عنوان: **طراحی ValidationAttribute دلخواه و هماهنگ سازی آن با ASP.NET MVC**
نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۷/۲۱ ۱۵:۳۵
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, MVC, ASP.Net MVC, jQuery

در سری پست‌های آقای مهندس یوسف نژاد با عنوان Globalization در ASP.NET MVC روشی برای پیاده سازی کار با Resourceها در ASP.NET با استفاده از دیتابیس شرح داده شده است. یکی از کمبودهایی که در این روش وجود داشت عدم استفاده از این نوع Resourceها از طریق Attributeها در ASP.NET MVC بود. برای استفاده از این روش در یک پروژه به این مشکل برخورد کردم و پس از تحقیق و بررسی چند پست سرانجام در این پست پاسخ خود را پیدا کرده و با ترکیب این روش با روش آقای یوسف نژاد موفق به پیاده سازی Attribute دلخواه شدم.
در این پست و با استفاده از سری پست‌های آقای مهندس یوسف نژاد در این زمینه، یک Attribute جهت هماهنگ سازی با سیستم اعتبار سنجی ASP.NET MVC در سمت سرور و سمت کلاینت (با استفاده از jQuery Validation) بررسی خواهد شد.

قبل از شروع مطالعه سری پست‌های MVC و Entity Framework الزامی است.

برای انجام این کار ابتدا مدل زیر را در برنامه خود ایجاد می‌کنیم.

```
using System;

public class SampleModel
{
    public DateTime StartDate { get; set; }
    public string Data { get; set; }
    public int Id { get; set; }
}
```

با استفاده از این مدل در برنامه در زمان ثبت داده‌ها هیچ گونه خطایی صادر نمی‌شود. برای اینکه بتوان از سیستم خطای پیش فرض ASP.NET MVC کمک گرفت می‌توان مدل را به صورت زیر تغییر داد.

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

public class SampleModel
{
    [Required(ErrorMessage = "Start date is required")]
    public DateTime StartDate { get; set; }

    [Required(ErrorMessage = "Data is required")]
    public string Data { get; set; }

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

حال ویو این مدل را طراحی می‌کنیم.

```
@model SampleModel
@{
    ViewBag.Title = "Index";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<section>
    <header>
        <h3>SampleModel</h3>
```

```
    </header>
    @Html.ValidationSummary(true, null, new { @class = "alert alert-error alert-block" })

    @using (Html.BeginForm("SaveData", "Sample", FormMethod.Post))
    {
        <p>
            @Html.LabelFor(x => x.StartDate)
            @Html.TextBoxFor(x => x.StartDate)
            @Html.ValidationMessageFor(x => x.StartDate)
        </p>
        <p>
            @Html.LabelFor(x => x.Data)
            @Html.TextBoxFor(x => x.Data)
            @Html.ValidationMessageFor(x => x.Data)
        </p>
        <input type="submit" value="Save"/>
    }
</section>
```

و بخش کنترلر آن را به صورت زیر پیاده سازی می‌کنیم.

```
public class SampleController : Controller
    {
        //
        // GET: /Sample/

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

        public ActionResult SaveData(SampleModel item)
        {
            if (ModelState.IsValid)
            {
                //save data
            }
            else
            {
                ModelState.AddModelError("","لطفا خطاهای زیر را برطرف نمایید");
                RedirectToAction("Index", item);
            }
            return View("Index");
        }
    }
```

حال با اجرای این کد و زدن دکمه Save صفحه مانند شکل پایین خطاها را نمایش خواهد داد.

تا اینجای کار روال عادی همیشگی است. اما برای طراحی Attribute دلخواه جهت اعتبار سنجی (مثلا برای مجبور کردن کاربر به وارد کردن یک فیلد با پیام دلخواه و زبان دلخواه) باید یک کلاس جدید تعریف کرده و از کلاس RequiredAttribute ارث ببرد. در پارامتر ورودی این کلاس جهت کار با Resourceهای ثابت در نظر گرفته شده است اما برای اینکه فیلد دلخواه را از دیتابیس بخواند این روش جوابگو نیست. برای انجام آن باید کلاس RequiredAttribute بازنویسی شود.

کلاس طراحی شده باید به صورت زیر باشد:

```
public class VegaRequiredAttribute : RequiredAttribute, IClientValidatable
    {
#region Fields (2)

        private readonly string _resourceId;
        private String _resourceString = String.Empty;

#endregion Fields

#region Constructors (1)

        public VegaRequiredAttribute(string resourceId)
        {
            _resourceId = resourceId;
            ErrorMessage = _resourceId;
            AllowEmptyStrings = true;
        }

#endregion Constructors

#region Properties (1)

        public new String ErrorMessage
        {
            get { return _resourceString; }
            set { _resourceString = GetMessageFromResource(value); }
        }

#endregion Properties

#region Methods (2)
```

```
// Public Methods (1)

        public IEnumerable<ModelClientValidationRule> GetClientValidationRules(ModelMetadata metadata,
ControllerContext context)
        {

            yield return new ModelClientValidationRule
            {
                ErrorMessage = GetMessageFromResource(_resourceId),
                ValidationType = "required"
            };
        }
// Private Methods (1)

        private string GetMessageFromResource(string resourceId)
        {
            var errorMessage = HttpContext.GetGlobalResourceObject(_resourceId, "Yes") as string;
            return errorMessage ?? ErrorMessage;
        }

#endregion Methods
    }
```

در این کلاس دو نکته وجود دارد.

1- ابتدا دستور

```
HttpContext.GetGlobalResourceObject(_resourceId, "Yes") as string;
```

که عنوان کلید Resource را از سازنده کلاس گرفته (کد اقای یوسف نژاد) رشته معادل آن را از دیتابیس بازیابی میکند.

2- ارث بری از اینترفیس IClientValidatable، در صورتی که از این اینترفیس ارث بری نداشته باشیم. Validator طراحی شده در طرف کلاینت کار نمی‌کند. بلکه کاربر با کلیک بروی دکمه مورد نظر داده‌ها را به سمت سرور ارسال می‌کند. در صورت وجود خطا در پست بک خطا نمایش داده خواهد شد. اما با ارث بری از این اینترفیس و پیاده سازی متد GetClientValidationRules می‌توان تعریف کرد که در طرف کلاینت با استفاده از Unobtrusive jQuery پیام خطای مورد نظر به کنترل ورودی مورد نظر (مانند تکست باکس) اعمال می‌شود. مثلا در این مثال خصوصیت data-val-required به input هایی که قبلا در مدل ما Reqired تعریف شده اند اعمال می‌شود.

حال در مدل تعریف شده می‌توان به جای Required می‌توان از VegaRequiredAttribute مانند زیر استفاده کرد. (همراه با نام کلید مورد نظر در دیتابیس)

```
public class SampleModel
{
    [VegaRequired("RequiredMessage")]
    public DateTime StartDate { get; set; }

    [VegaRequired("RequiredMessage")]
    public string Data { get; set; }

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

ورودی Validator مورد نظر نام کلیدی است به زبان دلخواه که عنوان آن RequiredMessage تعریف شده است و مقدار آن در دیتابیس مقداری مانند " **تکمیل این فیلد الزامی است** " است. با این کار در زمان اجرا با استفاده از این ولیدتور ابتدا کلید مورد نظر با توجه به زبان فعلی از دیتابیس بازیابی شده و در متادیتابی مدل ما قرار می‌گیرد. به جای استفاده از Resourceها می‌توان پیام‌های خطای دلخواه را در دیتابیس ذخیره کرد و در مواقع ضروری جهت جلوگیری از تکرار از آنها استفاده نمود.

با اجرای برنامه اینبار خروجی به شکل زیر خواهد بود.

جهت فعال ساری اعتبار سنجی سمت کلاینت ابتدا باید اسکریپت‌های زیر به صفحه اضافه شود.

```
<script src="@Url.Content("~/Scripts/jquery-1.9.1.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>
```

سپس در فایل web.config تنظیمات زیر باید اضافه شود

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

سپس برای اعمال Validator طراحی شده باید توسط کدهای جاوا اسکریپت زیر داده‌های مورد نیاز سمت کلاینت رجیستر شوند.

```
<script type="text/javascript">
        jQuery.validator.addMethod('required', function (value, element, params) {
            if (value == null | value == "") {
                return false;
            } else {
                return true;
            }

        }, '');

        jQuery.validator.unobtrusive.adapters.add('required', {}, function (options) {
            options.rules['required'] = true;
            options.messages['required'] = options.message;
        });
    </script>
```

البته برای مثال ما قسمت بالا به صورت پیش فرض رجیستر شده است اما در صورتی که بخواهید یک ولیدتور دلخواه و غیر استاندارد بنویسید روال کار به همین شکل است.

موفق و موید باشید.

منابع ^ و ^ و ^ و ^

نظرات خوانندگان

نویسنده: امیر خلیلی
تاریخ: ۱۳۹۲/۰۸/۰۴ ۸:۵۹

یک مشکل اساسی وجود داره
هنگامی که با استفاده از جاواسکریپت اعتبارسنجی انجام میشه کاربر یا اون شخصی که قراره کدهای مخرب را وارد کنه میتونه استفاده از فایل‌های JS را در مرورگش غیر فعال کنه

و اینگونه اعتبار سنجی انجام نمیشه !

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۰۴ ۹:۹

اعتبارسنجی در ASP.NET MVC دو مرحله‌ای است. مرحله اول آن فقط سمت کاربر است. مرحله دوم به تفسیر IClientValidatable در سمت سرور ختم می‌شود.