عنوان: CheckBoxList برای فیلد Enum Flags مدل در ASP.Net MVC
نویسنده: پژمان پارسائی
تاریخ: ۱۳۹۲/۰۷/۳۰ ۲۳:۰
آدرس: www.dotnettips.info
گروه‌ها: C#, ASP.Net MVC, Enum, ASP.Net MVC 4, MVC

قبلا مطالبی در سایت راجع به نوع داده شمارشی یا Enum و همچنین CheckBoxList و RadioButtonList وجود دارد. اما در این مطلب قصد دارم تا یک روش متفاوت را برای تولید و بهره گیری از CheckBoxList با استفاده از نوع داده‌های شمارشی برای شما ارائه کنم.
فرض کنید بخواهید به کاربر این امکان را بدهید تا بتواند چندین گزینه را برای یک فیلد انتخاب کند. به عنوان یک مثال ساده فرض کنید گزینه ای از مدل، پارچه‌های مورد علاقه یک نفر هست. کاربر می‌تواند چندین پارچه را انتخاب کند. و این فرض را هم بکنید که به لیست پارچه‌ها گزینه دیگری اضافه نخواهد شد. پارچه (Fabric) را مثلا می‌توانیم به صورت زیر تقسیم بندی کنیم :

پنبه (Cotton)
ابریشم (Silk)
پشم (Wool)
ابریشم مصنوعی (Rayon)
پارچه‌های دیگر (Other)

با توجه به اینکه دیگر قرار نیست به این لیست گزینه دیگری اضافه شود می‌توانیم آنرا به صورت یک نوع داده شمارشی (Enum) تعریف کنیم. مثلا بدین صورت:

```
public enum Fabric
{
    [Description("پنبه")]
    Cotton,

    [Description("ابریشم")]
    Silk,

    [Description("پشم")]
    Wool,

    [Description("ابریشم مصنوعی")]
    Rayon,

    [Description("پارچه‌های دیگر")]
    Other
}
```

حال فرض کنید View Model زیر فیلدی از نوع نوع داده شمارشی Fabric دارد:

```
public class MyViewModel
{
    public Fabric Fabric { get; set; }
}
```

توجه داشته باشید که فیلد Fabric از کلاس MyViewModel باید چند مقدار را در خود نگهداری کند. یعنی می‌تواند هر کدام از گزینه‌های Cotton، Silk، Wool، Rayon، Other به صورت جداگانه یا ترکیبی باشد. اما در حال حاضر با توجه به اینکه یک فیلد Enum معمولی فقط می‌تواند یک مقدار را در خودش ذخیره کند قابلیت ذخیره ترکیبی مقادیر در فیلد Fabric از View Model بالا وجود ندارد.

اما راه حل این مشکل استفاده از پرچم (Flags) در تعریف نوع داده شمارشی هست. با استفاده از پرچم نوع داده شمارشی بالا به صورت زیر باید تعریف شود:

```
[Flags]
public enum Fabric
```

```
{
    [Description("پنبه")]
    Cotton = 1,

    [Description("ابریشم")]
    Silk = 2,

    [Description("پشم")]
    Wool = 4,

    [Description("ابریشم مصنوعی")]
    Rayon = 8,

    [Description("پارچه‌های دیگر")]
    Other = 128
}
```

همان طور که می‌بینید از عبارت [Flags] قبل از تعریف enum استفاده کرده ایم. همچنین هر کدام از مقادیر ممکن این نوع داده شمارشی با توانهایی از 2 تنظیم شده اند. در این صورت یک نمونه از این نوع داده می‌تواند چندین مقدار را در خودش ذخیره کند.

برای آشنایی بیشتر با این موضوع به کدهای زیر نگاه کنید:

```
Fabric cotWool = Fabric.Cotton | Fabric.Wool;
int cotWoolValue = (int) cotWool;
```

به وسیله عملگر | می‌توان چندین مقدار را در یک نمونه از نوع Fabric ذخیره کرد. مثلا متغیر cotWool هم دارای مقدار Fabric.Cotton و هم دارای مقدار Fabric.Wool هست. مقدار عددی معادل متغیر cotWool برابر 5 هست که از جمع مقدار عددی Fabric.Cotton و Fabric.Wool به دست آمده است.

حال فرض کنید فیلد Fabric از View Model ذکر شده (کلاس MyViewModel) را به صورت لیستی از چک باکس‌ها نمایش دهیم. مثل زیر:

شکل (الف)

سپس بخواهیم تا کاربر بعد از انتخاب گزینه‌های مورد نظرش از لیست بالا و پست کردن فرم مورد نظر، بایندر وارد عمل شده و فیلد Fabric را بر اساس گزینه هایی که کاربر انتخاب کرده مقداردهی کند.

برای این کار از پروژه MVC Enum Flags کمک خواهیم گرفت. این پروژه شامل یک Html Helper برای تبدیل یه Enum به یک CheckBoxList و همچنین شامل Model Binder مربوطه هست. البته بعضی از کدهای Html Helper آن احتیاج به تغییر داشت که

آنرا انجام دادم ولی بایندر آن بسیار خوب کار می‌کند.

خوب html helper مربوط به آن به صورت زیر می‌باشد:

```
public static IHtmlString CheckBoxesForEnumFlagsFor<TModel, TEnum>(this HtmlHelper<TModel> htmlHelper,
Expression<Func<TModel, TEnum>> expression)
{
    ModelMetadata metadata = ModelMetadata.FromLambdaExpression(expression, htmlHelper.ViewData);
    Type enumModelType = metadata.ModelType;

    // Check to make sure this is an enum.
    if (!enumModelType.IsEnum)
    {
        throw new ArgumentException("This helper can only be used with enums. Type used was: " +
enumModelType.FullName.ToString() + ".");
    }

    // Create string for Element.
    var sb = new StringBuilder();

    foreach (Enum item in Enum.GetValues(enumModelType))
    {
        if (Convert.ToInt32(item) != 0)
        {
            var ti = htmlHelper.ViewData.TemplateInfo;
            var id = ti.GetFullHtmlFieldId(item.ToString());

            //Derive property name for checkbox name
            var body = expression.Body as MemberExpression;
            var propertyName = body.Member.Name;
            var name = ti.GetFullHtmlFieldName(propertyName);

            //Get currently select values from the ViewData model
            TEnum selectedValues = expression.Compile().Invoke(htmlHelper.ViewData.Model);

            var label = new TagBuilder("label");
            label.Attributes["for"] = id;
            label.Attributes["style"] = "display: inline-block;";
            var field = item.GetType().GetField(item.ToString());

            // Add checkbox.
            var checkbox = new TagBuilder("input");
            checkbox.Attributes["id"] = id;
            checkbox.Attributes["name"] = name;
            checkbox.Attributes["type"] = "checkbox";
            checkbox.Attributes["value"] = item.ToString();

            if ((selectedValues as Enum != null) && ((selectedValues as Enum).HasFlag(item)))
            {
                checkbox.Attributes["checked"] = "checked";
            }
            sb.AppendLine(checkbox.ToString());

            // Check to see if DisplayName attribute has been set for item.
            var displayName = field.GetCustomAttributes(typeof(DisplayNameAttribute), true)
                .FirstOrDefault() as DisplayNameAttribute;
            if (displayName != null)
            {
                // Display name specified.  Use it.
                label.SetInnerText(displayName.DisplayName);
            }
            else
            {
                // Check to see if Display attribute has been set for item.
                var display = field.GetCustomAttributes(typeof(DisplayAttribute), true)
                    .FirstOrDefault() as DisplayAttribute;
                if (display != null)
                {
                    label.SetInnerText(display.Name);
                }
                else
                {
                    label.SetInnerText(item.ToDescription());
                }
            }
            sb.AppendLine(label.ToString());

            // Add line break.
            sb.AppendLine("<br />");
```

```
        }
    }

    return new HtmlString(sb.ToString());
}
```

در کدهای بالا از متد الحاقی ToDescription نیز برای تبدیل معادل انگلیسی به فارسی یک مقدار از نوع داده شمارشی استفاده کرده ایم.

```
public static string ToDescription(this Enum value)
{
    var attributes =
(DescriptionAttribute[])value.GetType().GetField(value.ToString()).GetCustomAttributes(typeof(Descripti
onAttribute), false);
    return attributes.Length > 0 ? attributes[0].Description : value.ToString();
}
```

برای استفاده از این Html Helper در View کد زیر را می‌نویسیم:

```
@Html.CheckBoxesForEnumFlagsFor(x => x.Fabric)
```

که باعث تولید خروجی که در تصویر (الف) نشان داده شد می‌شود. و همچنین مدل بایندر مربوط به آن به صورت زیر هست:

```
public class FlagEnumerationModelBinder : DefaultModelBinder
{
    public override object BindModel(ControllerContext controllerContext, ModelBindingContext
bindingContext)
    {
        if (bindingContext == null) throw new ArgumentNullException("bindingContext");

        if (bindingContext.ValueProvider.ContainsPrefix(bindingContext.ModelName))
        {
            var values = GetValue<string[]>(bindingContext, bindingContext.ModelName);

            if (values.Length > 1 && (bindingContext.ModelType.IsEnum &&
bindingContext.ModelType.IsDefined(typeof(FlagsAttribute), false)))
            {
                long byteValue = 0;
                foreach (var value in values.Where(v => Enum.IsDefined(bindingContext.ModelType, v)))
                {
                    byteValue |= (int)Enum.Parse(bindingContext.ModelType, value);
                }

                return Enum.Parse(bindingContext.ModelType, byteValue.ToString());
            }
            else
            {
                return base.BindModel(controllerContext, bindingContext);
            }
        }

        return base.BindModel(controllerContext, bindingContext);
    }

    private static T GetValue<T>(ModelBindingContext bindingContext, string key)
    {
        if (bindingContext.ValueProvider.ContainsPrefix(key))
        {
            ValueProviderResult valueResult = bindingContext.ValueProvider.GetValue(key);
            if (valueResult != null)
            {
                bindingContext.ModelState.SetModelValue(key, valueResult);
                return (T)valueResult.ConvertTo(typeof(T));
            }
        }
        return default(T);
    }
}
```

این مدل بایندر را باید به این صورت در متد Application_Start فایل Global.asax فراخوانی کنیم:

```
ModelBinders.Binders.Add(typeof(Fabric), new FlagEnumerationModelBinder());
```

مشاهده می‌کنید که در اینجا دقیقا مشخص کرده ایم که این مدل بایندر برای نوع داده شمارشی Fabric هست. اگر نیاز دارید تا این بایندر برای نوع داده‌های شمارشی دیگری نیز به کار رود نیاز هست تا این خط کد را برای هر کدام از آنها تکرار کنید. اما راه حل بهتر این هست که کلاسی به صورت زیر تعریف کنیم و تمامی نوع داده‌های شمارشی که باید از بایندر بالا استفاده کنند را در یک پراپرتی آن برگشت دهیم. مثلا بدین صورت:

```
public class ModelEnums
{
    public static IEnumerable<Type> Types
    {
        get
        {
            var types = new List<Type> { typeof(Fabric) };
            return types;
        }
    }
}
```

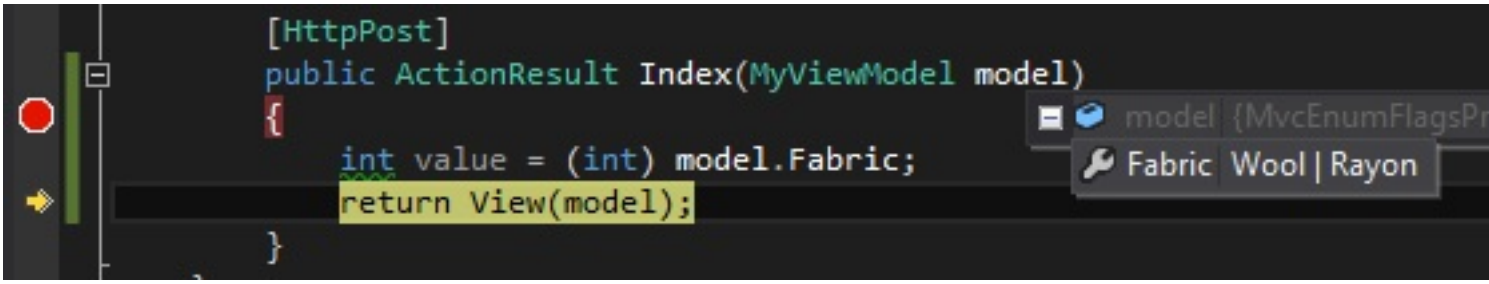
سپس به متد Application_Start رفته و کد زیر را اضافه می‌کنیم:

```
foreach (var type in ModelEnums.Types)
{
   ModelBinders.Binders.Add(type, new FlagEnumerationModelBinder())
}
```

اگر گزینه‌های پشم و ابریشم مصنوعی را از CheckBoxList تولید شده انتخاب کنیم، بدین صورت:

شکل (ب)

و سپس فرم را پست کنید، موردی شبیه زیر مشاهده می‌کنید:



شکل (ج)

همچنین مقدار عددی معادل در این جا برابر 12 می‌باشد که از جمع دو مقدار Wool و Rayon به دست آمده است. بدین ترتیب در یک فیلد از مدل، گزینه‌های انتخابی توسط کاربر قرار گرفته شده اند.

پروژه مربوط به این مثال را از لینک زیر دریافت کنید:
MvcEnumFlagsProjectSample.zip

پی نوشت : پوشه‌های bin و obj و packages جهت کاهش حجم پروژه از آن حذف شده اند. برای بازسازی پوشه packages لطفا به مطلب بازسازی کامل پوشه packages بسته‌های NuGet به صورت خودکار مراجعه کنید.

نظرات خوانندگان

نویسنده: علی
تاریخ: ۱۳۹۲/۰۸/۰۱ ۱۸:۵۱

من یک گرید تلریک دارم که یک فیلد چکباکس کنارش هست و مثلا لیستی از یوزرها رو این گرید شامل میشه .
آیتم‌های انتخابی رو چطوری میتونم به کنترلرم ارسال کنم (ورودی کنترلرمو چی بگیرم)...

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۰۱ ۱۸:۵۸

سؤال شما بیشتر به مطلب « CheckBoxList در ASP.NET MVC » مرتبط است تا enum ایی که ویژگی flag دارد. ضمنا گرید تلریک مستندات خوبی دارد که بهتر است به آن مراجعه کنید (مثال Ajax CheckBoxes هست که کدهای View و کنترلر آن نیز پیوست شدند).

نویسنده: علیرضا
تاریخ: ۱۳۹۲/۰۸/۰۲ ۱۲:۱۵

استفاده از Description برای Enumها جالب ولی به نظر من کمی مشکل سازه و مشکل اصلی اون Code Wired کردن شرح هر جزء Enum هست. اگر با یک پروژه دوزبانه طرف باشیم چی؟ اگر کلا استراتژی ترجمه رشته‌ها در پروژه ما این طور باشه که از منابع خارجی مثل DB یا اسمبلی‌های Resource استفاده کنیم چی؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۰۲ ۱۲:۳۲

مراجعه کنید به پیشنیازهای این مباحث تکمیلی. مانند:
تهیه سایت‌های چند زبانه و بومی سازی نمایش اطلاعات در ASP.NET MVC
Globalization در ASP.NET MVC
اولی در مورد کار با ریسورس‌ها است و بومی سازی ویژگی‌ها نیز در آن لحاظ شده و دومی تهیه یک فریم ورک است برای کار با بانک اطلاعاتی و تامین منبع داده از این طریق