکل از کجاست؟ ضمنا نسخه vs من 2013 و از mvc5 استفاده میکنم

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۹/۱۵ ۱۱:۳۳

- نسخه سایت خودش سی روزه هست. باید یک فایل مجوز هم براش دریافت کنی، اینجا:
https://www.twitterbootstrapmvc.com/Download یا نسخه پیوست شده از متن رو دریافت و استفاده کن.
- برای MVC5 در اینجا: http://www.nuget.org/packages/TwitterBootstrapMVC5/
- برای MVC4 در اینجا: http://www.nuget.org/packages/TwitterBootstrapMVC/
سایر موارد مرتبط

نویسنده: رضا رضایی
تاریخ: ۱۳۹۲/۰۹/۱۶ ۱۹:۳۸

سلام ، از NuGet قابل دریافته ولی نیاز به لایسنز داره که باید بخری و نسخه مجانیش فقط سی روزست ، فایل‌های که پیوست کردم این مشکل و نداره و آخرین نسخست ،راه درستش اینه که اول Bootstrap 3 را کامل تو برنامت پیکربندی کنی و از جواب دادنش اطمینان حاصل کنی بعد فقط کافیه فایل TwitterBootstrapMVC.dll و T4MVCExtensions.dll رفرنس بزنی ، بعد از اون باید در قسمت معرفی فضاهای نام در مسیر web.config / View این دو مورد و اضافه کنی

```
<add namespace="TwitterBootstrapMVC" />
<add namespace="TwitterBootstrap3" />
```

بعد از اون براحتی جواب می‌ده
یادتون نره که فایل Portable.Licensing.dll هم هنگام اجرا باد در کنار دو فایل بالا وجود داشته باشد

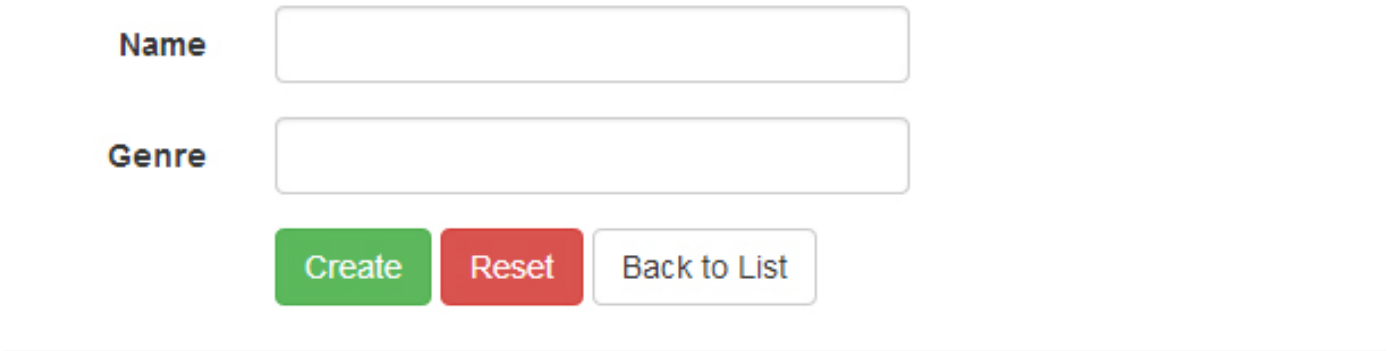
نویسنده: مجتبی فخاری
تاریخ: ۱۳۹۲/۱۲/۲۳ ۱۷:۴۷

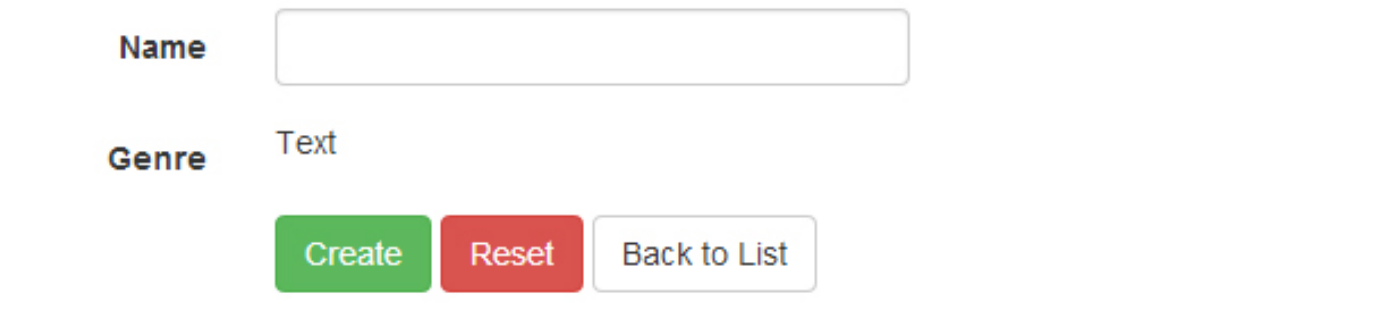
با سلام
آیا با Bootstrap.rtl 3 هم جواب میده ؟
آخه من همین فایل شما را که برای mvc4 است دانلود کرده و add References هم نمودم و فضاهای نام را هم در View / web.config اضافه نمودم ولی اصلاً کار نمیده و برام Intelisence رو که میاره داخلش BootStrap نداره.

عنوان: **قالب‌های سفارشی برای HtmlHelperها**
نویسنده: علی اکبر کورش فر
تاریخ: ۱۳۹۳/۰۵/۱۶ ۱۳:۳۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, HTML helpers

در ابتدای بحث، برای آشنایی بیشتر با HTML Helperها به مطالعه این مقاله بپردازین.
در این مقاله قرار است برای یک HTML Helper خاص، قالب نمایشی اختصاصی خودمان را طراحی کنیم و به نحوی HTML Helper موجود را سفارشی سازی کنیم. به عنوان مثال می‌خواهیم خروجی یک EditorFor() برای یک نوع خاص، به حالت دلخواهی باشد که ما خودمان آن را تولیدش کردیم؛ یا اصلا نه. حتی می‌شود برای خروجی یک EditorFor() که خصوصیتی از جنس string را می‌خواهیم به آن انتساب دهیم، به جای تولید input، یک مقدار متنی را برگردانیم. به این حالت:

```
<div>
            @Html.LabelFor(model => model.Name, htmlAttributes: new { @class = "control-label col-md-2"
})
            <div>
                @Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class = "form-
control" } })
                @Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-danger" })
            </div>
        </div>

        <div>
            @Html.LabelFor(model => model.Genre, htmlAttributes: new { @class = "control-label col-md-
2" })
            <div>
                @Html.EditorFor(model => model.Genre, new { htmlAttributes = new { @class = "form-
control" } })
                @Html.ValidationMessageFor(model => model.Genre, "", new { @class = "text-danger" })
            </div>
        </div>
```





در ادامه یک پروژه‌ی عملی را شروع کرده و در آن کاری را که می‌خواهیم، انجام می‌دهیم. پروژه‌ی ما به این شکل می‌باشد که قرار است در آن به ثبت کتاب بپردازیم و برای هر کتاب هم یک سبک داریم و قسمت سبک کتاب‌های ما یک Enum است که از قبل می‌خواهیم مقدارهایش را تعریف کنیم.

مدل برنامه

```
public class Books
    {
        public int Id { get; set; }
        [Required]
        [StringLength(255)]
        public string Name { get; set; }
        public Genre Genre { get; set; }
    }
```

```
public enum Genre
    {
        [Display(Name = "Non Fiction")]
        NonFiction,
        Romance,
        Action,
        [Display(Name = "Science Fiction")]
        ScienceFiction
    }
```

در داخل کلاس Books یک خصوصیت از جنس Genre برای سبک کتاب‌ها داریم و در داخل نوع شمارشی Genre، سبک‌های ما تعریف شده‌اند. همچنین هر کدام از سبک‌ها هم به ویژگی Display مزین شده‌اند تا بتونیم بعدا از مقدار آنها استفاده کنیم.

کنترلر برنامه

```
public class BookController : Controller
    {
        // GET: Book
        public ActionResult Index()
        {
            return View(DataAccess.DataContext.Book.ToList());
        }

        public ActionResult Create()
        {
            return View();
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Books model)
        {
            if (!ModelState.IsValid)
                return View(model);

            try
            {
                DataAccess.DataContext.Book.Add(model);
                DataAccess.DataContext.SaveChanges();
                return RedirectToAction("Index");
            }
            catch (Exception ex)
            {
                ModelState.AddModelError("", ex.Message);
                return View(model);
            }
        }

        public ActionResult Edit(int id)
        {
            try
            {
                var book = DataAccess.DataContext.Book.Find(id);
                return View(book);
```

```
            }
            catch (Exception ex)
            {
                return View("Error");
            }
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(Books model)
        {
            if (!ModelState.IsValid)
                return View(model);

            try
            {
                DataAccess.DataContext.Book.AddOrUpdate(model);
                DataAccess.DataContext.SaveChanges();
                return RedirectToAction("Index");
            }
            catch (Exception ex)
            {
                ModelState.AddModelError("", ex.Message);
                return View(model);
            }
        }

        public ActionResult Details(int id)
        {
            try
            {
                var book = DataAccess.DataContext.Book.Find(id);
                return View(book);
            }
            catch (Exception ex)
            {
                return View("Error");
            }
        }
    }
```

در قسمت کنترلر هم کار خاصی جز عملیات اصلی نوشته نشده‌است. لیست کتاب‌ها را از پایگاه داده بیرون آوردیم و از طریق اکشن Index به نمایش گذاشتیم. با اکشن‌های Create، Edit و Details هم کارهای روتین مربوط به خودشان را انجام دادیم. نکته‌ی قابل تذکر، DataAccess می‌باشد که کلاسی است که با آن ارتباط برقرار شده با EF و سپس اطلاعات واکشی و تزریق می‌شوند.

View مربوط به اکشن Create برنامه

```
@using Book.Entities
@model Book.Entities.Books

@{
    ViewBag.Title = "Create";
}
<h2>New Book</h2>
@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div>
        <h4>Books</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div>
            @Html.LabelFor(model => model.Name, htmlAttributes: new { @class = "control-label col-md-2"
})
            <div>
                @Html.EditorFor(model => model.Name, new { htmlAttributes = new { @class = "form-
control" } })
                @Html.ValidationMessageFor(model => model.Name, "", new { @class = "text-danger" })
            </div>
        </div>

        <div>
```

```
            @Html.LabelFor(model => model.Genre, htmlAttributes: new { @class = "control-label col-md-
2" })
            <div>
                @Html.EditorFor(model => model.Genre, new { htmlAttributes = new { @class = "form-
control" } })
                @Html.ValidationMessageFor(model => model.Genre, "", new { @class = "text-danger" })
            </div>
        </div>

        <div>
            <div>
                <input type="submit" value="Create" />
                <input type="reset" value="Reset" />
                @Html.ActionLink("Back to List", "Index", null, new {@class="btn btn-default"})
            </div>
        </div>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

View برنامه هم همان ویویی است که خود Visual Studio برای ما ساخته‌است. به جز یک سری دست‌کاری‌هایی داخل سی‌اس‌اس، هدف از گذاشتن View مربوط به Create این بود که قرار است بر روی این قسمت کار کنیم. اگر پروژه رو اجرا کنید و به قسمت Create بروید، مشاهده خواهید کرد که برای Genre یک input ساخته شده‌است که کاربر باید در آن مقدار وارد کند. ولی اگر یادتان باشد، ما سبک‌های نگارشی خودمان را در نوع شمارشی Genre ایجاد کرده بودیم. پس عملا باید یک لیست به کاربر نشان داده شود که تا از آن لیست، نوع را انتخاب کند. می‌توانیم بیایم همینجا در داخل View مربوطه، به‌جای استفاده از HTML Helper پیش‌فرض، از DropDownList یا EnumFor استفاده کنیم و به طریقی این لیست را ایجاد کنیم. ولی چون قرار است در این مثال به شرح موضوع مقاله خودمان بپردازیم، این کار را انجام نمی‌دهیم.

در حقیقیت می‌خوایم متد EditorFor را طوری سفارشی سازی کنیم که برای نوع شمارشی Genre، به صورت خودکار یک لیست ایجاد کرده و برگرداند. از نسخه‌ی سوم ASP.NET MVC به بعد این امکان برای توسعه دهنده‌ها فراهم شده‌است. شما می‌توانید در پوشه‌ی Shared داخل پوشه Views برنامه، پوشه‌ای را به اسم EditorTemplates ایجاد کنید؛ همینطور DisplayTemplates و برای نوع خاصی که می‌خواهید سفارشی‌سازی را برای آن انجام دهید، یک PartialView بسازید.

```
Views/Shared/DisplayTemplates/<type>.cshtml
```

یک PartialView در داخل پوشه EditorTemplates به نام Genre.cshtml ایجاد کنید. برای اینکه مشاهده کنید چطور کار می‌کند، کافی‌است یک فایل متنی اینجا تهیه کرده و بعد پروژه را اجرا کرده و به قسمت Create روید تا تغییرات را مشاهده کنید. بله! به‌جای inputی که از قبل وجود داشت، فقط متن شما آنجا نوشته شده‌است. (به عکسی که در بالا قرار دارد هم می‌تونید نگاه کنید)

کاری که الان میخواهیم انجام دهیم این است که یک SelectListItem ایجاد کرده تا مقدارهای نوع Genreمان داخلش باشد و بتوانیم به راحتی برای ساختن DropDownList از آن استفاده کنیم. برای این کار Helper مخصوص خودمان را ایجاد می‌کنیم. پوشه‌ای به اسم Helpers در کنار پوشه‌های Controllers، Models ایجاد می‌کنیم و در داخل آن کلاسی به اسم EnumHelpers می‌سازیم.

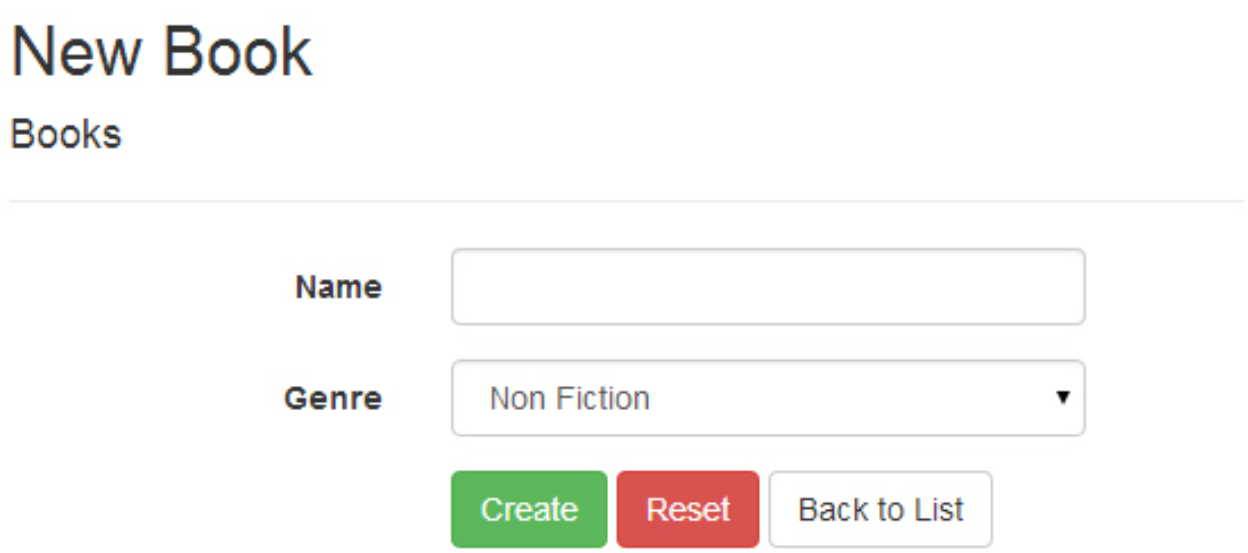
```
public static class EnumHelpers
    {
        public static IEnumerable<SelectListItem> GetItems(
            this Type enumType, int? selectedValue)
        {
            if (!typeof(Enum).IsAssignableFrom(enumType))
            {
                throw new ArgumentException("Type must be an enum");
            }

            var names = Enum.GetNames(enumType);
            var values = Enum.GetValues(enumType).Cast<int>();

            var items = names.Zip(values, (name, value) =>
```

```
                    new SelectListItem
                    {
                        Text = GetName(enumType, name),
                        Value = value.ToString(),
                        Selected = value == selectedValue
                    }
                );
            return items;
        }

        static string GetName(Type enumType, string name)
        {
            var result = name;

            var attribute = enumType
                .GetField(name)
                .GetCustomAttributes(inherit: false)
                .OfType<DisplayAttribute>()
                .FirstOrDefault();

            if (attribute != null)
            {
                result = attribute.GetName();
            }

            return result;
        }
    }
```

در توضیح کد بالا عنوان کرد که متدها به‌صورت متدهای الحاقی به نوع Type نوشته شدند. کار خاصی در بدنه‌ی متدها انجام نشده‌است. در بدنه‌ی متد اول لیست آیتم‌ها را تولید کردیم. در هنگام ساخت SelectListItem برای گرفتن Text، متد GetName را صدا زدیم. برای اینکه بتوانیم مقدار ویژگی Display که در هنگام تعریف نوع شمارشی استفاده کردیم را بدست بیاریم، باید چک کنیم ببینیم که آیا این آیتم به این ویژگی مزین شده‌است یا نه. اگر شده بود مقدار را می‌گیریم و به خصوصیت Text متد اول انتساب می‌دهیم.

```
@using Book.Entities
@using Book.Web.Helpers
@{
    var items = typeof(Genre).GetItems((int?)Model);
}

@Html.DropDownList("", items, new {@class="form-control"})
```

کدهایی که در بالا مشاهده می‌کنید کدهایی می‌باشند که قرار است داخل PartialViewی Genre قرار دهیم که در پوشه‌ی EditorTemplates ساختیم. ابتدا آمدیم آیتم‌ها را گرفتیم و بعد به DropDownList دادیم تا لیست نوع را برای ما بسازد. حالا اگه برنامه را اجرا کنید می‌بینید که EditorFor برای شما یه لیست از نوع شمارشی ساخته و حالا قابل استفاده هست.



کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید

HtmlHelpersEditor.rar