عنوان: ASP.NET MVC #13
نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۱/۲۲ ۲۲:۳۵:۰۰
آدرس: www.dotnettips.info
گروه‌ها: MVC

اعتبار سنجی اطلاعات ورودی در فرم‌های ASP.NET MVC

زمانیکه شروع به دریافت اطلاعات از کاربران کردیم، نیاز خواهد بود تا اعتبار اطلاعات ورودی را نیز ارزیابی کنیم. در ASP.NET MVC، به کمک یک سری متادیتا، نحوه‌ی اعتبار سنجی، تعریف شده و سپس فریم ورک بر اساس این ویژگی‌ها، به صورت خودکار اعتبار اطلاعات انتساب داده شده به خواص یک مدل را در سمت کلاینت و همچنین در سمت سرور بررسی می‌نماید.
این ویژگی‌ها در اسمبلی System.ComponentModel.DataAnnotations.dll قرار دارند که به صورت پیش فرض در هر پروژه جدید ASP.NET MVC لحاظ می‌شود.

یک مثال کاربردی

مدل زیر را به پوشه مدل‌های یک پروژه جدید خالی ASP.NET MVC اضافه کنید:

```
using System;
using System.ComponentModel.DataAnnotations;

namespace MvcApplication9.Models
{
    public class Customer
    {
        public int Id { set; get; }

        [Required(ErrorMessage = "Name is required.")]
        [StringLength(50)]
        public string Name { set; get; }

        [Display(Name = "Email address")]
        [Required(ErrorMessage = "Email address is required.")]
        [RegularExpression(@"\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*",
                           ErrorMessage = "Please enter a valid email address.")]
        public string Email { set; get; }

        [Range(0, 10)]
        [Required(ErrorMessage = "Rating is required.")]
        public double Rating { set; get; }

        [Display(Name = "Start date")]
        [Required(ErrorMessage = "Start date is required.")]
        public DateTime StartDate { set; get; }
    }
}
```

سپس کنترلر جدید زیر را نیز به برنامه اضافه نمائید:

```
using System.Web.Mvc;
using MvcApplication9.Models;

namespace MvcApplication9.Controllers
{
    public class CustomerController : Controller
    {
        [HttpGet]
        public ActionResult Create()
        {
            var customer = new Customer();
            return View(customer);
```

```
        }

        [HttpPost]
        public ActionResult Create(Customer customer)
        {
            if (this.ModelState.IsValid)
            {
                //todo: save data
                return Redirect("/");
            }
            return View(customer);
        }
    }
}
```

بر روی متد Create کلیک راست کرده و گزینه Add view را انتخاب کنید. در صفحه باز شده، گزینه Create a strongly typed view را انتخاب کرده و مدل را Customer انتخاب کنید. همچنین قالب Scaffolding را نیز بر روی Create قرار دهید.

توضیحات تکمیلی

همانطور که در مدل برنامه ملاحظه می‌نمائید، به کمک یک سری متادیتا یا اصطلاحا data annotations، تعاریف اعتبار سنجی، به همراه عبارات خطایی که باید به کاربر نمایش داده شوند، مشخص شده است. ویژگی Required مشخص می‌کند که کاربر مجبور است این فیلد را تکمیل کند. به کمک ویژگی StringLength، حداکثر تعداد حروف قابل قبول مشخص می‌شود. با استفاده از ویژگی RegularExpression، مقدار وارد شده با الگوی عبارت باقاعده مشخص گردیده، مقایسه شده و در صورت عدم تطابق، پیغام خطایی به کاربر نمایش داده خواهد شد. به کمک ویژگی Range، بازه اطلاعات قابل قبول، مشخص می‌گردد.
ویژگی دیگری نیز به نام System.Web.Mvc.Compare مهیا است که برای مقایسه بین مقادیر دو خاصیت کاربرد دارد. برای مثال در یک فرم ثبت نام، عموما از کاربر درخواست می‌شود که کلمه عبورش را دوبار وارد کند. ویژگی Compare در یک چنین مثالی کاربرد خواهد داشت.
در مورد جزئیات کنترلر تعریف شده در قسمت 11 مفصل توضیح داده شد. برای مثال خاصیت this.ModelState.IsValid مشخص می‌کند که آیا کارmodel binding موفق بوده یا خیر و همچنین اعتبار سنجی‌های تعریف شده نیز در اینجا تاثیر داده می‌شوند. بنابراین بررسی آن پیش از ذخیره سازی اطلاعات ضروری است.
در حالت HttpGet صفحه ورود اطلاعات به کاربر نمایش داده خواهد شد و در حالت HttpPost، اطلاعات وارد شده دریافت می‌گردد. اگر دست آخر، ModelState معتبر نبود، همان اطلاعات نادرست وارد شده به کاربر مجددا نمایش داده خواهد شد تا فرم پاک نشود و بتواند آن‌ها را اصلاح کند.
برنامه را اجرا کنید. با مراجعه به مسیر http://localhost/customer/create، صفحه ورود اطلاعات کاربر نمایش داده خواهد شد. در اینجا برای مثال در قسمت ورود اطلاعات آدرس ایمیل، مقدار abc را وارد کنید. بلافاصله خطای اعتبار سنجی عدم اعتبار مقدار ورودی نمایش داده می‌شود. یعنی فریم ورک، اعتبار سنجی سمت کاربر را نیز به صورت خودکار مهیا کرده است.
اگر علاقمند باشید که صرفا جهت آزمایش، اعتبار سنجی سمت کاربر را غیرفعال کنید، به فایل web.config برنامه مراجعه کرده و تنظیم زیر را تغییر دهید:

```
<appSettings>
    <add key="ClientValidationEnabled" value="true"/>
```

البته این تنظیم تاثیر سراسری دارد. اگر قصد داشته باشیم که این تنظیم را تنها به یک view خاص اعمال کنیم، می‌توان از متد زیر کمک گرفت:

```
@{ Html.EnableClientValidation(false); }
```

در این حالت اگر مجددا برنامه را اجرا کرده و اطلاعات نادرستی را وارد کنیم، باز هم همان خطاهای تعریف شده، به کاربر نمایش داده خواهد شد. اما اینبار یکبار رفت و برگشت اجباری به سرور صورت خواهد گرفت، زیرا اعتبار سنجی سمت کاربر (که درون مرورگر و توسط کدهای جاوا اسکریپتی اجرا می‌شود)، غیرفعال شده است. البته امکان غیرفعال کردن جاوا اسکریپت توسط کاربر نیز وجود دارد. به همین جهت بررسی خودکار سمت سرور، امنیت سیستم را بهبود خواهد بخشید.

نحوه تعریف عناصر مرتبط با اعتبار سنجی در Viewهای برنامه نیز به شکل زیر است:

```
<script src="@Url.Content("~/Scripts/jquery.validate.min.js")" type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/jquery.validate.unobtrusive.min.js")"
type="text/javascript"></script>

@using (Html.BeginForm()) {
    @Html.ValidationSummary(true)
    <fieldset>
        <legend>Customer</legend>

        <div class="editor-label">
            @Html.LabelFor(model => model.Name)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Name)
            @Html.ValidationMessageFor(model => model.Name)
        </div>
```

همانطور که ملاحظه می‌کنید به صورت پیش فرض از jQuery validator در سمت کلاینت استفاده شده است. فایل jquery.validate.unobtrusive متعلق به تیم ASP.NET MVC است و کار آن وفق دادن سیستم موجود، با jQuery validator می‌باشد (validation adapter). در نگارش‌های قبلی، از کتابخانه‌های اعتبار سنجی مایکروسافت استفاده شده بود، اما از نگارش سه به بعد، jQuery به عنوان کتابخانه برگزیده مطرح است.
Unobtrusive همچنین در اینجا به معنای مجزا سازی کدهای جاوا اسکریپتی، از سورس HTML صفحه و استفاده از ویژگی‌های data-* مرتبط با HTML5 برای معرفی اطلاعات مورد نیاز اعتبار سنجی است:

```
<input data-val="true" data-val-required="The Birthday field is required." id="Birthday"
name="Birthday" type="text" value="" />
```

اگر خواستید این مساله را بررسی کنید، فایل web.config قرار گرفته در ریشه اصلی برنامه را باز کنید. در آنجا مقدار UnobtrusiveJavaScriptEnabled را false کرده و بار دیگر برنامه را اجرا کنید. در این حالت کلیه کدهای اعتبار سنجی، به داخل سورس View رندر شده، تزریق می‌شوند و مجزا از آن نخواهند بود.
نحوه‌ی تعریف این اسکریپت‌ها نیز جالب توجه است. متد Url.Content، یک متد سمت سرور می‌باشد که در زمان اجرای برنامه، مسیر نسبی وارد شده را بر اساس ساختار سایت اصلاح می‌کند. حرف ~ بکارگرفته شده، در ASP.NET به معنای ریشه سایت است. بنابراین مسیر نسبی تعریف شده از ریشه سایت شروع و تفسیر می‌شود.
اگر از این متد استفاده نکنیم، مجبور خواهیم شد که مسیرهای نسبی را به شکل زیر تعریف کنیم:

```
<script src="../../Scripts/customvaildation.js" type="text/javascript"></script>
```

در این حالت بسته به محل قرارگیری صفحات و همچنین برنامه در سایت، ممکن است آدرس فوق صحیح باشد یا خیر. اما استفاده از متد Url.Content، کار مسیریابی نهایی را خودکار می‌کند.
البته اگر به فایل Views/Shared/_Layout.cshtml، مراجعه کنید، تعریف و الحاق کتابخانه اصلی jQuery در آنجا انجام شده است.

بنابراین می‌توان این دو تعریف دیگر مرتبط با اعتبار سنجی را به آن فایل هم منتقل کرد تا همه‌جا در دسترس باشند.
توسط متد Html.ValidationSummary، خطاهای اعتبار سنجی مدل که به صورت دستی اضافه شده باشند نمایش داده می‌شود.
این مورد در قسمت 11 توضیح داده شد (چون پارامتر آن true وارد شده، فقط خطاهای سطح مدل را نمایش می‌دهد).
متد Html.ValidationMessageFor، با توجه به متادیتای یک خاصیت و همچنین استثناهای صادر شده حین model binding خطایی را به کاربر نمایش خواهد داد.

اعتبار سنجی سفارشی

ویژگی‌های اعتبار سنجی از پیش تعریف شده، پر کاربردترین‌ها هستند؛ اما کافی نیستند. برای مثال در مدل فوق، StartDate نباید کمتر از سال 2000 وارد شود و همچنین در آینده هم نباید باشد. این موارد اعتبار سنجی سفارشی را چگونه باید با فریم ورک، یکپارچه کرد؟
حداقل دو روش برای حل این مساله وجود دارد:
الف) نوشتن یک ویژگی اعتبار سنجی سفارشی
ب) پیاده سازی اینترفیس IValidatableObject

تعریف یک ویژگی اعتبار سنجی سفارشی

```
using System;
using System.ComponentModel.DataAnnotations;

namespace MvcApplication9.CustomValidators
{
    public class MyDateValidator : ValidationAttribute
    {
        public int MinYear { set; get; }

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

            var date = (DateTime)value;
            if (date > DateTime.Now || date < new DateTime(MinYear, 1, 1))
                return false;

            return true;
        }
    }
}
```

برای نوشتن یک ویژگی اعتبار سنجی سفارشی، با ارث بری از کلاس ValidationAttribute شروع می‌کنیم. سپس باید متد IsValid آن‌را تحریف کنیم. اگر این متد false برگرداند به معنای شکست اعتبار سنجی می‌باشد.
در ادامه برای بکارگیری آن خواهیم داشت:

```
[Display(Name = "Start date")]
[Required(ErrorMessage = "Start date is required.")]
[MyDateValidator(MinYear = 2000,
                 ErrorMessage = "Please enter a valid date.")]
public DateTime StartDate { set; get; }
```

اکنون مجددا برنامه را اجرا نمائید. اگر تاریخ غیرمعتبری وارد شود، اعتبار سنجی سمت سرور رخ داده و سپس نتیجه به کاربر نمایش داده می‌شود.

اعتبار سنجی سفارشی به کمک پیاده سازی اینترفیس IValidatableObject

یک سؤال: اگر اعتبار سنجی ما پیچیده‌تر باشد چطور؟ مثلا نیاز باشد مقادیر دریافتی چندین خاصیت با هم مقایسه شده و سپس بر این اساس تصمیم گیری شود. برای حل این مشکل می‌توان از اینترفیس IValidatableObject کمک گرفت. در این حالت مدل تعریف شده باید اینترفیس یاد شده را پیاده سازی نماید. برای مثال:

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using MvcApplication9.CustomValidators;

namespace MvcApplication9.Models
{
    public class Customer : IValidatableObject
    {
        //... same as before

        public IEnumerable<ValidationResult> Validate(ValidationContext validationContext)
        {
            var fields = new[] { "StartDate" };
            if (StartDate > DateTime.Now || StartDate < new DateTime(2000, 1, 1))
                yield return new ValidationResult("Please enter a valid date.", fields);

            if (Rating > 4 && StartDate < new DateTime(2003, 1, 1))
                yield return new ValidationResult("Accepted date should be greater than 2003", fields);
        }
    }
}
```

در اینجا در متد Validate، فرصت خواهیم داشت تا به مقادیر کلیه خواص تعریف شده در مدل دسترسی پیدا کرده و بر این اساس اعتبار سنجی بهتری را انجام دهیم. اگر اطلاعات وارد شده مطابق منطق مورد نظر نباشند، کافی است توسط yield return new ValidationResult، یک پیغام را به همراه فیلدهایی که باید این پیغام را نمایش دهند، بازگردانیم.
به این نوع مدل‌ها، self validating models هم گفته می‌شود.

یک نکته:

از MVC3 به بعد، حین کار با ValidationAttribute، امکان تحریف متد IsValid به همراه پارامتری از نوع ValidationContext نیز وجود دارد. به این ترتیب می‌توان به اطلاعات سایر خواص نیز دست یافت. البته در این حالت نیاز به استفاده از Reflection خواهد بود و پیاده سازی IValidatableObject، طبیعی‌تر به نظر می‌رسد:

```
protected override ValidationResult IsValid(object value, ValidationContext validationContext)
{
    var info = validationContext.ObjectType.GetProperty("Rating");
    //...
    return ValidationResult.Success;
}
```

فعال سازی سمت کلاینت اعتبار سنجی‌های سفارشی

اعتبار سنجی‌های سفارشی تولید شده تا به اینجا، تنها سمت سرور است که فعال می‌شوند. به عبارتی باید یکبار اطلاعات به سرور ارسال شده و در بازگشت، نتیجه عملیات به کاربر نمایش داده خواهد شد. اما ویژگی‌های توکاری مانند Required و Range و امثال آن، علاوه بر سمت سرور، سمت کاربر هم فعال هستند و اگر جاوا اسکریپت در مرورگر کاربر غیرفعال نشده باشد، نیازی به ارسال اطلاعات یک فرم به سرور جهت اعتبار سنجی اولیه، نخواهد بود.
در اینجا باید سه مرحله برای پیاده سازی اعتبار سنجی سمت کلاینت طی شود:
الف) ویژگی سفارشی اعتبار سنجی تعریف شده باید اینترفیس IClientValidatable را پیاده سازی کند.

ب) سپس باید متد jQuery validation متناظر را پیاده سازی کرد.
ج) و همچنین مانند تیم ASP.NET MVC، باید unobtrusive adapter خود را نیز پیاده سازی کنیم. به این ترتیب متادیتای ASP.NET MVC به فرمتی که افزونه jQuery validator آن‌را درک می‌کند، وفق داده خواهد شد.

در ادامه، تکمیل کلاس سفارشی MyDateValidator را ادامه خواهیم داد:

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

namespace MvcApplication9.CustomValidators
{
    public class MyDateValidator : ValidationAttribute, IClientValidatable
    {
       // ... same as before

        public IEnumerable<ModelClientValidationRule> GetClientValidationRules(
                    ModelMetadata metadata,
                    ControllerContext context)
        {
            var rule = new ModelClientValidationRule
            {
                ValidationType = "mydatevalidator",
                ErrorMessage = FormatErrorMessage(metadata.GetDisplayName())
            };
            yield return rule;
        }
    }
}
```

در اینجا نحوه پیاده سازی اینترفیس IClientValidatable را ملاحظه می‌نمائید. ValidationType، نام متدی خواهد بود که در سمت کلاینت، کار بررسی اعتبار داده‌ها را به عهده خواهد گرفت.
سپس برای مثال یک فایل جدید به نام customvaildation.js به پوشه اسکریپت‌های برنامه با محتوای زیر اضافه خواهیم کرد:

```
/// <reference path="jquery-1.5.1-vsdoc.js" />
/// <reference path="jquery.validate-vsdoc.js" />
/// <reference path="jquery.validate.unobtrusive.js" />

jQuery.validator.addMethod("mydatevalidator",
 function (value, element, param) {
     return Date.parse(value) < new Date();
});

jQuery.validator.unobtrusive.adapters.addBool("mydatevalidator");
```

توسط referenceهایی که مشاهده می‌کنید، intellisense جی‌کوئری در VS.NET فعال می‌شود.
سپس به کمک متد jQuery.validator.addMethod، همان مقدار ValidationType پیشین را معرفی و در ادامه بر اساس مقدار value دریافتی، تصمیم گیری خواهیم کرد. اگر خروجی false باشد، به معنای شکست اعتبار سنجی است.
همچنین توسط متد jQuery.validator.unobtrusive.adapters.addBool، این متد جدید را به مجموعه وفق دهنده‌ها اضافه می‌کنیم.
و در آخر این فایل جدید باید به View مورد نظر یا فایل master page سیستم اضافه شود:

```
<script src="@Url.Content("~/Scripts/customvaildation.js")" type="text/javascript"></script>
```

تغییر رنگ و ظاهر پیغام‌های اعتبار سنجی

اگر از رنگ پیش فرض قرمز پیغام‌های اعتبار سنجی خرسند نیستید، باید اندکی CSS سایت را ویرایش کرد که شامل اعمال تغییرات به موارد ذیل خواهد شد:

```
1. .field-validation-error
2. .field-validation-valid
3. .input-validation-error
4. .input-validation-valid
5. .validation-summary-errors
6. .validation-summary-valid
```

نحوه جدا سازی تعاریف متادیتا از کلاس‌های مدل برنامه

فرض کنید مدل‌های برنامه شما به کمک یک code generator تولید می‌شوند. در این حالت هرگونه ویژگی اضافی تعریف شده در این کلاس‌ها پس از تولید مجدد کدها از دست خواهند رفت. به همین منظور امکان تعریف مجزای متادیتاها نیز پیش بینی شده است:

```
[MetadataType(typeof(CustomerMetadata))]
public partial class Customer
{
    class CustomerMetadata
     {

     }
}

public partial class Customer : IValidatableObject
{
```

حالت کلی روش انجام آن هم به شکلی است که ملاحظه می‌کنید. کلاس اصلی، به صورت partial معرفی خواهد شد. سپس کلاس partial دیگری نیز به همین نام که در برگیرنده یک کلاس داخلی دیگر برای تعاریف متادیتا است، به پروژه اضافه می‌گردد. به کمک ویژگی MetadataType، کلاسی که قرار است ویژگی‌های خواص از آن خوانده شود، معرفی می‌گردد. موارد عنوان شده، شکل کلی این پیاده سازی است. برای نمونه اگر با WCF RIA Services کار کرده باشید، از این روش زیاد استفاده می‌شود. کلاس خصوصی تو در توی تعریف شده صرفا وظیفه ارائه متادیتاهای تعریف شده را به فریم ورک خواهد داشت و هیچ کاربرد دیگری ندارد.
در ادامه کلیه خواص کلاس Customer به همراه متادیتای آن‌ها باید به کلاس CustomerMetadata منتقل شوند. اکنون می‌توان تمام متادیتای کلاس اصلی Customer را حذف کرد.

اعتبار سنجی از راه دور (remote validation)

فرض کنید شخصی مشغول به پر کردن فرم ثبت نام، در سایت شما است. پس از اینکه نام کاربری دلخواه خود را وارد کرد و مثلا به فیلد ورود کلمه عبور رسید، در همین حال و بدون ارسال کل صفحه به سرور، به او پیغام دهیم که نام کاربری وارد شده، هم اکنون توسط شخص دیگری در حال استفاده است. این مکانیزم از ASP.NET MVC3 به بعد تحت عنوان Remote validation در دسترس است و یک درخواست Ajaxایی خودکار را به سرور ارسال خواهد کرد و نتیجه نهایی را به کاربر نمایش می‌دهد؛ کارهایی که به سادگی توسط کدهای جاوا اسکریپتی قابل مدیریت نیستند و نیاز به تعامل با سرور، در این بین وجود دارد. پیاده سازی آن

هم به نحو زیر است:
برای مثال خاصیت Name را در مدل برنامه به نحو زیر تغییر دهید:

```
[Required(ErrorMessage = "Name is required.")]
[StringLength(50)]
[System.Web.Mvc.Remote(action: "CheckUserNameAndEmail",
                               controller: "Customer",
                               AdditionalFields = "Email",
                               HttpMethod = "POST",
                               ErrorMessage = "Username is not available.")]
public string Name { set; get; }
```

سپس متد زیر را نیز به کنترلر Customer اضافه کنید:

```
[HttpPost]
[OutputCache(Location = OutputCacheLocation.None, NoStore = true)]
public ActionResult CheckUserNameAndEmail(string name, string email)
{
      if (name.ToLowerInvariant() == "vahid") return Json(false);
      if (email.ToLowerInvariant() == "name@site.com") return Json(false);
      //...
      return Json(true);
}
```

توضیحات:
توسط ویژگی System.Web.Mvc.Remote، نام کنترلر و متدی که در آن قرار است به صورت خودکار توسط jQuery Ajax فراخوانی شود، مشخص خواهند شد. همچنین اگر نیاز بود فیلدهای دیگری نیز به این متد کنترلر ارسال شوند، می‌توان آن‌ها را توسط خاصیت AdditionalFields، مشخص کرد.
سپس در کدهای کنترلر مشخص شده، متدی با پارامترهای خاصیت مورد نظر و فیلدهای اضافی دیگر، تعریف می‌شود. در اینجا فرصت خواهیم داشت تا برای مثال پس از بررسی بانک اطلاعاتی، خروجی Json ایی را بازگردانیم. return Json false به معنای شکست اعتبار سنجی است.
توسط ویژگی OutputCache، از کش شدن نتیجه درخواست‌های Ajaxایی جلوگیری کرده‌ایم. همچنین نوع درخواست هم جهت امنیت بیشتر، به HttpPost محدود شده است.
تمام کاری که باید انجام شود همین مقدار است و مابقی مسایل مرتبط با اعمال و پیاده سازی آن خودکار است.

استفاده از مکانیزم اعتبار سنجی مبتنی برمتادیتا در خارج از ASP.Net MVC

مباحثی را که در این قسمت ملاحظه نمودید، منحصر به ASP.NET MVC نیستند. برای نمونه توسط متد الحاقی زیر نیز می‌توان یک مدل را مثلا در یک برنامه کنسول هم اعتبار سنجی کرد. بدیهی است در این حالت نیاز خواهد بود تا ارجاعی را به اسمبلی System.ComponentModel.DataAnnotations، به برنامه اضافه کنیم و تمام عملیات هم دستی است و فریم ورک ویژه‌ای هم وجود ندارد تا یک سری از کارها را به صورت خودکار انجام دهد.

```
using System.ComponentModel.DataAnnotations;

namespace MvcApplication9.Helper
{
    public static class ValidationHelper
    {
        public static bool TryValidateObject(this object instance)
        {
            return Validator.TryValidateObject(instance, new ValidationContext(instance, null, null),
```

```
null);
            }
        }
}
```

نظرات خوانندگان

نویسنده: Queryplus
تاریخ: ۱۳۹۱/۰۱/۲۴ ۱۰:۵۰:۵۲

سلام مهندس نصیری
در قسمت remote validation زمانی که رویداد blur مربوط به کنترل مورد نظر اتفاق افتاد یک درخواست به صورت ajax ارسال می شود آیا امکان تغییر رویداد مورد نظر امکان پذیر می باشد

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۱/۲۴ ۱۲:۰۳:۵۳

بله. نیاز هست که options مربوط به jQuery validator مانند onclick، onkeyup، onfocusout و غیره مقدار دهی شوند: (^) و (^)
البته حالت پیش فرض آن جهت کم کردن بار ارسالی به سرور در نظر گرفته شده که صحیح است.

نویسنده: _mahtab_
تاریخ: ۱۳۹۱/۰۳/۲۹ ۱۴:۵۵

سلام مهندس
دارم یه برنامه با mvc مینویسم که یه WebGrid دارم که تو WebGird هم Edit اطلاعات رو انجام میدم.حالا به این نیاز دارم که وقتی که ویرایش اطلاعات انجام می‌شه Remote validation رو اعمال کنم.
ممنون میشم اگه راهنمایم کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۳/۲۹ ۱۶:۱

برخلاف وب فرم‌ها شما در MVC محدود به یک فرم در صفحه نیستید. این مساله سبب میشه که بتوان اعتبار سنجی یک مدل را به ازای هر فرم تعریف شده در صفحه به صورت جداگانه انجام داد. به این ترتیب بجای اینکه کل گرید را در یک فرم تعریف کنید، هر سطر آن باید در یک فرم قرار گیرد یا قسمتی که قرار است به سرور ارسال شود باید در یک فرم قرار گیرد. به این صورت اعتبار سنجی از راه دور توضیح داده شده در بالا بدون مشکل کار خواهد کرد چون الان به ازای هر قسمتی که قرار است ویرایش شود یک فرم دارید و اطلاعات مدل متناظر با آن فرم به یک action method ارسال خواهد شد. مابقی مسایل یکی است و فرقی نمی‌کند.

نویسنده: عزیزی
تاریخ: ۱۳۹۱/۰۴/۱۳ ۱۶:۰

برای اعتبارسنجی تاریخ شمسی در mvc و در بالای view model میتونید از این عبارت استفاده کنید:

```
"^\\d{4}/\\d{2}/\\d{2}$"
```

نویسنده: محسن
تاریخ: ۱۳۹۱/۰۴/۱۴ ۱۶:۳۴

سلام آقای نصیری. قبل از اینکه سوالمو بپرسم باید بگم ممنون از مطلب خوبتون
آقای نصیری همونطور که اینجا گفتین من توسط MetaData برای فیلد Username داخل پروژه RemoteValidation رو فعال کردم .
برای Insert مشکلی نیست و درست عمل میکنه. اما موقعی که میخوام اطلاعات یک User رو ویرایش کنم، مجدد این اعتبار سنجی

فعال میشه و میگه که این Username وجود داره. برای غیرفعال کردنش در ویرایش باید چیکار کنم؟ ممنون میشم راهنماییم کنید

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۱۴ ۱۶:۴۵

باید برای حالت ویرایش از یک ViewModel جدید و از یک متد اعتبار سنجی جدید که می‌تونه کوئری صحیح بر اساس کاربر جاری لاگین شده به سیستم و کاربرهای ثبت شده در بانک اطلاعاتی بگیره، استفاده کنید. غیرفعال کردنش معنی نداره. چون سمت سرور باید یکبار دیگر همین مراحل رو پیش از ثبت نهایی بررسی کنید و در صورت لزوم پیغام لازم رو به کاربر بدید.

نویسنده: محسن
تاریخ: ۱۳۹۱/۰۴/۲۰ ۱۰:۴۰

سلام آقای نصیری خسته نباشید
من الان واسه ثبت کاربر توی دیتابیس یه کلاس دارم به اسم CustomProfile که تمامی خصوصیت‌های یک کاربر رو توی یه شیئ از این کلاس میریزم و بعد توی کدهای لایه مدل، این شیئ رو توسط کلاس Membership و ProfileBase خود دات نت توی پایگاه داده میریزیم. الان منظور شما اینه که من باید برای ویرایش یک کلاس CustomProfile دیگه با همون مشخصات بسازم و متد اعتبار سنجی برای Username اون رو یه متد دیگه قرار بدم ؟ ببینید کدهای من اینه:

```
public class CustomProfile
    {
        [Required(ErrorMessage="*")]
        [StringLength(50)]
        [System.Web.Mvc.Remote(action: "CheckUserName",
                                       controller: "User",
                                       HttpMethod = "POST",
                                       ErrorMessage = "این نام کاربری وجود دارد")]
        public string Username{ get; set; }
        [Required(ErrorMessage = "*")]
        public string Password{ get; set; }
        [Required(ErrorMessage = "*")]
        public string FirstName{ get; set; }
        [Required(ErrorMessage = "*")]
        public string LastName{ get; set; }

        [Required(ErrorMessage = "*")]
        [StringLength(10, ErrorMessage = "کدملی باید 10 رقم باشد")]
        public string NationalCode { get; set; }
        [Required(ErrorMessage = "*")]
        public string Address { get; set; }

        [RegularExpression(@"\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*",
                           ErrorMessage = "آدرس ایمیل معتبر نیست")]
        [System.Web.Mvc.Remote(action: "CheckEmail",
                                       controller: "User",
                                       AdditionalFields = "Username",
                                       HttpMethod = "POST",
                                       ErrorMessage = "این ایمیل وجود دارد")]
        public string Email{ get; set; }

        [StringLength(11, ErrorMessage = "تلفن باید 11 رقم باشد")]
        public string PhoneNo{ get; set; }

        [StringLength(11, ErrorMessage = "موبایل باید 11 رقم باشد")]
        public string MobileNo{ get; set; }


        public string[] Roles{ get; set; }

        public string LastActivityDate{ get; set; }
        public string LastLoginDate { get; set; }
        public string CreationDate { get; set; }
        public bool IsLockedOut{ get; set; }
    }
```

این کد کلاسم بود. من توی View هام اومدم و این کلاس را به عنوان ViewModel خودم قرار دادم. و وقتی فرم Create و یا Edit رو Submit میکنم متد اعتبار سنجی من وارد عمل میشه

اینم کدViewها:

```
@model Sama.Models.CustomProfile
@{
    ViewBag.Title = "Create";
    Layout = "~/Views/Shared/_Layout.cshtml";

}
<script src="@Url.Content("~/Scripts/jquery.validate.min.js")" type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/jquery.validate.unobtrusive.min.js")"
type="text/javascript"></script>
@using (Html.BeginForm(actionName: "Create", controllerName: "User"))
    {
        @Html.ValidationSummary(true)

        <fieldset>
            <legend>کاربر جدید</legend>
            <div>
                <table>
                    <tr>
                        <td>
                            <div>
                                @Html.LabelFor(model => model.Username, "نام کاربری")
                            </div>
                        </td>
                        <td>
                            <div>
                                @Html.EditorFor(model => model.Username)
                                @Html.ValidationMessageFor(model => model.Username)
                            </div>
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <div>
                                @Html.LabelFor(model => model.Password, "کلمه عبور")
                            </div>
                        </td>
                        <td>
                            <div>
                                @Html.PasswordFor(model => model.Password)
                                @Html.ValidationMessageFor(model => model.Password)
                            </div>
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <div>
                                @Html.LabelFor(model => model.FirstName, "نام")
                            </div>
                        </td>
                        <td>
                            <div>
                                @Html.EditorFor(model => model.FirstName)
                                @Html.ValidationMessageFor(model => model.FirstName)
                            </div>
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <div>
                                @Html.LabelFor(model => model.LastName, "نام خانوادگی")
                            </div>
                        </td>
                        <td>
                            <div>
                                @Html.EditorFor(model => model.LastName)
                                @Html.ValidationMessageFor(model => model.LastName)
                            </div>
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <div>
                                @Html.LabelFor(model => model.NationalCode, "کد ملی")
                            </div>
                        </td>
                        <td>
                            <div>
                                @Html.EditorFor(model => model.NationalCode)
```

```
                            @Html.ValidationMessageFor(model => model.NationalCode)
                        </div>
                    </td>
                </tr>
                <tr>
                    <td>
                        <div>
                            @Html.LabelFor(model => model.Email, "ایمیل")
                        </div>
                    </td>
                    <td>
                        <div>
                            @Html.EditorFor(model => model.Email)
                            @Html.ValidationMessageFor(model => model.Email)
                        </div>
                    </td>
                </tr>
                <tr>
                    <td>
                        <div>
                            @Html.LabelFor(model => model.PhoneNo, "تلفن")
                        </div>
                    </td>
                    <td>
                        <div>
                            @Html.EditorFor(model => model.PhoneNo)
                            @Html.ValidationMessageFor(model => model.PhoneNo)
                        </div>
                    </td>
                </tr>
                <tr>
                    <td>
                        <div>
                            @Html.LabelFor(model => model.MobileNo, "موبایل")
                        </div>
                    </td>
                    <td>
                        <div>
                            @Html.EditorFor(model => model.MobileNo)
                            @Html.ValidationMessageFor(model => model.MobileNo)
                        </div>
                    </td>
                </tr>
                <tr>
                    <td>
                        <div>
                            @Html.LabelFor(model => model.Address, "آدرس")
                        </div>
                    </td>
                    <td>
                        <div>
                            @Html.TextAreaFor(model => model.Address, 5, 30, null)
                            @Html.ValidationMessageFor(model => model.Address)
                        </div>
                    </td>
                </tr>
                <tr>
                    <td>
                        نقش‌ها </td>
                    <td>
                        @Helper.CheckBoxList("Roles", (List<SelectListItem>)ViewBag.Roles)
                    </td>
                </tr>
                <tr>
                    <td>
                    </td>
                    <td>
                        <input type="submit" value="ذخیره" class="btn btn-primary" />
                    </td>
                </tr>
            </table>
        </div>
    </fieldset>

}
```

این View برای Create بود و برای Edit هم مشابه همینه

حالا اگر من درست متوجه شده باشم باید برای Edit ، از یک ViewModel دیگه مثلا CustomProfileEdit استفاده کنم و اونجا متد

اعتبار سنجی دیگه ای تعریف کنم. منظور شما همین بود ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۲۰ ۱۲:۵

در قسمت‌های قبل مطرح کردم که می‌شود attributes را کلا از کلاس مدل خارج کرد:

```
[MetadataType(typeof(Customer_Validation))]
public partial class Customer
{
}
public class Customer_Validation
{
}
```

به این ترتیب می‌شود از یک مدل با ارث بری چندین viewModel درست کرد. بعد متادیتای آن‌را به صورت جداگانه اعمال کرد.

نویسنده: محمدرضا
تاریخ: ۱۳۹۱/۰۵/۱۹ ۱۴:۴۹

میخواستم بدون آیا استفاده از Data Annotationها توی همون کلاسای Data Access Layer مثلاً وقتی از Code First استفاده میکنیم کار درستی هست؟
یا بهتره یه کلاس دیگه (ViewModel) ساخت و توی اون Data Annotationها رو تعریف کرد و بعد با Auto Mapper به هم Map کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۱۹ ۱۷:۳۴

بله. مشکلی نداره. در EF Code first از Data Annotationها حداقل به سه منظور استفاده میشه:
الف) کنترل ساختار دیتابیس تشکیل شده. مثلا طول فیلد رشته‌ای چقدر باشد.
ب) اعتبار سنجی سمت سرور. اگر فیلدی رو required تعریف کردید، هم به صورت not null در سمت بانک اطلاعاتی تشکیل خواهد شد و هم پیش از ثبت، توسط EF به صورت خودکار اعتبار سنجی می‌شود.
ج) تعریف روابط بین جداول. مثلا می‌شود توسط آن‌ها کلید خارجی را تعریف کرد و مواردی از این دست.

ViewModel هم باید Data Annotation مختص به خودش را داشته باشد. حداقل روی اعتبار سنجی سمت کلاینت می‌تونه تاثیرگذار باشه چون به صورت خودکار توسط MVC اعمال می‌شود.

نویسنده: رضا
تاریخ: ۱۳۹۱/۰۵/۱۹ ۱۸:۳۱

ممنونم آقای نصیری. من برای کنترل ساختار دیتابیس و تعریف روابط برای هر Entity، به طور کامل از EntityTypeConfiguration استفاده میکنم.
حالا با این وجود کار درست و اصولی برای اعتبار سنجی سمت کلاینت کدومه؟
استفاده از ViewModel برای به کار بردن Data Annotationها و Map کردن به کلاسی که برای EF درست کردیم، یا روی همون کلاس‌های EF Code First؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۱۹ ۱۸:۳۹

امکانات Data Annotationها کار تیم EF یا به عبارتی طراحان اصلی دات نت است. بنابراین کمی زیاده روی است که عنوان کنیم غیراصولی هستند. شما در NHibernate چندین و چند روش تعریف نگاشت‌ها و روابط را دارید. از فایل‌های XML تا Data Annotationها تا Fluent NH تا Mapping by code اخیر آن و غیره. تمام این‌ها هست برای برآوردن سلایق مختلف.
در EF Code first هم به همین ترتیب. شما حق انتخاب دارید. به شخصه از ترکیب هر دو حالت Data Annotationها و Fluent

API استفاده می‌کنم.
ViewModel فقط بحث مدیریت صحیح ارتباط با کلاینت است (نمایش اطلاعات View) و برعکس (دریافت اطلاعات از کاربر).
بنابراین اگر از ViewModel استفاده می‌کنید، نیاز است از Data Annotationها استفاده کنید تا اعتبار سنجی سمت کاربر (که به صورت خودکار توسط MVC اعمال و مدیریت می‌شود) کار کند.

نویسنده: طاهریان
تاریخ: ۱۳۹۱/۰۷/۱۱ ۲۲:۲۴

سلام
برداشت من از صحبت شما این هست که زمانی که از ViewModel استفاده می‌کنیم DataAnotation رو باید داخل ViewModel تعریف کنیم.
با این کار اصل DRP زیر سوال نمیره؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۷/۲۰ ۰:۲۶

آ یا استفاده از ویژگی‌ها (Attributes یا Data Annotations) خلاف SRP یا Single Responsibility Principle کلاس‌ها است؟
پاسخ: خیر.
توضیحات:
این سؤال از اینجا ناشی می‌شود که چون برای مثال با استفاده از ویژگی StringLength می‌توان طول یک خاصیت رشته‌ای را مشخص کرد و اگر طول وارد شده توسط کاربر بیش از مقدار تعیین شده باشد، استثنایی صادر می‌شود یا مکانیزم‌های اعتبار سنجی سمت کاربر فعال خواهند شد، بنابراین کلاس تعریف شده در این حالت بیش از یک مسئولیت را عهده دار شده است.
این طرز برداشت صحیح نیست، زیرا ویژگی‌ها (Attributes) هیچ نوع کدی را به کلاس جاری تزریق نمی‌کنند؛ بلکه این فریم ورک خارجی مورد استفاده است که قابلیت پردازش و تصمیم‌گیری نهایی را بر اساس متادیتای تعیین شده، دارد. برای مثال این ویژگی‌ها را در یک برنامه ساده کنسول تعریف کنید. هیچ اتفاقی رخ نخواهد داد. زیرا در این حالت متادیتای تعریف شده توسط کتابخانه خارجی خاصی مورد استفاده قرار نمی‌گیرد.
با استفاده از ویژگی‌ها عملکردی به کلاس مورد نظر اضافه نمی‌شود؛ بلکه فقط نشانه گذاری صورت می‌گیرد. این نشانه گذاری هم تنها در صورتیکه یک کتابخانه خارجی که قابلیت درک آن‌را دارد وجود خارجی داشته باشد، مورد استفاده قرار خواهد گرفت.
بنابراین استفاده از ویژگی‌ها ناقض SRP نیست و در اینجا مسئولیت پردازش نشانه گذاری‌های انجام شده به یک کلاس یا فریم ورک خارجی واگذار می‌شود.

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۳۹۱/۱۱/۰۵ ۱۱:۲۷

با سلام.
در حالت زیر در هنگام submit همواره صفحه رفرش می‌شود و بعد از رفرش صفحه خطاها را نشان می‌دهد و اعتبارسنجی سمت کلاینت کار نمی‌کند؟

```
@model Models.Account
@{
    Layout = null;
    ViewBag.Title = "ورود به سیستم";
}

<script src="@Url.Content("~/Scripts/jquery-1.7.1.min.js")" type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/jquery.validate.min.js")" type="text/javascript"></script>
<script src="@Url.Content("~/Scripts/jquery.validate.unobtrusive.min.js")"
type="text/javascript"></script>

@using (Html.BeginForm())
{
     @Html.ValidationSummary(true)
    <fieldset>
        <legend>Login</legend>
```

```
        <table style="font-size: 8pt">
            <tr>
                <td style="width: 100px; text-align: left">نام کاربری :</td>
                <td>@Html.EditorFor(model => model.Username)</td>
            </tr>
            <tr>
                <td></td>
                <td style="color: red">@Html.ValidationMessageFor(model => model.Username)</td>
            </tr>
            <tr>
                <td style="width: 100px; text-align: left">کلمه عبور :</td>
                <td>@Html.EditorFor(model => model.Password)</td>
            </tr>
            <tr>
                <td></td>
                <td style="color: red">@Html.ValidationMessageFor(model => model.Password)</td>
            </tr>

            <tr>
                <td></td>
                <td>
                    <input type="submit" value="ورود به سیستم" /></td>
            </tr>
        </table>
    </fieldset>
}
```

```
public class Account
    {
        [Required(ErrorMessage = "نام کاربری باید وارد شود.")]
        [StringLength(20)]
        public string Username { get; set; }
        [Required(ErrorMessage = "کلمه عبور باید وارد شود.")]
        [DataType(DataType.Password)]
        public string Password { get; set; }
    }
```

```
[HttpGet]
        public ActionResult LogOn(string returnUrl)
        {
            if (User.Identity.IsAuthenticated) //remember me
            {
                if (shouldRedirect(returnUrl))
                {
                    return Redirect(returnUrl);
                }
                return Redirect(FormsAuthentication.DefaultUrl);
            }
            return View(); // show the login page
        }
```

```
[HttpPost]
        public ActionResult LogOn(Account loginInfo, string returnUrl)
        {
            if (this.ModelState.IsValid)
            {
                List<User> users = _userService.GetUser(loginInfo.Username, loginInfo.Password);
                if (users != null && users.Count == 1)
                {
                    FormsAuthentication.SetAuthCookie(loginInfo.Username,false);//
loginInfo.RememberMe);
                    //-- کاربر برنامه ریزی
                    if (users.First().UserType_Id == 1)
                    {
                        return RedirectToAction("Index", "Programming", new { u = loginInfo.Username
});
                    }
                    else if (users.First().UserType_Id == 2)
                    {

                    }
                    else if (users.First().UserType_Id == 3)
                    {

                    }
                    else if (users.First().UserType_Id == 4)
                    {
```

```
                    }
                }
            }
            this.ModelState.AddModelError("", "نام کاربری یا کلمه عبور اشتباه وارد شده اند.");
            ViewBag.Error = "";
            return View(loginInfo);
        }
```

```
<appSettings>
    <add key="webpages:Version" value="2.0.0.0" />
    <add key="webpages:Enabled" value="false" />
    <add key="PreserveLoginUrl" value="true" />
    <add key="ClientValidationEnabled" value="true" />
    <add key="UnobtrusiveJavaScriptEnabled" value="true" />
  </appSettings>
```

با تشکر.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۰۵ ۱۲:۱۵

احتمالا اسکریپت‌های شما درست load نشده.
به web developer tools مرورگر خودتون مراجعه کرده و خطاهای اسکریپتی رو بررسی کنید.

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۳۹۱/۱۱/۰۶ ۱۰:۳۵

با سلام آقای نصیری.
حق با شماست ولی نمی‌دانم چرا اسکریپت‌ها درست لود نمی‌شوند.

من هر وقت بر روی لینک اسکریپت‌ها در web developers کلیک میکنم به جای باز کردن محتوای فایل js دوباره همان صفحه لاگین را نشان می‌دهد و آدرسی شبیه این ایجاد می‌کند:

```
http://localhost:2215/Account/LogOn?ReturnUrl=%2fScripts%2fjquery-1.7.1.js
```

البته من اعتبارسنجی اجباری در تمام صفحات استفاده کردم:

```
<authorization>
      <deny users="?" />
    </authorization>
```

بسیار متشکرم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۰۶ ۱۱:۳

- برای سازگاری بیشتر با MVC، تنظیم وب کانفیگ فوق را حذف کنید.
- فیلتر Authorize را به صورت Global در فایل global.asax.cs اضافه کنید.
- یک سری مسیرهای مشخص را از سیستم Routing حذف کنید مانند:

```
routes.IgnoreRoute("Content/{*pathInfo}");
routes.IgnoreRoute("Scripts/{*pathInfo}");
```

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۳۹۱/۱۱/۰۶ ۱۲:۴۲

مهندس جان سلام.
مشکل کاملا برطرف شد. بسیار متشکرم از راهنمایی شما. یا حق.

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۳۹۱/۱۱/۱۵ ۱۶:۳۲

با سلام.
چرا ModelClientValidationRule در کدهای من قابل شناسایی نیست. از mvc 4 استفاده می‌کنم.

با تشکر.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۱۵ ۱۶:۴۵

منتقل شده به اسمبلی System.Web.Webpages نگارش 2 آن.

نویسنده: سعید یزدانی
تاریخ: ۱۳۹۱/۱۱/۱۸ ۲۳:۱۶

چرا ما در پیاده سازی متد Validate ورودی رو از نوع IEnumerable قرار دادیم

نویسنده: سعید یزدانی
تاریخ: ۱۳۹۱/۱۱/۱۸ ۲۳:۴۰

میخواستم بدونم پارامتر دوم OutputCache چی کار می‌کنه

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۱۸ ۲۳:۵۳

خروجی آن البته. به دلیل امضای متد تعریف شده در اینترفیس آن .

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۱۱/۱۸ ۲۳:۵۵

یک قسمت کامل به بحث caching اطلاعات اختصاص داده شده.

نویسنده: ahmad
تاریخ: ۱۳۹۲/۰۳/۲۳ ۱۹:۲۷

با سلام و تشکر فراوان
اعتبار سنجی سمت کلاینت در برنامه من مشکل دارد همه موارد کاملا مانند مثال است در فایرباگ breakpoint گذاشتم وارد این خط نمی‌شود

```
return Date.parse(value) < new Date();
```

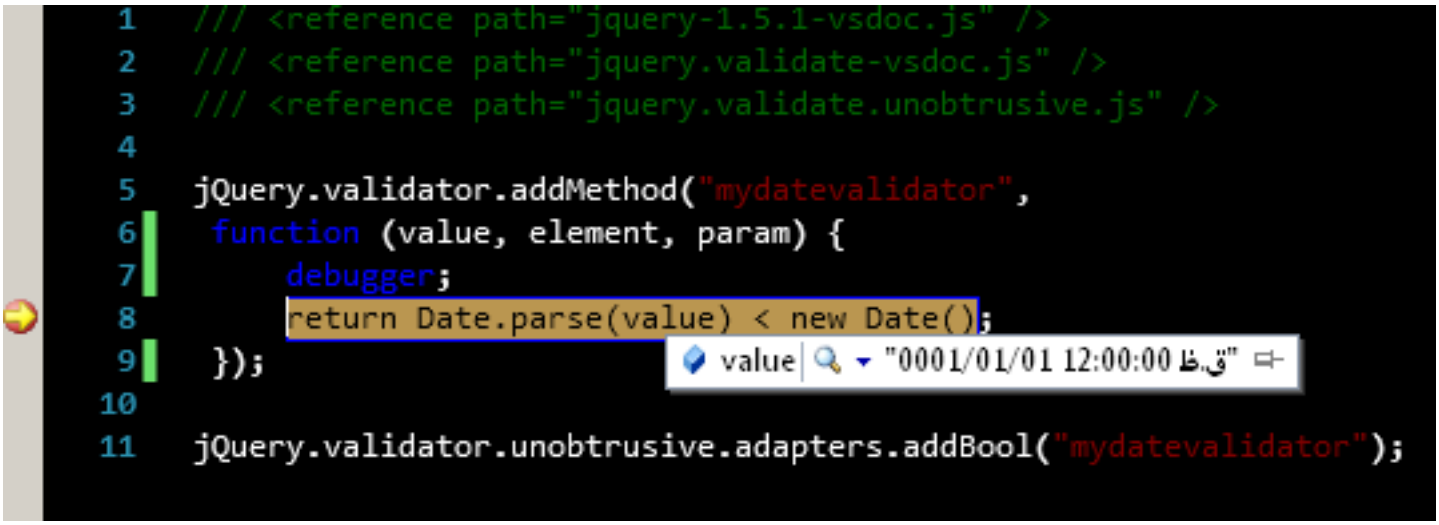
و بعد از این خط به بیرون می‌رود

```
jQuery.validator.addMethod("mydatevalidator",
 function (value, element, param) {
```

ایا مشکل از اینجاست یا جای دیگر در ضمن اگر خط بالا هم در document.readyنباشد هم این خط هم فراخوانی نمی‌شود

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۲۳ ۱۹:۵۹

مشکلی مشاهده نشد: (برای فعال سازی دیباگر توکار VS.NET فقط کافی است سطر debugger را اضافه کنید)



کدهای کامل این سری همانطور که در مطالب قبلی هم عنوان شده از آدرس ذیل قابل دریافت است:

MVC_Samples

مراجعه کنید به پوشه MVC-13 آن که حاوی کدهای قسمت 13 است.

نویسنده: Leila_gh
تاریخ: ۱۳۹۲/۰۵/۰۲ ۹:۱۹

با سلام و تشکر فراوان

اعتبار سنجی سمت کلاینت در مثال 13 که فرمودید در سیستم من کار نکرد اما این مثال code project کار می‌کند.

ولی بعد از اینکه دوباره مثال را در vs2012 mvc4 باز نویسی کردم کار نکرد (دقیقا مانند مثال code project) پس از تغییر متداز (-) split به (/) split و تغییر

```
mcvrTwo.ValidationParameters.Add("param", DateTime.Now.ToString("dd/MM/yyyy"));
```

به (فقط فرمت تاریخ تغییر کرد)

```
mcvrTwo.ValidationParameters.Add("param", DateTime.Now.ToString("dd-MM-yyyy"));
```

و حذف

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

کار کرد.

چرا به این تغییرات نیاز بود؟ (در صورتی که مثال code project کار می‌کرد، آیا به دلیل تفاوت ورژن است)

این مثال با مثال شما چه تفاوتی دارد که مثال شما در سیستم من اجرا نشد؟

این هم فایل نهایی من بعد از تغییر که کار کرد: MvcApplication-JsValidation.zip

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۵/۰۲ ۱۱:۲۳

کتابخانه‌های جی‌کوئری و کلیه مشتقات آن تقریبا ماهی یکبار شخم زده می‌شوند. اگر به کنسول پاور شل نیوگت در VS.NET مراجعه و دستور update-package را صادر کنید، متوجه حجیم عظیمی از تغییرات خواهید شد. تمام وابستگی‌ها هم اخیرا تغییر کردند. بنابراین اگر از یک مجموعه ناهماهنگ استفاده کنید، درسته ... ممکنه چیزی کار نکند.
مثلا jquery.validate.unobtrusive با jquery.validate و jquery «باید» هماهنگ باشند و اگر نیستند دستور update-package را فراموش نکنید. ضمنا پس از به روز رسانی، «باید» ارجاعات به فایل‌های جدید را در viewهای برنامه درست کنید. مثلا پروژه به نگارش جدید jQuery به روز شده، اما مدخل آن در view شما هنوز به نگارش قدیمی اشاره می‌کند. این‌ها یعنی تداخل و نتیجه آن کار نکردن افزونه‌های جدید است.
به علاوه اگر تعاریف اسکریپت‌ها را دستی در ابتدای فایل تعریف کردید دیگر نباید از Scripts.Render در پایین صفحه استفاده کنید. چون این مورد هم سبب می‌شود یکبار دیگر اسکریپت‌ها در صفحه پیوست شوند.

ضمنا من مجددا همین مثال بحث فوق را (نه مثال یک سایت ثالث را) در MVC4 بازنویسی کردم و بدون مشکل کار می‌کند:
Sample13Mvc4.zip

نویسنده: رضا گرمارودی
تاریخ: ۱۳۹۲/۱۲/۱۱ ۱۰:۱۵

سلام
در پروژه اصلی مشکلی نیست ولی وقتی در DomainClasses کلاس CustomValidator را اضافه می‌کنم خطای زیر و می‌گیرم

```
The type name 'ModelClientValidationRule' could not be found. This type has been forwarded
to assembly 'System.Web.WebPages, Version=3.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35'. Consider adding a reference to that assembly.
```

در App.Config هم تنظیمات زیر و اضافه کردم ولی باز هم نشد

```
<runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="System.Web.WebPages" publicKeyToken="31bf3856ad364e35" />
        <bindingRedirect oldVersion="1.0.0.0-3.0.0.0" newVersion="3.0.0.0" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۱۱ ۱۱:۴۲

«Consider adding a reference to that assembly» به معنای نیاز به افزودن ارجاعی به آن اسمبلی، یعنی System.Web.Webpages dll است.

نویسنده: رضا ریاضی
تاریخ: ۱۳۹۳/۱۰/۱۷ ۸:۵۱

با سلام؛ در صورتی که در فرم ورودی اطلاعات dropdownlist داشته باشم اعتبار سنجی را برای این کنترل انجام نمیدهد. از روش اعتبار سنجی سمت کاربر و با استفاده ازMetadata استفاده می‌کنم . دلیل اینکه برای سایر کنترل‌ها انجام می‌شود و dropdownlist انجام نمی‌شود چیست ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۰/۱۷ ۹:۳۷

```
public class MyViewModel
{
    [Required]
    public string CategoryId { get; set; }

    public IEnumerable<Category> Categories { get; set; }
}
```

Controller:

```
public ActionResult Index()
{
   var model = new MyViewModel
   {
       Categories = GetCategories()
   }
   return View(model);
}
```

View:

```
@Html.DropDownListFor(
    x => x.CategoryId,
    new SelectList(Model.Categories, "ID", "CategoryName"),
    "-- Please select a category --"
)
@Html.ValidationMessageFor(x => x.CategoryId)
```