عنوان: **بررسی سایز و پسوند فایل آپلود شده قبل از ارسال به سرور**
نویسنده: شاهین کیاست
تاریخ: ۱۳۹۱/۰۵/۱۲ ۱۳:۴۳
آدرس: www.dotnettips.info
برچسب‌ها: jQuery, Validation

می توان قبل از اینکه کاربر فایلی را سمت سرور ارسال کند پسوند فایل را چک کرد و از یک رفت و برگشت بیهوده به سرور جلوگیری کرد .

یک پیاده سازی ساده به کمک jQuery :

```
var ext = $('#my_file_field').val().split('.').pop().toLowerCase();
if($.inArray(ext, ['gif','png','jpg','jpeg']) == -1) {
    alert('invalid extension!');
}
```

در کد بالا ابتدا پسوند فایل انتخاب شده توسط کاربر پیدا می‌شود ، سپس به کمک متد inArray با آرایه ای از پسوندهای دلخواه ما مقایسه می‌شود و در صورت معتبر نبودن پیغامی به کاربر نشان می‌دهد.

کد بالا را می‌توان در رویداد Change کنترل فایل یا در هنگام Post شدن Form اطلاعات قرار داد.
کاملا واضح است که این روش را می‌توان به راحتی دور زد و نباید به آن اکتفا کرد.
مثال این روش را اینجا بررسی کنید.

بررسی سایز فایل :

در اکثر برنامه‌های تحت وب کاربرها محدود به Upload فایل تا سایز خاصی هستند ، این مورد را هم می‌توان قبل از Upload کنترل و اعتبارسنجی کرد :

```
//binds to onchange event of your input field
$('#myFile').bind('change', function() {

  //this.files[0].size gets the size of your file.
  alert(this.files[0].size);

});
```

این روش تا زمانی که کاربر JavaScript را غیر فعال نکرده قابل اطمینان است.

نظرات خوانندگان

نویسنده: پژمان
تاریخ: ۱۳۹۱/۰۵/۱۲ ۱۸:۱۱

سلام؛ با تشکر.

اگر در همینجا بخواهیم مانع ارسال فایل بشویم چکار باید کرد؟ مثلا الان فقط یک پیغام نمایش می‌دهد، اما مانع ارسال نمی‌شود.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۱۲ ۲۳:۴۸

می‌تونید بجای alert قابلیت کلیک رو از یک دکمه (ارسال فایل به سرور) بگیرید:

```
$("#btn1").unbind("click");
```

نویسنده: شهروز
تاریخ: ۱۳۹۱/۰۵/۱۵ ۱۵:۵۸

سلام
ببخشید قسمت دوم که نمایش سایز فایل است را من هر کاری کردم نتونستم انجام بدم.
یه HTML ساده ساختم و کدهای شما رو روش قرار دادم ولی هیچ اتفاقی نیافتاد.
البته اینم بگم که رو سایت jsfiddle جواب میده و مشکلی نیست ، می‌خواهم بدونم که به جز Library jQuery آیا چیز دیگری باید اضافه کنم ؟
ممنون میشم اگه فایل تستی هم قرار بدید

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۱۵ ۱۶:۲۰

این رو با فایرفاکس تست کردم جواب داد ولی با IE به نظر کار نمی‌کنه.

نویسنده: شاهین کیاست
تاریخ: ۱۳۹۱/۰۵/۱۵ ۱۶:۴۳

HTML 5 File API در IE 9 پشتیبانی نمیشه ، اما انگار تا حدودی در IE 10 پشتیبانی میشه .
فکر می‌کنم یک راه برای پشتیبانی در IE استفاده از ActiveXObject هست ، اما خب مشکلی که هست به صورت پیشفرض غیر فعال هست. این در صورت فعال بودن ActiveX باید کار کنه.

نویسنده: شاهین کیاست
تاریخ: ۱۳۹۱/۰۵/۱۵ ۱۶:۴۵

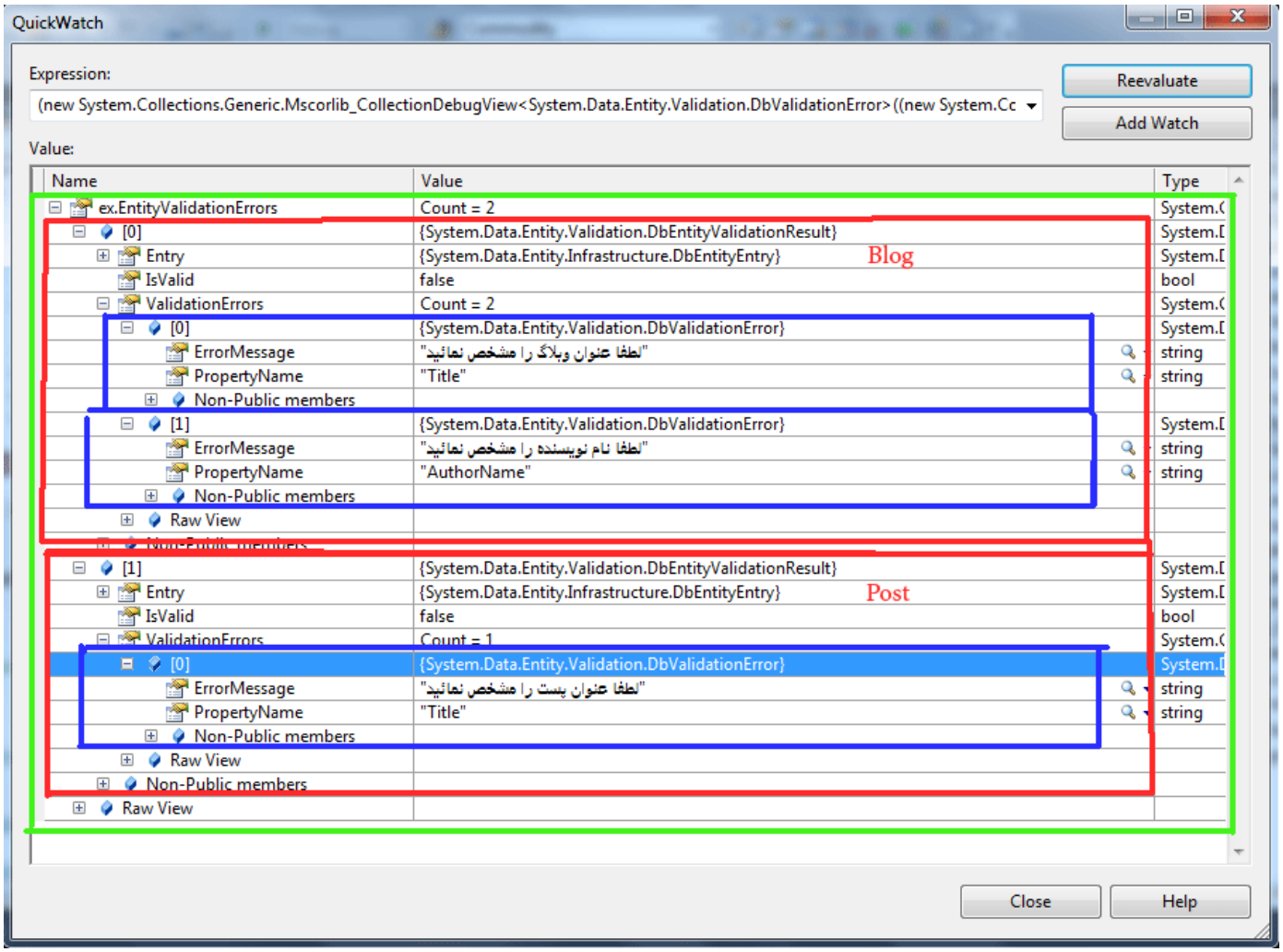
@شهروز سلام ،
اگر از IE استفاده می‌کنید رجوع کنید به پاسخی که زیر کامنت آقای نصیری نوشتم.
اگر نه به کمک FireBug در FireFox متن خطا را پیدا کنید.
آخر وقت امروز یک مثال در انتهای پست اضافه می‌کنم.

نویسنده: شهروز
تاریخ: ۱۳۹۱/۰۵/۱۶ ۹:۵۴

سلام
بله درسته من با IE چک کرده بودم ، بعدش با Chrome چک کردم درست بود پس فکر کنم IE هنوز ساپورتش نمی‌کنه
ممنون

عنوان: **اعتبارسنجی در Entity framework Code first قسمت اول**
نویسنده: ایمان محمدی
تاریخ: ۱۳۹۱/۰۵/۱۳ ۱:۳
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, Validation

در سری مباحث آموزشی EntityFramework وحیدی نصیری عزیز بصورت مختصر با اعتبار سنجی داده‌ها آشنا شدیم، در این آموزش سه قسمتی سعی می‌کنیم شناخت بیشتری از اعتبار سنجی داده در EF بدست بیاریم.
در EF CodeFirst بصورت پیش فرض پس از فراخوانی متد SaveChanges() اعتبار سنجی داده‌ها انجام می‌پذیرد؛ در صورتیکه که اعتبار سنجی با موفقیت انجام نشود با استثنای DbEntityValidationException روبرو می‌شویم که در اینجا از خاصیت EntityValidationErrors جهت اعلام خطاهای اعتبارسنجی استفاده می‌شود. EntityValidationErrors مجموعه‌ای از خطاهای مربوط به هر موجودیت می‌باشد و هر EntityValidationErrors شامل یک خاصیت ValidationErrors که خود مجموعه‌ای از خطاهای مربوط به propertyها می‌باشد. در تصویر زیر به خوبی می‌تونید این قضیه رو مشاهده کنید.



برای درک بهتر موضوع به ساختار کلاس‌های اعتبارسنجی در تصویر بعدی دقت کنید.

اعتبار سنجی در چه قسمت هایی اتفاق می‌افتد:

```
1.Property

2.Entity

3.Context
```

الف - Property :

در بخش سوم آموزش EF Code First با متادیتای مورد استفاده در EF و طرز استفاده از آنها آشنا شدیم.

```
[Required(ErrorMessage = "لطفا نام نویسنده را مشخص نمائید")]
        public string AuthorName { set; get; }

        [StringLength(100,MinimumLength=3,ErrorMessage="حداقل سه حرف و حداکثر 100 حرف وارد نمایید.")]
        public string AuthorName { set; get; }
```

همانطور که در ادامه می‌بینید برای اعتبار سنجی فقط به متادیتاهای واقع در DataAnnotations.dll نیاز داریم. بقیه متادیتاها مربوط به نگاشت روابط موجودیتها و نحوه ذخیره اون در دیتابیس می‌باشد.

```
Validation Attributes
AssemblySystem.ComponentModel.DataAnnotations.dll
NamespaceSystem.ComponentModel.DataAnnotations
StringLength
RegularExpression
DataType
Required
Range
CustomValidation
```

```
Mapping Attributes
AssemblyEntityFramework.dll
NamespaceSystem.ComponentModel.DataAnnotations
Key
Column,Table
ComplexType
ConCurrency
TimeStamp
DatabaseGenerated
ForeignKey
InverseProperty
MaxLength
MinLength
NotMapped
```

ب- Entity :

برای اعتبار سنجی یک موجودیت باید اینترفیس IValidatableObject پیاده سازی شود:

```
public class Blog : IValidatableObject
    {

        public int blogID { set; get; }

        [Required(ErrorMessage = "لطفا عنوان وبلاگ را مشخص نمائید")]
        public string Title { set; get; }
        [Required(ErrorMessage = "لطفا نام نویسنده را مشخص نمائید")]
        [StringLength(100,MinimumLength=3,ErrorMessage="حداقل سه حرف و حداکثر 100 حرف وارد نمایید.")]
        public string AuthorName { set; get; }


        public IEnumerable<ValidationResult> Validate(ValidationContext ValidationContext)
        {
             if (this.AuthorName =="بدون نام")
                yield return new ValidationResult
                    ("این نام برای نام نویسنده مجاز نمی‌باشد.", new[] { "AuthorName"});

            if (this.AuthorName == this.Title)
                yield return new ValidationResult
                    (" نام نویسنده وبلاگ و عنوان وبلاگ نمی‌تواند همسان باشد.!", new[] { "AuthorName",
"Title" });
        }
    }
```

نکته: در بررسی هم نام بودن نام نویسنده و نام وبلاگ هردو خاصیت ("AuthorName", "Title") رو درج کردیم اینکار باعث ایجاد دو خطای اعتبارسنجی می‌شود.

پ- Context :

برای اعتبار سنجی در سطح Context باید متد ()ValidateEntity واقع در کلاس DbContext را تحریف کنیم. این قسمت در بخش دوم مقاله کامل شرح داده خواهد شد.

```
protected override DbEntityValidationResult ValidateEntity(DbEntityEntry entityEntry,
System.Collections.Generic.IDictionary<object, object> items)
        {
            return base.ValidateEntity(entityEntry, items);
        }
```

نحوه فراخوانی اعتبار سنجی ها:

```
// اعتبار سنجی یک خاصیت
ICollection<DbValidationError> ValidationProperty = Context.Entry(Blog).Property(p =>
p.AuthorName).GetValidationErrors();

// اعتبار سنجی یک موجودیت
DbEntityValidationResult ValidationEntity = Context.Entry(Blog).GetValidationResult();

// اعتبار سنجی همه موجودیت‌ها
 IEnumerable<DbEntityValidationResult> ValidationContext = Context.GetValidationErrors();
```

نکته : در اعتبار سنجی Context بصورت پیش فرض فقط موجودیت‌های جدید و یا تغییر یافته اعتبار سنجی می‌شوند.

```
EntityState.Added ||  EntityState.Modified
```

ترتیب فراخوانی اعتبار سنجی‌ها :

ابتدا اعتبار سنجی روی Property انجام می‌گیرد در صورتی که خطایی وجود نداشته باشد اعتبار سنجی مرحله بعد یعنی موجودیت‌ها بررسی می‌شود. اگر در مرحله اعتبار سنجی خاصیت‌ها خطایی وجود داشته باشد اعتبار سنجی موجودیت انجام نمی‌گیرد. ترتیب اعتبار سنجی در مرحله Context بستگی به نحوه پیاده سازی ما دارد که در بخش دوم آموزش شرح داده خواهد شد.

سوال:
اعتبار سنجی چند زبانی رو چگونه تعریف کنیم؟

متد GetValidationErrors() رو در الگوی UOW , Repository چگونه پیاده سازی کنیم؟

آیا اعتبار سنجی در کنار موجودیت‌ها از نظر معماری چند لایه کار درستی می‌باشد؟

با پاسخ دادن به سوالات بالا در قالب نظر و یا مقاله به تکمیل موضوع کمک کنید و فضای آموزشی سایت رو رونق ببخشید.

نظرات خوانندگان

نویسنده: daneshjoo
تاریخ: ۱۳۹۲/۰۲/۰۹ ۲۱:۵۱

سلام؛ من دارم با wpf کار می‌کنم و همین طور که می‌دونید در این تکنولوژی اعتبار سنجی خوبی داره حالا سوال من اینه که چطور می‌تونم اعتبار سنجی EF 5 رو با اعتبار سنجی WPF تلفیق کنم و چیز جامع و یکپارچه ای ازش در بیاد

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۰۹ ۲۲:۱۰

مراجعه کنید به قسمت پنجم سری MVVM که در مورد نحوه استفاده از data annotations برای اعتبار سنجی در WPF مطلب داره (محل دریافت دوم کل سری).

نویسنده: ایمان محمدی
تاریخ: ۱۳۹۲/۰۲/۱۰ ۸:۴۷

من از ValidationHelper که شما قرار دادید در کلاس زیر استفاده کردم و baseentity از کلاس زیر مشتق شده تا تمام موجودیت‌ها اینترفیس IDataErrorInfo رو برای wpf پیاده کرده باشند.

```
public abstract class DataErrorInfo :ObservableObject, IDataErrorInfo
    {
        [Browsable(false)]
        public string Error
        {
            get
            {
                var errors = ValidationHelper.GetErrors(this);
                return string.Join(Environment.NewLine, errors);
            }
        }

        public string this[string columnName]
        {
            get
            {
                var errors = ValidationHelper.ValidateProperty(this, columnName);
                return string.Join(Environment.NewLine, errors);
            }
        }
    }
```

نویسنده: ایمان محمدی
تاریخ: ۱۳۹۲/۰۲/۲۲ ۰:۵۲

ValidationHelper مورد نیاز جهت کلاس DataErrorInfo :

```
public class ValidationHelper
    {
        public static IList<ValidationResult> GetErrors(object instance)
        {
            var res = new List<ValidationResult>();
            Validator.TryValidateObject(instance,
                new ValidationContext(instance, null, null), res, true);
            return res;
        }



        public static IList<ValidationResult> ValidateProperty(object value, string propertyName)
        {
```

```
            if (string.IsNullOrEmpty(propertyName))
                throw new ArgumentException("Invalid property name", propertyName);

            var propertyValue = value.GetType().GetProperty(propertyName).GetValue(value, null);

            var results = new List<ValidationResult>();
            var context = new ValidationContext(value, null, null) { MemberName = propertyName };
            Validator.TryValidateProperty(propertyValue, context, results);
            return results;
        }
    }
```

نویسنده: ایمان
تاریخ: ۱۳۹۲/۱۱/۱۴ ۱۲:۲۹

سلام خسته نباشید

قسمت دوم این مقاله رو ننوشتین تو سایت؟

خیلی خوبه و من واقعا بهش احتیاج دارم چون دارم الگوی Repositor,Uow رو پیاده سازی میکنم و تو اعتبار سنجی هاش به مشکل خوردم و نمیدونم تو یه بیزینس بزرگ چجوری باید اون رو پیاده سازی کرد که به مشکل نخوره

نویسنده: ابوالفضل موسوی
تاریخ: ۱۳۹۳/۰۴/۱۹ ۱۶:۱۸

سلام . یه سوال داشتم . توی codefirst می‌خوام فیلدهای جدول تولید شده نالیبل باشند و از اعتبار سنجی سمت کلاینت و سرور(MVC) هم استفاده کنم [Requierd] . اگه امکانش هست راهنمای کنید ؟

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۴/۱۹ ۱۶:۳۵

از ViewModel باید استفاده و اعتبارسنجی رو به ViewModel اعمال کنید.

نویسنده: ابوالفضل موسوی
تاریخ: ۱۳۹۳/۰۴/۲۰ ۱۲:۴۹

این کاری که شما گفتین دوباره کاریه! و فیلدهای که من دارم و همچنین جداول خیلی زیادن. من تونستم NULL بودن و اعتبار سنجی سمت سرور رو انجام بدم؛ با کدهایی که زیر قرار میدم . اما چطوری با جی کوری ایجکس باید این ولیدیشنو سمن کلاینت نیز فرا خوامی بکنم ؟

```
[AttributeUsage(AttributeTargets.Field | AttributeTargets.Property, AllowMultiple = false, Inherited =
true)]
    public class RequierdExAttribute : ValidationAttribute
    {
        public RequierdExAttribute(string ErrorMessage)
            : base()
        {
            this.ErrorMessage = ErrorMessage;
        }

        public override bool IsValid(object value)
        {
            if (value == null) return false;
```

```
            return true;
        }
```

حالا بجایِ Requierd روی فیلدها از RequierdِEx استفاده میکنم که فیلد مورد نظر در دیتا بیس نال پذیر ساخته میشه . اما مشکل اعتبار سنجی سمت کلاینته؟

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۴/۲۰ ۱۳:۰

- کاری که می‌خوای، منطقا زیر سؤاله. هم قرار نال پذیر باشه. هم کاربر باید اجبارا پرش کنه! یعنی چی اینکار؟!

- در مورد ویژگی‌های اعتبار سنجی سفارشی و مدیریت کدهای سمت کلاینت اون‌ها مطلب در سایت موجوده:

طراحی ValidationAttribute دلخواه و هماهنگ سازی آن با ASP.NET MVC

MVC 13 قسمت اعتبارسنجی سفارشی

نویسنده: ابوالفضل موسوی
تاریخ: ۱۳۹۳/۰۴/۲۰ ۱۵:۴۰

تا حالا شده که شما بخوای یه فیلد از جدولو در یه مقطعی پر کنی بعد فیلدهای بعدی رو در جاهای دیگه پر کنید ؟ خوب وقتی مقدار فیلدها نال پذیر نباشه نمیشه در مرحله ای اول اون فیلدو ذخیره کرد ! حالا چون ولیدیشن‌های MVC رو هم نیاز دارم پس باید requird هم باشه فیلد ها... تا در مراحل بعدی بگم کدام فیلدها ورودشون اجباریه ! منطقش درسته ! این کاری که می‌گم برای ذخیره سازی چند جدول به صورت همزمانه ! که چند جدول با هم ارتباط هم دارن ! توی database first کار میکنه اما توی code first به دلیل این مشکل کار نمی‌کرد ! من فقط می‌خوام وقتی داده‌ی در جدول اولی ذخیره شد ایدی اون در جدول دومی ذخیره بشه به عنوان کلید خارجی بعد اطلاعات دیگه بعدا پر بشه ! همین ! مرسی از لینک‌های که قرارا دادین برسی کردم لینک دوم کاری که من انجام دادمو اورده سمت کلاینت . و تقریبا مشکلم حل شد . تشکر

عنوان: **مقابله با پسوردهایی که ساده حدس زده می‌شوند**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۸/۱۰ ۲۱:۵
آدرس: www.dotnettips.info
برچسب‌ها: MVC, Security, Validation

هر ساله لیستی از پرکاربردترین کلمات عبور کاربران در دنیا، منتشر می‌شود که یک نمونه از آن‌را در اینجا می‌توان مشاهده کرد:
« Splashdata، توسعه دهنده نرم افزارهای امنیتی، فهرست سالانه خود را از رایج‌ترین رمزهای عبور منتشر کرده است. »

می‌شود از این لیست برای بهبود پروسه ثبت نام در یک سایت استفاده کرد و همان زمان که کاربر کلمه عبور ضعیفی را وارد کرده است، به او پیغام داد که «کلمه عبور وارد شده را راحت می‌توان حدس زد!»

```
using System.Linq;

namespace SecurityModule
{
    public static class SafePassword
    {
        public static ISet<string> BadPasswords = new HashSet<string>
        {
            "password",
            "password1",
            "123456",
            "12345678",
            "1234",
            "qwerty",
            "12345",
            "dragon",
            "******",
            "baseball",
            "football",
            "letmein",
            "monkey",
            "696969",
            "abc123",
            "mustang",
            "michael",
            "shadow",
            "master",
            "jennifer",
            "111111",
            "2000",
            "jordan",
            "superman",
            "harley",
            "1234567",
            "iloveyou",
            "trustno1",
            "sunshine",
            "123123",
            "welcome"
        };

        public static bool IsSafePasword(this string data)
        {
            if (string.IsNullOrWhiteSpace(data)) return false;
            if (data.Length < 5) return false;
            if (BadPasswords.Contains(data.ToLowerInvariant())) return false;
            if (data.AreAllCharsEuqal()) return false;

            return true;
        }

        public static bool AreAllCharsEuqal(this string data)
        {
            if (string.IsNullOrWhiteSpace(data)) return false;
            data = data.ToLowerInvariant();
            var firstElement = data.ElementAt(0);
            var euqalCharsLen = data.ToCharArray().Count(x => x == firstElement);
            if (euqalCharsLen == data.Length) return true;
            return false;
        }
    }
}
```

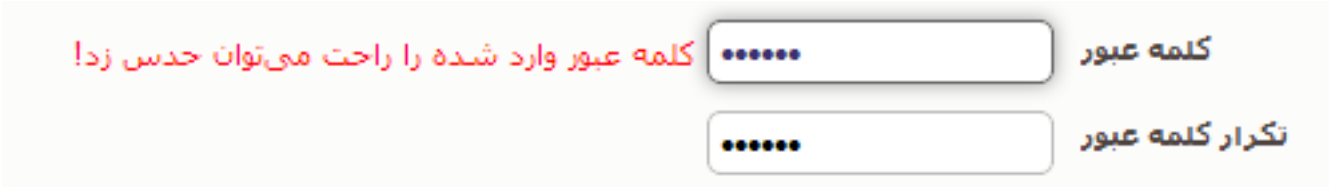
متد الحاقی IsSafePasword فوق بررسی می‌کند که آیا کلمه عبور انتخابی:
- خالی نیست.
- بیشتر از 5 کاراکتر طول دارد.
- تمام حروف بکارگرفته شده در آن یکسان نیستند.

و در ASP.NET MVC با استفاده از قابلیت Remote validation آن استفاده از این متد به نحو زیر خواهد بود:

```
public partial class RegisterController : Controller
{
        //...

        [HttpPost]
        [OutputCache(Location = OutputCacheLocation.None, NoStore = true)]
        public virtual ActionResult CheckPassword(string password1)
        {
            return Json(password1.IsSafePasword());
        }
}
```

ابتدا یک اکشن متد به کنترلر ثبت نام در سایت به نحو فوق اضافه خواهد شد.
سپس قسمتی از ViewModel متناظر با صفحه ثبت نام سایت، به شکل زیر اضافه و تعریف می‌گردد:

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

namespace MyBlog.Models
{
    public class RegisterViewModel
    {
        //...

        [Display(Name = "کلمه عبور")]
        [Required(ErrorMessage = "لطفا کلمه عبور خود را وارد نمائید")]
        [DataType(DataType.Password)]
        [StringLength(50, MinimumLength = 5, ErrorMessage = "حداقل طول کلمه عبور 5 حرف است")]
        [Remote(action: "CheckPassword",  controller: "Register",  HttpMethod = "POST",
                    ErrorMessage = "کلمه عبور وارد شده را راحت می‌توان حدس زد!")]
        public string Password1 { get; set; }
    }
}
```



عنوان: **متدی برای بررسی صحت کد ملی وارد شده**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۱/۰۸/۱۹ ۱۵:۵۰
آدرس: www.dotnettips.info
گروه‌ها: C#, Validation

در بعضی از سایت‌ها به عنوان داده ورودی کد ملی فرد دریافت می‌شود در این پست می‌خواهیم بررسی کنیم که آیا کد ملی وارد شده از نظر صحت درسا وارد شده است یا خیر.
قبل از نوشتن متد قالب کد ملی را شرح می‌دهیم.

کد ملی شماره ای است 10 رقمی که از سمت چپ سه رقم کد شهرستان ، شش رقم بعدی کد منحصر به فرد برای فرد دارنده و رقم آخر آن هم یک رقم کنترل است که از روی 9 رقم سمت چپ بدست می‌آید. برای بررسی کنترل کد کافی است مجدد از روی 9 رقم سمت چپ رقم کنترل را محاسبه کنیم

از آنجایی که درسیستم کد ملی معمولا قبل از کد تعدادی صفر وجود دارد.(رقم اول و رقم دوم از سمت چپ کد ملی ممکن است صفر باشد) و در بسیاری از موارد ممکن است کاربر این صفرها را وارد نکرده باشد و یا نرم افزار این صفرها را ذخیره نکرده باشد بهتر است قبل از هر کاری در صورتی که طول کد بزرگتر مساوی 8 و کمتر از 10 باشد به تعداد لازم (یک تا دو تا صفر) به سمت چپ عدد اضافه کنید. ساختار کد ملی در زیر نشان داده شده است.

ساختار کد ملی										
ارقام کد	رقم کنترل	9 رقم سمت چپ کد ملی								
موقعیت	1	2	3	4	5	6	7	8	9	10

کدهای ملی که همه ارقام آنها مثل هم باشند معتبر نیستند مثل:

```
۰۰۰۰۰۰۰۰۰۰
۱۱۱۱۱۱۱۱۱۱
۲۲۲۲۲۲۲۲۲۲
۳۳۳۳۳۳۳۳۳۳
۴۴۴۴۴۴۴۴۴۴
۵۵۵۵۵۵۵۵۵۵
۶۶۶۶۶۶۶۶۶۶
۷۷۷۷۷۷۷۷۷۷
۸۸۸۸۸۸۸۸۸۸
۹۹۹۹۹۹۹۹۹۹
```

روش اعتبار سنجی کد ملی :

دهمین رقم شماره ملی را (از سمت چپ) به عنوان TempAدر نظر می‌گیریم.

یک مقدار TempB در نظر می‌گیریم و آن را برابر با =

(اولین رقم * ۱۰) + (دومین رقم * ۹) + (سومین رقم * ۸) + (چهارمین رقم * ۷) + (پنجمین رقم * ۶) + (ششمین رقم * ۵) + (هفتمین رقم * ۴) + (هشتمین رقم * ۳) + (نهمین رقم * ۲)

قرار می‌دهیم.

مقدار TempC را برابر با = 11*(TempB/11) – TempB قرار می‌دهیم.

اگر مقدار TempC برابر با صفر باشد و مقدار TempA برابر TempC باشد کد ملی صحیح است.

اگر مقدار TempC برابر با ۱ باشد و مقدار TempA برابر با ۱ باشد کد ملی صحیح است.

اگر مقدار TempC بزرگتر از ۱ باشد و مقدار TempA برابر با ۱۱ – TempC باشد کد ملی صحیح است.

در ادامه متد نوشته شده به زبان C#.NET را مشاهده می‌کنید.

```
public static class Helpers
    {
        /// <summary>
        /// تعیین معتبر بودن کد ملی
        /// </summary>
        /// <param name="nationalCode">کد ملی وارد شده</param>
        /// <returns>
        /// و در صورتی که کد ملی اشتباه باشد خروجی <c>true</c> در صورتی که کد ملی صحیح باشد خروجی
<c>false</c> بود خواهد
        /// </returns>
        /// <exception cref="System.Exception"></exception>
        public static Boolean IsValidNationalCode(this String nationalCode)
        {
           //در صورتی که کد ملی وارد شده تهی باشد

            if (String.IsNullOrEmpty(nationalCode))
                throw new Exception("لطفا کد ملی را صحیح وارد نمایید");


            //در صورتی که کد ملی وارد شده طولش کمتر از 10 رقم باشد
            if (nationalCode.Length != 10)
                throw new Exception("طول کد ملی باید ده کاراکتر باشد");

            //در صورتی که کد ملی ده رقم عددی نباشد
            var regex = new Regex(@"\d{10}");
            if (!regex.IsMatch(nationalCode))
                throw new Exception("کد ملی تشکیل شده از ده رقم عددی می‌باشد؛ لطفا کد ملی را صحیح وارد
نمایید");

            //در صورتی که رقم‌های کد ملی وارد شده یکسان باشد
            var allDigitEqual =
new[]{"0000000000","1111111111","2222222222","3333333333","4444444444","5555555555","6666666666","77777
77777","8888888888","9999999999"};
            if (allDigitEqual.Contains(nationalCode)) return false;


            //عملیات شرح داده شده در بالا
            var chArray = nationalCode.ToCharArray();
            var num0 = Convert.ToInt32(chArray[0].ToString())*10;
            var num2 = Convert.ToInt32(chArray[1].ToString())*9;
            var num3 = Convert.ToInt32(chArray[2].ToString())*8;
            var num4 = Convert.ToInt32(chArray[3].ToString())*7;
            var num5 = Convert.ToInt32(chArray[4].ToString())*6;
            var num6 = Convert.ToInt32(chArray[5].ToString())*5;
            var num7 = Convert.ToInt32(chArray[6].ToString())*4;
            var num8 = Convert.ToInt32(chArray[7].ToString())*3;
            var num9 = Convert.ToInt32(chArray[8].ToString())*2;
            var a = Convert.ToInt32(chArray[9].ToString());

            var b = (((((((num0 + num2) + num3) + num4) + num5) + num6) + num7) + num8) + num9;
            var c = b%11;

            return (((c < 2) && (a == c)) || ((c >= 2) && ((11 - c) == a)));
        }
    }
```

نحوه استفاده از این متد به این صورت می‌باشد:

```
if(TextBoxNationalCode.Text.IsValidNationalCode ())
     //some code

//OR

if(Helpers.IsValidNationalCode (TextBoxNationalCode.Text))
     //some code
```

موفق ومومید باشید

نظرات خوانندگان

نویسنده: صابر احمدی
تاریخ: ۱۳۹۱/۰۸/۲۱ ۹:۷

نحوه استفاده از این کد بدون using system.linq چگونه است؟

نویسنده: صابر فتح اللهی
تاریخ: ۱۳۹۱/۰۸/۲۱ ۹:۴۸

سلام
من که منظور شمارو نفهمیدم.
اما اگر منظورتون این خطه:

```
var allDigitEqual =
new[]{"0000000000","1111111111","2222222222","3333333333","4444444444","5555555555","6666666666","77777
77777","8888888888","9999999999"};
            if (allDigitEqual.Contains(nationalCode)) return false;
```

که شما می‌تونی به جای اینکار پارامتر allDigitEqual را با مقادیر "1....1" و "2....2" و غیره از نظر تساوی با if بررسی کنی البته باید این شرطهارو OR کنید.
یعنی این قسمت به این شکل بنویسید.

```
if (nationalCode == "1111111111" ||
    nationalCode == "0000000000" ||
    nationalCode == "2222222222" ||
    nationalCode == "3333333333" ||
    nationalCode == "4444444444" ||
    nationalCode == "5555555555" ||
    nationalCode == "6666666666" ||
    nationalCode == "7777777777" ||
    nationalCode == "8888888888" ||
    nationalCode == "9999999999")
         return false;
```

امیدوارم منظور شمارو درست فهمیده باشم.

نویسنده: سعید
تاریخ: ۱۳۹۱/۰۹/۱۶ ۱۹:۴۰

سلام آقای مهندس فتح الهی لازم دونستم بابت ارائه این کد از شما تشکر کنم
موفق باشید

نویسنده: مهدی موسوی
تاریخ: ۱۳۹۲/۰۱/۰۶ ۲۰:۲۸

سلام.
من از این مطلب به مطلب شما رسیدم. اونطور که قبلا خونده بودم، کد 1111111111 کد ملی معتبری هستش بنابراین نمیشه در مورد کدهای مشابه نیز زیاد مطمئن بود چرا که ممکنه اونها به افراد دیگه نیز Assign شده باشه. نکته دوم اینکه این بخش از کد شما واقعا آزاردهنده هستش:

```
var chArray = nationalCode.ToCharArray();
var num0 = Convert.ToInt32(chArray[0].ToString())*10;
```

```
var num2 = Convert.ToInt32(chArray[1].ToString())*9;
var num3 = Convert.ToInt32(chArray[2].ToString())*8;
var num4 = Convert.ToInt32(chArray[3].ToString())*7;
var num5 = Convert.ToInt32(chArray[4].ToString())*6;
var num6 = Convert.ToInt32(chArray[5].ToString())*5;
var num7 = Convert.ToInt32(chArray[6].ToString())*4;
var num8 = Convert.ToInt32(chArray[7].ToString())*3;
var num9 = Convert.ToInt32(chArray[8].ToString())*2;
var a = Convert.ToInt32(chArray[9].ToString());

var b = (((((((num0 + num2) + num3) + num4) + num5) + num6) + num7) + num8) + num9;
var c = b%11;

return (((c < 2) && (a == c)) || ((c >= 2) && ((11 - c) == a)));
```

شما می‌تونید کد فوق رو بدین شکل بازنویسی کنید:

```
int result = 0, controlNr = (int)(input[9] - 48);
for (int i = 0; i < input.Length - 1; i++)
    result += (input[i] - 48) * (10 - i);

int remainder = result % 11;
bool isValid = controlNr == (remainder < 2 ? remainder : 11 - remainder);
```

با فرض اینکه input، رشته حاوی کد ملی باشه. در مورد اون بخش از کد که اعداد 1111111111 و 2222222222 و ... رو مقایسه کرده اید نیز، می‌تونید از Regular Expressionها بهره ببرید تا کدتون بسیار خواناتر بشه.

موفق باشید.

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۱/۰۶ ۲۳:۳

بله این قسمت کد آزار دهنده هست و می‌تواند بهتر شود، در ضمن کدهای یکسان طبق الگوریتم خود سازمان ثبت احوال نامعتبر هست در لینکی که دادین به همین موضوع نیز اشاره شده است.
نکته دوم اینکه ممنون میشم Regular Expression مورد نظر بنویسین، چون من زیاد باهاش کار نکردم.
موفق وموید باشید

نویسنده: مهدی موسوی
تاریخ: ۱۳۹۲/۰۱/۰۷ ۸:۱۸

سلام.
با این Pattern میشه اون اعداد رو جدا کرد:

```
(\d)\1{9}
```

به بیان دیگه، کافیه تا بدین شکل عمل کنیم:

```
if (Regex.IsMatch(input, @"(\d)\1{9}"))
{
    //Invalid Code...
}
```

موفق باشید.

پ.ن.: برای افرادی که هنوز متوجه موضوع نشدن باید عرض کنم که میشه جای این سه خط،

```
var allDigitEqual = new[] { "0000000000", "1111111111", "2222222222", "3333333333", "4444444444",
"5555555555", "6666666666", "7777777777", "8888888888", "9999999999" };
if (allDigitEqual.Contains(nationalCode))
    return false;
```

اینو نوشت:

```
if (Regex.IsMatch(nationalCode, @"(\d)\1{9}"))
    return false;
```

نویسنده: ابراهیم بیاگوی
تاریخ: ۱۳۹۲/۰۱/۱۳ ۱۷:۳۱

سلام. من کدتون رو بهینه و بازنویسی کردم. خوشحال می‌شم تست کنید ببینید رفتار کد عوض نشده باشه:

```
public static bool IsValidIranianNationalCode(string input)
{
    // input has 10 digits that all of them are not equal
    if (!System.Text.RegularExpressions.Regex.IsMatch(input, @"^(?!(\d)\1{9})\d{10}$"))
        return false;

    var check = Convert.ToInt32(input.Substring(9, 1));
    var sum = Enumerable.Range(0, 9)
        .Select(x => Convert.ToInt32(input.Substring(x, 1)) * (10 - x))
        .Sum() % 11;

    return sum < 2 && check == sum || sum >= 2 && check + sum == 11;
}
```

gist

در regex از negative lookahead استفاده شده که بررسی‌شه هر ده عدد یکی نباشن، از Substring استفاده‌شده که با توجه به پیاده‌سازی کلاس String به نظر من خیلی بهنیه هست، پرانتزها هم حذف شدند چون بدون پرانتز با توجه به اولویت عملگرها همان معنی را می‌دهد.

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۱/۱۴ ۸:۲۹

به نظر که درست هست، از همکاری شما متشکرم

نویسنده: سجـــــــاد ف
تاریخ: ۱۳۹۲/۰۶/۲۸ ۸:۴۶

با سلام
اینم واسه **شناسه ملی اشخاص حقوقی** منبع

```
public bool IsValidIranianLegalCode(string input)
        {
            //input has 11 digits that all of them are not equal
            if (!Regex.IsMatch(input, @"^(?!(\d)\1{10})\d{11}$"))
                return false;

            var check = Convert.ToInt32(input.Substring(10, 1));
            int dec = Convert.ToInt32(input.Substring(9, 1)) + 2;
            int[] Coef = new int[10] { 29, 27, 23, 19, 17, 29, 27, 23, 19, 17 };

            var sum = Enumerable.Range(0, 10)
                        .Select(x => (Convert.ToInt32(input.Substring(x, 1)) + dec) * Coef[x])
```

```
                        .Sum() % 11;

            return sum == check;
        }
```

نویسنده: افشین
تاریخ: ۱۳۹۲/۰۸/۰۶ ۱۳:۵۹

عجیبه !
کد ملی 1000000001 که همه میدونیم، اشتباهه رو معتبر می‌دونه !

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۸/۰۷ ۰:۵۷

طبق فرمول باید صحیح باشه منم چن سایت آنلاین بررسی کردم حق با شماست دارم بررسی می‌کنم

نویسنده: مهمان
تاریخ: ۱۳۹۲/۰۸/۱۰ ۱۲:۵۰

عذر میخوام،
چرا کدملی 1000000001 اشتباه است؟ طبق فرمولی که معرفی شده که صحیحه.
(باقیمانده 1*10 تقسیم بر 11 برابر با 10 میشه و چون این عدد بزرگتر از 1 است باید از 11 کسر بشه که نتیجه میشه 1)

آیا فرمول دقیق‌تری برای تشخیص صحت کدملی وجود دارد؟
به چه علت و بر چه اساسی کد ملی مذکور اشتباه است؟
آیا اداره ثبت احوال فرمول دقیق تشخیص صحت کدملی را در ارائه داده است؟

و ضمنا آیا بررسی یکسان بودن تمام ارقام ضروری است؟ بنظر میرسه نیازی اصلا به این بررسی وجود نداشته باشد. مثلا برای کدملی که هر ده رقم صفر است هیچگاه رقم کنترل صحیح نخواهد بود.
و همچنین در این لینک هیچ اشاره ای به این موضوع نشده است.

با تشکر

نویسنده: مهمان
تاریخ: ۱۳۹۲/۰۸/۱۰ ۱۳:۲۰

راستی موضوعی را فراموش کردم مطرح کنم.
واقعیتش من نیاز دارم کدملی را دریافت کنم که واقعی باشه اونم به معنای واقعی کلمه. عرض می‌کنم.

همانطور که میدانید (مطابق با تصویری که در ویکیا قرار دارد و قبلا لینکش را قرار دادم) سه رقم اول کد ملی (از چپ) مربوط به کد شهرستان محل صدور شناسنامه فرد است. حالا باید ما لیست کدهای شهرستان‌ها را داشته باشیم تا بتونیم کدملی را بپذیریم که مربوط باشه به یک شهرستانی که وجود خارجی داره (به عبارتی از بین تمام ترکیبات ممکن، اگر اشتباه نکنم 10 به قوه‌ی 3 که به بیان دیگر 1 تا 999 که میشه 1000 مورد؛ من اطلاعی از تعداد شهرستان‌های ایران ندارم ولی چیزی که در اینجا می‌بینم بعید میرسه به هزار برسه البته از صحت و سقم آن بی اطلاع هستم)

نکته‌ی بعدی مرتبط هست به کدمنحصربفرد (شش رقم از راست بدون در نظر گرفتن رقم کنترل) این هم فکر کنم با یک قاعده و قانونی مقداردهی شده است یا شایدم نه بصورت ترتیبی باشه مثلا از 100001 (یکصد و یک هزار) آغاز شده و یکی یکی افزایش پیدا کرده.

به هر حال اگر اطلاعات راجب این موضوع دارید لطفا دریغ نفرمایید.
منتظر پاسخ دوستان هستم.
با تشکر.

نویسنده: م منفرد
تاریخ: ۱۳۹۲/۰۸/۱۰ ۱۳:۳۹

سلام
دوست عزیز ویکیپدیا منبع معتبری در این زمینه نمی‌باشد چون هر فردی می‌تواند مقاله در آن ثبت کند و افرادی که سابقه بیشتری داشته باشند می‌توانند آنها را اصلاح و... نمایند

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۸/۱۱ ۱۲:۲۵

دوست گلم سلام
طبق فرمول‌های ارائه شده با بررسی کد ملی به طریق بالا که اگر اینترنت هم جستجو بزنین فرمول هاش ارائه شده این اعداد با روش بالا بررسی شده و در صورت عدم اعتبار کد ملی به شما خروجی میده. روشی که شما می‌خوای پیش بگیری اینه که باید برین و لیست کد تمام شهرستان‌های ایران بگیرین و توی برنامه به کار ببندین. که به نظرم روش جالبی نیست

نویسنده: افشین
تاریخ: ۱۳۹۲/۰۸/۲۶ ۱۳:۵۹

با استعلام از سازمان ثبت احوال متوجه شدیم که کد ملی 172942284 کاملا صحیحه ولی کد شما اون رو نادرست می‌دونه.

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۸/۲۶ ۱۹:۱۲

کدها اصلاح شد

```
namespace ConsoleApplicationTest
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("0172942284 => {0}", "0172942284".IsValidNationalCode());
            Console.WriteLine("1000000001 => {0}", "1000000001".IsValidNationalCode());
        }
    }

    public static class Helpers
    {
        public static Boolean IsValidNationalCode(this String nationalCode)
        {
            if (String.IsNullOrEmpty(nationalCode))
                throw new Exception("لطفا کد ملی را صحیح وارد نمایید");

            if (nationalCode.Length != 10)
                throw new Exception("طول کد ملی باید ده کاراکتر باشد");

            var regex = new Regex(@"[^0-9]
```

```
");
            if (!regex.IsMatch(nationalCode))
                throw new Exception("کد ملی تشکیل شده از ده رقم عددی می‌باشد؛ لطفا کد ملی را صحیح وارد
نمایید");

            if (!Regex.IsMatch(nationalCode, @"^(?!(\d)\1{9})\d{10}$"))
                return false;
```

```
            var check = Convert.ToInt32(nationalCode.Substring(9, 1));
            var result = Enumerable.Range(0, 9)
                .Select(x => Convert.ToInt32(nationalCode.Substring(x, 1)) * (10 - x))
                .Sum() % 11;

            int remainder = result % 11;
            return check == (remainder < 2 ? remainder : 11 - remainder);

        }
    }
}
```

پ.ن. لازم به ذکر است از کدهای اقای بیاگوی استفاده شده است.

نویسنده: ابراهیم بیاگوی
تاریخ: ۱۳۹۲/۰۸/۲۶ ۲۱:۴۴

اگر کارت‌های ملی که شمارهٔ شناسنامهٔ ۹ و ۸ رقمی دارند را دیده باشید متوجه می‌شوید که صفرهای اول جز شمارهٔ ملی **است** . به هر حال اگر به هر دلیل صفرهای اول را به گونه‌ای پاک‌کرده‌اید می‌توان کد را اینگونه نوشت که کدهای ۹ یا ۸ رقمی را پس از افزوده‌شدن صفر مانند کدهای ۱۰ رقمی صحت‌سنجی کند:

```
public static bool IsValidIranianNationalCode(string input)
        {
            input = input.PadLeft(10, '0');

            if (!Regex.IsMatch(input, @"^\d{10}$"))
                return false;

            var check = Convert.ToInt32(input.Substring(9, 1));
            var sum = Enumerable.Range(0, 9)
                .Select(x => Convert.ToInt32(input.Substring(x, 1)) * (10 - x))
                .Sum() % 11;

            return sum < 2 && check == sum || sum >= 2 && check + sum == 11;
        }
```

راستی اعداد یکسان نامعتبر **نیست** : http://www.fardanews.com/fa/news/127747
برای زبان‌های دیگر این کد https://gist.github.com/ebraminio/5292017

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۸/۲۷ ۱۰:۱۰

ظاهرا در صورتی که عدد قرینه باشه جواب درست در میاد

نویسنده: ژ محمدی
تاریخ: ۱۳۹۲/۰۹/۰۳ ۷:۵

سلام
اگر بخواهیم صحت کد ملی را از طریق jqueryvalidation چک کنیم کد ان به چه صورت خواهد بود
ممنون میشم این را هم تکه کد را هم اضافه کنید

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۹/۰۳ ۹:۲۱

آقای بیاگوی چند نظر بالاتر کدهای جاوا اسکریپتی آن‌را قرار دادند (قسمت برای زبان‌های دیگر این کد ...).

نویسنده: ژ محمدی
تاریخ: ۱۳۹۲/۰۹/۰۸ ۱۸:۳۲

سلام؛ دوتا سوال داشتم. آیا ابتدای شناسه ملی اشخاص حقوقی میتواند با 4 تا صفر شروع شود؟ میشه کد جاوا اسکریپت آن را هم بدهید ؟

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۹/۰۹ ۱۳:۲۹

سلام
می‌تونین چک کنین فکر نکنم معتبر باشه
کد جاوا اسکریپت تشخیص صحت کد ملی، منبعش این سایته

```
function checkCodeMeli(code)
{

  var L=code.length;

  if(L<8 || parseInt(code,10)==0) return false;
  code=('0000'+code).substr(L+4-10);
  if(parseInt(code.substr(3,6),10)==0) return false;
  var c=parseInt(code.substr(9,1),10);
  var s=0;
  for(var i=0;i<9;i++)
    s+=parseInt(code.substr(i,1),10)*(10-i);
  s=s%11;
  return (s<2 && c==s) || (s>=2 && c==(11-s));
return true;
}
```

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۹/۰۹ ۱۳:۵۶

اینم برای تشخیص صحت کد ملی اشخاص حقوقی با استفاده از جاوا اسکریپت

```
function checkCodeMeli(code)
{

  var L=code.length;

  if(L<11 || parseInt(code,10)==0) return false;

  if(parseInt(code.substr(3,6),10)==0) return false;
  var c=parseInt(code.substr(10,1),10);
  var d=parseInt(code.substr(9,1),10)+2;
  var z=new Array(29,27,23,19,17);
  var s=0;
  for(var i=0;i<10;i++)
    s+=(d+parseInt(code.substr(i,1),10))*z[i%5];
  s=s%11;if(s==10) s=0;
  return (c==s);

}
```

نویسنده: ژ محمدی
تاریخ: ۱۳۹۲/۰۹/۱۱ ۲۳:۲۵

در مورد کد اقتصادی چطور ؟ آن را هم میدونید به چه صورت هست ؟ و کد اعتبار سنجی به چه صورت هست

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۹/۱۲ ۱۵:۴۰

فعلا نمیدونم بررسی نکردم

نویسنده: جواد
تاریخ: ۱۳۹۲/۱۲/۲۳ ۱۹:۵۷

لطف کنید کد تشخیص کد ملی را به زبان vb 2008 برام بزارید ممنون

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۱۲/۲۳ ۲۲:۳۶

سلام
من از یک مبدل آنلاین استفاده کردم به این نتیجه رسیدم.

```
Namespace ConsoleApplicationTest
Class Program
Private Shared Sub Main(args As String())
Console.WriteLine("0172942284 => {0}", "0172942284".IsValidNationalCode())
Console.WriteLine("1000000001 => {0}", "1000000001".IsValidNationalCode())
End Sub
End Class

Public NotInheritable Class Helpers
Private Sub New()
End Sub
<System.Runtime.CompilerServices.Extension> _
Public Shared Function IsValidNationalCode(nationalCode As [String]) As [Boolean]
If [String].IsNullOrEmpty(nationalCode) Then
Throw New Exception("لطفا کد ملی را صحیح وارد نمایید")
End If

If nationalCode.Length <> 10 Then
Throw New Exception("طول کد ملی باید ده کاراکتر باشد")
End If

Dim regex__1 = New Regex("[^0-9]<span> </span>")
If Not regex__1.IsMatch(nationalCode) Then
Throw New Exception("کد ملی تشکیل شده از ده رقم عددی می‌باشد؛ لطفا کد ملی را صحیح وارد نمایید")
End If

If Not Regex.IsMatch(nationalCode, "^(?!(\d)\1{9})\d{10}$") Then
Return False
End If

Dim check = Convert.ToInt32(nationalCode.Substring(9, 1))
Dim result = Enumerable.Range(0, 9).[Select](Function(x) Convert.ToInt32(nationalCode.Substring(x, 1))
* (10 - x)).Sum() Mod 11

Dim remainder As Integer = result Mod 11
Return check = (If(remainder < 2, remainder, 11 - remainder))

End Function
End Class
End Namespace
```

نویسنده: ناشناس
تاریخ: ۱۳۹۳/۰۴/۰۱ ۱۵:۰

با سلام،

تو عبارت "مقدار TempC را برابر با = 11*(TempB/11) - TempB قرار می‌دهیم. " منظور اینه که C باید برابر با باقیمانده تقسیم صحیح TempB بر 11 باشه دیگه؟

چون 11*(TempB/11) - TempB که مقدارش 0 میشه به نظر.

ممنون

نویسنده: مهیار
تاریخ: ۱۳۹۳/۰۵/۰۶ ۱۵:۶

عذرخواهی می‌کنم ولی کد ملی من در این متد نا معتبر است !
و یا مثل این کد ملی "0081037511" که معتبر است ولی حتی متد اصلاح شده شما درون exception زیر می‌افتد

```
throw new Exception("کد ملی تشکیل شده از ده رقم عددی می‌باشد؛ لطفا کد ملی را صحیح وارد نمایید");
```

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۵/۰۸ ۱۲:۱۰

خوب کاربر رو اینقدر عذاب ندید. همون قسمت بررسی با Regex کافی هست. بیشتر نیازی نیست.

```
input = input.PadLeft(10, '0');
if (!Regex.IsMatch(input, @"^\d{10}$"))
        return false;
```

نویسنده: ایمان محمدی
تاریخ: ۱۳۹۴/۰۵/۱۸ ۱:۳۹

این مورد را فراموش کردید: "اگر باقیمانده برابر 10 باشد ، باقیمانده را برابر 0 قرار می‌دهیم"

```
sum = sum == 10 ? 0 : sum;
return sum == check;
```

عنوان: **FluentValidation #1**
نویسنده: محمّد زارع
تاریخ: ۱۳۹۱/۰۸/۲۰ ۱۰:۵
آدرس: www.dotnettips.info
برچسب‌ها: Validation, FluentValidation

FluentValidation یک پروژه سورس باز برای اعتبارسنجی Business Objectها با استفاده از Fluent Interface و Lambada Expressions می‌باشد.

جهت نصب این کتابخانه دستور زیر را در Package Manager Console وارد نمایید:

```
PM> Install-Package FluentValidation
```

ایجاد یک Validator

برای تعریف مجموعه قوانین اعتبارسنجی برای یک موجودیت ابتدا بایستی یک کلاس ایجاد کرد که از AbstractValidator<T> مشتق می‌شود که T در اینجا برابر موجودیتی است که می‌خواهیم اعتبارسنجی کنیم. به عنوان مثال کلاس مشتری به صورت زیر را در نظر بگیرید:

```
public class Customer
{
      public int Id { get; set; }
      public string Surname { get; set; }
      public string Forename { get; set; }
      public decimal Discount { get; set; }
      public string Address { get; set; }
}
```

مجموعه قوانین اعتبارسنجی با استفاده از متد RuleFor و داخل متد سازنده کلاس Validator تعریف می‌شوند. به عنوان مثال برای اطمینان از اینکه مقدار خاصیت Surname برابر Null نباشد باید به صورت زیر عمل کرد:

```
using FluentValidation;

public class CustomerValidator : AbstractValidator<Customer>
{
      public CustomerValidator
      {
           RuleFor(customer => customer.Surname).NotNull();
      }
}
```

اعتبارسنجی زنجیره ای برای یک خاصیت

برای اعتبارسنجی یک خاصیت، می‌توان از چندین Validator باهم نیز استفاده کرد:

```
RuleFor(customer => customer.Surname).NotNull().NotEqual("foo");
```

در اینجا خاصیت Surname نباید Null باشد و همچنین مقدار آن نباید برابر "Foo" باشد.
برای اجراکردن اعتبارسنجی، ابتدا یک نمونه از کلاس Validator مان را ساخته و شیء ای را که می‌خواهیم اعتبارسنجی کنیم به متد Validate آن میفرستیم:

```
Customer customer = new Customer();
CustomerValidator validator = new CustomerValidator();
ValidationResult results = validator.Validate(customer);
```

خروجی متد Validate، یک ValidationResult است که شامل دو خاصیت زیر می‌باشد:

IsValid: از نوع bool برای تعیین اینکه اعتبارسنجی موفقیت آمیز بوده یا خیر.
Errors: یک مجموعه از ValidationFailure که جزئیات تمام اعتبارسنجی‌های ناموفق را شامل می‌شود.

به عنوان مثال قطعه کد زیر، جزئیات اعتبارسنجی‌های ناموفق را نمایش می‌دهد:

```
Customer customer = new Customer();
CustomerValidator validator = new CustomerValidator();

ValidationResult results = validator.Validate(customer);

if(! results.IsValid)
{
     foreach(var failure in results.Errors)
     {
          Console.WriteLine("Property " + failure.PropertyName + " failed validation. Error was: " +
failure.ErrorMessage);
     }
}
```

پرتاب استثناها (Throwing Exceptions)

به جای برگرداندن ValidationResult شما میتوانید با کمک متد ValidateAndThrow به FluentValidation بگویید که هنگام اعتبارسنجی ناموفق یک استثنا پرتاب کند:

```
Customer customer = new Customer();
CustomerValidator validator = new CustomerValidator();
validator.ValidateAndThrow(customer);
```

در این صورت Validator یک ValidationException را پرتاب خواهد کرد که دربردارنده‌ی پیام‌های خطا در خاصیت Errors خود می‌باشد.

استفاده از Validatorها برای Complex Properties

جهت درک این ویژگی تصور کنید که کلاس‌های مشتری و آدرس و همچنین کلاس‌های مربوط به اعتبارسنجی آن‌ها را به صورت زیر داریم:

```
public class Customer
{
     public string Name { get; set; }
     public Address Address { get; set; }
}

public class Address
{
     public string Line1 { get; set; }
     public string Line2 { get; set; }
     public string Town { get; set; }
     public string County { get; set; }
     public string Postcode { get; set; }
}

public class AddressValidator : AbstractValidator<Address>
{
     public AddressValidator()
     {
          RuleFor(address => address.Postcode).NotNull();
          //etc
     }
}

public class CustomerValidator : AbstractValidator<Customer>
{
```

```
     public CustomerValidator()
     {
          RuleFor(customer => customer.Name).NotNull();
          RuleFor(customer => customer.Address).SetValidator(new AddressValidator())
      }
}
```

در این صورت وقتی متد Validate کلاس CustomerValidator را فراخوانی نمایید AddressValidator نیز فراخوانی خواهد شد و نتیجه این اعتبارسنجی به صورت یکجا در یک ValidationResult برگشت داده خواهد شد.

استفاده از Validatorها برای مجموعه‌ها (Collections)

Validatorها همچنین می‌توانند بر روی خاصیت هایی که شامل مجموعه ای از یک شیء دیگر هستند نیز استفاده شوند. به عنوان مثال یک مشتری که دارای لیستی از سفارشات است را در نظر بگیرید:

```
public class Customer
{
     public IList<Order> Orders { get; set; }
}

public class Order
{
     public string ProductName { get; set; }
     public decimal? Cost { get; set; }
}

var customer = new Customer();
customer.Orders = new List<Order>
{
     new Order { ProductName = "Foo" },
     new Order { Cost = 5 }
};
```

کلاس OrderValidator نیز به صورت زیر خواهد بود:

```
public class OrderValidator : AbstractValidator<Order>
{
     public OrderValidator()
     {
          RuleFor(x => x.ProductName).NotNull();
          RuleFor(x => x.Cost).GreaterThan(0);
     }
}
```

این Validator می‌تواند داخل CustomerValidator مورد استفاده قرار بگیرد (با استفاده از متد SetCollectionValidator):

```
public class CustomerValidator : AbstractValidator<Customer>
{
     public CustomerValidator()
     {
          RuleFor(x => x.Orders).SetCollectionValidator(new OrderValidator());
     }
}
```

می توان با استفاده از متد Where یا Unless روی اعتبارسنجی شرط گذاشت:

```
RuleFor(x => x.Orders).SetCollectionValidator(new OrderValidator()).Where(x => x.Cost != null);
```

گروه بندی قوانین اعتبارسنجی

RuleSetها به شما این امکان را می‌دهند تا بعضی از قوانین اعتبارسنجی را داخل یک گروه قرار دهید تا با یکدیگر اجرا شوند. در حالی که دیگر قوانین نادیده گرفته می‌شوند.
برای مثال تصور کنید شما سه خاصیت در کلاس Person دارید که شامل (Id, Surname, Forename) می‌باشند و همچنین یک قانون برای هرکدام از آن ها. میتوان قوانین مربوط به Surname و Forename را در یک RuleSet مجزا به نام Names قرار داد:

```
public class PersonValidator : AbstractValidator<Person>
{
     public PersonValidator()
     {
          RuleSet("Names", () =>
          {
               RuleFor(x => x.Surname).NotNull();
               RuleFor(x => x.Forename).NotNull();
          });

          RuleFor(x => x.Id).NotEqual(0);
      }
}
```

در اینجا دو خاصیت Surname و Forename با یکدیگر داخل یک RuleSet به نام Names گروه شده اند. برای اعتبارسنجی جداگانه این گروه نیز به صورت زیر می‌توان عمل کرد:

```
var validator = new PersonValidator();
var person = new Person();
var result = validator.Validate(person, ruleSet: "Names");
```

این ویژگی به شما این امکان را می‌دهد تا یک Validator پیچیده را به چندین قسمت کوچکتر تقسیم کرده و توانایی اعتبارسنجی این قسمت‌ها را به صورت جداگانه داشته باشید.

عنوان: **FluentValidation #2**
نویسنده: محمّد زارع
تاریخ: ۱۳۹۱/۰۸/۲۰ ۱۳:۵
آدرس: www.dotnettips.info
برچسب‌ها: Validation, FluentValidation

کتابخانه FluentValidation به صورت پیش فرض دارای تعدادی Validation می‌باشد که برای اکثر کارهای ابتدایی کافی می‌باشد.

NotNull	اطمینان از اینکه خاصیت مورد نظر Null نباشد
NotEmpty	اطمینان از اینکه خاصیت مورد نظر Null یا رشته خالی نباشد (یا مقدار پیش فرض نباشد، مثلا 0 برای int)
NotEqual	اطمینان از اینکه خاصیت مورد نظر برابر مقدار تعیین شده نباشد (یا برابر مقدار خاصیت دیگری نباشد)
Equal	اطمینان از اینکه خاصیت مور نظر برابر مقدار تعیین شده باشد (یا برابر مقدار خاصیت دیگری باشد)
Length	اطمینان از اینکه طول رشته‌ی خاصیت مورد نظر در محدوده خاصی باشد
LessThan	اطمینان از اینکه مقدار خاصیت مورد نظر کوچکتر از مقدار تعیین شده باشد (یا کوچکتر از خاصیت دیگری)
LessThanOrEqual	اطمینان از اینکه مقدار خاصیت مورد نظر کوچکتر یا مساوی مقدار تعیین شده باشد (یا کوچکتر مساوی مقدار خاصیت دیگری)
GreaterThan	اطمینان از اینکه مقدار خاصیت مورد نظر بزرگتر از مقدار تعیین شده باشد (یا بزرگتر از مقدار خاصیت دیگری)
GreaterThanOrEqual	اطمینان از اینکه مقدار خاصیت مورد نظر بزرگتر مساوی مقدار تعیین شده باشد (یا بزرگتر مساوی مقدار خاصیت دیگری)
Matches	اطمینان از اینکه مقدار خاصیت مورد نظر با عبارت باقائده (Regular Expression) تنظیم شده مطابقت داشته باشد
Must	اعتبارسنجی یک predicate با استفاده از Lambda Expressions. اگر عبارت Lambda مقدار true برگرداند اعتبارسنجی با موفقیت انجام شده و اگر false برگرداند، اعتبارسنجی با شکست مواجه شده است.
Email	اطمینان از اینکه مقدار خاصیت مورد نظر یک آدرس ایمیل معتبر باشد
CreditCard	اطمینان از اینکه مقدار خاصیت مورد نظر یک Credit Card باشد

همان طور که در جدول بالا ملاحظه می‌کنید بعضی از اعتبارسنجی‌ها را می‌توان با استفاده از مقدار خاصیت‌های دیگر انجام داد. برای درک این موضوع مثال زیر را در نظر بگیرید:

```
RuleFor(customer => customer.Surname).NotEqual(customer => customer.Forename);
```

در مثال بالا مقدار خاصیت Surname نباید برابر مقدار خاصیت Forename باشد.

برای تعیین اینکه در هنگام اعتبارسنجی چه پیامی به کاربر نمایش داده شود نیز می‌توان از متد WithMessage استفاده کرد:

```
RuleFor(customer => customer.Surname).NotNull().WithMessage("Please ensure that you have entered your
Surname");
```

اعتبارسنجی تنها در مواقع خاص

با استفاده از شرط‌های When و Unless می‌توان تعیین کرد که اعتبارسنجی فقط در مواقعی خاص انجام شود. به عنوان مثال در قطعه کد زیر با استفاده از متد When، تعیین می‌کنیم که اعتبارسنجی روی خاصیت CustomerDiscount تنها زمانی اتفاق بیفتد که خاصیت IsPreferredCustomer برابر true باشد.

```
RuleFor(customer => customer.CustomerDiscount).GreaterThan(0).When(customer =>
customer.IsPreferredCustomer);
```

متد Unless نیز برعکس متد When می‌باشد.
اگر نیاز به تعیین یک شرط یکسان برای چند خاصیت باشد، میتوان به جای تکرار شرط برای هرکدام از خاصیت‌ها به صورت زیر عمل کرد:

```
When(customer => customer.IsPreferred, () => {
   RuleFor(customer => customer.CustomerDiscount).GreaterThan(0);
   RuleFor(customer => customer.CreditCardNumber).NotNull();
});
```

تعیین نحوه برخورد با اعتبارسنجی‌های زنجیره ای

در قطعه کد زیر ملاحظه می‌کنید که از دو Validator برای یک خاصیت استفاده شده است. (*NotNull* و *NotEqual*)

```
RuleFor(x => x.Surname).NotNull().NotEqual("foo");
```

قطعه کد بالا بررسی می‌کند که مقدار خاصیت Surname، ابتدا برابر Null نباشد و پس از آن برابر رشته "Foo" نیز نباشد. در این حالت (حالت پیش فرض) اگر اعتبارسنجی اول (NotNull) با شکست مواجه شود، اعتبارسنجی دوم (NotEqual) نیز انجام خواهد شد. برای جلوگیری از این حالت می‌توان از CascadeMode به صورت زیر استفاده کرد:

```
RuleFor(x => x.Surname).Cascade(CascadeMode.StopOnFirstFailure).NotNull().NotEqual("foo");
```

اکنون اگر اعتبارسنجی NotNull با شکست مواجه شود، دیگر اعتبارسنجی دوم انجام نخواهد شد. این ویژگی در مواردی کاربرد دارد که یک زنجیره پیچیده از اعتبارسنجی‌ها داریم که شرط انجام هرکدام از آنها موفقیت در اعتبارسنجی‌های قبلی است.
اگر نیاز بود تا CascadeMode را برای تمام خاصیت‌های یک کلاس Validator تعیین کنیم می‌توان به صورت خلاصه از روش زیر استفاده کرد:

```
public class PersonValidator : AbstractValidator<Person> {
  public PersonValidator() {
    // First set the cascade mode
    CascadeMode = CascadeMode.StopOnFirstFailure;
```

```
    // Rule definitions follow
    RuleFor(...)
    RuleFor(...)
   }
}
```

سفارشی سازی اعتبارسنجی برای ایجاد اعتبارسنجی سفارشی دو راه وجود دارد:
راه اول ایجاد یک کلاس که از PropertyValidator مشتق می‌شود. برای توضیح نحوه استفاده از این راه، تصور کنید که میخواهیم یک اعتبارسنج سفارشی درست کنیم تا چک کند که یک لیست حتماً کمتر از 10 آیتم داخل خود داشته باشد. در این صورت کدی که بایستی نوشته شود به صورت زیر خواهد بود:

```
using System.Collections.Generic;
using FluentValidation.Validators;

public class ListMustContainFewerThanTenItemsValidator<T> : PropertyValidator {

public ListMustContainFewerThanTenItemsValidator()
: base("Property {PropertyName} contains more than 10 items!") {

}

protected override bool IsValid(PropertyValidatorContext context) {
var list = context.PropertyValue as IList<T>;

if(list != null && list.Count >= 10) {
return false;
}

return true;
}
}
```

کلاسی که از PropertyValidator مشتق می‌شود بایستی متد IsValid آن را override کند. متد IsValid یک PropertyValidatorContext را به عنوان ورودی می‌گیرد و یک boolean را که مشخص کننده نتیجه اعتبارسنجی است، بر می‌گرداند. همان طور که در مثال بالا ملاحظه می‌کنید پیغام خطا نیز در constructor مشخص شده است.
برای استفاده از این Validator سفارشی نیز می‌توان از متد SetValidator به صورت زیر استفاده نمود:

```
public class PersonValidator : AbstractValidator<Person> {
    public PersonValidator() {
       RuleFor(person => person.Pets).SetValidator(new
ListMustContainFewerThanTenItemsValidator<Pet>());
    }
}
```

راه دیگر استفاده از آن تعریف یک Extension Method می‌باشد که در این صورت می‌توان از آن به صورت زنجیره ای مانند دیگر Validatorها استفاده نمود:

```
public static class MyValidatorExtensions {
   public static IRuleBuilderOptions<T, IList<TElement>> MustContainFewerThanTenItems<T, TElement>(this
IRuleBuilder<T, IList<TElement>> ruleBuilder) {
      return ruleBuilder.SetValidator(new ListMustContainFewerThanTenItemsValidator<TElement>());
   }
}
```

اکنون برای استفاده از Extension Method می‌توان به راحتی مانند زیر عمل کرد:

```
public class PersonValidator : AbstractValidator<Person> {
    public PersonValidator() {
```

```
        RuleFor(person => person.Pets).MustContainFewerThanTenItems();
    }
}
```

راه دوم استفاده از متد Custom می‌باشد. برای توضیح نحوه استفاه از این متد مثال قبل (چک کردن تعداد آیتم‌های لیست) را به صورت زیر بازنویسی می‌کنیم:

```
public class PersonValidator : AbstractValidator<Person> {
   public PersonValidator() {
       Custom(person => {
           return person.Pets.Count >= 10
              ? new ValidationFailure("More than 10 pets is not allowed.")
              : null;
       });
   }
}
```

توجه داشته باشید که متد Custom تنها برای اعتبارسنجی‌های خیلی پیچیده طراحی شده است و در اکثر مواقع می‌توان خیلی راحت‌تر و تمیزتر از (Must PredicateValidator) برای اعتبارسنجی سفارشی مان استفاده کرد، مانند مثال زیر:

```
public class PersonValidator : AbstractValidator<Person> {
   public PersonValidator() {
      RuleFor(person => person.Pets).Must(HaveFewerThanTenPets).WithMessage("More than 9 pets is not
allowed");
   }

   private bool HaveFewerThanTenPets(IList<Pet> pets) {
      return pets.Count < 10;
   }
}
```

پ.ن.
در این دو مقاله سعی شد تا ویژگی‌های FluentValidation به صورت انتزاعی توضیح داده شود. در قسمت بعد نحوه استفاده از این کتابخانه در یک برنامه ASP.NET MVC نشان داده خواهد شد.

نظرات خوانندگان

نویسنده: alireza
تاریخ: ۱۳۹۲/۰۶/۳۰ ۱۴:۴۲

با سلام میشه مقایسه ای با validation تو کار ماکروسافت داشته باشید؟
با تشکر

نویسنده: محمّد زارع
تاریخ: ۱۳۹۲/۰۷/۰۲ ۹:۴۴

کنترل بهتر روی قوانین اعتبارسنجی.

جداسازی قوانین اعتبارسنجی از کلاس‌های ViewModel یا Model.

پشتیبانی خوب از Client Side Validation.
UnitTesting ساده‌تر نسبت به DataAnnotations.
ساده‌تر بودن نوشتن Custom Validator برای موارد خاص.
اعمال اعتبارسنجی شرطی با FluentValidation راحت‌تر است.
امکان اعتبارسنجی گروهی و ...

عنوان: **استفاده از FluentValidation در ASP.NET MVC**
نویسنده: محمّد زارع
تاریخ: ۱۳۹۱/۰۸/۲۰ ۱۶:۵
آدرس: www.dotnettips.info
برچسب‌ها: MVC, Validation, FluentValidation

برای هماهنگی این کتابخانه با ASP.NET MVC نیاز به نصب FluentValidation.Mvc3 یا FluentValidation.Mvc4 از طریق Nuget یا دانلود کتابخانه از سایت CodePlex می‌باشد. بعد از نصب کتابخانه، نیاز به تنظیم FluentValidationModelValidatorProvider داخل متد Application_Start (فایل Global.asax) داریم:

```
protected void Application_Start() {
    AreaRegistration.RegisterAllAreas();

    RegisterGlobalFilters(GlobalFilters.Filters);
    RegisterRoutes(RouteTable.Routes);

    FluentValidationModelValidatorProvider.Configure();
}
```

تصور کنید دو کلاس Person و PersonValidator را به صورت زیر داریم:

```
[Validator(typeof(PersonValidator))]
    public class Person {
    public int Id { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    public int Age { get; set; }
}

public class PersonValidator : AbstractValidator<Person> {
    public PersonValidator() {
        RuleFor(x => x.Id).NotNull();
        RuleFor(x => x.Name).Length(0, 10);
        RuleFor(x => x.Email).EmailAddress();
        RuleFor(x => x.Age).InclusiveBetween(18, 60);
    }
}
```

همان طور که ملاحظه می‌کنید، در بالای تعریف کلاس Person با استفاده از ValidatorAttribute مشخص کرده ایم که از PersonValidator جهت اعتبارسنجی استفاده کند.
در آخر می‌توانیم Controller و View ی برنامه مان را درست کنیم:

```
public class PeopleController : Controller {
    public ActionResult Create() {
        return View();
    }

    [HttpPost]
    public ActionResult Create(Person person) {

        if(! ModelState.IsValid) { // re-render the view when validation failed.
            return View("Create", person);
        }

        TempData["notice"] = "Person successfully created";
        return RedirectToAction("Index");
    }
}
```

```
@Html.ValidationSummary()

@using (Html.BeginForm()) {
Id: @Html.TextBoxFor(x => x.Id) @Html.ValidationMessageFor(x => x.Id)
<br />
Name: @Html.TextBoxFor(x => x.Name) @Html.ValidationMessageFor(x => x.Name)
<br />
Email: @Html.TextBoxFor(x => x.Email) @Html.ValidationMessageFor(x => x.Email)
<br />
Age: @Html.TextBoxFor(x => x.Age) @Html.ValidationMessageFor(x => x.Age)
```

```
<input type="submit" value="submit" />
}
```

اکنون DefaultModelBinder موجود در MVC برای اعتبارسنجی شیء Person از FluentValidationModelValidatorProvider استفاده خواهد کرد.
توجه داشته باشید که FluentValidation با اعتبارسنجی سمت کاربر ASP.NET MVC به خوبی کار خواهد کرد منتها نه برای تمامی اعتبارسنجی ها. به عنوان مثال تمام قوانینی که از شرط‌های When/Unless استفاده کرده اند، Validatorهای سفارشی، و قوانینی که در آن‌ها از Must استفاده شده باشد، اعتبارسنجی سمت کاربر نخواهند داشت. در زیر لیست Validator هایی که با اعتبارسنجی سمت کاربر به خوبی کار خواهند کرد آمده است:

NotNull/NotEmpty

Matches

InclusiveBetween

CreditCard

Email

EqualTo

Length

صفت CustomizeValidator

با استفاده از CustomizeValidatorAttribute می‌توان نحوه اجرای Validator را تنظیم کرد. به عنوان مثال اگر میخواهید که Validator تنها برای یک RuleSet مخصوص انجام شود می‌توانید مانند زیر عمل کنید:

```
public ActionResult Save([CustomizeValidator(RuleSet="MyRuleset")] Customer cust) {
  // ...
}
```

مواردی که تا اینجا گفته شد برای استفاده در یک برنامه‌ی ساده MVC کافی به نظر می‌رسد، اما از اینجا به بعد مربوط به مواقعی است که نخواهیم از Attributeها استفاده کنیم و کار را به IoC بسپاریم.

استفاده از Validator Factory با استفاده از یک IoC Container

Validator Factory چیست؟ Validator Factory یک کلاس می‌باشد که وظیفه ساخت نمونه از Validatorها را بر عهده دارد. اینترفیس IValidatorFactory به صورت زیر می‌باشد:

```
public interface IValidatorFactory {
   IValidator<T> GetValidator<T>();
   IValidator GetValidator(Type type);
}
```

ساخت Validator Factory سفارشی:

برای ساخت یک Validator Factory شما می‌توانید به طور مستقیم IValidatorFactory را پیاده سازی نمایید یا از کلاس ValidatorFactoryBase به عنوان کلاس پایه استفاده کنید (که مقداری از کارها را برای شما انجام داده است). در این مثال نحوه ایجاد یک Validator Factory که از StructureMap استفاده می‌کند را بررسی خواهیم کرد. ابتدا نیاز به ثبت Validatorها در

StructureMap داریم:

```
ObjectFactory.Configure(cfg => cfg.AddRegistry(new MyRegistry()));

public class MyRegistry : Registry {
    public MyRegistry() {
        For<IValidator<Person>>()
    .Singleton()
    .Use<PersonValidator>();
    }
}
```

در اینجا StructureMap را طوری تنظیم کرده ایم که از یک Registry سفارشی استفاده کند. در داخل این Registry به StructureMap میگوییم که زمانی که خواسته شد تا یک نمونه از IValidator<Person> ایجاد کند، PersonValidator را بر گرداند. متد CreateInstance نوع مناسب را نمونه سازی می‌کند (CustomerValidator) و آن را بازمی گرداند (یا Null بر می‌گرداند اگر نوع مناسبی وجود نداشته باشد)

استفاده از AssemblyScanner

FluentValidation دارای یک AssemblyScanner می‌باشد که کار ثبت Validatorها داخل یک اسمبلی را راحت‌تر می‌سازد. با استفاده از AssemblyScanner کلاس MyRegistery ما شبیه قطعه کد زیر خواهد شد:

```
public class MyRegistry : Registry {
   public MyRegistry() {

     AssemblyScanner.FindValidatorsInAssemblyContaining<MyValidator>()
       .ForEach(result => {
            For(result.InterfaceType)
               .Singleton()
               .Use(result.ValidatorType);
       });
   }
}
```

حالا زمان استفاده از factory ساخته شده در متد Application_Start برنامه می‌باشد:

```
protected void Application_Start() {
    RegisterRoutes(RouteTable.Routes);

    //Configure structuremap
    ObjectFactory.Configure(cfg => cfg.AddRegistry(new MyRegistry()));
    ControllerBuilder.Current.SetControllerFactory(new StructureMapControllerFactory());

    //Configure FV to use StructureMap
    var factory = new StructureMapValidatorFactory();

    //Tell MVC to use FV for validation
    ModelValidatorProviders.Providers.Add(new FluentValidationModelValidatorProvider(factory));
    DataAnnotationsModelValidatorProvider.AddImplicitRequiredAttributeForValueTypes = false;
}
```

اکنون FluentValidation از StructureMap برای نمونه سازی Validatorها استفاده خواهد کرد و کار اعتبارسنجی مدل‌ها به FluentValidaion سپرده شده است.

نظرات خوانندگان

نویسنده: محمد صاحب
تاریخ: ۱۳۹۱/۰۸/۲۱ ۸:۵۱

با تشکر مجدد از شما...
برای مواردی که سمت کلاینت ساپورت نمیشن راه حل ی وجود داره؟
بهتره این اعتبارسنجی‌ها تو کدوم لایه نوشته بشن؟

نویسنده: میثم زارع
تاریخ: ۱۳۹۱/۰۸/۲۱ ۹:۴۷

در مورد سوال اول : Custom validators with client side و Enable custom validator client side in FV
در مورد سوال دوم هم، اکثر مواقع روی ViewModel یا بهتر بگم InputModel انجام میشه، هر چند اگر نیاز بود میشه روی خود کلاس Entity مورد نظر هم ایجاد کرد.
اینجا خود سازنده کتابخانه توضیح داده که چطور ازش استفاده میکنه:
http://fluentvalidation.codeplex.com/discussions/355068

نویسنده: نارینه
تاریخ: ۱۳۹۱/۱۱/۰۱ ۱۳:۲۶

StructureMapValidatorFactory که در Application_Start() استفاده شده است، د رکجا و به چه شکلی تعریف گردیده؟
امکان دارد نمونه کد کامل را جهت استفاده قرار دهید؟

نویسنده: محمد صاحب
تاریخ: ۱۳۹۱/۱۱/۰۱ ۱۴:۳۷

```
public class StructureMapValidatorFactory : ValidatorFactoryBase {
public override IValidator CreateInstance(Type validatorType) {
return ObjectFactory.TryGetInstance(validatorType) as IValidator;
}
}
```

اطلاعات بیشتر

عنوان: **اعتبارسنجی سایتهای چند زبانه در ASP.NET MVC - قسمت اول**
نویسنده: محسن اسماعیل پور
تاریخ: ۱۳۹۲/۰۳/۰۸ ۰:۵
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net MVC, jQuery, Validation, Localization, Globalization, MVC

اگر در حال تهیه یک سایت چند زبانه هستید و همچنین سری مقالات Globalization در ASP.NET MVC رو دنبال کرده باشید میدانید که با تغییر Culture فایلهای Resource مورد نظر بارگذاری و نوشته‌های سایت تغییر میابند ولی با تغییر Culture رفتار اعتبارسنجی در سمت سرور نیز تغییر و اعتبارسنجی بر اساس Culture فعلی سایت انجام میگیرد. بررسی این موضوع را با یک مثال شروع میکنیم.

یک پروژه وب بسازید سپس به پوشه Models یک کلاس با نام ValueModel اضافه کنید. تعریف کلاس به شکل زیر هست:

```
public class ValueModel
{
    [Required]
    [Display(Name = "Decimal Value")]
    public decimal DecimalValue { get; set; }

    [Required]
    [Display(Name = "Double Value")]
    public double DoubleValue { get; set; }

    [Required]
    [Display(Name = "Integer Value")]
    public int IntegerValue { get; set; }

    [Required]
    [Display(Name = "Date Value")]
    public DateTime DateValue { get; set; }
}
```

به سراغ کلاس HomeController بروید و کدهای زیر را اضافه کنید:

```
[HttpPost]
public ActionResult Index(ValueModel valueModel)
{
    if (ModelState.IsValid)
    {
        return Redirect("Index");
    }

    return View(valueModel);
}
```

Culture را به fa-IR تغییر میدهیم، برای اینکار در فایل web.config در بخش system.web کد زیر اضافه نمایید:

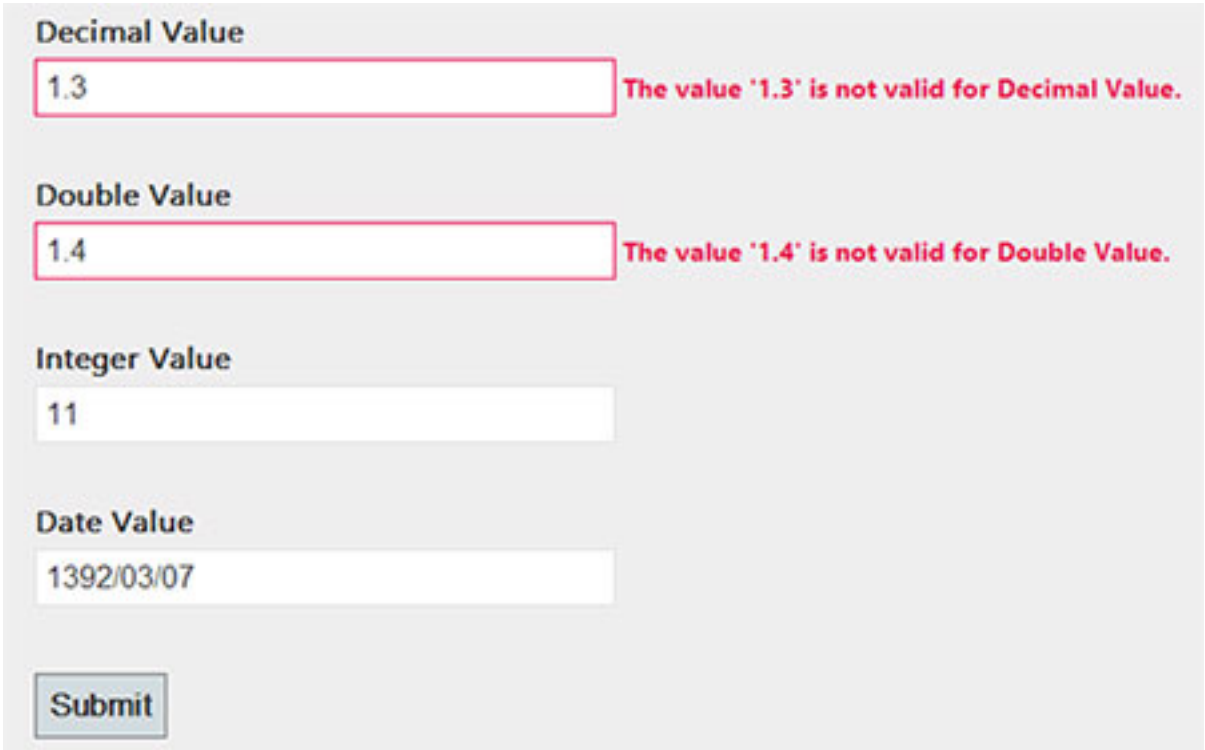
```
<globalization culture="fa-IR" uiCulture="fa-IR" />
```

و در نهایت به سراغ فایل Index.cshtml بروید کدهای زیر رو اضافه کنید:

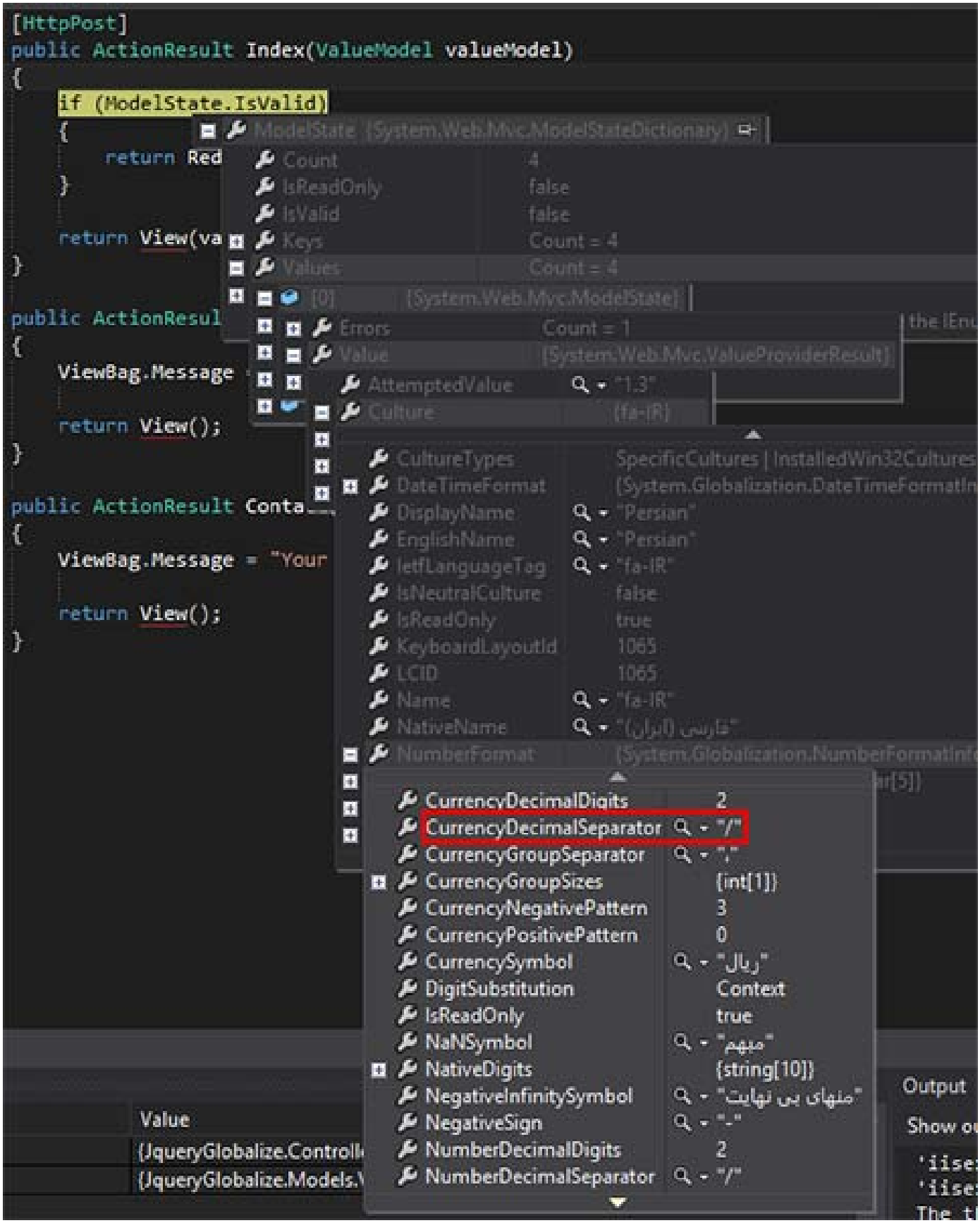
```
@using (Html.BeginForm())
{
    <ol>
        <li>
            @Html.LabelFor(m => m.DecimalValue)
            @Html.TextBoxFor(m => m.DecimalValue)
            @Html.ValidationMessageFor(m => m.DecimalValue)
        </li>
```

```
        <li>
            @Html.LabelFor(m => m.DoubleValue)
            @Html.TextBoxFor(m => m.DoubleValue)
            @Html.ValidationMessageFor(m => m.DoubleValue)
        </li>
        <li>
            @Html.LabelFor(m => m.IntegerValue)
            @Html.TextBoxFor(m => m.IntegerValue)
            @Html.ValidationMessageFor(m => m.IntegerValue)
        </li>
        <li>
            @Html.LabelFor(m => m.DateValue)
            @Html.TextBoxFor(m => m.DateValue)
            @Html.ValidationMessageFor(m => m.DateValue)
        </li>
        <li>
            <input type="submit" value="Submit"/>
        </li>
    </ol>
}
```

پرژه را اجرا نمایید و در ۲ تکست باکس اول ۲ عدد اعشاری را و در ۲ تکست باکس آخر یک عدد صحیح و یک تاریخ وارد نمایید و سپس دکمه Submit را بزنید. پس از بازگشت صفحه از سمت سرور در در ۲ تکست باکس اول با این پیامها روبرو میشوید که مقادیر وارد شده نامعتبر میباشند.



اگر پروژه رو در حالت دیباگ اجرا کنیم و نگاهی به داخل ModelState بیاندازیم، میبینیم که کاراکتر جدا کننده قسمت اعشاری برای fa-IR '/' میباشد که در اینجا برای اعداد مورد نظر کاراکتر '.' وارد شده است.



برای فایق شدن بر این مشکل یا باید سمت سرور اقدام کرد یا در سمت کلاینت. در بخش اول راه حل سمت کلاینت را بررسی مینماییم.

در سمت کلاینت برای اینکه کاربر را مجبور به وارد کردن کاراکترهای مربوط به Culture فعلی سایت نماییم باید مقادیر وارد شده را اعتبارسنجی و در صورت معتبر نبودن مقادیر پیام مناسب نشان داده شود. برای اینکار از کتابخانه jQuery Globalize استفاده میکنیم. برای اضافه کردن jQuery Globalize از طریق کنسول nuget فرمان زیر اجرا نمایید:

```
PM> Install-Package jquery-globalize
```

پس از نصب کتابخانه اگر به پوشه Scripts نگاهی بیاندازید میبینید که پوشای با نام jquery.globalize اضافه شده است. درداخل پوشه زیر پوشی دیگری با نام cultures وجود دارد که در آن Culture های مختلف وجود دارد و بسته به نیاز میتوان از آنها استفاده کرد. دوباره به سراغ فایل Index.cshtm بروید و فایلهای جاوا اسکریپتی زیر را به صفحه اضافه کنید:

```
<script src="~/Scripts/jquery.validate.js"> </script>
<script src="~/Scripts/jquery.validate.unobtrusive.js"> </script>
<script src="~/Scripts/jquery.globalize/globalize.js"> </script>
<script src="~/Scripts/jquery.globalize/cultures/globalize.culture.fa-IR.js"> </script>
```

در فایل globalize.culture.fa-IR.js کاراکتر جدا کننده اعشاری '.' در نظر گرفته شده است که مجبور به تغییر آن هسیتم. برای اینکار فایل را باز کرده و numberFormat را پیدا کنید و آن را به شکل زیر تغییر دهید:

```
numberFormat: {
    pattern: ["n-"],
    ".": "/",
    currency: {
        pattern: ["$n-", "$ n"],
        ".": "/",
        symbol: "ریال"
    }
},
```

و در نهایت کدهای زیر را به فایل Index.cshtml اضافه کنید و برنامه را دوباره اجرا نمایید :

```
Globalize.culture('fa-IR');
$.validator.methods.number = function(value, element) {
    if (value.indexOf('.') > 0) {
        return false;
    }
    var splitedValue = value.split('/');
    if (splitedValue.length === 1) {
        return !isNaN(Globalize.parseInt(value));
    } else if (splitedValue.length === 2 && $.trim(splitedValue[1]).length === 0) {
        return false;
    }
    return !isNaN(Globalize.parseFloat(value));
};
};
```

در خط اول Culture را ست مینمایم و در ادامه نحوه اعتبارسنجی را در unobtrusive validation تغییر میدهیم . از آنجایی که برای اعتبارسنجی عدد وارد شده از تابع parseFloat استفاده میشود، کاراکتر جدا کننده قسمت اعشاری قابل قبول برای این تابع '.' است پس در داخل تابع دوباره '/' به '.' تبدیل میشود و سپس اعتبارسنجی انجام میشود از اینرو اگر کاربر '.' را نیز وارد نماید قابل قبول است به همین دلیل با این خط کد if (value.indexOf('.') > 0) وجود نقطه را بررسی میکنیم تا در صورت وجود '.' پیغام خطا نشان داده شود.در خط بعدی بررسی مینماییم که اگر عدد وارد شده اعشاری نباشد از تابع parseInt استفاده نماییم. در خط بعدی این حالت را بررسی مینماییم که اگر کاربر عددی همچون ۱۲/ وارد کرد پیغام خطا صادر شود.

برای اعتبارسنجی تاریخ شمسی متاسفانه توابع کمکی برای تبدیل تاریخ در فایل globalize.culture.fa-IR.js وجود ندارد ولی اگر نگاهی به فایلهای Culture عربی بیاندازید همه دارای توابع کمکی برای تبدیل تاریخ هجری به میلادی هستند به همین دلیل امکان اعتبارسنجی تاریخ شمسی با استفاده از jQuery Globalize میسر نمیباشد. من خودم تعدادی توابع کمکی را به globalize.culture.fa-IR.js اضافه کردام که از تقویم فارسی آقای علی فرهادی برداشت شده است و با آنها کار اعتبارسنجی را انجام میدهیم. لازم به ذکر است این روش ۱۰۰% تست نشده است و شاید راه کاملا اصولی نباشد ولی به هر حال در اینجا توضیح میدهم. در فایل globalize.culture.fa-IR.js قسمت Gregorian_Localized را پیدا کنید و آن را با کدهای زیر جایگزین کنید:

```
Gregorian_Localized: {
    firstDay: 6,
    days: {
        names: ["یکشنبه", "دوشنبه", "سه شنبه", "چهارشنبه", "پنجشنبه", "جمعه", "شنبه"],
        namesAbbr: ["یکشنبه", "دوشنبه", "سه شنبه", "چهارشنبه", "پنجشنبه", "جمعه", "شنبه"],
        namesShort: ["ی", "د", "س", "چ", "پ", "ج", "ش"]
    },
    months: {
        names: ["ژانویه", "فوریه", "مارس", "آوریل", "می", "ژوئن", "ژوئیه", "اوت", "سپتامبر", "اکتبر", "نوامبر", "دسامبر", ""],
        namesAbbr: ["ژانویه", "فوریه", "مارس", "آوریل", "می", "ژوئن", "ژوئیه", "اوت", "سپتامبر", "اکتبر", "نوامبر", "دسامبر", ""]
    },
    AM: ["ق.ظ", "ق.ظ", "ق.ظ"],
    PM: ["ب.ظ", "ب.ظ", "ب.ظ"],
    patterns: {
        d: "yyyy/MM/dd",
        D: "yyyy/MM/dd",
        t: "hh:mm tt",
        T: "hh:mm:ss tt",
        f: "yyyy/MM/dd hh:mm tt",
        F: "yyyy/MM/dd hh:mm:ss tt",
        M: "dd MMMM"
    },
    JalaliDate: {
        g_days_in_month: [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31],
        j_days_in_month: [31, 31, 31, 31, 31, 31, 30, 30, 30, 30, 30, 29]
    },
    gregorianToJalali: function (gY, gM, gD) {
        gY = parseInt(gY);
        gM = parseInt(gM);
        gD = parseInt(gD);
        var gy = gY - 1600;
        var gm = gM - 1;
        var gd = gD - 1;

        var gDayNo = 365 * gy + parseInt((gy + 3) / 4) - parseInt((gy + 99) / 100) + parseInt((gy + 399) / 400);

        for (var i = 0; i < gm; ++i)
            gDayNo += Globalize.culture().calendars.Gregorian_Localized.JalaliDate.g_days_in_month[i];
        if (gm > 1 && ((gy % 4 == 0 && gy % 100 != 0) || (gy % 400 == 0)))
            /* leap and after Feb */
            ++gDayNo;
        gDayNo += gd;

        var jDayNo = gDayNo - 79;

        var jNp = parseInt(jDayNo / 12053);
        jDayNo %= 12053;

        var jy = 979 + 33 * jNp + 4 * parseInt(jDayNo / 1461);

        jDayNo %= 1461;

        if (jDayNo >= 366) {
            jy += parseInt((jDayNo - 1) / 365);
            jDayNo = (jDayNo - 1) % 365;
        }

        for (var i = 0; i < 11 && jDayNo >= Globalize.culture().calendars.Gregorian_Localized.JalaliDate.j_days_in_month[i]; ++i) {
            jDayNo -= Globalize.culture().calendars.Gregorian_Localized.JalaliDate.j_days_in_month[i];
        }
        var jm = i + 1;
        var jd = jDayNo + 1;

        return [jy, jm, jd];
    },
    jalaliToGregorian: function (jY, jM, jD) {
```

```
        jY = parseInt(jY);
        jM = parseInt(jM);
        jD = parseInt(jD);
        var jy = jY - 979;
        var jm = jM - 1;
        var jd = jD - 1;

        var jDayNo = 365 * jy + parseInt(jy / 33) * 8 + parseInt((jy % 33 + 3) / 4);
        for (var i = 0; i < jm; ++i) jDayNo +=
Globalize.culture().calendars.Gregorian_Localized.JalaliDate.j_days_in_month[i];

        jDayNo += jd;

        var gDayNo = jDayNo + 79;

        var gy = 1600 + 400 * parseInt(gDayNo / 146097); /* 146097 = 365*400 + 400/4 - 400/100 +
400/400 */
        gDayNo = gDayNo % 146097;

        var leap = true;
        if (gDayNo >= 36525) /* 36525 = 365*100 + 100/4 */ {
            gDayNo--;
            gy += 100 * parseInt(gDayNo / 36524); /* 36524 = 365*100 + 100/4 - 100/100 */
            gDayNo = gDayNo % 36524;

            if (gDayNo >= 365)
                gDayNo++;
            else
                leap = false;
        }

        gy += 4 * parseInt(gDayNo / 1461); /* 1461 = 365*4 + 4/4 */
        gDayNo %= 1461;

        if (gDayNo >= 366) {
            leap = false;

            gDayNo--;
            gy += parseInt(gDayNo / 365);
            gDayNo = gDayNo % 365;
        }

        for (var i = 0; gDayNo >=
Globalize.culture().calendars.Gregorian_Localized.JalaliDate.g_days_in_month[i] + (i == 1 && leap) ;
i++)
            gDayNo -= Globalize.culture().calendars.Gregorian_Localized.JalaliDate.g_days_in_month[i] +
(i == 1 && leap);
        var gm = i + 1;
        var gd = gDayNo + 1;

        return [gy, gm, gd];
    },
    checkDate: function (jY, jM, jD) {
        return !(jY < 0 || jY > 32767 || jM < 1 || jM > 12 || jD < 1 || jD >
            (Globalize.culture().calendars.Gregorian_Localized.JalaliDate.j_days_in_month[jM - 1] + (jM
== 12 && !((jY - 979) % 33 % 4))));
    },
    convert: function (value, format) {
        var day, month, year;

        var formatParts = format.split('/');
        var dateParts = value.split('/');
        if (formatParts.length !== 3 || dateParts.length !== 3) {
            return false;
        }

        for (var j = 0; j < formatParts.length; j++) {
            var currentFormat = formatParts[j];
            var currentDate = dateParts[j];
            switch (currentFormat) {
                case 'dd':
                    if (currentDate.length === 2 || currentDate.length === 1) {
                        day = currentDate;
                    } else {
                        year = currentDate;
                    }
                    break;
                case 'MM':
                    month = currentDate;
                    break;
                case 'yyyy':
```

```
                    if (currentDate.length === 4) {
                        year = currentDate;
                    } else {
                        day = currentDate;
                    }
                    break;
                default:
                    return false;
            }
        }

        year = parseInt(year);
        month = parseInt(month);
        day = parseInt(day);
        var isValidDate = Globalize.culture().calendars.Gregorian_Localized.checkDate(year, month,
day);
        if (!isValidDate) {
            return false;
        }

        var grDate = Globalize.culture().calendars.Gregorian_Localized.jalaliToGregorian(year, month,
day);
        var shDate = Globalize.culture().calendars.Gregorian_Localized.gregorianToJalali(grDate[0],
grDate[1], grDate[2]);

        if (year === shDate[0] && month === shDate[1] && day === shDate[2]) {
            return true;
        }

        return false;
    }
},
```

روال کار در تابع convert به اینصورت است که ابتدا تاریخ وارد شده را بررسی مینماید تا معتبر بودن آن معلوم شود به عنوان مثال اگر تاریخی مثل 31/12/1392 وارد شده باشد و در ادامه برای بررسی بیشتر تاریخ یک بار به میلادی و تاریخ میلادی دوباره به شمسی تبدیل میشود و با تاریخ وارد شده مقایسه میشود و در صورت برابری تاریخ معتبر اعلام میشود. در فایل Index.cshtml کدهای زیر اضافی نمایید:

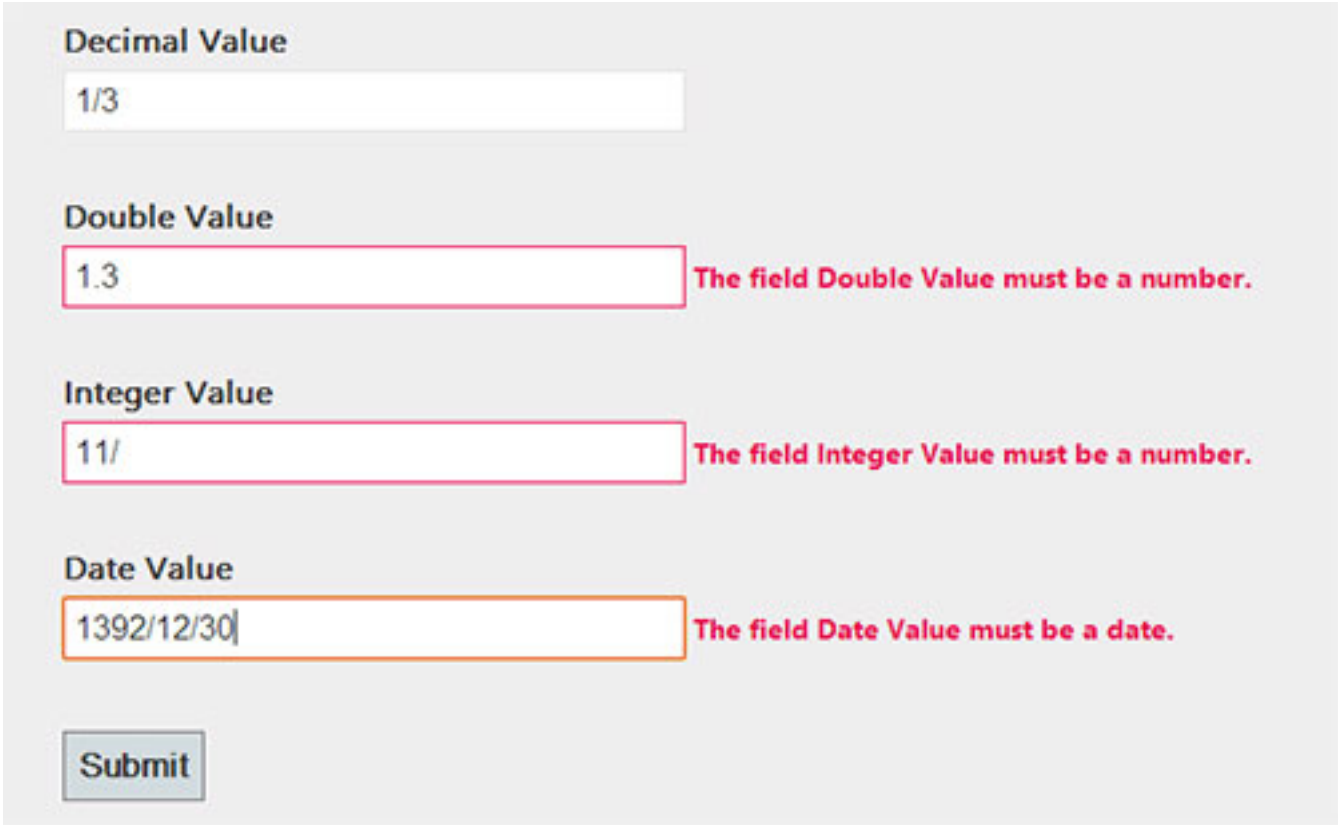
```
$.validator.methods.date = function (value, element) {
    return Globalize.culture().calendars.Gregorian_Localized.convert(value, 'yyyy/MM/dd');
};
```

برای اعتبارسنجی تاریخ میتوانید از ۲ فرمت استفاده کنید:

۱ - yyyy/MM/dd

۲ - dd/MM/yyyy

البته از توابع اعتبارسنجی تاریخ میتوانید به صورت جدا استفاده نمایید و لزومی ندارد آنها را همراه با jQuery Globalize بکار ببرید. در آخر خروجی کار به این شکل است:



در کل استفاده از jQuery Globalize برای اعتبارسنجی در سایتهای چند زبانه به نسبت خوب میباشد و برای هر زبان میتوانید از culture مورد نظر استفاده نمایید. در قسمت دوم این مطلب به بررسی بخش سمت سرور میپردازیم.

عنوان: **مشکل اعتبار سنجی jQuery validator در Bootstrap tabs**
نویسنده: ابوالفضل رجب پور
تاریخ: ۱۳۹۲/۰۹/۲۴ ۹:۱۰
آدرس: www.dotnettips.info
گروه‌ها: MVC, jQuery, Validation, Twitter Bootstrap

با استفاده از چهارچوب بوت استرپ می‌توان رابطهای کاربر استانداردی ساخت که قبلا دوستان در این سایت مطالبی را در این باب نوشته اند.
در مطلب صفحات مودال در بوت استرپ 3 در مورد ترکیب قالب بوت استرپ با سیستم اعتبارسنجی MVC و jQuery validation و نمایش فرم‌های مودال بوت استرپ صحبت شده و بسیار کامل هست.

مشکل:
هنگام کار با بوت استرپ اگر از tabهای آن در فرم برنامه استفاده کنید، هنگام validate کردن فرم متوجه می‌شوید که **فقط** کنترل‌های تب جاری اعتبارسنجی میشوند و بدون توجه به سایر کنترل هایی که در تب‌های دیگر هستند و احیانا در وضعیت عدم اعتبار هستند، فرم اعتبارسنجی و کامل اعلام شده و اطلاعات ارسال می‌شود.
دلیل:
دلیل این اتفاق این هست که jQuery validation به طور پیش فرض کنترل‌هایی که در صفحه مخفی هستند را اعتبارسنجی نمی‌کند و نادیده می‌گیرد.

راه حل:

با تغییر رفتار پیش فرض سیستم اعتبارسنجی می‌شود این مساله را حل کرد. با اضافه کردن **ignore: ""** اعلام می‌شود که کنترل‌های مخفی هم اعتبارسنجی شوند.
در نهایت کد کامل ترکیب قالب بوت استرپ با سیستم اعتبارسنجی جی کوئری به صورت زیر می‌شود. (فقط همان **ignore: ""** اضافه شده است).

```
function enableBootstrapStyleValidation() {
    $.validator.setDefaults({
        ignore: "",
        highlight: function (element, errorClass, validClass) {
            if (element.type === 'radio') {
                this.findByName(element.name).addClass(errorClass).removeClass(validClass);
            } else {
                $(element).addClass(errorClass).removeClass(validClass);
                $(element).closest('.form-group').removeClass('has-success').addClass('has-error');
            }
            $(element).trigger('highlited');
        },
        unhighlight: function (element, errorClass, validClass) {
            if (element.type === 'radio') {
                this.findByName(element.name).removeClass(errorClass).addClass(validClass);
            } else {
                $(element).removeClass(errorClass).addClass(validClass);
                $(element).closest('.form-group').removeClass('has-error').addClass('has-success');
            }
            $(element).trigger('unhighlited');
        }
    });
}
```

البته می‌توان این تغییر رفتار پیش فرض را فقط به فرم خاصی هم اعمال کرد. به این صورت

```
$("#form").validate({
...
rules: {...},
messages: {...},
ignore: "",
...
});
```

نظرات خوانندگان

نویسنده: م راد
تاریخ: ۱۳۹۲/۱۰/۰۷ ۱۳:۳۸

با تشکر از ارسال مطلب مفیدتون،
اما موضوعی که پیش میاد اینکه چطوری به ازای هر تب ابتدا اعتبارسنجی فیلدهای اون تب انجام شه و بعد تب بعدی نمایش داده شه؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۰/۰۷ ۱۹:۳۳

خوب این همون حالت پیش فرضی هست که توضیح دادن: «هنگام validate کردن فرم متوجه می‌شوید که **فقط** کنترل‌های تب جاری اعتبارسنجی میشوند و بدون توجه به سایر کنترل هایی که در تب‌های دیگر هستند». این tab جاری منظور همون به ازای هر tab جداگانه هست.

برای نمایش خودکار تب بعدی هم باید کدنویسی کنید. فقط کافی هست که کلاس active رو به div تب مورد نظر اضافه کنید:

```
$("#home").removeClass("active");  // this deactivates the home tab
$("#profile").addClass("active");  // this activates the profile tab
```

عنوان: **اعتبارسنجی در فرم‌های ASP.NET MVC با Remote Validation**
نویسنده: علی اکبر کورش فر
تاریخ: ۱۳۹۳/۰۵/۱۱ ۱۲:۵۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, Validation, Attributes

بعد از آمدن نسخه‌ی سوم ASP.NET MVC مکانیسمی به نام Remote Validation به آن اضافه شد که کارش اعتبارسنجی از راه دور بود. فرض کنید نیاز است در یک فرم، قبل از اینکه کل فرم به سمت سرور ارسال شود، مقداری بررسی شده و اعتبارسنجی آن انجام گیرد و این اعتبارسنجی چیزی نیست که بتوان سمت کاربر و بدون فرستاده شدن مقداری به سمت سرور صورت گیرد.
نمونه بارز این مسئله صفحه عضویت اکثر سایت‌هایی هست که روزانه داریم با آن‌ها کار می‌کنیم. فیلد نام کاربری توسط شما پر شده و بعد از بیرون آمدن از آن فیلد، سریعا مشخص می‌شود که آیا این نام کاربری قابل استفاده برای شما هست یا خیر.
به‌صورت معمول برای انجام این کار باید با جاوا اسکریپت، مدیریتی روی فیلد مربوطه انجام دهیم. مثلا با بیرون آمدن فوکوس از روی فیلد، با Ajax نام کاربری وارد شده را به سمت سرور بفرستیم، چک کنیم و بعد از اینکه جواب برگشت بررسی کنیم که الان آیا این نام کاربری قبلا گرفته شده یا نه.
انجام این کار به‌راحتی با مزین‌کردن خصوصیت (Property) مربوطه موجود در مدل برنامه به Attribute یا ویژگی Remote و داشتن یک Action در Controller مربوطه که کارش بررسی وجود یوزرنیم هست امکان پذیر است. ادامه بحث را با مثال همراه می‌کنم.
به عنوان مثال در سیستمی که قرار هست محصولات ما را ثبت کند، باید بیایم و قبل از اینکه محصول جدید به ثبت برسد این عملیات چک‌کردن را انجام دهیم تا کالای تکراری وارد سیستم نشود. شناسه اصلی که برای هر محصول وجود دارد بارکد هست و ما آن را میخواهیم مورد بررسی قرار دهیم.

مدل برنامه

```
public class ProductModel
    {
        public int Id { get; set; }

        [Display(Name = "نام کالا")]
        [Required(ErrorMessage = "{0} یک فیلد اجباری است و باید آن را وارد کنید.")]
        [StringLength(50, ErrorMessage = "طول {0} باید کمتر از {1} کاراکتر باشد.")]
        public string Name { get; set; }

        [Display(Name = "قیمت")]
        [Required(ErrorMessage = "{0} یک فیلد اجباری است و باید آن را وارد کنید.")]
        [DataType(DataType.Currency)]
        public double Price { get; set; }

        [Display(Name = "بارکد")]
        [Required(ErrorMessage = "{0} یک فیلد اجباری است و باید آن را وارد کنید.")]
        [StringLength(50, ErrorMessage = "طول {0} باید کمتر از {1} کاراکتر باشد.")]
        [Remote("IsProductExist", "Product", HttpMethod = "POST", ErrorMessage = "این بارکد از قبل در
سیستم وجود دارد.")]
        public string Barcode { get; set; }
    }
```

همونطور که می‌بینید خصوصیت Barcode را مزین کردیم به ویژگی Remote. این ویژگی دارای ورودی‌های خاص خودش هست. وارد کردن نام اکشن و کنترلر مربوطه برای انجام این چک‌کردن از مهم‌ترین قسمت‌های اصلی هست. چیزهایی دیگه‌ای هم هست که می‌توانیم آن‌ها را مقداردهی کنیم. مثل HttpMethod، ErrorMessage و یا AdditionFields. HttpMethod که همان طریقه‌ی ارسال درخواست به سرور هست. ErrorMessage هم همان خطایی هست که در زمان رخداد قرار است نشان داده شود. AdditionFields هم خصوصیتی را مشخص می‌کند که ما می‌خوایم به‌همراه فیلد مربوطه به سمت سرور بفرستیم. مثلا می‌تونیم به‌همراه بارکد، نام کالا را هم برای بررسی‌های مورد نیازمان بفرستیم.

کنترلر برنامه

```
[HttpPost]
        [OutputCache(Location = OutputCacheLocation.None, NoStore = true)]
        public ActionResult IsProductExist(string barcode)
```

```
        {
            if (barcode == "123456789") return Json(false); // اگر محصول وجود داشت
            return Json(true);
        }
```

در اینجا به نمایش قسمتی از کنترلر برنامه می‌پردازیم. اکشنی که مربوط می‌شود به چک‌کردن مقدارهای لازم و در پایان آن یک خروجی Json را برمی‌گردانیم که مقدار true یا false دارد. در حقیقت مقدار را به این صورت برمی‌گردانیم که اگر مقدار ورودی در پایگاه داده وجود دارد، false را برمی‌گرداند و اگر وجود نداشت true. همین‌طور آمدیم از کش شدن درخواست‌هایی که با Ajax آمده با ویژگی OutputCache جلوگیری کردیم.

نظرات خوانندگان

نویسنده: مجتبی
تاریخ: ۱۳۹۳/۰۵/۱۲ ۱۳:۳

مشکل این روش این است که بعد از یک بار اعتبار سنجی دفعه بعد با زدن هر کلید داخل تکست باکس می‌خواد بره به بانک و اطلاعات را چک کنه.

نویسنده: علی اکبر کورش فر
تاریخ: ۱۳۹۳/۰۵/۱۴ ۸:۳۴

بله. البته. ولی به نظر شما چه نیازی هستش وقتی که فیلد مربوطه پر شد دوباره به فیلد برگرده. این فقط در حالتی هستش که کاربر بخواد مقدار رو تغییر بده. پس اولویت داره استفاده از این کار در برابر استفاده نکردن

نویسنده: جوکار
تاریخ: ۱۳۹۳/۰۶/۰۱ ۱۵:۴۸

روش بسیار جالبی بود.اما یک مشکل که در این روش با آن روبرو میشویم این است که هنگام ویرایش یک رکورد موجود در بانک ، اگر قرار نباشد فیلد مورد نظر بروز رسانی شود ، اکشن متد ذکر شده دست ما را می‌بندد و اجازه آپدیت را نمی‌دهد.
شاید یک راه ایجاد یک viewModel جداگانه برای آپدیت باشد که در آن از remote attribute صرف نظر شود ، اما این راه حل زیاد جالب به نظر نمی‌رسد!
آیا راه حل مناسب‌تری وجود دارد؟

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۶/۰۲ ۱۹:۶

برای حالت ویرایش AdditionFields آن کاربرد داره. مثلا فیلد Id رو اینجا میشه ارسال کرد تا مشخص باشه حالت ویرایش هست. در حالت ثبت معمولی، خوب هنوز Id رکورد مشخص نیست و نال هست.

نویسنده: علی خسروی
تاریخ: ۱۳۹۳/۰۷/۲۲ ۱۲:۵۷

با سلام و تشکر؛ در صورتی که قصد داشته باشیم هنگام remote یک loader هم نشون بدیم باید چکار کرد.

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۷/۲۲ ۱۴:۱۲

از روال‌های رخدادگردان عمومی ajaxStart و ajaxComplete استفاده کنید.

نویسنده: م علی خسروی
تاریخ: ۱۳۹۳/۰۷/۲۳ ۹:۷

با تشکر
در واقع من سه input دارم که می‌خام remote بشند برای هر کدام هم یه loader در کنارش قرار دارم ، چه طوری میشه فهمید الان کدوم input در صفحه ajax رو start کرده تا loader اون نمایش داده بشه

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۲۳ ۱۰:۴۷

اگر به فایل jquery.validate.js مراجعه کنید، در قسمت remote آن، متد startRequest پیش از شروع عملیات Ajax و متد

stopRequest پس از پایان کار فراخوانی می‌شوند.

```
prototype: {
startRequest: function( element ) {
  //...
},
stopRequest: function( element, valid ) {
  //...
},
```

این دو متد را باید برای نمایش loading بازنویسی کرد. برای مثال:

```
var originalStartRequest = $.validator.prototype.startRequest;
$.validator.prototype.startRequest = function (element) {
    // یافتن عنصر در حال بررسی
    var container = $('form').find("[data-valmsg-for='" + element.name + "']");
// افزودن کلاس نمایش منتظر بمانید
    container.addClass('loading');

// فراخوانی متد اصلی برای انجام کارهای درونی افزونه
    originalStartRequest.apply(this, arguments);
};

var originalStopRequest = $.validator.prototype.stopRequest;
$.validator.prototype.stopRequest = function (element) {
    // یافتن عنصر در حال بررسی
    var container = $('form').find("[data-valmsg-for='" + element.name + "']");
// حذف  کلاس نمایش منتظر بمانید
    container.removeClass('loading');

// فراخوانی متد اصلی برای انجام کارهای درونی افزونه
    originalStopRequest.apply(this, arguments);
};
```

در اینجا loading به span مخفی data-valmsg-for اضافه می‌شود.

```
<span class="field-validation-valid" data-valmsg-replace="true" data-valmsg-for="Url"></span>
```

نمونه‌ی این بازنویسی در مطلب « اعتبار سنجی سمت کاربر wysiwyg-editorها در ASP.NET MVC » هم انجام شده‌است.

نویسنده: حامد رشنو
تاریخ: ۱۳۹۳/۰۸/۳۰ ۱۹:۴۱

با سلام
من از این روش استفاده کردم خیلی جالبه، فقط یک مشکل ، توی سیستم خودم خوب جواب میده ولی وقتی سایت رو پابلیش میکنم و روی هاست میزارم کار نمیکنه. نمیدونم مشکل از کجاست.

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۸/۳۰ ۲۰:۱۱

خطاهای ممکن را به این صورت بررسی کنید: نحوه استفاده از افزونه Firebug برای دیباگ برنامه‌های ASP.NET مبتنی بر jQuery

نویسنده: جواد جوادی
تاریخ: ۱۳۹۴/۰۴/۰۴ ۱۲:۱۹

سلام وقت بخیر ; در صورتی که مدل در پوشه Model برنامه باشد این روش اوکی است .. در صورتی که مدل برنامه در یک Calss Library دیگر باشد این روش عمل نمی‌کند . من در این حالت با JQuery چک می‌کنم .. آیا با Remote Validation این امکان وجود دارد که اگر مدل در یک کلاس Library دیگر باشد ارتباط برقرار کند ؟

با تشکر و سپاس فراوان از شما

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۴/۰۴ ۱۲:۵۷

« آشنایی با نحوه‌ی دیباگ برنامه‌های جاوا اسکریپتی »

نویسنده: بالازاده
تاریخ: ۱۳۹۴/۰۴/۰۵ ۱۶:۴۶

با بررسی فیلد مورد نظر در خروجی html تولید شده، می‌توانید صحت عملکرد برنامه را بررسی کنید.
مثال زیر در این زمینه می‌باشد که مدل آن در یک class library دیگر است (البته در اینجا به جای استفاده از نام اکشن و نام کنترلر از نام روت استفاده شده است)
حالت اول: مدل برنامه در حالتی که فقط فیلد مورد نظر باید بررسی شود (ایجاد کاربر):

```
namespace Project.Models
{
    public class EmployeeCreateModel
    {
        [Required]
        [Display(Name = "آدرس ایمیل")]
        [EmailAddress(ErrorMessage = "معتبر وارد کنید {0} لطفاً.")]
        [Remote("UserExistByEmailValidation",
            HttpMethod = "POST",
            ErrorMessage = "ایمیل وارد شده هم اکنون توسط یکی از کاربران مورد استفاده است.")]
        public string Email { get; set; }

     ...

     }
}
```

- حالت دوم: مدل برنامه در حالتی که به جز فیلد مورد نظر باید یک فیلد دیگر نیز مورد بررسی قرار گیرد (ویرایش کاربر):

```
namespace Project.Models
{
    public class EmployeeEditModel
    {
        public int Id { get; set; }

        [Required]
        [Display(Name = "آدرس ایمیل")]
        [EmailAddress(ErrorMessage = "معتبر وارد کنید {0} لطفاً.")]
        [Remote("EmailExistForOtherUserValidation",
            AdditionalFields = "Id",
            HttpMethod = "POST",
            ErrorMessage = "ایمیل وارد شده هم اکنون توسط یکی از کاربران مورد استفاده است.")]
        public string Email { get; set; }

        ....

    }
}
```

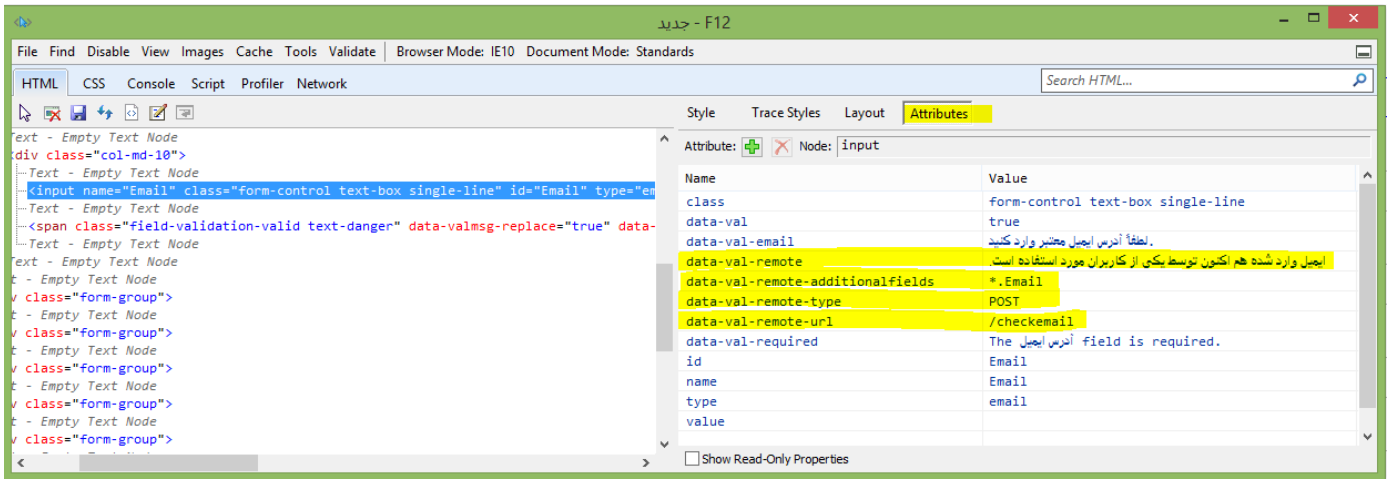
کنترلر چک کننده (partial بودن کلاس و virtual بودن اکشن‌ها به دلیل استفاده از T4MVC است):

```
namespace Project.Web.Controllers
{
    [RoutePrefix("UserValidation")]
    [Route("{Action}")]
    [OutputCache(Location = OutputCacheLocation.None, NoStore = true)]
    public partial class UserValidationController : Controller
    {
        readonly IUserService<User> _userService;
        readonly IUnitOfWork _uow;
```

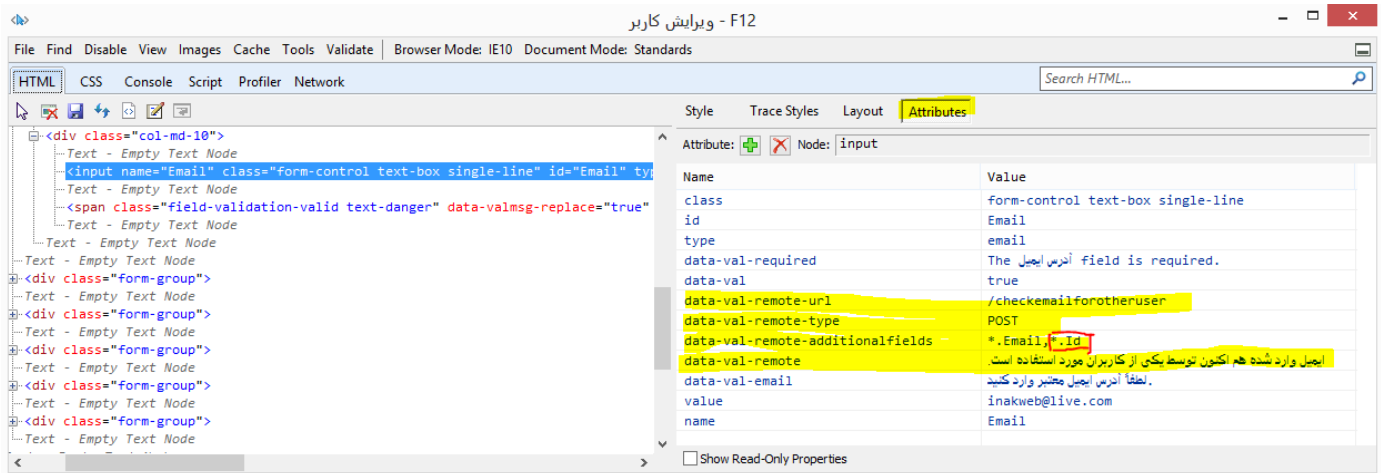
```
        public UserValidationController(IUnitOfWork uow, IUserService<User> userService)
        {
            _userService = userService;
            _uow = uow;
        }

        [HttpPost]
        [Route("~/CheckEmail", Name = "UserExistByEmailValidation")]
        public virtual JsonResult CheckEmail(string email)
        {
            return Json(!_userService.UserExistsByEmail(email));
        }

        [HttpPost]
        [Route("~/CheckEmailForOtherUser", Name = "EmailExistForOtherUserValidation")]
        public virtual JsonResult CheckEmailForOtherUser(string email, int id)
        {
            return Json(!_userService.EmailExistForOtherUser(email, id));
        }
    }
}
```

فیلد مورد نظر در خروجی Html تولید شده، باید به صورت زیر باشد:

- حالت اول:



- حالت دوم (فیلد Id هم ارسال می‌گردد):



در صورتی که خروجی درست بود، باید scriptها را مورد بررسی قرار دهید که یکی از متداول‌ترین آن‌ها

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

می‌باشد.

عنوان: **کار با وب سرویس جاوایی تشخیص ایمیل‌های موقتی در دات نت**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۰/۲۷ ۱۲:۴۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MVC, Security, WCF, xml, Validation, Java

یکی از وب سرویس‌های سایت name api ، امکان تشخیص موقتی بودن ایمیل مورد استفاده‌ی جهت ثبت نام در یک سایت را فراهم می‌کند. آدرس WSDL آن نیز در اینجا قرار دارد. اگر مطابق معمول استفاده از سرویس‌های وب در دات نت، بر روی ارجاعات پروژه کلیک راست کرده و گزینه‌ی Add service refrence را انتخاب کنیم و سپس آدرس WSDL یاد شده را به آن معرفی کنیم، بدون مشکل ساختار این وب سرویس دریافت و برای استفاده‌ی از آن به یک چنین کدی خواهیم رسید:

```
var client = new SoapDisposableEmailAddressDetectorClient();

var context = new soapContext
{
    //todo: get your API key here: http://www.nameapi.org/en/register/
    apiKey = "test"
};
var result = client.isDisposable(context, "DaDiDoo@mailinator.com");

if (result.disposable.ToString() == "YES")
{
    Console.WriteLine("YES! It's Disposable!");
}
```

متد isDisposable ارائه شده‌ی توسط این وب سرویس، دو پارامتر context که در آن باید API Key خود را مشخص کرد و همچنین آدرس ایمیل مورد بررسی را دریافت می‌کند. اگر به همین ترتیب این پروژه را اجرا کنید، با خطای Bad request از طرف سرور متوقف خواهید شد:

```
Additional information: The remote server returned an unexpected response: (400) Bad Request.
```

اگر به خروجی این وب سرویس در فیدلر مراجعه کنیم، چنین شکلی را خواهد داشت:

```
<html><head><title>Bad Request</title></head><body><h1>Bad Request</h1><p>No api-key
provided!</p></body></html>
```

عنوان کرده‌است که api-key را، در درخواست وب خود ذکر نکرده‌ایم.
اگر همین وب سرویس را توسط امکانات سایت http://wsdlbrowser.com بررسی کنید، بدون مشکل کار می‌کند. اما تفاوت در کجاست؟
خروجی ارسالی به سرور، توسط سایت http://wsdlbrowser.com به این شکل است:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://disposableemailaddressdetector.email.services.v4_0.soap.server.nameapi.org/">
  <SOAP-ENV:Body>
    <ns1:isDisposable>
      <context>
        <apiKey>test</apiKey>
      </context>
      <emailAddress>sdsdg@site.com</emailAddress>
    </ns1:isDisposable>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

و نمونه‌ی تولید شده‌ی توسط WCF (امکان Add service reference در حقیقت یک WCF Client را ایجاد می‌کند) به صورت زیر می‌باشد:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <isDisposable
```

```
xmlns="http://disposableemailaddressdetector.email.services.v4_0.soap.server.nameapi.org/">
          <context xmlns=""><apiKey>test</apiKey></context>
          <emailAddress xmlns="">DaDiDoo@mailinator.com</emailAddress>
      </isDisposable>
  </s:Body>
</s:Envelope>
```

از لحاظ اصول XML، خروجی تولیدی توسط WCF هیچ ایرادی ندارد. از این جهت که نام فضای نام مرتبط با http://schemas.xmlsoap.org/soap/envelope/ را به s تنظیم کرده‌است و سپس با استفاده از این نام، Envelope را تشکیل داده‌است. اما ... این وب سرور جاوایی دقیقا با نام SOAP-ENV کار می‌کند و فضای نام ns1 بعدی آن. کاری هم به اصول XML ندارد که باید بر اساس نام xmlns ذکر شده، کار Parse ورودی دریافتی صورت گیرد و نه بر اساس یک رشته‌ی ثابت از پیش تعیین شده. بنابراین باید راهی را پیدا کنیم تا بتوان این s را تبدیل به SOAP-ENV کرد.

برای این منظور به سه کلاس ذیل خواهیم رسید:

```
public class EndpointBehavior : IEndpointBehavior
{
    public void AddBindingParameters(ServiceEndpoint endpoint, BindingParameterCollection
bindingParameters)
    { }

    public void ApplyDispatchBehavior(ServiceEndpoint endpoint, EndpointDispatcher endpointDispatcher)
    { }

    public void Validate(ServiceEndpoint endpoint)
    { }

    public void ApplyClientBehavior(ServiceEndpoint endpoint, ClientRuntime clientRuntime)
    {
        clientRuntime.MessageInspectors.Add(new ClientMessageInspector());
    }
}

public class ClientMessageInspector : IClientMessageInspector
{
    public void AfterReceiveReply(ref Message reply, object correlationState)
    { }

    public object BeforeSendRequest(ref Message request, System.ServiceModel.IClientChannel channel)
    {
        request = new MyCustomMessage(request);
        return request;
    }
}

/// <summary>
/// To customize WCF envelope and namespace prefix
/// </summary>
public class MyCustomMessage : Message
{
    private readonly Message _message;

    public MyCustomMessage(Message message)
    {
        _message = message;
    }

    public override MessageHeaders Headers
    {
        get { return _message.Headers; }
    }

    public override MessageProperties Properties
    {
        get { return _message.Properties; }
    }

    public override MessageVersion Version
    {
        get { return _message.Version; }
    }

    protected override void OnWriteStartBody(XmlDictionaryWriter writer)
    {
```

```
        writer.WriteStartElement("Body", "http://schemas.xmlsoap.org/soap/envelope/");
    }

    protected override void OnWriteBodyContents(XmlDictionaryWriter writer)
    {
        _message.WriteBodyContents(writer);
    }

    protected override void OnWriteStartEnvelope(XmlDictionaryWriter writer)
    {
        writer.WriteStartElement("SOAP-ENV", "Envelope", "http://schemas.xmlsoap.org/soap/envelope/");
        writer.WriteAttributeString("xmlns", "ns1", null, value:
"http://disposableemailaddressdetector.email.services.v4_0.soap.server.nameapi.org/");
    }
}
```

که پس از تعریف client به نحو ذیل معرفی می‌شوند:

```
var client = new SoapDisposableEmailAddressDetectorClient();
client.Endpoint.Behaviors.Add(new EndpointBehavior());
```

توسط EndpointBehavior سفارشی، می‌توان به متد **Envelope** OnWriteStart دسترسی یافت و سپس s آن‌را با SOAP-ENV درخواستی این وب سرویس جایگزین کرد. اکنون اگر برنامه را اجرا کنید، بدون مشکل کار خواهد کرد و دیگر پیام یافت نشدن API-Key را صادر نمی‌کند.

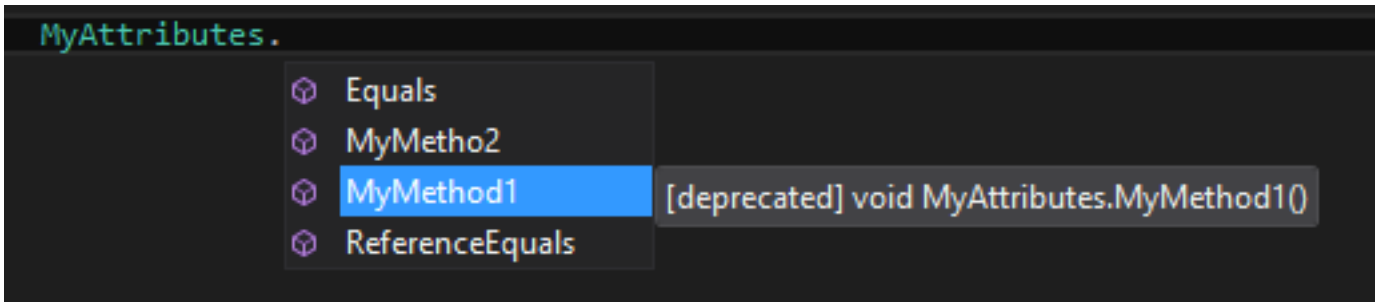
کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید.

عنوان: **ساخت Attribute های دلخواه یا خصوصی سازی شده**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۳/۰۲ ۲۲:۴۵
آدرس: www.dotnettips.info
گروه‌ها: C#, Validation, Reflection

در قسمت‌های مختلفی از منابع آموزشی این سایت از متادیتاها attributes استفاده شده و در برخی آموزش هایی چون EF و MVC حداقل یک قسمت کامل را به خود اختصاص داده‌اند. متادیتاها کلاس‌هایی هستند که به روشی سریع و کوتاه در بالای یک Type معرفی شده و ویژگی‌هایی را به آن اضافه می‌کنند. به عنوان مثال متادیتای زیر را ببینید. این متادیتا در بالای یک متد در یک کلاس تعریف شده است و این متد را منسوخ شده اعلام می‌کند و به برنامه نویس می‌گوید که در نسخه‌ی جاری کتابخانه، این متد که احتمال میرود در نسخه‌های پیشین کاربرد داشته است، الان کارآیی خوبی برای استفاده نداشته و بهتر است طبق مستندات آن کلاس، از یک متد جایگزین که برای آن فراهم شده است استفاده کند.

```
public static class  MyAttributes
    {
        [Obsolete]
        public static void MyMethod1()
        {

        }

        public static void MyMetho2()
        {

        }
    }
```

همانطور که ملاحظه می‌کنید می‌توانید اخطار آن را مشاهده کنید:



البته توصیه می‌کنم از ابزارهایی چون Resharper در کارهایتان استفاده کنید، تا طعم کدنویسی را بهتر بچشید. نحوه‌ی نمایش آن در Resharper به مراتب واضح‌تر و گویاتر است:

حال در این بین این سؤال پیش می‌آید که چگونه ما هم می‌توانیم متادیتاهایی را با سلیقه‌ی خود ایجاد کنیم.
برای تهیه‌ی یک متادیتا از کلاس system.attribute استفاده می‌کنیم:

```
public  class  MyMaxLength:Attribute
    {

    }
```

در چنین حالتی شما یک متادیتا ساخته‌اید که می‌توان از آن به شکل زیر استفاده کرد:

```
[MyMaxLength]
    public class GetCustomProperties
    {
//...
     }
```

ولی اگر بخواهید توسط این متادیتا اطلاعاتی را دریافت کنید، می‌توانید به روش زیر عمل کنید. در اینجا من دوست دارم یک متادیتا به اسم MyMaxLength را ایجاد کرده تا جایگزین MaxLength دات نت کنم، تا طبق میل من رفتار کند.

```
public  class  MyMaxLength:Attribute
    {
        private int max;
        public string ErrorText = "";

        public MyMaxLength(int max)
        {
            this.max = max;
            ErrorText = string.Format("max Length is {0} chars", max);
        }
    }
```

در کد بالا، یک متادیتا با یک پارامتر اجباری در سازنده تعریف شده است. این کلاس هم می‌تواند مثل سایر کلاس‌ها سازنده‌های مختلفی داشته باشد تا چندین شکل تعریف متادیتا داشته باشیم. متغیر ErrorText به عنوان یک پارامتر معرفی نشده، ولی از آن جا که public تعریف شده است می‌تواند مورد استفاده‌ی مستقیم قرار بگیرد و استفاده‌ی از آن نیز اختیاری است. نحوه‌ی معرفی این متادیتا نیز به صورت زیر است:

```
[MyMaxLength(30)]
    public class GetCustomProperties
    {
//...
     }

//or
 [MyMaxLength(30,ErrorText = "شما اجازه ندارید بیش از 30 کاراکتر وارد نمایید")]
    public class GetCustomProperties
    {
//...
     }
```

در حالت اول از آنجا که متغیر ErrorText اختیاری است، تعریف نشده‌است. پس در نتیجه با مقدار (x=max) Max length is chars پر خواهد شد ولی در حالت دوم برنامه نویس متن خطا را به خود کلاس واگذار نکرده است و آن را طبق میل خود تغییر داده است.

اجباری کردن Type

هر متادیتا می‌تواند مختص یک نوع Type باشد که این نوع می‌تواند یک کلاس، متد، پراپرتی یا ساختار و ... باشد. نحوه‌ی محدود سازی آن توسط یک متادیتا مشخص می‌شود:

```
[System.AttributeUsage(System.AttributeTargets.Class | System.AttributeTargets.Struct)]
    public  class  MyMaxLength:Attribute
    {
        private int max;
        public string ErrorText = "";

        public MyMaxLength(int max)
        {
            this.max = max;
            ErrorText = string.Format("max Length is {0} chars", max);
        }
    }
```

الان این کلاس توسط متادیتای AttributeUsage که پارامتر ورودی آن Enum است محدود به دو ساختار کلاس و Struct شده است. البته در ویژوال بیسیک با نام Structure معرفی شده است. اگر ساختار شمارشی AttributeTarget را مشاهده کنید، لیستی از نوع‌ها را چون All (همه موارد) ، دلیگیت، سازنده، متد و ... را مشاهده خواهید کرد و از آن جا که این متادیتای ما کاربردش در پراپرتی‌ها خلاصه می‌شود، از متادیتای زیر بر روی آن استفاده می‌کنیم:

```
[AttributeUsage(AttributeTargets.Property)]
```

```
public class User
    {
        [MyMaxLength(30, ErrorText = "شما اجازه ندارید بیش از 30 کاراکتر وارد نمایید")]
        public string Name { get; set; }
}
```

یکی دیگر از ویژگی‌های AttributeUsage خصوصیتی به اسم AllowMultiple است که اجازه می‌دهد بیش از یک بار این متادیتا، بر روی یک نوع استفاده شود:

```
[AttributeUsage(AttributeTargets.Property,AllowMultiple = true)]
    public  class  MyMaxLength:Attribute
    {
        //....
     }
```

که تعریف چندگانه آن به شکل زیر می‌شود:

```
[MyMaxLength(40, ErrorText = "شما اجازه ندارید بیش از 40 کاراکتر وارد نمایید")]
        [MyMaxLength(50, ErrorText = "شما اجازه ندارید بیش از 50 کاراکتر وارد نمایید")]
        [MyMaxLength(30, ErrorText = "شما اجازه ندارید بیش از 30 کاراکتر وارد نمایید")]
        public string Name { get; set; }
```

در این مثال ما فقط اجازه‌ی یکبار استفاده را خواهیم داد؛ پس مقدار این ویژگی را false قرار می‌دهم.

آخرین ویژگی که این متادیتا در دسترس ما قرار میدهد، استفاده از خصوصیت ارث بری است که به طور پیش فرض با True مقداردهی شده است. موقعی که شما یک متادیتا را به ویژگی ارث بری مزین کنید، در صورتی که آن کلاس که برایش متادیتا تعریف می‌کنید به عنوان والد مورد استفاده قرار بگیرد، فرزند آن هم به طور خودکار این متادیتا برایش منظور می‌گردد. به مثال‌های زیر دقت کنید:
دو عدد متادیتا تعریف شده که یکی از آن‌ها ارث بری در آن فعال شده و دیگری خیر.

```
public class MyAttribute : Attribute
{
    //...
}
```

```
[AttributeUsage(AttributeTargets.Method, Inherited = false)]
public class YourAttribute : Attribute
{
    //...
}
```

هر دو متادیتا بر سر یک متد در یک کلاسی که بعدا از آن ارث بری می‌شود تعریف شده اند.

```
public class MyClass
{
    [MyAttribute]
    [YourAttribute]
    public virtual void MyMethod()
    {
        //...
    }
}
```

در کد زیر کلاس بالا به عنوان والد معرفی شده و متد کلاس فرزند الان شامل متادیتایی به اسم MyAttribute است، ولی متادیتای YourAttribute بر روی آن تعریف نشده است.

```
public class YourClass : MyClass
{
    public override void MyMethod()
    {
        //...
    }

}
```

الان که با نحوه‌ی تعریف یکی از متادیتاها آشنا شدیم، این بحث پیش می‌آید که چگونه Type مورد نظر را تحت تاثیر این متادیتا قرار دهیم. الان چگونه میتوانم حداکثر متنی که یک پراپرتی می‌گیرد را کنترل کنم. در اینجا ما از مفهومی به نام Reflection استفاده می‌کنیم. با استفاده از این مفهوم ما میتوانیم به تمامی قسمت‌های یک Type دسترسی داشته باشیم. متاسفانه دسترسی مستقیمی از داخل کلاس متادیتا به نوع مورد نظر نداریم. کد زیر تمامی پراپرتی‌های یک کلاس را چک میکند و سپس ویژگی‌های هر پراپرتی را دنبال کرده و در صورتیکه متادیتای مورد نظر به آن پراپرتی ضمیمه شده باشد، حالا می‌توانید عملیات را انجام دهید. کد زیر میتواند در هر جایی نوشته شود. داخل کلاسی که به آن متادیتا ضمیمه می‌کنید یا داخل تابع Main در اپلیکشین‌ها و هر جای دیگر. مقدار True که به متد GetCustomAttributes پاس می‌شود باعث می‌شود تا متادیتاهای ارث بری شده هم لحاظ گردند.

```
Type type = typeof (User);

                foreach (PropertyInfo property in type.GetProperties())
                {
                    foreach (Attribute attribute in property.GetCustomAttributes(true))
                    {
                        MyMaxLength max = attribute as MyMaxLength;
                        if (max != null)
                        {
                            string Max = max.ErrorText;
                            //انجام عملیات
                        }
                    }
                }
```

البته یک ترفند جهت دسترسی به کلاس‌ها از داخل کلاس متادیتا وجود دارد و آن هم این هست که نوع را از طریق پارامتر به سمت متادیتا ارسال کنید. هر چند این کار زیبایی ندارد ولی به هر حال روش خوبی برای کنترل از داخل کلاس متادیتا و هچنین منظم سازی و دسته بندی و کم کردن کد دارد.

```
[MyMaxLength(30, typeof(User))]
```

عنوان: **ایجاد اعتبار سنجی های شرطی با Foolproof**
نویسنده: عثمان رحیمی
تاریخ: ۱۳۹۴/۰۳/۱۶ ۱۱:۳۰
آدرس: www.dotnettips.info
گروه‌ها: MVC, Validation, Data Annotation

ابتدا کلاس زیر را در نظر بگیرید:

```
public class UserVM
 {
        public string Name { get; set; }
        public bool  Gender { get; set; }
        public string Soldier { get; set; }
    }
```

قصد داریم یک سری اعتبار سنجی را بر روی خصوصیات کلاس فوق ایجاد کنیم. می‌خواهیم اگر کاربر جنسیت مرد را انتخاب کرد، حتما مقداری برای فیلد محل خدمت خود که در این کلاس Soldier می‌باشد، انتخاب کند. شاید انتخاب اول برای انجام چنین کاری، کنترل کردن آن در سمت کاربر با استفاده از جاوا اسکریپت باشد که می‌بایست یک رویداد را برای چک باکس جنسیت تعریف کنیم و بر اساس اینکه مرد انتخاب شده یا زن، ادامه کار را انجام دهیم.

روش اول: نوشتن یک کلاس سفارشی برای اعتبار سنجی کلاس فوق

```
public class SoldierValidation : ValidationAttribute
{
    public override bool IsValid(object value)
    {
        UserVM app = value as UserVM ;
        if (app.Gender && app.Soldier.Length==0)
        {
            ErrorMessage = "لطفا محل خدمت را وارد نمایید";

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

و سپس اعمال به کلاس مورد نظر همانند زیر :

```
[SoldierValidation ]
   public class UserVM
    {
        public string Name { get; set; }
        public bool  Gender { get; set; }
        public string Soldier { get; set; }
    }
```

تا اینجای کار، اگر کاربر از DropDown و یا RadioButton، آقا را انتخاب کرده باشد و View مورد نظر را برای Update و یا Insert ارسال کند، با خطای «لطفا محل خدمت را وارد نمایید» مواجه خواهد شد. تا به اینجا به مقصود مورد نظرمان رسیدیم.

روش دوم

لازم نیست چرخ رو دوباره اختراع کنید (البته در بعضی مواقع لازم است)

استفاده از Foolproof MVC:

Foolproof یک سری Annotation هایی را در اختیار شما قرار می‌دهد که با استفاده از آنها می‌توانید اعتبار سنجی‌های شرطی را انجام دهید؛ دقیقا همانند کاری که در بالا برای آن یک Validation سفارشی نوشتیم. البته Foolproof فقط به این مورد ختم نمی‌شود. در ادامه با چند مورد از آنها آشنا خواهیم شد.

ابتدا از طریق NuGet اقدام به نصب **Foolproof** نمایید:

```
PM> Install-Package foolproof
```

سپس اینبار همان مثال خود را با FoolPfoof انجام می‌دهیم:

```
public class UserVM
    {
        public string Name { get; set; }
        public bool  Gender { get; set; }
        [RequiredIfTrue("Gender ")]
        public string Soldier { get; set; }
    }
```

با استفاده از RequiredIfTrue دقیقا به همان مقصود خواهیم رسید که از ورودی، اسم فیلدی را می‌گیرد که می‌خواهیم آن را چک کنیم.

حال بپردازیم به چندین Annotation دیگر که در Foolproof وجود دارند:

GreatThan: همانطور که از نام آن پیداست، برای موقعیکه می‌خواهیم این فیلد بزرگتر از فیلد مورد نظرمان باشد:

```
public class EventViewModel
{
    public string Name { get; set; }
    public DateTime Start { get; set; }

    [Required]
    [GreaterThan("Start")]
    public DateTime End { get; set; }
}
```

جهت آشنایی بیشتر، در ادامه فقط لیست Annotationهای موجود در این پکیج قرار داده شده است .

```
[Is]
[EqualTo]
[NotEqualTo]
[GreaterThan]
[LessThan]
[GreaterThanOrEqualTo]
[LessThanOrEqualTo]
```

و

```
[RequiredIf]
[RequiredIfNot]
[RequiredIfTrue]
[RequiredIfFalse]
[RequiredIfEmpty]
[RequiredIfNotEmpty]
[RequiredIfRegExMatch]
[RequiredIfNotRegExMatch]
```

نظرات خوانندگان

نویسنده: مرتضی
تاریخ: ۱۳۹۴/۰۳/۱۶ ۱۲:۳۸

تشکر از مطلبتون

foolproof خیلی وقته بروزرسانی نشده

بهتره از https://github.com/JeremySkinner/FluentValidation استفاده کرد

عنوان: **تبدیل خودکار استثنای HttpRequestValidationException به یک ModelError در ASP.NET MVC**
نویسنده: سالمیان
تاریخ: ۱۳۹۴/۰۴/۳۰ ۱:۴۰
آدرس: www.dotnettips.info
گروه‌ها: MVC, Security, Validation

فرم هایی که اطلاعاتی را از یک کاربر دریافت کرده و به سمت سرور Post می‌کنند، از مهمترین اجزای لاینفک یک وب سایت می‌باشند. بی شک همه‌ی ما از چنین فرمهایی حتی در یک پروژه‌ی هرچند کوچک استفاده کرده‌ایم. ممکن است هنگام ارسال این فرمها، کاربری شیطنت به خرج داده و درون یکی از فیلدهای فرم، از عبارت‌های HTML و یا یک اسکریپت استفاده کرده باشد. در ASP.Net + MVC از مکانیزم ValidationRequest برای مقابله با چنین حملاتی (XSS) استفاده شده است.
یک فرم با یک Textbox در یک صفحه قرار دهید و درون Textbox یک تگ HTML بنویسید و آنرا به سرور ارسال کنید، در حالت پیش فرض اتفاقی که خواهد افتاد اینست که با یک صفحه‌ی خطا روبرو خواهید شد که به شما هشدار می‌دهد داده‌های ارسال شده، دارای مقادیر غیرمجازی می‌باشند. شما می‌توانید این مکانیزم اعتبارسنجی-امنیتی را غیرفعال کنید. برای غیرفعال کردن آن هم در لینکی که در فوق معرفی گردید توضیحات کافی داده شده است. ولی توصیه میشود برای جلوگیری از حملات XSS هیچگاه این مکانیزم را غیرفعال نکنید، مگر اینکه هیچ داده‌ای را بدون Encode کردن در صفحه نمایش ندهید.
راه‌های زیادی برای هندل کردن این خطا و مطلع کردن کاربر وجود دارند و معمولا از صفحه‌ی خطای سفارشی برای اینکار استفاده میشود. اما ایده‌ی بهتری نیز برای مقابله با این خطا وجود دارد!
فرض کنید اطلاعات فرم شما از طریق Ajax به سرور ارسال می‌شود و نتیجه بصورت Json به مروگر برگشت داده می‌شود. در حالت معمول در صورت رخ دادن خطای فوق، سرور کد 500 را برگشت می‌دهد و تنها راه هندل کردن آن در رویداد error شئ Ajax شما می‌باشد؛ آن‌هم با یک پیغام ساده. ولی من ترجیح می‌دهم به جای صادر کردن خطای 500، در صورت وقوع این خطا آن‌را بصورت یک خطای ModelState به کاربر نمایش دهم. به نظر من این‌کار وجهه‌ی بهتری دارد و ضمنا لازم نیست در هر رویدادی این خطا را هندل کنید. برای رسیدن به این هدف هم چندین راه وجود دارد که ساده‌ترین آن‌ها اینست که یک ModelBinder سفارشی ایجاد کنید و با هندل کردن این خطا، یک پیام خطا را به ModelState اضافه کنید و بعد به MVC بگویید که از این ModelBinder برای پروژه‌ی من استفاده کن. با این روش هرگاه کاربر داده‌ای حاوی کدهای HTML درون فیلدهای فرم وارد کرده باشد، بدون هیچ کار اضافه‌ای وضعیت ModelState شما Invalid می‌شود و خطای مدل به کاربر نمایش داده خواهد شد.
ابتدا کلاسی برای ModelBinder سفارشی خود ایجاد کنید و از کلاس DefaultModelBinder ارث بری کنید. سپس با بازنویسی متد BindModel آن منطق خود را پیاده سازی کنید:

```
using System.Web;
using System.Web.Mvc;
using System.Web.Helpers;
using System.Globalization;

namespace Parsnet
{
    public class ParsnetModelBinder : DefaultModelBinder
    {
        public override object BindModel(ControllerContext controllerContext, ModelBindingContext
bindingContext)
        {
            try
            {
                return base.BindModel(controllerContext, bindingContext);
            }
            catch (HttpRequestValidationException ex)
            {
                var modelState = new ModelState();
                modelState.Errors.Add("اطلاعات ارسالی شما دارای کدهای HTML می‌باشند.");
                var key = bindingContext.ModelName;
                var value =
controllerContext.RequestContext.HttpContext.Request.Unvalidated().Form[key];
                modelState.Value = new ValueProviderResult(value, value, CultureInfo.InvariantCulture);
                bindingContext.ModelState.Add(key, modelState);
            }

            return null;
        }
    }
}
```

در این کلاس ابتدا سعی میکنیم بطور عادی کار را به متد BindModel کلاس پایه بسپاریم. اگر داده‌های ارسال شده حاوی کد HTML

نباشد، بدون هیچ خطایی Model Binding صورت می‌گیرد. ولی در صورتیکه کاربر در فیلدی، از کدهای HTML استفاده کرده باشد، یک خطای HttpRequestValidationException رخ خواهد داد که با هندل کردن آن هدف خود را تامین می‌کنیم.
برای اینکار در قسمت catch بلوک مدیریت خطا، ابتدا یک نمونه از کلاس ModelState را میسازیم. بعد پیام خطای مورد نظر خود را به Errorsهای آن اضافه میکنیم. حال باید یک زوج کلید/مقدار برای این ModelState تعریف کنیم و به bindingContext اضافه کنیم. کلید ما در اینجا نام مدل جاری و مقدار آن هم نام فیلدی از فرم است که سبب بروز این خطا شده است.
حالا نهایت کاری که باید انجام دهیم اینست که در رویداد Application_Start این مدل بایندیگ سفارشی را جایگزین مدل بایندیگ پیش فرض کنیم:

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

کار تمام است. از حالا به بعد در صورتیکه کاربر در فیلدهای هر فرمی از سایت شما از کدهای HTML استفاده کند دیگر خطایی رخ نمی‌دهد و فقط ModelState در وضعیت Invalid قرار میگیرد که میتوانید با قرار دادن یک ValidationMessage یا ValidationSummary به راحتی خطا را به کاربر نشان دهید.