عنوان: **آشنایی با مفاهیم نوع داده Enum و توسعه آن - قسمت یکم**
نویسنده: علیرضا اسم‌رام
تاریخ: ۱۳۹۱/۰۵/۰۲ ۰:۸
آدرس: www.dotnettips.info
برچسب‌ها: C#, Enum

نوع داده شمارشی یا Enum، جهت تعاریف مقادیر ثابت و قابل شمارش در برنامه، بسیار کاربرد دارد. مقادیری که در این نوع داده تعریف می‌شوند بطور خودکار از عدد 0 شماره گذاری می‌شوند و به ترتیب یکی به آن‌ها اضافه می‌شود. برای مثال حالت زیر را در نظر بگیرید:

```
public enum Grade
    {
        Failing,        // = 0
        BelowAverage,   // = 1
        Average,        // = 2
        VeryGood,       // = 3
        Excellent       // = 4
    }
```

در این حالت متد ToString() نوع داده Enum عنوان مقادیر ثابت را بر می‌گرداند.

جهت برگشت مقدار عددی و شماره مقادیر ثابت‌های تعریف شده از متد ToString() با فرمت D (شماره مقدار را بصورت Decimal نشان می‌دهد) و فرمت X جهت نمایش بصورت هگزا می‌توان استفاده کرد.

روش عرف برای نمایش مقدار عددی استفاده از تبدیل نوع صریح به int است.

به منظور درک بهتر موضوع، از یک برنامه کنسول استفاده می‌کنیم تا این نوع داده شمارشی را در آن استفاده کنیم.

```
static void Main(string[] args)
{
    Grade grade = Grade.Average;
    Console.WriteLine(grade.ToString());    // Print Avarage
    Console.WriteLine(grade.ToString("D")); // Print 2
    Console.WriteLine(grade.ToString("X")); // Print 00000002
    Console.WriteLine((int) grade);         //Print 2
    Console.ReadKey();
}
```

تغییر شماره (اندیس) مقادیر ثابت تعریف شده:

جهت تغییر شماره مقادیر کافیست بصورت زیر عمل کنیم:

```
public enum Grade
{
    Failing = 5,
    BelowAverage = 10,
    Average = BelowAverage + 5, // = 15
    VeryGood = 18,
    Excellent = 20
}
```

همانطور که در بالا می‌بینید برای مقدار Average بصورت ترکیبی عمل شده است.

بصورت پیش فرض کامپایلر سی شارپ از Int32 جهت نگهداری اعضای یک Enum استفاده می‌کند. هر چند غیر معقول به نظر می‌رسد اما شما می‌توانید این نوع را به byte - sbyte - short - ushort - uint - long تغییر دهید.

```
public enum Grade : byte
{
    Failing = 5,
    BelowAverage = 10,
    Average = BelowAverage + 5, // = 15
    VeryGood = 18,
    Excellent = 20
```

```
}
```

بدیهی است در این حالت خروجی دستور زیر 0F خواهد بود:

```
Console.WriteLine(grade.ToString("X")); // Print 0F
```

همچنین به خروجی دستورات زیر در حالت فوق توجه کنید:

```
Console.WriteLine("Underlying type: {0}", Enum.GetUnderlyingType(grade.GetType())); // Print
System.Byte

Console.WriteLine("Type Code      : {0}", grade.GetTypeCode()); // Print Byte
```

و البته این:

```
Console.WriteLine("Value : {0}", (int)grade); // Print 15
```

در قسمت دوم این مطلب با استفاده از فضای نام System.Reflection و Extension Methodها و Custom Attribute کمی مقادیر Enum را توسعه خواهیم داد.

نظرات خوانندگان

نویسنده: احمد احمدی
تاریخ: ۱۳۹۱/۰۵/۰۲ ۲:۴۶

سلام - بحث Enum خیلی بحث جالبی است .
فقط فراموش کردید برچسب Enum را هم به این قسمت از مقالتون اضافه کنید .
با تشکر ...

نویسنده: علیرضا اسم‌رام
تاریخ: ۱۳۹۱/۰۵/۰۲ ۹:۳۲

سلام. با تشکر از شما. این موضوع را در قسمت سوم و باز هم با کمک متدهای الحاقی اجرا می‌کنیم. در نهایت امیدوارم یک کلاس از متدهای الحاقی جهت کار با Enumها داشته باشیم.
ممنون از یادآوری شما.

نویسنده: KishIsland
تاریخ: ۱۳۹۱/۱۲/۲۷ ۱۵:۲۰

سپاس.مفید بود

عنوان: **آشنایی با مفاهیم نوع داده Enum و توسعه آن - قسمت دوم**
نویسنده: علیرضا اسم‌رام
تاریخ: ۱۳۹۱/۰۵/۰۲ ۰:۳۲
آدرس: www.dotnettips.info
برچسب‌ها: C#, Extension Method, Enum

اگر با نوع داده Enum آشنایی ندارید قسمت یکم این مطلب را بخوانید.

```
public enum Grade
{
    Failing = 5,
    BelowAverage = 10,
    Average = BelowAverage + 5, // = 15
    VeryGood = 18,
    Excellent = 20
}
```

بازنویسی متد ToString(): امکان بازنویسی متد ()ToString در نوع Enum وجود ندارد. بنابراین برای چاپ عبارت Very Good به جای VeryGood تکنیک زیر جالب به نظر می‌رسد. هر چند استفاده از آرایه و ترکیب اندیس آن با Enum و یا استفاده از HashTable راه هایی است که در ابتدا به ذهن ما خطور می‌کند اما لطفاً به ادامه مطلب توجه فرمایید!
با در نظر گرفتن مثال قبل، یک Custom Attribute به نوع داده شمارشی اضافه می‌کنیم. برای این منظور بصورت زیر عمل می‌کنیم.

1. ایجاد کلاس Description که از کلاس Attribute مشتق شده است و تعریف خصوصیت Text:

```
class Description : Attribute
{
    public string Text;
    public Description(string text)
    {
        Text = text;
    }
}
```

2. به سراغ نوع Enum تعریف شده رفته و جهت استفاده از صفت جدید که در مرحله قبل پیاده سازی کردیم، تغییرات را به شکل زیر اعمال می‌کنیم:

```
public enum Grade
{
    [Description("Mardood")]
    Failing = 5,

    [Description("Ajab Shansi")]
    BelowAverage = 10,

    [Description("Bad Nabood")]
    Average = BelowAverage + 5,

    [Description("Khoob Bood")]
    VeryGood = 18,

    [Description("Gol Kashti")]
    Excellent = 20
}
```

تنها کاری که باقی مانده یاری گرفتن از متدهای الحاقی (Extension Methods) جهت خواندن مقدار Description است:

```
public static class ExtensionMethodCls
{
    public static string GetDescription(this Enum enu)
    {

        Type type = enu.GetType();
```

```
        MemberInfo[] memInfo = type.GetMember(enu.ToString());

        if (memInfo != null && memInfo.Length > 0)
        {

            object[] attrs = memInfo[0].GetCustomAttributes(typeof(Description), false);

            if (attrs != null && attrs.Length > 0)
                return ((Description)attrs[0]).Text;
        }

        return enu.ToString();
    }
}
```

حال نوع Enum ما کمی توسعه یافته است و توسط متد GetDescription می توان متن دلخواه و متناسب با مقدار را نمایش داد:

```
Console.WriteLine(grade.GetDescription());  // Print Bad Nabood
```

کد کامل مثال بررسی شده نیز بصورت زیر خواهد بود:

```
using System;
using System.Reflection;

namespace CSharpEnum
{

    class Description : Attribute
    {
        public string Text;
        public Description(string text)
        {
            Text = text;
        }
    }

    public enum Grade
    {
        [Description("Mardood")]
        Failing = 5,

        [Description("Ajab Shansi")]
        BelowAverage = 10,

        [Description("Bad Nabood")]
        Average = BelowAverage + 5,

        [Description("Khoob Bood")]
        VeryGood = 18,

        [Description("Gol Kashti")]
        Excellent = 20
    }

    public static class ExtensionMethodCls
    {
        public static string GetDescription(this Enum enu)
        {

            Type type = enu.GetType();

            MemberInfo[] memInfo = type.GetMember(enu.ToString());

            if (memInfo != null && memInfo.Length > 0)
            {

                object[] attrs = memInfo[0].GetCustomAttributes(typeof(Description), false);

                if (attrs != null && attrs.Length > 0)
                    return ((Description)attrs[0]).Text;
            }

            return enu.ToString();
        }
```

```
    }

    class Program
    {
        static void Main(string[] args)
        {
            const Grade grade = Grade.Average;
            Console.WriteLine("Underlying type: {0}", Enum.GetUnderlyingType(grade.GetType()));
            Console.WriteLine("Type Code      : {0}", grade.GetTypeCode());
            Console.WriteLine("Value          : {0}", (int)grade);
            Console.WriteLine("-------------------------------------");
            Console.WriteLine(grade.ToString()); // name of the constant
            Console.WriteLine(grade.ToString("G")); // name of the constant
            Console.WriteLine(grade.ToString("F")); // name of the constant
            Console.WriteLine(grade.ToString("x")); // value is hex
            Console.WriteLine(grade.ToString("D")); // value in decimal
            Console.WriteLine("-------------------------------------");
            Console.WriteLine(grade.GetDescription());  // Print Bad Nabood
            Console.ReadKey();
        }
    }
}
```

با استفاده از این تکنیک (مخصوصاً ما فارسی زبان ها) به راحتی می‌توانیم از مقادیر Enum استفاده بهتری ببریم. برای مثال اگر بخواهیم یک مقدار Enum را بصورت فارسی در یک Drop Down List نمایش دهیم این تکنیک بسیار مفید خواهد بود.

نظرات خوانندگان

نویسنده: حسین
تاریخ: ۱۳۹۱/۰۵/۰۲ ۱۲:۱۹

سلام
کدهایی که به این صورت نوشته میشن چی بهشون گفته میشه؟ چی هستن؟
خیلی جاها این کدها رو دیدم به صورت‌های مختلف
کمی تو گوگل سرچ کردم نتیجه مطلوبی نگرفتم یعنی نمیدونم دنبال چی بگردم
لطفا راهنمایی کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۰۲ ۱۲:۲۳

به Attribute ایی که در اینجا توسعه داده شده (یا از آن استفاده شده)، اصطلاحا data annotation هم گفته می‌شود. یک سری از فریم ورک‌ها به صورت توکار قادر به استفاده از آن‌ها هستند مانند ASP.NET MVC برای نمایش توضیحات مرتبط یا نمایش برچسب‌ها به صورت خودکار.
مطالب فوق رو می‌تونید پایه طراحی این نوع کتابخانه‌ها درنظر بگیرید.

نویسنده: حسین
تاریخ: ۱۳۹۱/۰۵/۰۲ ۱۲:۳۰

کدهایی که تو linq هم استفاده میشن از همین دسته اند؟
مثلا

```
[global::System.Data.Linq.Mapping.ColumnAttribute(Storage="_ID", AutoSync=AutoSync.Always, DbType="Int
NOT NULL IDENTITY", IsDbGenerated=true)]
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۰۲ ۱۲:۴۶

بله. به این‌ها Attribute، meta-data و یا data annotation گفته می‌شود.

نویسنده: ایمان محمدی
تاریخ: ۱۳۹۱/۰۵/۰۶ ۱۴:۹

نیازی به تعریف کلاس Description نیست.
کافیه از فضای نام System.ComponentModel استفاده کنید و در مقدار بازگشتی متد GetDescription بجای

```
return ((Description)attrs[0]).Text;
```

بنوسید

```
return ((DescriptionAttribute)attrs[0]).Description;
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۰۶ ۱۴:۲۰

هدف این مطلب بررسی زیر ساخت این نوع طراحی بوده نه صرفا مصرف کننده محض بودن. تشابه اسمی اهمیتی نداره. روش انجام کار مهم بوده در اینجا.

نویسنده: ایمان محمدی
تاریخ: ۱۳۹۱/۰۵/۰۶ ۱۶:۵

وقتی کلاس Description در فضای نام System.ComponentModel وجود داره دلیلی نداره کلاس مشابه ای تعریف کنیم.
بخاطر اینکه مصرف کننده محض نباشیم یک متد الحاقی به نام GetEnumList() اضافه کردم که لیست اعضای یک Enum رو برای استفاده در کمبو باکس و ... بر می‌گرودنه :
ابتدا کلاس زیر به کلاس ExtensionMethodCls اضافه می‌کنیم :

```
public class EnumObject
        {
            public Enum ValueMember { get; set; }
            public int intValueMember
            {
                get { return int.Parse(ValueMember.ToString("D")); }
            }
            public string stringValueMember
            {
                get { return ValueMember.ToString(""); }
            }
            public string DisplayMember
            {
                get { return ValueMember.GetDescription(); }
            }
        }
```

و متد الحاقی زیر رو برای گرفتن لیست تعریف می‌کنیم:

```
public static List<EnumObject> GetEnumList(this Enum enu)
        {
            List<EnumObject> li = new List<EnumObject>();
            foreach (var item in enu.GetType().GetEnumValues())
            {
                li.Add(new EnumObject { ValueMember = (Enum)item });
            }
            return li;
        }
```

نحوه استفاده :

```
comboBox1.DataSource = Grade.VeryGood.GetEnumList();
          comboBox1.DisplayMember = "DisplayMember";
          comboBox1.ValueMember = "ValueMember";
```

همون طوری که در کد بالا می‌بینید برای گرفتن لیست مجبور شدیم یکی از اعضای enum رو انتخاب کنیم (Grade. **VeryGood** .GetEnumList()) شاید انتخاب یکی از اعضا و بعد درخواست لیست اعضا رو کردن کار قشنگی نباشه به همین دلیل متد زیر رو تعریف کردیم :

```
public static List<EnumObject> EnumToList<T>()
        {
            Type enumType = typeof(T);
            if (enumType.BaseType != typeof(Enum))
                throw new ArgumentException("T must be of type System.Enum");

            List<EnumObject> li = new List<EnumObject>();
            foreach (var item in enumType.GetEnumValues())
            {
                li.Add(new EnumObject { ValueMember = (Enum)item });
            }
```

```
            return li;
        }
```

نحوه استفاده :

```
comboBox1.DataSource =ExtensionMethodCls.EnumToList<Grade>();
          comboBox1.DisplayMember = "DisplayMember";
          comboBox1.ValueMember = "ValueMember";
```

کد کامل :

```
public static class ExtensionMethodCls
    {
        public class EnumObject
        {
            public Enum ValueMember { get; set; }
            public int intValueMember
            {
                get { return int.Parse(ValueMember.ToString("D")); }
            }
            public string stringValueMember
            {
                get { return ValueMember.ToString(""); }
            }
            public string DisplayMember
            {
                get { return ValueMember.GetDescription(); }
            }
        }



        public static List<EnumObject> EnumToList<T>()
        {
            Type enumType = typeof(T);
            if (enumType.BaseType != typeof(Enum))
                throw new ArgumentException("T must be of type System.Enum");

            List<EnumObject> li = new List<EnumObject>();
            foreach (var item in enumType.GetEnumValues())
            {
                li.Add(new EnumObject { ValueMember = (Enum)item });
            }
            return li;
        }

        public static List<EnumObject> GetEnumList(this Enum enu)
        {
            List<EnumObject> li = new List<EnumObject>();
            foreach (var item in enu.GetType().GetEnumValues())
            {
                li.Add(new EnumObject { ValueMember = (Enum)item });
            }
            return li;
        }

        public static string GetDescription(this Enum enu)
        {

            Type type = enu.GetType();

            MemberInfo[] memInfo = type.GetMember(enu.ToString());

            if (memInfo != null && memInfo.Length > 0)
            {

                object[] attrs = memInfo[0].GetCustomAttributes(typeof(DescriptionAttribute), false);

                if (attrs != null && attrs.Length > 0)
                    return ((DescriptionAttribute)attrs[0]).Description;
            }

            return enu.ToString();

        }
```

```
    }
```

نویسنده: علیرضا اسم‌رام
تاریخ: ۱۳۹۱/۰۵/۰۷ ۹:۴۵

با سلام خدمت شما دوست عزیز. همانطور که آقای نصیری اشاره کردند هدف این مطلب و البته این سری از مطالب آشنایی قدم به قدم با مفاهیم این نوع داده بوده است.
انقیاد کنترل‌ها با اشیاء از نوع Enum موضوع بعدی این سری از مطالب بود. البته باز هم از شما تشکر می‌کنم بخاطر این نظر.
اگر این قابلیت در سایت ایجاد شود که بتوان به نظرات لینک داد بهتر است.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۰۷ ۱۰:۰

علامت آبی رنگ # کنار هر نظر، لینک مستقیم به همان نظر است.

نویسنده: علیرضا اسم‌رام
تاریخ: ۱۳۹۱/۰۵/۰۷ ۲۲:۸

مرسی. خیلی خیلی خوب...

نویسنده: سعید یزدانی
تاریخ: ۱۳۹۱/۱۱/۱۷ ۲۰:۵۴

باسلام؛ لطفا یه توضیحی در باره‌ی MemberInfo بدید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۱۸ ۹:۱۱

مراجعه کنید به MSDN و مثال‌های آن در همان صفحه.

عنوان: **پر کردن مقادیر Enum در کلاینت**
نویسنده: مهدی پایروند
تاریخ: ۱۳۹۲/۰۲/۱۱ ۱۲:۱۵
آدرس: www.dotnettips.info
برچسب‌ها: JSON, jQuery, jQuery-Tips, Enum, dropdown

در برنامه‌های وب امروز نیازی به فراخوانی ثوابت که در طول حیات برنامه انگشت شمار تغیر میکنند نیست و با توجه به استفاده از فرامین و متدهای سمت کلاینت احتیاج هست تا این ثوابت بار اول لود صفحه به کلاینت پاس داده شوند.

میتوان در این گونه موارد از قابلیت‌های گوناگونی استفاده کرد که در اینجا ما با استفاده از یک فیلد مخفی و json مقدار را به کلاینت پاس میدهیم و در این مثال در سمت کلاینت نیز دراپ دان را با این مقادیر پر میکنیم:

```
public enum PersistType
    {
        Persistable = 1,
        NotPersist = 2,
        AlwaysPersist = 3
    }
```

لیست را باید قبل از پر کردن در فیلد مخفی به json بصورت serialize شده تبدیل کرد، برای این منظور از JavaScriptSerializer موجود در اسمبلی‌های دات نت در متد زیر استفاده شده:

```
public static string ConvertEnumToJavascript(Type t)
        {
            if (!t.IsEnum) throw new Exception("Type must be an enumeration");

            var values = System.Enum.GetValues(t);
            var dict = new Dictionary<int, string>();

            foreach (object obj in values)
            {
                string name = System.Enum.GetName(t, obj);
                dict.Add(Convert.ToInt32(System.Enum.Format(t, obj, "D")), name);
            }

            return new JavaScriptSerializer().Serialize(dict);
        }
```

با توجه به اینکه در سمت کلاینت مقدار json ذخیره شده در فیلد مخفی را میتوان به صورت آبجکت برخورد کرد پس یک متد در سمت کلاینت این آبجکت را در loop قراد داده و درمتغییری در فایل جاوا اسکریپت نگهداری میکنیم:

```
var Enum_PersistType = null;

function SetEnumTypes() {
    Enum_PersistType = JSON.parse($('#hfJsonEnum_PersistType').val());
}
```

و در هر قسمت که نیاز به مقدار enum بود با توجه به ایندکس مقدار را برای نمایش ازاین متغییر بیرون میکشیم:

```
function GetPersistTypeTitle_Concept(enumId) {
    return Enum_PersistType[enumId];
}
```

برای مثال در dropdown در سمت کلاینت این نوع استفاده شده و در حالتی از صفحه فقط برای نمایش عنوان آن احتیاج به دریافت آن از سمت سرور باشد میتوان از این روش کمک گرفت

و یا در سمت javascript میتوان با استفاده از jQuery مقادیر متغییر را در dropdown پر کرد.

```
function FillDropdown() {
    $("#ddlPersistType").html("");
    $.each(Enum_PersistType, function (key, value) {
        $("#ddlPersistType").append($("<option></option>").val(key).html(value));
    });
}
```

FillDropdownListOnClient.zip

نظرات خوانندگان

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۱۲ ۱۱:۵۴

با تشکر از شما.

فایل Newtonsoft.Json.dll در پروژه شما هست. JavaScriptSerializer توکار دات نت ازش استفاده نمی‌کنه. فقط از اسمبلی System.Web.Extensions.dll هست که استفاده می‌کنه.

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۲/۰۲/۱۲ ۱۱:۵۷

ممنون از شما، برای ادامه این سری لازم میشده که در آینده اضافه میشه.
شما میتونید در صورت دلخواه کد قسمت serialize رو با این کتابخانه بنویسید:

```
//return new JavaScriptSerializer().Serialize(dict);
            return Newtonsoft.Json.JsonConvert.SerializeObject(dict);
```

نویسنده: سام ناصری
تاریخ: ۱۳۹۲/۰۲/۱۲ ۱۳:۴۴

به نظر من موضوع رو خیلی پیچیده کردی. برای تولید json لازم نیست که از کتابخانه خاصی استفاده کنی با همون استرینگ منیوپولیشن ساده هم میشه این کار را کرد:

```
public static class EnumHelper
{
    public static Dictionary<string,EnumValueType> ToDictionary<EnumType,EnumValueType>()
    {
        return
Enum.GetValues(typeof(EnumType)).Cast<EnumValueType>().ToDictionary(i=>Enum.GetName(typeof(EnumType),i)
,i=>i);
    }
    public static string ToJson<EnumType>()
    {
        return "{" + string.Join(",",
ToDictionary<EnumType,int>().Select(i=>string.Format(@"{{""{0}"":{1}}}",i.Key,i.Value)).ToArray())+ "}"
;
    }
}
```

کلاس ساده بالا به سادگی json مورد نیاز را تولید میکند.
در ضمن برای اینجکت کردنش به صفحه هم لازم نیست که از فیلد مخفی استفاده کنی. به جاش json را مستقیم در محل مورد نظر رندر کن:
در Asp.Net MVC Razor:

```
var x = @EnumHelper.ToJson<MyEnum()
```

در Asp.Net Web Forms :

```
var x = <%=EnumHelper.ToJson<MyEnum>()%>
```

همچنین همانطور که در مثالهای فوق نشان داده ام حتی لازم نیست از JSON.Parse استفاده کنی.
البته من اینها را بر اساس ذهنیاتم خیلی سریع نوشتم و کدهای فوق را تست نکرده ام که ببینم درست کار میکنند یا نه. اما منظورم این بود که بپرسم چرا از فیلد مخفی استفاده کردی و چرا از JSON.Parse استفاده کردی و اینکه چرا از JavaScriptSerializer استفاده کردی؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۱۲ ۱۴:۲۶

در حالت کلی بهتره که از JavaScriptSerializer استفاده بشه چون می‌تونه یک سری escape حروف خاص رو لحاظ کنه.

نویسنده: سام ناصری
تاریخ: ۱۳۹۲/۰۲/۱۲ ۱۵:۶

موضوع این مقاله درباره Enum است و نه ارسال داده‌های کلی به کلاینت. پس آیا فکر میکنید در اینجا چیزی برای اسکیپ شدن وجود داره؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۱۲ ۱۶:۳۲

در مورد پیچیدگی صحبت کردید. راه شما به مراتب پیچیده‌تر است از روش مطرح شده و خوانایی کمتری داره. به علاوه هدف از ارائه مقالات بهتره ارائه راه حل‌هایی باشه تا حد امکان عمومی تا این که یک سری هک خاص مطرح بشه فقط مختص به یک روش خاص که فقط در یک مساله مشخص قابل استفاده باشه. بعد هم اگر کسی این هک رو جای دیگری استفاده کرد، چون نمی‌دونه یک سری از کاراکترها باید escape بشن، در ضمن کار گیر میفته. دید دادن برای حل مساله اینجا شاید بیشتر مطرح باشه تا حل مساله با یک هک ساده که فقط همینجا قابل استفاده است. همچنین زمانیکه یک سری متد تست شده داخل فریم ورک هست چرا باید رفت سراغ هک؟

ضمنا در ASP.NET MVC نیاز دارید که یک Html.Raw رو هم اضافه کنید و گرنه اطلاعات درج شده در صفحه encode می‌شن و در متغیر جاوا اسکریپتی قابل استفاده نخواهند بود.

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۲/۰۲/۱۲ ۲۳:۸

در موردی مطلبی که آقای ناصری فرمودند باید بگم زمانیکه برنامه به سمت چند زبانه میره اهمبت پیدا میکنه که میتونید برای مثال مقدار یا لیستی از مقادیر متنی برای زبان خاصی رو با resource خودش به فیلد مخفی پاس بدید و در نمایش پیغام‌های مختلف سمت کلاینت اسنفاده کنید. مثل متن پیغام هایی که خاص ارتباط ajax میباشد که به زبان‌های مختلف ارائه کرد.

نویسنده: سام ناصری
تاریخ: ۱۳۹۲/۰۲/۱۳ ۴:۳۳

من کلاً نمیفهمم. در ضمن من سه تا سوال مطرح کردم(پاراگراف آخر کامنتم) که من باز هم نمیفهمم این جواب کدومشونه.

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;
        }
    }
}
```

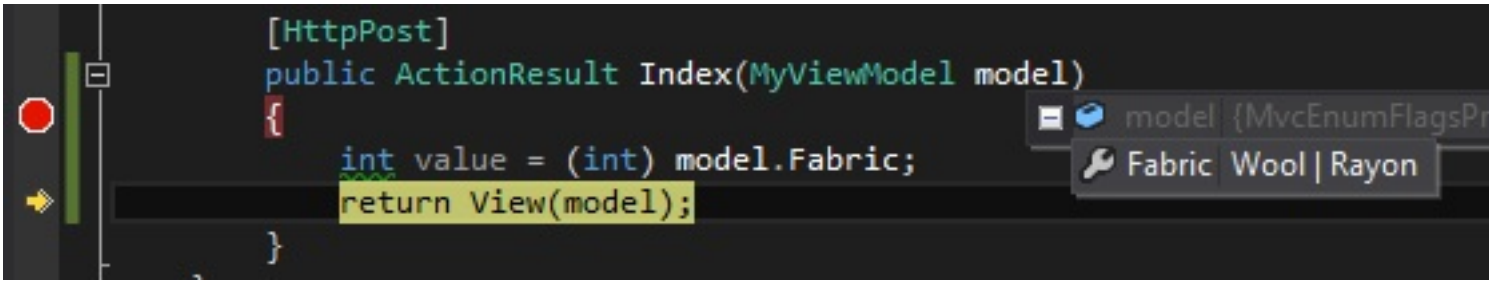
سپس به متد Application_Start رفته و کد زیر را اضافه می‌کنیم:

```
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

عنوان: **سفارشی سازی Binding یک خصوصیت از طریق Attributes**
نویسنده: محمد رعیت پیشه
تاریخ: ۱۳۹۲/۰۸/۲۵ ۲۲:۰
آدرس: www.dotnettips.info
برچسب‌ها: MVC, Enum, custom model binder

اگر با MVC کار کرده باشید حتما با ModelBinding آن آشنا هستید؛ DefaultModelBinder توکار آن که در اکثر مواقع، باری زیادی را از روی دوش برنامه نویسان بر می‌دارد و کار را برای آنان راحتتر می‌کند. اما در بعضی مواقع این مدل بایندر پیش فرض ممکن است پاسخگوی نیاز ما در بایند کردن یک خصوصیت از یک مدل خاص نباشد، برای همین ما نیاز داریم که کمی آن را سفارشی سازی کنیم.

برای این کار ما دو راه داریم:

1) یک مدل بایندر جدید را با پیاده سازی IModelBinder تهیه کنیم. (در این حالت ما مجبوریم که مدل بایندر را از ابتدا جهت بایند کردن کلیه مقادیر شی مدل خود، بازنویسی کنیم و در واقع امکان انتساب آن‌را در سطح فقط یک خصوصیت نداریم.) (نحوه پیاده سازی قبلا در اینجا مطرح شده)

2) ModelBinder پیش فرض را جهت پاسخگویی به نیازمان توسعه دهیم. (که در این مطلب قصد آموزشش را داریم.)

فرض کنید که می‌خواهید بر اساس یک Enum در صفحه، یک DropDownFor معادل را قرار بدید که به طور خودکار رشته انتخاب شده را به یک خصوصیت مدل که از نوع بایت هست بایند بکند.

از طریق کد زیر یک DropDownListFor برای Enum مورد نظر در مدل ایجاد کنیم:

```
@Html.DropDownListFor(model => model.AccountType, new
SelectList(Enum.GetNames(typeof(Enums.AccountType))))
```

و کلاس Enum مورد نظر :

```
public enum AccountType : byte
{
   0 = مدیر,
   1 = کاربر_حقیقی,
   2 = کاربر_حقوقی,
}
```

حالا برای اینکه این مقدار انتخابی به صورت خودکار و از طریق امکان binding توکار خود MVC به خصوصیت AccountType مقدار دهی شود باید یک PropertyBindAttribute سفارشی بنویسیم، برای اینکار یک کلاس جدید با نام CustomBinding می‌سازیم و کدهای زیر را به آن اضافه می‌کنیم :

```
namespace MvcApplication1.Models
{
    [AttributeUsage(AttributeTargets.Property, AllowMultiple = false)]
    public abstract class PropertyBindAttribute : Attribute
    {
        public abstract bool BindProperty(ControllerContext controllerContext,
        ModelBindingContext bindingContext, PropertyDescriptor propertyDescriptor);
    }

    public class ExtendedModelBinder : DefaultModelBinder
    {
        protected override void BindProperty(ControllerContext controllerContext,
            ModelBindingContext bindingContext, PropertyDescriptor propertyDescriptor)
        {
            if (propertyDescriptor.Attributes.OfType<PropertyBindAttribute>().Any())
            {
                var modelBindAttr =
```

```
propertyDescriptor.Attributes.OfType<PropertyBindAttribute>().FirstOrDefault();

                if (modelBindAttr.BindProperty(controllerContext, bindingContext, propertyDescriptor))
                    return;
            }

            base.BindProperty(controllerContext, bindingContext, propertyDescriptor);
        }
    }
}
```

در کد بالا ما تمام کلاس هایی را که از PropertyBindAttribute مشتق شده باشند را به DefaultModelBinder اضافه می‌کنیم.
این کد فقط یک بار نوشته می‌شود و از این به بعد هر بایندر سفارشی که بسازیم به بایندر پیشفرض اضافه خواهد شد.

حالا از طریق کدهای زیر، ما بایندر سفارشیِ خصوصیتِ خودمان را به کلاس اضافه می‌کنیم :

```
public class AccountTypeBindAttribute : PropertyBindAttribute
    {
        public override bool BindProperty(ControllerContext controllerContext,
            ModelBindingContext bindingContext, PropertyDescriptor propertyDescriptor)
        {
            if (propertyDescriptor.PropertyType == typeof(byte))
            {
                HttpRequestBase request = controllerContext.HttpContext.Request;

                byte accountType = (byte)Enum.Parse(typeof(Enums.AccountType),
request.Form["AccountType"]);
                propertyDescriptor.SetValue(bindingContext.Model, accountType);

                return true;
            }
            return false;
        }
    }
```

در کد بالا ما مقدار رشته‌ای را که از DropDownListFor ارسال شده، به مقدار عددی متناظر تعریف شده آن در Enum تبدیل می‌کنیم و آن را به خصوصیت مورد نظر بازگشت می‌دهیم، از این به بعد فقط برای فیلدی که به شکل زیر نشانه گذاری شده باشد، از این کلاس بایندر سفارشی استفاده می‌کنیم و مدل بایندر پیش فرض هم کار خود را خواهد کرد و بقیه مقادیر را بایند خواهد کرد.
اطلاعات بیشتر

```
[AccountTypeBindAttribute]
public byte AccountType { get; set; }
```

حالا باید این کلاس گسترش یافته ModelBinder را به عنوان بایندر پیش فرض MVC قرار بدهیم، برای اینکار کد زیر را به فایل Global.asax.cs اضافه کنید:

```
ModelBinders.Binders.DefaultBinder = new ExtendedModelBinder();
```

کار ما دیگر تمام است و تا اینجای کار همه چیز به‌درستی کار می‌کند ... تا اینکه شما تصمیم میگیرید که از jquery.validate.unobtrusive برای اعتبار سنجی سمت کاربر استفاده کنید و می‌بینید به DropDownListFor شما هم ایراد میگیرید که حتما باید از نوع عددی باشد
نوع کاربر : The field must be a number. برای حل این مشکل هم باید به صورت دستی validation سمت کاربر رو برای این DropDownListFor غیرفعال کرد. برای این منظور باید کدهای DropDownListFor که در صفحه گذاشتید را به شکل زیر تغییر بدید:

```
@Html.DropDownListFor(model => model.AccountType, new
SelectList(Enum.GetNames(typeof(Enums.AccountType))),new Dictionary<string, object>() {{ "data-val",
"false" }})
```

نظرات خوانندگان

نویسنده: مهران بادامی
تاریخ: ۱۳۹۲/۰۸/۲۶ ۲۰:۲۴

سلام
من یهBinding نوشتم برای تاریخ که شمسی از کاربر گرفته به میلادی میدهد برای دخیره تو DB ولی وقتی میخواهم تاریخ که تو دیتابیسم میلادی هست را نشان بدهم همون میلادی نشون میده برای نمایش به شمسی چه کنم؟

نویسنده: محمد رعیت پیشه
تاریخ: ۱۳۹۲/۰۸/۲۸ ۱۰:۲۸

_ یک راه:
ایجاد یک HTML Helper سفارشی.
http://www.dotnettips.info/post/811/asp-net-mvc-8