عنوان: **افزونه‌ای برای کپسوله سازی نکات ارسال یک فرم ASP.NET MVC به سرور توسط jQuery Ajax**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۳ ۱۱:۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, jQuery, MVC, JavaScript

اگر مطالب سایت جاری را مطالعه و دنبال کرده باشید، تاکنون به صورت پراکنده نکات زیادی را در مورد استفاده از jQuery Ajax تهیه و ارائه کرده‌ایم. در این مطلب قصد داریم تا این نکات را نظم بخشیده و جهت استفاده مجدد، به صورت یک افزونه کپسوله سازی کنیم.

در کدها و افزونه‌ای که در ادامه ارائه خواهند شد، این مسایل درنظر گرفته شده است:

- چگونه اعتبار سنجی سمت کاربر را در حین استفاده از Ajax فعال کنیم.
- چگونه از چندبار کلیک کاربر در حین ارسال فرم به سرور جلوگیری نمائیم.
- چگونه Complex Types قابل تعریف در EF Code first را نیز در اینجا مدیریت کنیم.
- نحوه تعریف صحیح آدرس‌های کنترلرها چگونه باید باشد.
- نحوه اعلام وضعیت لاگین شخص به او، در صورت بروز مشکل.
- ارسال صحیح anti forgery token در حین اعمال Ajax ایی.
- بررسی Ajax بودن درخواست رسیده و تهیه یک فیلتر سفارشی مخصوص آن.
- از کش شدن اطلاعات Ajax ایی جلوگیری شود.

ابتدا معرفی مدل برنامه

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

namespace jQueryMvcSample01.Models
{
    public class User
    {
        [Required(ErrorMessage = "(*)"), DisplayName("نام")]
        public string Name { set; get; }

        public PhoneInfo PhoneInfo { set; get; }
    }

    public class PhoneInfo
    {
        [Required(ErrorMessage = "(*)"), DisplayName("تلفن")]
        public string Phone { get; set; }

        [Required(ErrorMessage = "(*)"), DisplayName("پیش شماره")]
        public string Ext { get; set; }
    }
}
```

همانطور که ملاحظه می‌کنید، خاصیت PhoneInfo، تو در تو یا به نوعی Complex است. اگر از ابزارهای Scafolding توکار VS.NET برای تولید View متناظر استفاده کنیم، فیلد تو در توی PhoneInfo را لحاظ نخواهد کرد، اما ... مهم نیست. تعریف دستی آن هم کار می‌کند.

کدهای کنترلر برنامه

```
using System.Web.Mvc;
using jQueryMvcSample01.Models;
using jQueryMvcSample01.Security;

namespace jQueryMvcSample01.Controllers
{
    public class HomeController : Controller
```

```
    {
        [HttpGet]
        public ActionResult Index()
        {
            return View(); //نمایش فرم
        }

        [HttpPost]
        [AjaxOnly] //فقط در حالت ای‌جکس قابل دسترسی باشد
        [ValidateAntiForgeryToken]
        public ActionResult Index(User user)
        {
            if (this.ModelState.IsValid)
            {
                // ذخیره سازی در بانک اطلاعاتی ...
                System.Threading.Thread.Sleep(3000);

                return Content("ok");//اعلام موفقیت آمیز بودن کار
            }

            return Content(null);//ارسال خطا
        }
    }
}
```

در اینجا در متد Index، اطلاعات شیء User به صورت Ajaxایی دریافت شده و پس از آن برای مثال قابلیت ذخیره سازی را خواهد داشت.

چند نکته در اینجا حائز اهمیت هستند:

الف) استفاده از ویژگی AjaxOnly (که کدهای آن‌را در پروژه پیوست می‌توانید مشاهده نمائید)، جهت صرفا پردازش درخواست‌های Ajaxایی.

ب) استفاده از ویژگی ValidateAntiForgeryToken در حین اعمال اجکسی. اگر سایت‌های مختلف را در اینباره جستجو کنید، عموما برای پردازش آن در حین استفاده از jQuery Ajax بسیار مشکل دارند.

ج) استفاده از return Content برای اعلام نتیجه کار. اگر اطلاعات ثبت شد، یک ok یا هر عبارت دیگری که علاقمند بودید ارسال گردیده و در غیراینصورت null بازگشت داده می‌شود.

کدهای افزونه PostMvcFormAjax

```
// <![CDATA[
(function ($) {
    $.fn.PostMvcFormAjax = function (options) {
        var defaults = {
            postUrl: '/',
            loginUrl: '/login',
            beforePostHandler: null,
            completeHandler: null,
            errorHandler: null
        };
        var options = $.extend(defaults, options);

        var validateForm = function (form) {
            //فعال سازی دستی اعتبار سنجی جی‌کوئری
            var val = form.validate();
            val.form();
            return val.valid();
        };

        return this.each(function () {
            var form = $(this);
            //اگر فرم اعتبار سنجی نشده، اطلاعات آن ارسال نشود
            if (!validateForm(form)) return;

            //در اینجا می‌توان مثلا دکمه‌ای را غیرفعال کرد
            if (options.beforePostHandler)
                options.beforePostHandler(this);

            //اطلاعات نباید کش شوند
            $.ajaxSetup({ cache: false });

            $.ajax({
```

```
                type: "POST",
                url: options.postUrl,
                data: form.serialize(), //تمام فیلدهای فرم منجمله آنتی فرجری توکن آن‌را ارسال می‌کند
                complete: function (xhr, status) {
                    var data = xhr.responseText;
                    if (xhr.status == 403) {
                        window.location = options.loginUrl; //در حالت لاگین نبودن شخص اجرا می‌شود
                    }
                    else if (status === 'error' || !data) {
                        if (options.errorHandler)
                            options.errorHandler(this);
                    }
                    else {
                        if (options.completeHandler)
                            options.completeHandler(this);
                    }
                }

            });
        });
    };
})(jQuery);
// ]]>
```

چند نکته مهم در تهیه این افزونه رعایت شده:

الف) فعال سازی دستی اعتبار سنجی جی‌کوئری، از این جهت که این نوع اعتبار سنجی به صورت پیش فرض تنها در حالت postback و ارسال کامل صفحه به سرور فعال می‌شود.

ب) استفاده از متد serialize جهت پردازش یکباره کل اطلاعات و فیلدهای یک فرم.

نکته مهم این متد ارسال فیلد مخفی anti forgery token نیز می‌باشد. فقط باید دقت داشت که این فیلد در حالتی که dataType به json تنظیم شود و همچنین از متد serialize استفاده گردد، در ASP.NET MVC پردازش نمی‌گردد (خیلی مهم!). به همین جهت در اینجا dataType تنظیمات jQuery Ajax حذف شده است.

ج) تنظیم cache به false در تنظیمات ابتدایی jQuery Ajax تا اطلاعات ارسالی و دریافتی کش نشوند و مشکل ساز نگردند.

د) بررسی xhr.status == 403 که توسط SiteAuthorizeAttribute (جایگزین بهتر فیلتر Authorize توکار ASP.NET MVC که کدهای آن در پروژه پیوست قابل دریافت است) و هدایت کاربر به صفحه لاگین

تعریف View ایی که از اشیاء تو در تو استفاده می‌کند و همچنین از افزونه فوق برای ارسال اطلاعات بهره خواهد برد:

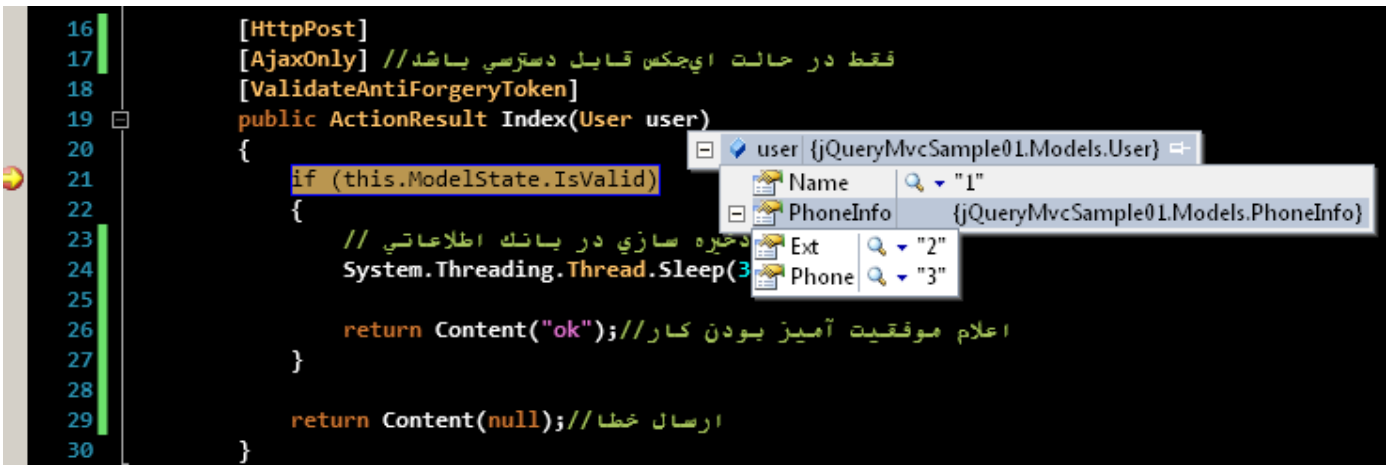
```
@model jQueryMvcSample01.Models.User
@{
    ViewBag.Title = "تعریف کاربر";
    var postUrl = Url.Action(actionName: "Index", controllerName: "Home");
}
@using (Html.BeginForm(actionName: "Index", controllerName: "Home",
                       method: FormMethod.Post,
                       htmlAttributes: new { id = "UserForm" }))
{
    @Html.ValidationSummary(true)
    @Html.AntiForgeryToken()

    <fieldset>
        <legend>تعریف کاربر</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>
        <div class="editor-label">
            @Html.LabelFor(model => model.PhoneInfo.Ext)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.PhoneInfo.Ext)
            @Html.ValidationMessageFor(model => model.PhoneInfo.Ext)
        </div>
        <div class="editor-label">
            @Html.LabelFor(model => model.PhoneInfo.Phone)
        </div>
```

```
        <div class="editor-field">
            @Html.EditorFor(model => model.PhoneInfo.Phone)
            @Html.ValidationMessageFor(model => model.PhoneInfo.Phone)
        </div>
        <p>
            <input type="submit" id="btnSave" value="ارسال" />
        </p>
    </fieldset>
}
@section JavaScript
{
    <script type="text/javascript">
        $(document).ready(function () {
            $("#btnSave").click(function (event) {
                //جلوگیری از پست بک به سرور
                event.preventDefault();

                var button = $(this);

                $("#UserForm").PostMvcFormAjax({
                    postUrl: '@postUrl',
                    loginUrl: '/login',
                    beforePostHandler: function () {
                        //غیرفعال سازی دکمه ارسال
                        button.attr('disabled', 'disabled');
                        button.val("...");
                    },
                    completeHandler: function () {
                        //فعال سازی مجدد دکمه ارسال
                        alert('انجام شد');
                        button.removeAttr('disabled');
                        button.val("ارسال");
                    },
                    errorHandler: function () {
                        alert('خطایی رخ داده است');
                    }
                });
            });
        });
    </script>
}
```

همانطور که عنوان شد، مهم نیست که اشیاء تو در تو توسط ابزار Scafolding پشتیبانی نمی‌شود. این نوع خواص را به همان نحو متداول ذکر زنجیره وار خواص می‌توان معرفی و استفاده کرد:

```
@Html.EditorFor(model => model.PhoneInfo.Phone)
```

هم اعتبار سنجی سمت کلاینت آن کار می‌کند و هم اطلاعات آن به اشیاء و خواص متناظر به خوبی نگاشت خواهد شد:



در ادامه نحوه استفاده از افزونه PostMvcFormAjax را مشاهده می‌کنید. چند نکته نیز در اینجا حائز اهمیت هستند:
الف) توسط htmlAttributes یک id برای فرم تعریف کرده‌ایم تا در افزونه PostMvcFormAjax مورد استفاده قرار گیرد.
ب) postUrl و loginUrl را همانند متغیر تعریف شده در ابتدای View توسط Url.Action باید تعریف کرد تا در صورتیکه سایت ما در ریشه اصلی قرار نداشت، باز هم به صورت خودکار مسیر صحیحی محاسبه و ارائه گردد.
ج) نحوه غیرفعال سازی و فعال سازی دکمه submit را در روال‌های beforePostHandler و completeHandler ملاحظه می‌کنید.
این مساله برای جلوگیری از کلیک‌های مجدد یک کاربر ناشکیبا و جلوگیری از ثبت اطلاعات تکراری بسیار مهم است.
د) کل این اطلاعات، در یک section به نام JavaScript ثبت شده است. این section در فایل layout برنامه به صورت زیر مورد استفاده قرار خواهد گرفت و به این ترتیب مقدار دهی خواهد شد:

```
<head>
    <title>@ViewBag.Title</title>
    <link href="@Url.Content("Content/Site.css")" rel="stylesheet" type="text/css" />
    <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.unobtrusive-ajax.min.js")"
type="text/javascript"></script>
    <script src="@Url.Content("~/Scripts/jquery.validate.unobtrusive.min.js")"
type="text/javascript"></script>
    <script src="@Url.Content("~/Scripts/jquery.PostMvcFormAjax.js")" type="text/javascript"></script>
    @RenderSection("JavaScript", required: false)
</head>
```

دریافت کدهای کامل این قسمت
jQueryMvcSample01.zip

نظرات خوانندگان

نویسنده: molana11
تاریخ: ۱۳۹۲/۰۱/۰۴ ۱۶:۵۵

با سلام.
دستور val.form چه کاری انجام میدهد؟
با تشکر.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۴ ۱۷:۶

مطابق مستندات افزونه jQuery Validator که در ASP.NET MVC استفاده می‌شود، متد Validate فقط یک سری روال رویدادگردان را تنظیم می‌کند تا در زمان postback کامل به سرور فعال شوند. برای اینکه در اینجا postback کامل به سرور نداریم (عملیات Ajax ایی است)، نیاز است عملیات اعتبارسنجی را دستی فراخوانی کنیم و اینکار توسط متد form آن انجام می‌شود .

نویسنده: molana11
تاریخ: ۱۳۹۲/۰۱/۱۴ ۱۵:۲۶

با سلام.
اطلاعات کنترلر من بصورت زیر است:

```
using MvcApplication3.Models;
...

namespace MvcApplication3.Controllers
{
    public class StudentController : Controller
    {
        public ActionResult Index()
        {
            //var data = new StudentsList();
            return View();
        }
        public ActionResult DataList()
        {
            var data = new StudentsList();
            return PartialView("Pv_DataList", data);
        }

        #region Edit

        [HttpGet]
        public ActionResult Edit(int? id)
        {
            var data = new StudentsList().FirstOrDefault(p => p.Id == id);
            return PartialView("Pv_Edit", data);
        }
        [HttpPost]
        [AjaxOnly]
        [ValidateAntiForgeryToken]
        public ActionResult Edit(StudentModel model)
        {
            Thread.Sleep(1000);
            if (this.ModelState.IsValid)
            {
                return Json("ok", JsonRequestBehavior.AllowGet);
            }
            return Json("error");
        }

        #endregion
    }
}
```

اطلاعات ویو :

```
@using MvcApplication3.Models
@{
    ViewBag.Title = "Student Index";
}
<style>
    #div_StudentEditDialogContainer {
        padding: 15px;
        background-color: silver;
        border: 1px solid gray;
        -webkit-border-radius: pxpx;
        -moz-border-radius: pxpx;
        border-radius: 10px;
        display: none;
        position: absolute;
        -webkit-box-shadow: 0 1px 3px rgba(0,0,0,0.3);
        -moz-box-shadow: 0 1px 3px rgba(0,0,0,0.4);
        box-shadow: 0 1px 3px rgba(0,0,0,0.5);
    }
</style>
<h2>Student Index</h2>
<div id="div_StudentListViewContainer">
    @{ Html.RenderAction("DataList", "Student");}
</div>
<div id="div_StudentEditDialogContainer">
    @{ Html.RenderAction("Edit", "Student");}
</div>
<div id="div_StudentAddDialogContainer">
</div>
<div id="div_StudentRemoveDialogContainer">
</div>
<div id="div_StudentSearchDialogContainer">
</div>

@section JavaScript{
    <script type="text/javascript">
        $(document).ready(function () {
            $("#div_StudentListViewContainer table input[type='submit']").click(function (e) {
                //show edit dialog
                e.preventDefault();
                var id = $(this).parent().parent().attr('data-studentid');
                var url = '@Url.Action("Edit", "Student")';
                $.ajax({
                    type: "GET",
                    url: url,
                    data: { id: id },
                    beforeSend: function () {
                        //$(waitingPanel).css("display", "block");
                    },
                    success: function (html) {
                        if (html == "nodata") {
                            $("#div_StudentEditDialogContainer").html("دانشجویی با این مشخصات یافت
نشد!");
                        } else {
                            $("#div_StudentEditDialogContainer").css("display", "block");
                            $("#div_StudentEditDialogContainer").html("").append(html);
                        }
                    },
                    complete: function () {
                        //$(waitingPanel).css("display", "none");
                    }
                });
            });
        });
    </script>
}
```

و یک دیالوگ برای ویرایش را بصورت داینامیک در صفحه ظاهر میکنم، اما اعتبارسنجی سمت کاربر برای آن کار نمیکند:

```
@using MvcApplication3.Models
@model StudentModel
@{
    var postUrl = Url.Action(actionName: "Edit", controllerName: "Student");
}
```

```
@using (Html.BeginForm(actionName: "Edit", controllerName: "Student",
                       method: FormMethod.Post,
                       htmlAttributes: new { id = "frm_studentEdit" }))
{
    @Html.ValidationSummary(true)
    @Html.AntiForgeryToken()

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

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

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

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

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

    <p>
        <input type="submit" id="btn_Save" value="ذخیره" />
        <input type="submit" id="btn_Cancel" value="انصراف" />
    </p>
}

<script type="text/javascript">
    $(function () {
        $("#btn_Save").click(function (e) {
            e.preventDefault();
            var button = $(this);
            $("#frm_studentEdit").PostMvcFormAjax({
                postUrl: '@postUrl',
                loginUrl: '/login',
                beforePostHandler: function () {
                    //غیرفعال سازی دکمه ارسال
                    button.attr('disabled', 'disabled');
                    button.val("...");
                },
                completeHandler: function (data) {
                    //فعال سازی مجدد دکمه ارسال
                    button.removeAttr('disabled');
                    button.val("ذخیره");
                },
                errorHandler: function () {
                    alert('خطایی رخ داده است');
                }
            });
        });
        $("#btn_Cancel").click(function (e) {
            e.preventDefault();
            var button = $(this);
            $(".editDialog").parent("div").css("display", "none");
        });
```

```
    });
</script>
```

با تشکر.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۱۴ ۱۵:۴۸

برای تکمیل بحث مراجعه کنید به قسمت 21 سری MVC؛ بحث «افزودن فرم‌ها به کمک jQuery.Ajax و فعال سازی اعتبار سنجی سمت کلاینت» آن.

نویسنده: molana11
تاریخ: ۱۳۹۲/۰۱/۱۵ ۱۰:۱۷

با سلام.
اگر بخواهیم به جای return Content از return json به عنوان خروجی استفاده کنیم و مثلا مقدار "ok" را برگردانیم، در تابع completeHandler چطور این مقدار را دریافت و بررسی کنیم؟
باتشکر.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۱۵ ۱۱:۵۴

نیاز به کمی تغییر دارد. اگر خروجی اکشن متد شما به این نحو باشد:

```
return json(new { result = "ok" });
```

در سمت اسکریپت در $.ajax:

```
//...
success: function (data) {
                    if (data) {
                        alert("json data: " + data.result);
                    }
                }
```

نویسنده: محمد آزاد
تاریخ: ۱۳۹۲/۰۱/۱۷ ۱۷:۳۲

جناب نصیری با تشکر از مطلبتون.ی سوال داشتم اینکه استفاده از فرم ایجکس (Ajax.BeginForm) موجود در Razor بهتر و راحتر نیست؟یا اینکه نسبت به jquery محدودیت خاصی داره؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۱۷ ۱۸:۷

- Ajax.BeginForm هم در پشت صحنه از jQuery Ajax استفاده می‌کنه. فقط پیاده سازی نهائیش از شما مخفی شده.
- در نمونه بحث جاری کنترل بیشتری بر روی رویدادها خواهید داشت. مثلا قسمت 403 == xhr.status و هدایت کاربر به صفحه لاگین در صورت منقضی شدن اعتبارسنجی آن.
- Ajax.BeginForm برای کار کردن حتما نیاز به submit button داره. در مطلب جاری از یک span هم می‌تونید استفاده کنید و مشکلی نداره.
- در Ajax.BeginForm آنچنان کنترلی بر روی پردازش نهایی خروجی اکشن متد ندارید. مثلا در اینجا عنوان شد که اگر خروجی JSON بود و اگر دارای فیلد مشخصی با مقدار مشخصی بود نیاز است کار خاصی انجام شود. در حالت jQuery Ajax مستقیم،

پردازش JSON ساده‌تر است.

نویسنده: محمد آزاد
تاریخ: ۱۳۹۲/۰۱/۱۷ ۱۹:۱۹

- در نمونه بحث جاری کنترل بیشتری بر روی رویدادها خواهید داشت. مثلا قسمت xhr.status == 403 و هدایت کاربر به صفحه لاگین در صورت منقضی شدن اعتبارسنجی آن.

با توجه به اینکه گفتید فرم ایجکس داره همین روال رو در پشت صحنه ایجاد می‌کنه برای هدایت کاربر به صفحه لاگین نمیشه کد مربوطه رو تو OnComplete شی AjaxOptions نوشت؟

- Ajax.BeginForm برای کار کردن حتما نیاز به submit button داره. در مطلب جاری از یک span هم می‌تونید استفاده کنید و مشکلی نداره.

آیا نمیشه این سابمیت رو با جاوااسکریپت پیاده سازی کرد؟که بشه تو رویداد کلیک هر المنتی نوشت؟

- در Ajax.BeginForm آنچنان کنترلی بر روی پردازش نهایی خروجی اکشن متد ندارید. مثلا در اینجا عنوان شد که اگر خروجی JSON بود و اگر دارای فیلد مشخصی با مقدار مشخصی بود نیاز است کار خاصی انجام شود. در حالت jQuery Ajax مستقیم، پردازش JSON ساده‌تر است.

اینم نمیشه نتیجه نهایی رو تو OnComplete شی AjaxOptions داشته باشیم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۱۷ ۲۱:۳۵

سورس فایل jquery.unobtrusive-ajax.js به همراه پروژه‌های MVC موجود است. می‌توانید موارد مدنظر رو در اون بررسی کنید، مثلا نحوه انتقال اطلاعات یا نحوه گوش فرا دادن به دکمه submit در آن. سورس باز هم هست. تغییرش بدید، بعد نتایج رو برای حذف کدهای تکراری کپسوله کنید (اصل بحث جاری).

نویسنده: م کریمی
تاریخ: ۱۳۹۲/۰۳/۲۱ ۱۸:۴۰

با سلام خدمت آقای نصیری
با تشکر از مطلب کاربردی جاری ، یک سوال داشتم چطور می‌شه با این روش validationهای سمت سرور رو به کاربر نمایش بدهیم به طور مثال اگه در اینجا

```
[Required(ErrorMessage = "(*)"), DisplayName("نام")]
        [MaxLength(3,ErrorMessage="لطفا بیشتر از 3 کاراکتر وارد ننمایید")]
        public string Name { set; get; }
```

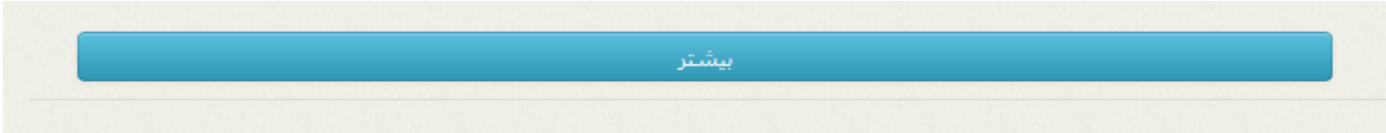
از MaxLength و یا یک سری Attهای خاصی استفاده بشه چطور میشه پیغام "لطفا بیشتر از 3 کاراکتر وارد ننمایید" را به کاربر نشان داد ، در حالت فعلی فقط پیغام خطایی رخ داده ظاهر می‌شه
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۳/۲۱ ۲۰:۵۶

این افزونه خروجی ساده متنی داره. اگر نیاز به بازگرداندن اطلاعات بیشتر و ساختار یافته‌ای هست، باید خروجی JSON براش طراحی کنید و بعد در سمت jQuery Ajax این ساختار مدنظر رو پردازش کنید. مثلا ساختاری بر اساس خواصی مانند لیست خطاها، لیست پیام‌ها و وضعیت عملیات. بعد قسمت complete افزونه فوق باید کلا بازنویسی شود.

عنوان: **پیاده سازی دکمه «بیشتر» یا «اسکرول نامحدود» به کمک jQuery در ASP.NET MVC**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۴ ۹:۳۶
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, jQuery, MVC, JavaScript

مدتی است در اکثر سایت‌ها و طراحی‌های جدید، به جای استفاده از روش متداول نمایش انتخاب صفحه 1، 2 ... 100، برای صفحه بندی اطلاعات، از روش اسکرول نامحدود یا infinite scroll استفاده می‌کنند. نمونه‌ای از آن‌را هم در سایت جاری با دکمه «بیشتر» در ذیل اکثر صفحات و مطالب سایت مشاهده می‌کنید.



در ادامه قصد داریم نحوه پیاده سازی آن‌را در ASP.NET MVC به کمک امکانات jQuery بررسی کنیم.

مدل برنامه

```
namespace jQueryMvcSample02.Models
{
    public class BlogPost
    {
        public int Id { set; get; }
        public string Title { set; get; }
        public string Body { set; get; }
    }
}
```

در این برنامه و مثال، قصد داریم لیستی از مطالب را توسط اسکرول نامحدود، نمایش دهیم. هر آیتم نمایش داده شده، ساختاری همانند کلاس BlogPost دارد.

منبع داده فرضی برنامه

```
using System.Collections.Generic;
using System.Linq;
using jQueryMvcSample02.Models;

namespace jQueryMvcSample02.DataSource
{
    public static class BlogPostDataSource
    {
        private static IList<BlogPost> _cachedItems;
        /// <summary>
        /// با توجه به استاتیک بودن سازنده کلاس، تهیه کش، پیش از سایر فراخوانی‌ها صورت خواهد گرفت
        /// باید دقت داشت که این فقط یک مثال است و چنین کشی به معنای
        /// تهیه یک لیست برای تمام کاربران سایت است
        /// </summary>
        static BlogPostDataSource()
        {
            _cachedItems = createBlogPostsInMemoryDataSource();
        }

        /// <summary>
        /// هدف صرفا تهیه یک منبع داده آزمایشی ساده تشکیل شده در حافظه است
        /// </summary>
        private static IList<BlogPost> createBlogPostsInMemoryDataSource()
        {
```

```
            var results = new List<BlogPost>();
            for (int i = 1; i < 30; i++)
            {
                results.Add(new BlogPost { Id = i, Title = "عنوان " + i, Body = "متن ... متن ... متن "
+ i });
            }
            return results;
        }

        /// <summary>
        /// پارامترهای شماره صفحه و تعداد رکورد به ازای یک صفحه برای صفحه بندی نیاز هستند
        /// شماره صفحه از یک شروع می‌شود
        /// </summary>
        public static IList<BlogPost> GetLatestBlogPosts(int pageNumber, int recordsPerPage = 4)
        {
            var skipRecords = pageNumber * recordsPerPage;
            return _cachedItems
                        .OrderByDescending(x => x.Id)
                        .Skip(skipRecords)
                        .Take(recordsPerPage)
                        .ToList();
        }
    }
}
```

برای اینکه برنامه نهایی را به سادگی بتوانید اجرا کنید، به عمد از بانک اطلاعاتی خاصی استفاده نشده و صرفا یک منبع داده فرضی تشکیل شده در حافظه، در اینجا مورد استفاده قرار گرفته است. بدیهی است قسمت cachedItems را به سادگی می‌توانید با یک ORM جایگزین کنید.

تنها نکته مهم آن، نحوه تعریف متد GetLatestBlogPosts می‌باشد که برای صفحه بندی اطلاعات بهینه سازی شده است. در اینجا توسط متدهای Skip و Take، تنها بازه‌ای از اطلاعات که قرار است نمایش داده شوند، دریافت می‌گردد. خوشبختانه این متدها معادل‌های مناسبی را در اکثر بانک‌های اطلاعاتی داشته و استفاده از آن‌ها بر روی یک بانک اطلاعاتی واقعی نیز بدون مشکل کار می‌کند و تنها بازه محدودی از اطلاعات را واکشی خواهد کرد که از لحاظ مصرف حافظه و سرعت کار بسیار مقرون به صرفه و سریع است.

کنترلر برنامه

```
using System.Linq;
using System.Web.Mvc;
using System.Web.UI;
using jQueryMvcSample02.DataSource;
using jQueryMvcSample02.Security;

namespace jQueryMvcSample02.Controllers
{
    public class HomeController : Controller
    {
        [HttpGet]
        public ActionResult Index()
        {
            آغاز کار با صفحه صفر است//
            var list = BlogPostDataSource.GetLatestBlogPosts(pageNumber: 0);
            return View(list); نمایش ابتدایی صفحه//
        }

        [HttpPost]
        [AjaxOnly]
        [OutputCache(Location = OutputCacheLocation.None, NoStore = true)]
        public virtual ActionResult PagedIndex(int? page)
        {
            var pageNumber = page ?? 0;
            var list = BlogPostDataSource.GetLatestBlogPosts(pageNumber);
            if (list == null || !list.Any())
                return Content("no-more-info"); //این شرط ما است برای نمایش عدم یافتن رکوردها

            return PartialView("_ItemsList", list);
        }

        [HttpGet]
        public ActionResult Post(int? id)
```

```
        {
            if (id == null)
                return Redirect("/");

            //todo: show the content here
            return Content("Post " + id.Value);
        }
    }
}
```

کنترلر برنامه را در اینجا ملاحظه می‌کنید. برای کار با اسکرول نامحدود، به ازای هر صفحه، نیاز به دو متد است:
الف) یک متد که بر اساس HttpGet کار می‌کند. این متد در اولین بار نمایش صفحه فراخوانی می‌گردد و اطلاعات صفحه آغازین را نمایش می‌دهد.
ب) متد دومی که بر اساس HttpPost کار کرده و محدود است به درخواستی‌های AjaxOnly همانند متد PagedIndex.
از این متد دوم برای پردازش کلیک‌های کاربر بر روی دکمه «بیشتر» استفاده می‌گردد. بنابراین تنها کاری که افزونه جی‌کوئری تدارک دیده شده ما باید انجام دهد، ارسال شماره صفحه است. سپس با استفاده از این شماره، بازه مشخصی از اطلاعات دریافت و نهایتا یک PartialView رندر شده برای افزوده شدن به صفحه بازگشت داده می‌شود.

دو View برنامه

همانطور که برای بازگشت اطلاعات نیاز به دو اکشن متد است، برای رندر اطلاعات نیز به دو View نیاز داریم:
الف) یک PartialView که صرفا لیستی از اطلاعات را مطابق سلیقه ما رندر می‌کند. از این PartialView در متد PagedIndex استفاده خواهد شد:

```
@model IList<jQueryMvcSample02.Models.BlogPost>
<ul>
    @foreach (var item in Model)
    {
        <li>
            <h5>
                @Html.ActionLink(linkText: item.Title,
                                 actionName: "Post",
                                 controllerName: "Home",
                                 routeValues: new  { id = item.Id },
                                 htmlAttributes: null)
            </h5>
            @item.Body
        </li>
    }
</ul>
```

ب) یک View کامل که در بار اول نمایش صفحه، مورد استفاده قرار می‌گیرد:

```
@model IList<jQueryMvcSample02.Models.BlogPost>
@{
    ViewBag.Title = "Index";
    var loadInfoUrl = Url.Action(actionName: "PagedIndex", controllerName: "Home");
}
<h2>
    اسکرول نامحدود</h2>
@{ Html.RenderPartial("_ItemsList", Model); }
<div id="MoreInfoDiv">
</div>
<div align="center" style="margin-bottom: 9px;">
    <span id="moreInfoButton" style="width: 90%;" class="btn btn-info">بیشتر</span>
</div>
<div id="ProgressDiv" align="center" style="display: none">
    <br />
    <img src="@Url.Content("~/Content/images/loadingAnimation.gif")" alt="loading..."  />
</div>
@section JavaScript
{
    <script type="text/javascript">
        $(document).ready(function () {
            $("#moreInfoButton").InfiniteScroll({
                moreInfoDiv: '#MoreInfoDiv',
```

```
                progressDiv: '#ProgressDiv',
                loadInfoUrl: '@loadInfoUrl',
                loginUrl: '/login',
                errorHandler: function () {
                    alert('خطایی رخ داده است');
                },
                completeHandler: function () {
                    // اگر قرار است روی اطلاعات نمایش داده شده پردازش ثانوی صورت گیرد
                },
                noMoreInfoHandler: function () {
                    alert('اطلاعات بیشتری یافت نشد');
                }
            });
        });
    </script>
}
```

چند نکته در اینجا حائز اهمیت است:

1) مسیر دقیق اکشن متد PagedIndex توسط متد Url.Action تهیه شده است.

2) در ابتدای نمایش صفحه، متد Html.RenderPartial کار نمایش اولیه اطلاعات را انجام خواهد داد.

3) از div خالی MoreInfoDiv، به عنوان محل افزوده شدن اطلاعات Ajax ایی دریافتی استفاده می‌کنیم.

4) دکمه بیشتر در اینجا تنها یک span ساده است که توسط css به شکل یک دکمه نمایش داده خواهد شد (فایل‌های آن در پروژه پیوست موجود است).

5) ProgressDiv در ابتدای نمایش صفحه مخفی است. زمانیکه کاربر بر روی دکمه بیشتر کلیک می‌کند، توسط افزونه جی‌کوئری ما نمایان شده و در پایان کار مجددا مخفی می‌گردد.

6) section JavaScript کار استفاده از افزونه InfiniteScroll را انجام می‌دهد.

و کدهای افزونه اسکرول نامحدود

```
// <![CDATA[
(function ($) {
    $.fn.InfiniteScroll = function (options) {
        var defaults = {
            moreInfoDiv: '#MoreInfoDiv',
            progressDiv: '#Progress',
            loadInfoUrl: '/',
            loginUrl: '/login',
            errorHandler: null,
            completeHandler: null,
            noMoreInfoHandler: null
        };
        var options = $.extend(defaults, options);

        var showProgress = function () {
            $(options.progressDiv).css("display", "block");
        }

        var hideProgress = function () {
            $(options.progressDiv).css("display", "none");
        }

        return this.each(function () {
            var moreInfoButton = $(this);
            var page = 1;
            $(moreInfoButton).click(function (event) {
                showProgress();
                $.ajax({
                    type: "POST",
                    url: options.loadInfoUrl,
                    data: JSON.stringify({ page: page }),
                    contentType: "application/json; charset=utf-8",
                    dataType: "json",
                    complete: function (xhr, status) {
                        var data = xhr.responseText;
                        if (xhr.status == 403) {
                            window.location = options.loginUrl;
                        }
                        else if (status === 'error' || !data) {
                            if (options.errorHandler)
```

```
                                options.errorHandler(this);
                        }
                        else {
                            if (data == "no-more-info") {
                                if (options.noMoreInfoHandler)
                                    options.noMoreInfoHandler(this);
                            }
                            else {
                                var $boxes = $(data);
                                $(options.moreInfoDiv).append($boxes);
                            }
                            page++;
                        }
                        hideProgress();
                        if (options.completeHandler)
                            options.completeHandler(this);
                    }
                });
            });
        });
    };
})(jQuery);
// ]]>
```

ساختار افزونه اسکرول نامحدود به این شرح است:
هربار که کاربر بر روی دکمه بیشتر کلیک می‌کند، progress div ظاهر می‌گردد. سپس توسط امکانات jQuery Ajax، شماره صفحه (بازه انتخابی) به اکشن متد صفحه بندی اطلاعات ارسال می‌گردد. در نهایت اطلاعات را از کنترلر دریافت و به moreInfoDiv اضافه می‌کند. در آخر هم شماره صفحه را یکی افزایش داده و سپس progress div را مخفی می‌کند.

دریافت مثال و پروژه کامل این قسمت
jQueryMvcSample02.zip

نظرات خوانندگان

نویسنده: Information
تاریخ: ۱۳۹۲/۰۱/۰۴ ۱۴:۴۲

به نظر شما این دکمه برای سئو سایت مضر نیست؟

نویسنده: ناصر فرجی
تاریخ: ۱۳۹۲/۰۱/۰۴ ۱۴:۴۵

آقای نصیری همون طور که میدونید این روش با seo مشکل داره. برای رفع این مشکل به نظر شما چه کاری میشه انجام داد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۴ ۱۵:۷

کاری که من انجام می‌دم قرار دادن دو لینک در بالای هر مطلب است (لینک به مطلب بعدی و مطلب قبلی):

به این ترتیب موتور جستجو از یک مطلب شروع می‌کند و به راحتی تا آخرین مطلب سایت را می‌تواند پیمایش کند.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۴ ۱۵:۱۴

الان اگر به سایت رسمی wordpress مراجعه کنید، اکثر وبلاگ‌های آن به این ترتیب طراحی و ارائه می‌شن. ولی برای حفظ SEO، از تعبیه لینک به مطلب بعدی و قبلی استفاده می‌کنند؛ تا موتور جستجو به سادگی راه خودش را پیدا کند.

نویسنده: ناصر فرجی
تاریخ: ۱۳۹۲/۰۱/۰۴ ۲۰:۵۲

ممنون از پاسختون. چون سایت شما حالتی مثل یک وبلاگ داره میشه از این روش استفاده کرده. اما خیلی از سایت‌ها نمیشه این امکان رو گذاشت! خوشحال میشم راجع به حل معضلی به اسم seo در ajax مقالات بیشتری رو از شما یا سایر دوستان ببینیم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۴ ۲۱:۲۶

نهایتا یکی از اهدف مهم SEO این است که یک ربات بتواند سایت را راحت پیمایش کند. خیلی از سایت‌های دیگر هم برای این منظور از مفهومی به نام « site map » استفاده می‌کنند که به اکثر مداخل مهم سایت لینک دارد.

نویسنده: بهمن خلفی
تاریخ: ۱۳۹۲/۰۱/۱۱ ۹:۳۵

با تشکر از مطالب مفید شما
تقریبا اکثر امکاناتی که در سایت خودتان استفاده کردید را در این دوره به اشتراک گذاشتید که بسیار ارزشمند و کاربردی هستند که این جای بسی تشکر و قدردانی را دارد
یک خواهش هم بنده داشتم آن هم اینکه در مورد نمایش و نحوه کاربردی کردن پیام همین سایت یک مطلب ارائه دهید (البته مطالب مشابهی در همین سایت ^ و ^ و ^ وجود دارند) اما به زیبایی کار شما نیستند؟

با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۱۱ ۹:۵۴

من در این سایت بجای alert معمولی از افزونه jQuery noty استفاده کردم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۲۹ ۱۹:۱۳

نگارش جدید این افزونه و مثال را از اینجا می‌توانید دریافت کنید: jQueryMvcSample02_V2.zip
تغییرات:
- اضافه شدن history مشاهده صفحات جدید به دکمه back مرورگر. برای اینکار از افزونه Path.Js کمک گرفته شد. این مورد همچنین سبب می‌شود بتوان آدرس صفحه جاری را ذخیره و بعدا بازیابی کرد.
- اضافه شدن دو دراپ داون برای مرتب سازی بر اساس یک سری فیلد به صورت صعودی و یا نزولی
- محو دکمه بیشتر در زمان کلیک بر روی آن جهت جلوگیری از کلیک‌های بیش از حد با سرعت دریافت پایین اینترنت.

عنوان: **پیاده سازی امتیاز دهی ستاره‌ای به مطالب به کمک jQuery در ASP.NET MVC**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۴ ۲۳:۴۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, jQuery, MVC, JavaScript

در این قسمت قصد داریم با نحوه پیاده سازی امتیاز دهی ستاره‌ای به مطالب، که نمونه‌ای از آن‌را در سایت جاری در قسمت‌های مختلف آن مشاهده می‌کنید، آشنا شویم.

مدل برنامه

در ابتدای کار نیاز است تا ساختاری را جهت ارائه لیستی از مطالب که دارای گزینه امتیاز دهی می‌باشند، تهیه کنیم:

```
namespace jQueryMvcSample03.Models
{
    public class BlogPost
    {
        public int Id { set; get; }
        public string Title { set; get; }
        public string Body { set; get; }

        /// <summary>
        /// اطلاعات رای گیری یک مطلب به صورت یک خاصیت تو در تو یا پیچیده
        /// </summary>
        public Rating Rating { set; get; }

        public BlogPost()
        {
            Rating = new Rating();
        }
    }
}
```

```
namespace jQueryMvcSample03.Models
{
    //[ComplexType]
    public class Rating
    {
        public double? TotalRating { get; set; }
        public int? TotalRaters { get; set; }
        public double? AverageRating { get; set; }
    }
}
```

اگر با EF Code first آشنا باشید، خاصیت Rating تعریف شده در اینجا می‌تواند از نوع ComplexType تعریف شود که شامل جمع امتیازهای داده شده، تعداد کل رای دهنده‌ها و همچنین میانگین امتیازهای حاصل است.

منبع داده فرضی برنامه

```
using System.Collections.Generic;
using System.Linq;
using jQueryMvcSample03.Models;

namespace jQueryMvcSample03.DataSource
{
    /// <summary>
    /// منبع داده فرضی
    /// </summary>
    public static class BlogPostDataSource
    {
        private static IList<BlogPost> _cachedItems;
        /// <summary>
        /// با توجه به استاتیک بودن سازنده کلاس، تهیه کش، پیش از سایر فراخوانی‌ها صورت خواهد گرفت
        /// باید دقت داشت که این فقط یک مثال است و چنین کشی به معنای
        /// تهیه یک لیست برای تمام کاربران سایت است
        /// </summary>
        static BlogPostDataSource()
        {
            _cachedItems = createBlogPostsInMemoryDataSource();
        }

        /// <summary>
        /// هدف صرفا تهیه یک منبع داده آزمایشی ساده تشکیل شده در حافظه است
        /// </summary>
        private static IList<BlogPost> createBlogPostsInMemoryDataSource()
        {
            var results = new List<BlogPost>();
            for (int i = 1; i < 30; i++)
            {
                results.Add(new BlogPost { Id = i, Title = "عنوان " + i, Body = "متن ... متن ... متن "
+ i, Rating = new Rating { TotalRaters = i + 1, AverageRating = 3.5 } });
            }
            return results;
        }

        /// <summary>
        /// پارامترهای شماره صفحه و تعداد رکورد به ازای یک صفحه برای صفحه بندی نیاز هستند
        /// شماره صفحه از یک شروع می‌شود
        /// </summary>
        public static IList<BlogPost> GetLatestBlogPosts(int pageNumber, int recordsPerPage = 4)
        {
            var skipRecords = pageNumber * recordsPerPage;
            return _cachedItems
                        .OrderByDescending(x => x.Id)
                        .Skip(skipRecords)
                        .Take(recordsPerPage)
                        .ToList();
        }
    }
}
```

در این مثال نیز از یک منبع داده فرضی تشکیل شده در حافظه استفاده خواهیم کرد تا امکان اجرای پروژه پیوستی را بدون نیاز به بانک اطلاعاتی خاصی و بدون نیاز به مقدمات برپایی آن، به سادگی داشته باشید.
در این منبع داده ابتدا لیستی از مطالب تهیه شده و سپس کش می‌شوند. در ادامه توسط متد GetLatestBlogPosts بازه‌ای از این اطلاعات قابل بازیابی خواهند بود که برای استفاده در حالات صفحه بندی اطلاعات بهینه سازی شده است.

آشنایی با طراحی افزونه jQuery Star Rating

افزودن CSS نمایش امتیازها در ذیل هر مطلب

```
/* star rating system */
.post_rating
{
direction: ltr;
```

```
}
.rating
{
text-indent: -99999px;
overflow: hidden;
background-repeat: no-repeat;
display: inline-block;
width: 8px;
height: 16px;
}
.rating.stars
{
background-image: url('Images/star_rating.png');
}
.rating.stars.active
{
cursor: pointer;
}
.star-left_off
{
background-position: -0px -0px;
}
.star-left_on
{
background-position: -16px -0px;
}
.star-right_off
{
background-position: -8px -0px;
}
.star-right_on
{
background-position: -24px -0px;
}
```

برای نمایش ستاره‌ها و کار با تصویر Images/star_rating.png (که در پروژه پیوست قرار دارد) ابتدا نیاز است CSS فوق را به پروژه خود اضافه نمائید.

افزودن افزونه jQuery Star rating

```
// <![CDATA[
(function ($) {
    $.fn.StarRating = function (options) {
        var defaults = {
            ratingStarsSpan: '.rating.stars',
            postInfoUrl: '/',
            loginUrl: '/login',
            errorHandler: null,
            completeHandler: null,
            onlyOneTimeHandler: null
        };
        var options = $.extend(defaults, options);

        return this.each(function () {
            var ratingStars = $(this);

            $(ratingStars).unbind('mouseover');
            $(ratingStars).mouseover(function () {
                var span = $(this).parent("span");
                var newRating = $(this).attr("value");
                setRating(span, newRating);
            });

            $(ratingStars).unbind('mouseout');
            $(ratingStars).mouseout(function () {
                var span = $(this).parent("span");
                var rating = span.attr("rating");
                setRating(span, rating);
            });

            $(ratingStars).unbind('click');
            $(ratingStars).click(function () {
                var span = $(this).parent("span");
                var newRating = $(this).attr("value");
```

```
                var text = span.children("span");
                var pID = span.attr("post");
                var type = span.attr("sectiontype");
                postData({ postID: pID, rating: newRating, sectionType: type });
                span.attr("rating", newRating);
                setRating(span, newRating);
            });

            function setRating(span, rating) {
                span.find(options.ratingStarsSpan).each(function () {
                    var value = parseFloat($(this).attr("value"));
                    var imgSrc = $(this).attr("class");
                    if (value <= rating)
                        $(this).attr("class", imgSrc.replace("_off", "_on"));
                    else
                        $(this).attr("class", imgSrc.replace("_on", "_off"));
                });
            }

            function postData(dataJsonArray) {
                $.ajax({
                    type: "POST",
                    url: options.postInfoUrl,
                    data: JSON.stringify(dataJsonArray),
                    contentType: "application/json; charset=utf-8",
                    dataType: "json",
                    complete: function (xhr, status) {
                        var data = xhr.responseText;
                        if (xhr.status == 403) {
                            window.location = options.loginUrl;
                        }
                        else if (status === 'error' || !data) {
                            if (options.errorHandler)
                                options.errorHandler(this);
                        }
                        else if (data == "nok") {
                            if (options.onlyOneTimeHandler)
                                options.onlyOneTimeHandler(this);
                        }
                        else {
                            if (options.completeHandler)
                                options.completeHandler(this);
                        }
                    }
                });
            }
        });
    };
})(jQuery);
// ]]>
```

اطلاعات فوق، فایل jquery.StarRating.js را تشکیل می‌دهند که باید به پروژه اضافه گردند.
کاری که این افزونه انجام می‌دهد ردیابی حرکت ماوس بر روی ستاره‌های نمایش داده شده و سپس ارسال سه پارامتر ذیل به اکشن متدی که توسط پارامتر postInfoUrl مشخص می‌گردد، پس از کلیک کاربر می‌باشد:

```
{ postID: pID, rating: newRating, sectionType: type }
```

همانطور که ملاحظه می‌کنید به ازای هر قطعه رای گیری که به صفحه اضافه می‌شود، Id مطلب، رای داده شده و نام قسمت جاری، به اکشن متدی خاص ارسال خواهند گردید. sectionType از این جهت اضافه گردیده است تا بتوانید با بیش از یک جدول کار کنید و از این افزونه در قسمت‌های مختلف سایت به سادگی بتوانید استفاده نمائید.
در اینجا از errorHandler برای نمایش خطاها، از completeHandler برای نمایش تشکر به کاربر و از onlyOneTimeHandler برای نمایش اخطار مثلا «یکبار بیشتر مجاز نیستید به ازای یک مطلب رای دهید»، می‌توان استفاده کرد.

بنابراین تا اینجا فایل layout برنامه تقریبا چنین مداخلی را خواهد داشت:

```
<head>
    <title>@ViewBag.Title</title>
    <link href="@Url.Content("Content/starRating.css")" rel="stylesheet" type="text/css" />
    <link href="@Url.Content("Content/Site.css")" rel="stylesheet" type="text/css" />
```

```
    <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.unobtrusive-ajax.min.js")"
type="text/javascript"></script>
    <script src="@Url.Content("~/Scripts/jquery.validate.unobtrusive.min.js")"
type="text/javascript"></script>
    <script src="@Url.Content("~/Scripts/jquery.StarRating.js")" type="text/javascript"></script>
    @RenderSection("JavaScript", required: false)
</head>
```

طراحی یک HTML helper برای نمایش ستاره‌های امتیاز دهی

ابتدا پوشه استاندارد app_code را به پروژه اضافه کرده و سپس فایلی را به نام StarRatingHelper.cshtml، با محتوای ذیل به آن اضافه نمائید:

```
@using System.Globalization
@helper AddStarRating(int postId,
                      double? average = 0, int? postRatingsCount = 0, string type = "BlogPost",
                      string tooltip = "لطفا جهت رای دادن کلیک نمائید")
    {
        string actIt = "active ";
        if (!average.HasValue) { average = 0; }
        if (!postRatingsCount.HasValue) { postRatingsCount = 0; }

    <span class='postRating' rating='@average' post='@postId' title='@tooltip' sectiontype='@type'>
        @for (double i = .5; i <= 5.0; i = i + .5)
        {
            string left;
            if (i <= average)
            {
                left = (i * 2) % 2 == 1 ? "left_on" : "right_on";
            }
            else
            {
                left = (i * 2) % 2 == 1 ? "left_off" : "right_off";
            }
            <span class='rating stars @(actIt)star-@left' value='@i'></span>
        }
         
        @if (postRatingsCount > 0)
        {
            var ratingInfo = string.Format(CultureInfo.InvariantCulture, "{1} توسط 5 از {0:0.00} امتیاز
نفر", average, postRatingsCount);
            <span>@ratingInfo</span>
        }
        else
        {
            <span></span>
        }
    </span>
}
```

از این Html helper برای تشکیل ساختار نمایش قطعه امتیاز دهی به یک مطلب استفاده خواهیم کرد که توسط افزونه جی‌کوئری فوق ردیابی می‌شود.

کنترلر ذخیره سازی اطلاعات دریافتی برنامه

```
using System.Web.Mvc;
using System.Web.UI;
using jQueryMvcSample03.DataSource;
using jQueryMvcSample03.Security;

namespace jQueryMvcSample03.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
```

```
        {
            var postsList = BlogPostDataSource.GetLatestBlogPosts(pageNumber: 0);
            return View(postsList); //نمایش صفحه اصلی
        }


        [HttpPost]
        [AjaxOnly]
        [OutputCache(Location = OutputCacheLocation.None, NoStore = true)]
        public ActionResult SaveRatings(int? postId, double? rating, string sectionType)
        {
            if (postId == null || rating == null || string.IsNullOrWhiteSpace(sectionType))
                return Content(null); //اعلام بروز خطا

            if (!this.HttpContext.CanUserVoteBasedOnCookies(postId.Value, sectionType))
                return Content("nok"); //اعلام فقط یکبار مجاز هستید رای دهید

            switch (sectionType) //قسمت‌های مختلف سایت که در جداول مختلفی قرار دارند نیز می‌توانند گزینه
امتیاز دهی داشته باشند
            {
                case "BlogPost":
                    //الان شماره مطلب و رای ارسالی را داریم که می‌توان نسبت به ذخیره آن اقدام کرد
                    //مثلا
                    //_blogPostsService.SaveRating(postId.Value, rating.Value);
                    break;

                //... سایر قسمت‌های دیگر سایت

                default:
                    return Content(null); //اعلام بروز خطا
            }

            return Content("ok"); //اعلام موفقیت آمیز بودن ثبت اطلاعات
        }

        [HttpGet]
        public ActionResult Post(int? id)
        {
            if (id == null)
                return Redirect("/");

            //todo: show the content here
            return Content("Post " + id.Value);
        }
    }
}
```

در اینجا کنترلری را که کار پردازش کلیک کاربر را بر روی امتیازی خاص انجام می‌دهد، ملاحظه می‌کنید.
امضای اکشن متد SaveRatings دقیقا بر اساس سه پارامتر ارسالی توسط jquery.StarRating.js که پیشتر توضیح داده شد، تعیین گردیده است. در این متد ابتدا بررسی می‌شود که آیا اطلاعاتی دریافت شده است یا خیر. اگر خیر، null را بازگشت خواهد داد. سپس توسط متد CanUserVoteBasedOnCookies بررسی می‌شود که آیا کاربر می‌تواند (خصوصا مجددا) رای دهد یا خیر. این افزونه برای رای دهی کاربران وارد نشده به سیستم نیز مناسب است. به همین جهت از کوکی‌ها برای ثبت اطلاعات رای دادن کاربران استفاده گردیده است. پیاده سازی متد CanUserVoteBasedOnCookies را در ادامه ملاحظه خواهید نمود.
در ادامه در متد SaveRatings، یک switch تشکیل شده است تا بر اساس نام قسمت مرتبط به رای گیری، اطلاعات را بتوان به سرویس خاصی در برنامه هدایت کرد. مثلا اطلاعات قسمت مطالب به سرویس مطالب و قسمت نظرات به سرویس نظرات هدایت شوند.

متدهایی برای کار با کوکی‌ها در ASP.NET MVC

```
using System;
using System.Web;

namespace jQueryMvcSample03.Security
{
    public static class CookieHelper
    {
        public static bool CanUserVoteBasedOnCookies(this HttpContextBase httpContext, int postId,
string sectionType)
        {
```

```
            string key = sectionType + "-" + postId;
            var value = httpContext.GetCookieValue(key);
            if (string.IsNullOrWhiteSpace(value))
            {
                httpContext.AddCookie(key, key);
                return true;
            }
            return false;
        }

        public static void AddCookie(this HttpContextBase httpContextBase, string cookieName, string
value)
        {
            httpContextBase.AddCookie(cookieName, value, DateTime.Now.AddDays(30));
        }

        public static void AddCookie(this HttpContextBase httpContextBase, string cookieName, string
value, DateTime expires)
        {
            var cookie = new HttpCookie(cookieName)
            {
                Expires = expires,
                Value = httpContextBase.Server.UrlEncode(value) // For Cookies and Unicode characters
            };
            httpContextBase.Response.Cookies.Add(cookie);
        }

        public static string GetCookieValue(this HttpContextBase httpContext, string cookieName)
        {
            var cookie = httpContext.Request.Cookies[cookieName];
            if (cookie == null)
                return string.Empty; //cookie doesn't exist

            // For Cookies and Unicode characters
            return httpContext.Server.UrlDecode(cookie.Value);
        }
    }
}
```

در اینجا یک سری متد الحاقی را ملاحظه می‌کنید که برای ثبت اطلاعات رای داده شده یک کاربر بر اساس Id مطلب و نام قسمت متناظر با آن در یک کوکی طراحی شده‌اند. بدیهی است اگر تمام قسمت‌های برنامه شما محافظت شده هستند و کاربران حتما نیاز است ابتدا به سیستم لاگین نمایند، می‌توانید این قسمت را حذف کرده و اطلاعات postId و SectionType را به ازای هر کاربر، جداگانه در بانک اطلاعاتی ثبت و بازیابی نمائید (دقیق‌ترین حالت ممکن؛ البته برای سیستمی بسته که حتما تمام قسمت‌های آن نیاز به اعتبار سنجی دارند).

پیشنهادی در مورد نحوه ذخیره سازی اطلاعات دریافتی

```
using jQueryMvcSample03.Models;

namespace jQueryMvcSample03.DataSource
{
    public interface IBlogPostsService
    {
        void SaveRating(int postId, double rating);
    }

    public class SampleService : IBlogPostsService
    {
        /// <summary>
        /// یک نمونه از متد ذخیره سازی اطلاعات پیشنهادی
        /// فقط برای ایده گرفتن
        /// بدیهی است محل قرارگیری اصلی آن در لایه سرویس برنامه شما خواهد بود
        /// </summary>
        public void SaveRating(int postId, double rating)
        {
            BlogPost post = null;
            //post = _blogCtx.Find(postId); // بر اساس شماره مطلب، مطلب یافت شده و فیلدهای آن تنظیم
می‌شوند
            if (post == null) return;

            if (!post.Rating.TotalRaters.HasValue) post.Rating.TotalRaters = 0;
            if (!post.Rating.TotalRating.HasValue) post.Rating.TotalRating = 0;
```

```
            if (!post.Rating.AverageRating.HasValue) post.Rating.AverageRating = 0;

            post.Rating.TotalRaters++;
            post.Rating.TotalRating += rating;
            post.Rating.AverageRating = post.Rating.TotalRating / post.Rating.TotalRaters;

            // todo: call save changes at the end.
        }
    }
}
```

همانطور که عنوان شد، سه داده Id مطلب، رای داده شده و نام قسمت متناظر به اکشن متد ارسال می‌شود. از نام قسمت، برای انتخاب سرویس ذخیره سازی اطلاعات استفاده خواهیم کرد. این سرویس می‌تواند شامل متدی به نام SaveRating، همانند کدهای فوق باشد که Id مطلب و عدد رای حاصل به آن ارسال می‌گردند. ابتدا بر اساس این Id، مطلب متناظر یافت شده و سپس اطلاعات Rating آن به روز خواهد شد. در پایان هم ذخیره سازی اطلاعات باید صورت گیرد.

Viewهای برنامه

قسمت پایانی کار ما در اینجا تهیه دو View است:
الف) یک Partial view که لیست مطالب را به همراه گزینه رای دهی به آن‌ها رندر می‌کند.
ب) View کاملی که از این Partial View استفاده کرده و همچنین افزونه jquery.StarRating.js را فراخوانی می‌کند.

```
@using System.Text.RegularExpressions
@model IList<jQueryMvcSample03.Models.BlogPost>
<ul>
    @foreach (var item in Model)
    {
        <li>
            <fieldset>
            <legend>مطلب @item.Id</legend>
                <h5>
                    @Html.ActionLink(linkText: item.Title,
                                actionName: "Post",
                                controllerName: "Home",
                                routeValues: new { id = item.Id },
                                htmlAttributes: null)
                </h5>
                @item.Body
                <div class="post_rating">
                    @Html.Raw(Regex.Replace(@StarRatingHelper.AddStarRating(item.Id,
item.Rating.AverageRating, item.Rating.TotalRaters, "BlogPost").ToHtmlString(), @">\s+<", "><"))
                </div>
            </fieldset>
        </li>
    }
</ul>
```

کدهای _ItemsList.cshtml را در اینجا ملاحظه می‌کند که در آن نحوه فراخوانی متد کمکی StarRatingHelper.AddStarRating ذکر شده است.
اگر به کدهای آن دقت کنید از Regex.Replace برای حذف فاصله‌های خالی و خطوط جدید بین تگ‌ها استفاده گردیده است. اگر اینکار انجام نشود، نیمه‌های ستاره‌های نمایش داده شده، با فاصله از یکدیگر رندر می‌شوند که صورت خوشایندی ندارد.

و نهایتا View ایی که از این اطلاعات استفاده می‌کنید ساختار زیر را خواهد داشت:

```
@model IList<jQueryMvcSample03.Models.BlogPost>
@{
    ViewBag.Title = "Index";
    var postInfoUrl = Url.Action(actionName: "SaveRatings", controllerName: "Home");
}
<h2>
    سیستم امتیاز دهی</h2>
@{ Html.RenderPartial("_ItemsList", Model); }
@section JavaScript
{
    <script type="text/javascript">
```

```
        $(document).ready(function () {
            $(".rating.stars.active").StarRating({
                ratingStarsSpan: '.rating.stars',
                postInfoUrl: '@postInfoUrl',
                loginUrl: '/login',
                errorHandler: function () {
                    alert('خطایی رخ داده است');
                },
                completeHandler: function () {
                    alert('با تشکر! رای شما با موفقیت ثبت شد');
                },
                onlyOneTimeHandler: function () {
                    alert('فقط یکبار می‌توانید به ازای هر مطلب رای دهید');
                }
            });
        });
    </script>
}
```

در این View لیستی از مطالب دریافت و به partial view طراحی شده برای نمایش ارسال می‌شود. سپس افزونه StarRating نیز تنظیم و به صفحه اضافه خواهد گردید. نکته مهم آن تعیین صحیح اکشن متدی است که قرار است اطلاعات را دریافت کند و نحوه مقدار دهی آن‌را توسط متغیر postInfoUrl مشاهده می‌کنید.

دریافت کدها و پروژه کامل این قسمت

jQueryMvcSample03.zip

نظرات خوانندگان

نویسنده: فرزین بیاتی
تاریخ: ۱۳۹۲/۱۲/۱۸ ۱۸:۱۲

سلام،برای من پیغام "خطایی رخ داده است" نمایش داده می‌شود و هیچگاه اکشن SaveRating صدا زده نمیشه،البته من از StarRatingHelper در یک view به جای partial view استفاده کردم (این view فقط یک مطلب از سایت من رو نشان میده و من میخوام کاربر بتونه به مطلب امتیاز بده) نمیدونم مشکل از کجاست؟ مثال شما رو دانلود کردم و به درستی کار میکنه،ممنون میشم راهنمایی بفرمایید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۱۸ ۱۸:۳۸

احتمال دارد مسیرها را صحیح تنظیم نکردید. مطلب زیر در مورد نحوه‌ی دیباگ برنامه‌های Ajax ایی مفید است:

« نحوه استفاده از افزونه Firebug برای دیباگ برنامه‌های ASP.NET مبتنی بر jQuery »

نویسنده: محمد صاحب
تاریخ: ۱۳۹۲/۱۲/۱۹ ۹:۳۳

آیا این دو اسکریپت با هم تداخل دارن؟ جدا جدا اجرا میشن ولی با هم سیستم امتیاز دهی از کار میفته
یا نحوه استفاده‌ی من غلطه؟

```
$(document).ready(function () {
            $("#moreInfoButton").InfiniteScroll({
                moreInfoDiv: '#MoreInfoDiv',
                progressDiv: '#ProgressDiv',
                loadInfoUrl: '/Media/PagedIndex',
                loginUrl: '/login',
                errorHandler: function () { alert('خطایی رخ داده است'); },
                completeHandler: function () { },
                noMoreInfoHandler: function () { alert('اطلاعات بیشتری یافت نشد'); }
            });


            $(".rating.stars.active").StarRating({
                ratingStarsSpan: '.rating.stars',
                postInfoUrl: '/Media/SaveRatings',
                loginUrl: '/login',
                errorHandler: function () { alert('خطایی رخ داده است'); },
                completeHandler: function () { alert('با تشکر! رای شما با موفقیت ثبت شد'); },
                onlyOneTimeHandler: function () { alert('فقط یکبار می‌توانید به ازای هر مطلب رای دهید');
}
            });
        }
        );
```

با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۱۹ ۹:۴۹

تداخلی ندارند. فقط محتوایی که از طریق Ajax به صفحه اضافه می‌شود، باید توسط متد on (پیشتر live بود) تحت نظر قرار گرفته شود و گرنه ندید گرفته می‌شود؛ چون در زمان document.ready وجود خارجی نداشته‌است.

نویسنده: ایمان اسلامی
تاریخ: ۱۳۹۳/۰۲/۱۸ ۱۵:۵۰

آیا میتوان از این مثال در asp.net web form هم استفاده کرد؟

نویسنده: وحید طجری
تاریخ: ۱۳۹۳/۰۶/۰۵ ۱۸:۵۹

بعد از آپدیت قسمت نظرات با ajax اسکریپت از کار میفته.توی این مثال از متد On چطوری باید استفاده کنیم؟ توی فایل jquery.StarRating باید تغییرات بدیم یا توی اسکریپتی که داخل صفحه مینویسیم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۶/۰۶ ۱:۷

الزاما نیازی به استفاده از متد On نیست. در رویدادگردان complete عملیات ajax ایی، متد StarRating را دوباره فراخوانی کنید.

عنوان: **ارائه کاربری ساده‌تر انتخاب چندین آیتم از یک لیست به کمک افزونه TagIt در ASP.NET MVC**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۵ ۱۳:۰
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, jQuery, MVC, JavaScript

چندی قبل مطلبی را در مورد بررسی تفصیلی رابطه چند به چند در این سایت مطالعه کردید. در آن مطلب صرفا به بحث ذخیره سازی اطلاعات دریافتی از کاربر اشاره شد. برای مثال اگر مطلبی چندین برچسب دارد، چگونه باید این‌ها را در بانک اطلاعاتی به نحو صحیحی ذخیره کرد.
در مطلب جاری قصد داریم با نحوه ارائه یک UI کاربر پسند برای این منظور آشنا شویم و سؤال مهم هم این است: «چگونه می‌توان کار کاربر را در حین وارد کردن تعدادی از برچسب‌های مرتبط با یک مطلب ساده‌تر کرد؟». برای این منظور یکی از راه حل‌هایی که در بسیاری از سایت‌ها مرسوم شده است، استفاده از افزونه‌هایی مانند jQuery TagIt می‌باشد که در ادامه با نحوه استفاده از آن در ASP.NET MVC آشنا خواهیم شد.

پیشنیازها:
دریافت افزونه TagIt
همچنین دریافت jQuery UI (افزونه TagIt برای نمایش لیست Auto Complete آیتم‌ها از jQuery UI در پشت صحنه استفاده می‌کند)

```
<head>
    <title>@ViewBag.Title</title>
    <link href="@Url.Content("~/Content/TagIt/jquery-ui-1.8.23.custom.css")" rel="stylesheet"
type="text/css" />
    <link href="@Url.Content("~/Content/TagIt/tagit-simple-blue.css")" rel="stylesheet" type="text/css"
/>
    <link href="@Url.Content("Content/Site.css")" rel="stylesheet" type="text/css" />
    <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.unobtrusive-ajax.min.js")"
```

```
type="text/javascript"></script>
    <script src="@Url.Content("~/Scripts/jquery.validate.unobtrusive.min.js")"
type="text/javascript"></script>
    <script src="@Url.Content("~/Content/TagIt/jquery-ui-1.8.23.custom.min.js")"
type="text/javascript"></script>
    <script src="@Url.Content("~/Content/TagIt/tagit.js")" type="text/javascript"></script>
    @RenderSection("JavaScript", required: false)
</head>
```

که نهایتا نیاز است یک چنین تعاریفی را به فایل layout برنامه اضافه کنیم.

آشنایی با مدل برنامه

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

namespace jQueryMvcSample04.Models
{
    public class BlogPostViewModel
    {
        [DisplayName("عنوان"), Required(ErrorMessage = "*")]
        public string Title { set; get; }

        [DisplayName("متن"), Required(ErrorMessage = "*")]
        public string Body { set; get; }

        /// <summary>
        /// آرایه‌ای محدود از برچسب‌های این مطلب خاص به صورت جی‌سون که پیشتر ثبت شده است
        /// هدف استفاده در حین ویرایش مطلب
        /// </summary>
        public string InitialTags { set; get; }

        /// <summary>
        /// آرایه‌ای جی‌سونی از تمام برچسب‌های موجود در سیستم
        /// هدف نمایش منوی انتخاب برچسب‌ها از لیست
        /// </summary>
        public string TagsSource { set; get; }

        /// <summary>
        /// آرایه‌ای از برچسب‌های وارد شده توسط کاربر در حین ثبت مطلب
        /// </summary>
        [DisplayName("برچسب‌ها"), Required(ErrorMessage = "*")]
        public string[] Tags { set; get; }

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

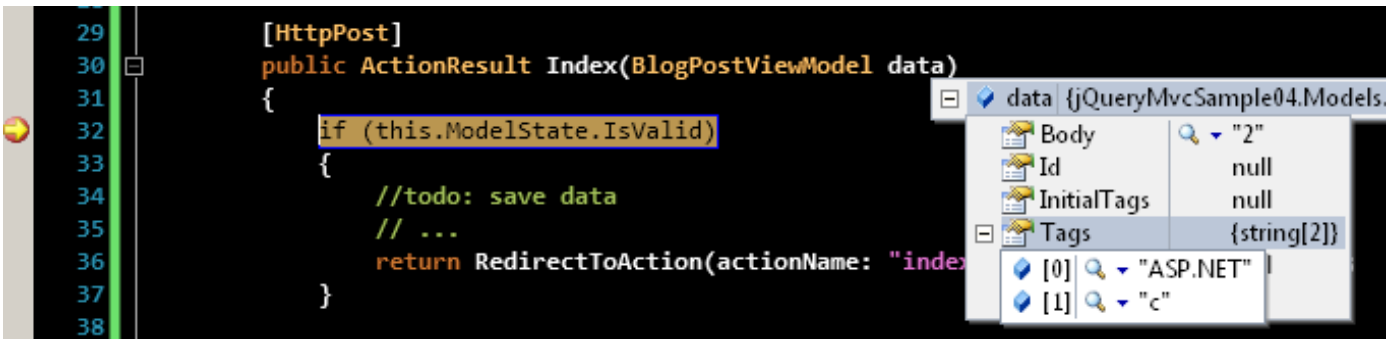
اگر به نام این کلاس دقت کنید، به ViewModel ختم شده است. از این لحاظ که حاوی خواصی می‌باشد که عموما جهت رندر کردن صحیح UI مورد استفاده قرار می‌گیرند و معادلی در سمت بانک اطلاعاتی نخواهند داشت.
افزونه TagIt برای نمایش اطلاعات خود به دو منبع داده نیاز دارد:
الف) TagsSource : لیستی است به فرمت JSON، از هر آنچه که در سیستم پیشتر به عنوان یک برچسب ثبت شده است. از این لیست برای نمایش منوی خودکار انتخاب آیتم‌ها استفاده می‌شود.
ب) InitialTags : لیستی است به فرمت JSON، از تمام برچسب‌های مرتبط با یک مطلب. از این اطلاعات در حین ویرایش یک مطلب استفاده خواهد شد.

در این ViewModel یک خاصیت دیگر به شکل آرایه، به نام Tags تعریف شده است که لیست برچسب‌های وارد شده توسط کاربر را دریافت خواهد کرد.

معرفی کنترلر برنامه

```
using System.Web.Mvc;
```

```
using jQueryMvcSample04.Extensions;
using jQueryMvcSample04.Models;

namespace jQueryMvcSample04.Controllers
{
    public class HomeController : Controller
    {
        [HttpGet]
        public ActionResult Index(int? id)
        {
            //در ابتدای کار تمام تگ‌های موجود در سیستم از بانک اطلاعاتی دریافت خواهند شد
            //از این تگ‌ها برای تشکیل منوی انتخاب برچسب‌ها استفاده می‌شود
            var tagsSource = new[] { "C#", "C++", "C", "ASP.NET", "MVC" }.ToJson();

            //همچنین صرفا برچسب‌های مطلب جاری که پیشتر ثبت شده‌اند نیز باید از بانک اطلاعاتی دریافت
گردند
            //از این برچسب‌ها برای ویرایش یک مطلب موجود استفاده خواهد شد
            var init = new[] { "ASP.NET" }.ToJson();

            var model = new BlogPostViewModel
            {
                TagsSource = tagsSource,
                InitialTags = init,
                Id = id
            };
            return View(model);
        }

        [HttpPost]
        public ActionResult Index(BlogPostViewModel data)
        {
            if (this.ModelState.IsValid)
            {
                //todo: save data
                // ...
                return RedirectToAction(actionName: "index", controllerName: "home");
            }

            //در صورت بروز خطا مجددا اطلاعات موجود نمایش داده خواهند شد
            data.TagsSource = new[] { "C#", "C++", "C", "ASP.NET", "MVC" }.ToJson();
            data.InitialTags = data.Tags.ToJson();
            return View(data);
        }
    }
}
```



با توجه به توضیحاتی که ارائه شد، کنترلر برنامه ساختار واضح‌تری را یافته است. در اولین بار نمایش صفحه، لیست منبع داده تگ‌ها و همچنین تگ‌های مرتبط با یک مطلب (در صورت وجود) به View ارائه خواهند شد.

از همین ViewModel، در عملیات Post نیز استفاده گردیده و اطلاعات دریافت می‌گردد.

تعریف متد الحاقی ToJson مورد استفاده را نیز در ادامه ملاحظه می‌نمائید:

```
using System.Linq;
```

```
using System.Web.Script.Serialization;

namespace jQueryMvcSample04.Extensions
{
    public static class JsonExt
    {
        public static string ToJson(this string[] initialTags)
        {
            if (initialTags == null || !initialTags.Any())
                return "[]";
            else
                return new JavaScriptSerializer().Serialize(initialTags);
        }
    }
}
```

و مرحله آخر تعریف View متناظر است

```
@model jQueryMvcSample04.Models.BlogPostViewModel
@{
    ViewBag.Title = "Index";
}
@using (Html.BeginForm())
{
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>ثبت مطلب جدید</legend>
        @Html.HiddenFor(model => model.Id)
        <div class="editor-label">
            @Html.LabelFor(model => model.Title)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Title)
            @Html.ValidationMessageFor(model => model.Title)
        </div>
        <div class="editor-label">
            @Html.LabelFor(model => model.Body)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Body)
            @Html.ValidationMessageFor(model => model.Body)
        </div>
        <div class="editor-label">
            @Html.LabelFor(model => model.Tags)
        </div>
        <div class="editor-field">
            <ul id="tagsList" dir="ltr" name="Tags">
            </ul>
            @Html.ValidationMessageFor(model => model.Tags)
        </div>
        <p>
            <input type="submit" value="Create" />
        </p>
    </fieldset>
}
@section JavaScript
{
    <script type="text/javascript">
    $(document).ready(function () {
            var tagsSource = @Html.Raw(Model.TagsSource);
            $('#tagsList').tagit({
                 tagSource: tagsSource,
                 select: true,
                 triggerKeys: ['enter', 'comma', 'tab'],
                 initialTags:  @Html.Raw(Model.InitialTags)
              });
});
    </script>
}
```

در این View دو نکته حائز اهمیت هستند:

الف) برای نمایش افزونه TagIt از یک ul با id مساوی tagsList استفاده شده است.
ب) خواص اضافی موجود در ViewModel که اطلاعات JSON ایی مورد نیاز را بازگشت می‌دهند در قسمت اسکریپت صفحه مورد استفاده قرار گرفته‌اند. در اینجا نیاز است از Html.Raw استفاده شود تا اطلاعات مرتبط با JSON اشتباها encode نشده و قابل استفاده باشند.

دریافت مثال و پروژه کامل این قسمت
jQueryMvcSample04.zip

نظرات خوانندگان

نویسنده: محسن.د
تاریخ: ۱۳۹۲/۰۱/۱۳ ۱۱:۶

بسیار عالی .
بعد از خوندن مطلب ، اول فکر کردم که استفاده از jQuery UI ضروری نیست و فقط در صورتی که بخوایم امکان autoComplete رو فعال کنیم باید رفرنسی به اون بدیم اما بعد از حذف کردن متوجه شدم که اینطور نیست و کلا این plugin هم به jQuery و هم به jQuery UI وابسته است و بدون UI کار نمی‌کنه .

عنوان: **تغییر ترتیب آیتم‌های یک لیست به کمک افزونه jquery.sortable در ASP.NET MVC**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۶ ۱:۹
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, jQuery, MVC, JavaScript

در این مطلب قصد داریم ترتیب عناصر نمایش داده شده توسط یک لیست را به کمک افزونه بسیار سبک وزن jquery.sortable تغییر داده و نتایج را در سمت سرور مدیریت کنیم. این افزونه بر اساس امکانات کشیدن و رها ساختن HTML5 کار می‌کند و با مرورگرهای IE8 به بعد سازگار است.

مدل‌های برنامه

```
using System.Collections.Generic;

namespace jQueryMvcSample05.Models
{
    public class Survey
    {
        public int Id { set; get; }
        public string Title { set; get; }

        public virtual ICollection<SurveyItem> SurveyItems { set; get; }
    }
}
```

```
namespace jQueryMvcSample05.Models
{
    public class SurveyItem
    {
        public int Id { set; get; }
        public string Title { set; get; }
        public int Order { set; get; }

        //[ForeignKey("SurveyId")]
        public virtual Survey Survey { set; get; }
        public int SurveyId { set; get; }
    }
}
```

به کمک این ساختار قصد داریم اطلاعات یک سیستم نظر سنجی را نمایش دهیم.
تعدادی نظر سنجی به همراه گزینه‌های آن‌ها تعریف خواهند شد (یک رابطه one-to-many است). سپس توسط افزونه sortable می‌خواهیم ترتیب قرارگیری گزینه‌های آن‌را مشخص کنیم یا تغییر دهیم.

منبع داده فرضی برنامه

```
using System.Collections.Generic;
using jQueryMvcSample05.Models;

namespace jQueryMvcSample05.DataSource
{
    /// <summary>
    /// یک منبع داده فرضی جهت دموی ساده‌تر برنامه
    /// </summary>
    public static class SurveysDataSource
    {
        private static IList<Survey> _surveysCache;
        static SurveysDataSource()
        {
            _surveysCache = createSurveys();
        }

        public static IList<Survey> SystemSurveys
        {
```

```
            get { return _surveysCache; }
        }

        private static IList<Survey> createSurveys()
        {
            var results = new List<Survey>();
            for (int i = 1; i < 6; i++)
            {
                results.Add(new Survey
                {
                    Id = i,
                    Title = "نظر سنجی " + i,
                    SurveyItems = new List<SurveyItem>
                    {
                       new SurveyItem{ Id = 1, SurveyId = i, Title = "گزینه 1", Order = 1 },
                       new SurveyItem{ Id = 2, SurveyId = i, Title = "گزینه 2", Order = 2 },
                       new SurveyItem{ Id = 3, SurveyId = i, Title = "گزینه 3", Order = 3 },
                       new SurveyItem{ Id = 4, SurveyId = i, Title = "گزینه 4", Order = 4 }
                    }
                });
            }
            return results;
        }
    }
}
```

در اینجا نیز از یک منبع داده فرضی تشکیل شده در حافظه جهت سهولت دموی برنامه استفاده خواهد شد.

کدهای کنترلر برنامه

```
using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using System.Web.UI;
using jQueryMvcSample05.DataSource;
using jQueryMvcSample05.Security;

namespace jQueryMvcSample05.Controllers
{
    public class HomeController : Controller
    {
        [HttpGet]
        public ActionResult Index()
        {
            var surveysList = SurveysDataSource.SystemSurveys;
            return View(surveysList);
        }

        [HttpPost]
        [AjaxOnly]
        [OutputCache(Location = OutputCacheLocation.None, NoStore = true)]
        public ActionResult SortItems(int? surveyId, string[] items)
        {
            if (items == null || items.Length == 0 || surveyId == null)
                return Content("nok");

            updateSurvey(surveyId, items);

            return Content("ok");
        }

        /// <summary>
        /// این متد جهت آشنایی با پروسه به روز رسانی ترتیب گزینه‌ها در اینجا قرار گرفته است
        /// بدیهی است محل قرارگیری آن باید در لایه سرویس برنامه اصلی باشد
        /// </summary>
        private static void updateSurvey(int? surveyId, string[] items)
        {
            var itemIds = new List<int>();
            foreach (var item in items)
            {
                itemIds.Add(int.Parse(item.Replace("item-row-", string.Empty)));
            }

            var survey = SurveysDataSource.SystemSurveys.FirstOrDefault(x => x.Id == surveyId.Value);
            if (survey == null)
```

```
                return;

            int order = 0;
            foreach (var itemId in itemIds)
            {
                order++;
                var surveyItem = survey.SurveyItems.FirstOrDefault(x => x.Id == itemId);
                if (surveyItem == null) continue;
                surveyItem.Order = order;
            }

            //todo: call save changes ....
        }
    }
}
```

و کدهای View متناظر

```
@model IList<jQueryMvcSample05.Models.Survey>
@{
    ViewBag.Title = "Index";
    var sortUrl = Url.Action(actionName: "SortItems", controllerName: "Home");
}
<h2>
    نظر سنجی‌ها</h2>
@foreach (var survey in Model)
{
    <fieldset>
        <legend>@survey.Title</legend>
        <div id="sortable-@survey.Id">
            @foreach (var surveyItem in survey.SurveyItems.OrderBy(x => x.Order))
            {
                <div id="item-row-@surveyItem.Id">
                    <span class="handles">::</span>
                    @surveyItem.Title
                </div>
            }
        </div>
    </fieldset>
}
<div>
    لطفا برای تغییر ترتیب آیتم‌های تعریف شده، از امکان کشیدن و رها کردن تعریف شده بر
    روی آیکون‌های :: در کنار هر آیتم استفاده نمائید.
</div>
@section JavaScript
{
    <script type="text/javascript">
        $(document).ready(function () {
            $('div[id^="sortable"]').sortable({ handle: 'span' }).bind('sortupdate', function (e, ui) {
                var sortableItemId = $(ui.item).parent().attr('id');
                var surveyId = sortableItemId.replace('sortable-', '');
                var items = [];
                $('#' + sortableItemId + ' div').each(function () {
                    items.push($(this).attr('id'));
                });
                //alert(items.join('&'));
                $.ajax({
                    type: "POST",
                    url: "@sortUrl",
                    data: JSON.stringify({ items: items, surveyId: surveyId }),
                    contentType: "application/json; charset=utf-8",
                    dataType: "json",
                    complete: function (xhr, status) {
                        var data = xhr.responseText;
                        if (xhr.status == 403) {
                            window.location = "/login";
                        } else if (status === 'error' || !data || data == "nok") {
                            alert('خطایی رخ داده است');
                        }
                        else {
                            alert('انجام شد');
                        }
                    }
                });
            });
```

```
        });
    </script>
}
```

توضیحات

در اینجا نیاز بود تا ابتدا کدهای کنترلر و View ارائه شوند، تا بتوان در مورد ارتباطات بین آن‌ها بهتر بحث کرد.
در ابتدای نمایش صفحه Home، رکوردهای نظرسنجی‌ها از منبع داده دریافت شده و به View ارسال می‌شوند. در View برنامه یک حلقه تشکیل گردیده و این موارد رندر خواهند شد.
هر نظر سنجی با یک div بیرونی که با id مساوی sortable شروع می‌شود، آغاز گردیده و گزینه‌های آن نظر سنجی نیز توسط divهایی با id مساوی item-row شروع خواهند گردید. هر کدام از این idها حاوی id رکوردهای متناظر هستند. از این idها در کدهای برنامه جهت یافتن یک نظر سنجی یا یک ردیف مشخص برای به روز رسانی ترتیب آن‌ها استفاده خواهیم کرد.
ادامه کار، به تنظیمات و اعمال افزونه sortable مرتبط می‌شود. توسط تنظیم ذیل به jQuery اعلام خواهیم کرد، هرجایی یک div با id شروع شده با sortable یافتی، افزونه sortable را به آن متصل کن:

```
$('div[id^="sortable"]').sortable
```

در ادامه در ناحیه و div ایی که عمل کشیدن و رها شدن رخ داده، id این div را بدست آورده و سپس کلیه row-itemهای آن را در آرایه‌ای به نام items قرار می‌دهیم:

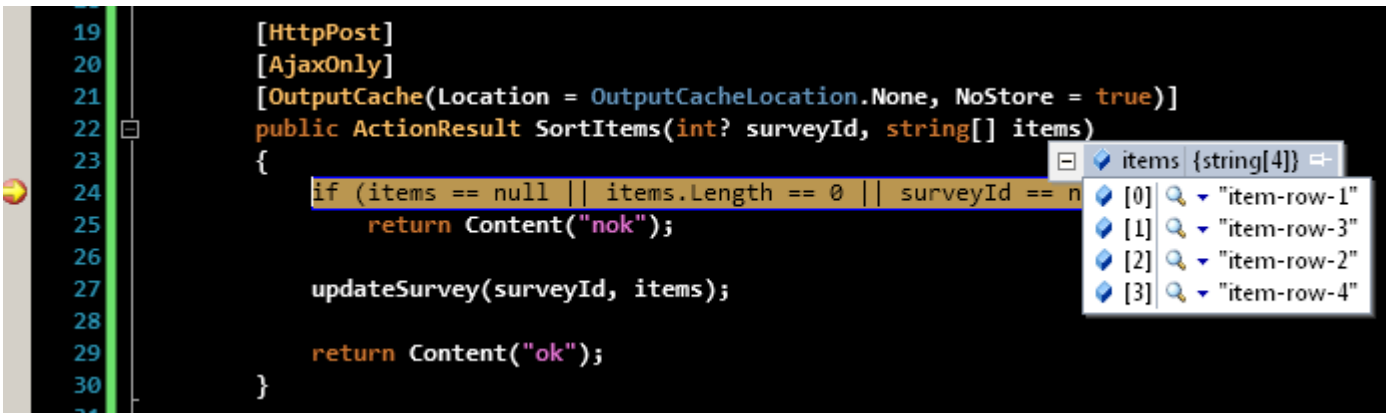
```
var sortableItemId = $(ui.item).parent().attr('id');
var surveyId = sortableItemId.replace('sortable-', '');
var items = [];
$('#' + sortableItemId + ' div').each(function () {
  items.push($(this).attr('id'));
});
```

اکنون که به id یک نظر سنجی و همچنین idهای ردیف‌های مرتب شده حاصل دسترسی داریم، آن‌ها را توسط jQuery Ajax به کنترلر برنامه ارسال می‌کنیم:

```
data: JSON.stringify({ items: items, surveyId: surveyId })
```

امضای اکشن متد SortItems نیز دقیقا بر همین مبنا تنظیم شده است:

```
public ActionResult SortItems(int? surveyId, string[] items)
```



اطلاعاتی که در اینجا دریافت می‌شوند در متد updateSurvey مورد استفاده قرار خواهند گرفت. بر اساس surveyId دریافتی،

نظرسنجی مرتبط را یافته و سپس به گزینه‌های آن دست خواهیم یافت. اکنون نوبت به پردازش آرایه items دریافت شده است. این آرایه بر اساس انتخاب کاربر مرتب شده است.

دریافت کدها و پروژه کامل این قسمت
jQueryMvcSample05.zip

نظرات خوانندگان

نویسنده: علی حق جو
تاریخ: ۱۳۹۲/۰۲/۳۰ ۱۰:۴۵

با سلام.
امکان مرتب سازی یک table با این کامپوننت وجود دارد؟
سعی من بی نتیجه بود. با تشکر.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۳۰ ۱۱:۱۱

بهتره به issue tracker مخصوص آن مراجعه کنید. یک نمونه از کار با جدول

عنوان: **ارسال کوئری استرینگ‌ها به همراه عملیات Ajax در jQuery**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۶ ۱۰:۳۹
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, jQuery, MVC, JavaScript

فرض کنید اطلاعات صفحه‌ای بر اساس کوئری استرینگ دریافتی رندر می‌شوند. مثلا یک گرید و یا حتی یک صفحه ویرایش اطلاعات. در حین فراخوانی متد Ajax در jQuery اطلاعات کوئری استرینگ‌های موجود به سرور ارسال نخواهند شد و صرفا اطلاعاتی که در پارامتر data آن صریحا ذکر می‌شوند، به سرور ارسال می‌گردند.
برای رفع این نقیصه با استفاده از قطعه کد زیر می‌توان کلیه کوئری استرینگ‌های صفحه را یافت و به شیءایی به نام urlParams به صورت خاصیت اضافه کرد:

```
var urlParams;
(window.onpopstate = function () {
    var match,
        pl     = /\+/g,  // Regex for replacing addition symbol with a space
        search = /([^&=]+)=?([^&]*)/g,
        decode = function (s) { return decodeURIComponent(s.replace(pl, " ")); },
        query  = window.location.search.substring(1);

urlParams = {};
    while (match = search.exec(query))
       urlParams[decode(match[1])] = decode(match[2]);
})();
```

سپس اگر قسمت ارسال اطلاعات متد ajax در حال فراخوانی چنین شکلی را داشته باشد:

```
$.ajax({
                    type: "POST",
                    url: "@sortUrl",
                    data: JSON.stringify({ items: items, surveyId: surveyId }),
```

نیاز خواهیم داشت تا urlParams را با آن یکی کنیم. اینکار نیز توسط متد extend خود jQuery قابل انجام است:

```
function jsonConcat(defaults, options) {
 /* merge defaults and options, without modifying defaults */
 return $.extend({}, defaults, options);
}
```

بنابراین در نهایت قسمت ارسال اطلاعات ما برای الحاق کلیه کوئری استرینگ‌های صفحه چنین شکلی را خواهد یافت:

```
var ajaxData = { items: items, surveyId: surveyId };
    $.ajax({
        type: "POST",
        url: "@sortUrl",
        data: JSON.stringify(jsonConcat(ajaxData,urlParams)),
```

و در سمت سرور امضای متدی که اطلاعات به آن ارسال می‌شوند، در این مثال خاص شامل items و surveyId به همراه نام کوئری استرینگ‌های مدنظر می‌تواند باشد و اطلاعات به صورت خودکار به آن‌ها بایند خواهند شد.

عنوان: **افزودن مشخص کننده فیلد اجباری در ASP.NET MVC به صورت خودکار توسط jQuery**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۶ ۱۳:۳۸
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, jQuery, MVC, JavaScript

در ASP.NET MVC زمانیکه خاصیتی با ویژگی Required مزین می‌شود، تا زمان اعتبار سنجی فرم، هیچ نشانی را از خود بروز نمی‌دهد. برای مثال علاقمندیم مانند شکل زیر، یک ستاره پس از فیلدهای اجباری ظاهر گردد:

انجام اینکار نیز با دو سطر کد نویسی توسط jQuery قابل انجام است.
در ادامه خروجی رندر شده فیلدی را که پرکردن آن الزامی است مشاهده می‌کنید:

```
<div class="editor-label">
            <label for="Title">عنوان</label>
        </div>
        <div class="editor-field">
            <input class="text-box single-line" data-val="true" data-val-required="*" id="Title"
name="Title" type="text" value="" />
            <span class="field-validation-valid" data-valmsg-for="Title" data-valmsg-
replace="true"></span>
        </div>
```

تنها کاری که باید صورت گیرد، یافتن مواردی است که حاوی data-val-required هستند و سپس افزودن یک ستاره پس از آن‌ها:

```
$(function() {
            $('[data-val-required]').after('<span class="required-indicator"> (*)</span>');
        });
```

برای اعمال رنگ به آن نیز می‌توان کمی css سایت را ویرایش کرد:

```
.required-indicator
{
color: red;
font-size: 1.2em;
font-weight: bold;
}
```

نظرات خوانندگان

نویسنده: مرتضی
تاریخ: ۱۳۹۳/۰۱/۰۵ ۱۲:۵۶

فقط با استفاده از CSS هم میشه

```
input[type=text]:required +  span::after {
            content: "*";
            color: red;
            font-size: large;
        }
```

عنوان: **حذف یک ردیف از اطلاعات به همراه پویانمایی محو شدن اطلاعات آن توسط jQuery در ASP.NET MVC**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۷ ۰:۳
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, jQuery, MVC, JavaScript

فرض کنید تعدادی ردیف در گزارشی نمایش داده شده‌اند. قصد داریم برای هر ردیف یک دکمه حذف را قرار دهیم. این حذف باید Ajax ایی باشد؛ به علاوه در حین حذف ردیف، پویانمایی محو آن ردیف را نیز سبب شود.

مدل و منبع داده برنامه

```
namespace jQueryMvcSample06.Models
{
    public class BlogPost
    {
        public int Id { set; get; }
        public string Title { set; get; }
        public string Body { set; get; }
    }
}
```

```
using System.Collections.Generic;
using jQueryMvcSample06.Models;

namespace jQueryMvcSample06.DataSource
{
    /// <summary>
    /// منبع داده فرضی جهت سهولت دموی برنامه
    /// </summary>
    public static class BlogPostDataSource
    {
        private static IList<BlogPost> _cachedItems;
        static BlogPostDataSource()
        {
            _cachedItems = createBlogPostsInMemoryDataSource();
        }

        /// <summary>
        /// هدف صرفا تهیه یک منبع داده آزمایشی ساده تشکیل شده در حافظه است
        /// </summary>
        private static IList<BlogPost> createBlogPostsInMemoryDataSource()
        {
            var results = new List<BlogPost>();
            for (int i = 1; i < 30; i++)
            {
                results.Add(new BlogPost { Id = i, Title = "عنوان " + i, Body = "متن ... متن ... متن "
+ i});
            }
            return results;
```

```
        }

        public static IList<BlogPost> LatestBlogPosts
        {
            get { return _cachedItems; }
        }
    }
}
```

در اینجا مدل برنامه که ساختار نمایش یک سری مطلب را تهیه می‌کند، ملاحظه می‌کنید؛ به علاوه یک منبع داده فرضی تشکیل شده در حافظه جهت سهولت دموی برنامه.

کنترلر برنامه

```
using System.Web.Mvc;
using System.Web.UI;
using jQueryMvcSample06.DataSource;
using jQueryMvcSample06.Security;

namespace jQueryMvcSample06.Controllers
{
    public class HomeController : Controller
    {
        [HttpGet]
        public ActionResult Index()
        {
            var postsList = BlogPostDataSource.LatestBlogPosts;
            return View(postsList);
        }

        [AjaxOnly]
        [HttpPost]
        [OutputCache(Location = OutputCacheLocation.None, NoStore = true)]
        public ActionResult DeleteRow(int? postId)
        {
            if (postId == null)
                return Content(null);

            //todo: delete post from db

            return Content("ok");
        }
    }
}
```

کنترلر برنامه بسیار ساده بوده و نکته خاصی ندارد. در حین اولین بار نمایش صفحه، لیست مطالب را به View مرتبط ارسال می‌کند. همچنین یک اکشن متد حذف ردیف‌های نمایش داده شده را نیز در اینجا تدارک دیده‌ایم. این اکشن متد از طریق ارسال اطلاعات به صورت Ajax، شماره مطلب را در اختیار برنامه قرار می‌دهد که توسط آن در ادامه برای مثال می‌توان این رکورد را از بانک اطلاعاتی حذف کرد. امضای متد DeleteRow بر اساس پارامترهای ارسالی توسط jQuery Ajax مشخص و تنظیم شده‌اند:

```
data: JSON.stringify({ postId: postId }),
```

View برنامه

```
@model IEnumerable<jQueryMvcSample06.Models.BlogPost>
@{
    ViewBag.Title = "Index";
    var postUrl = Url.Action(actionName: "DeleteRow", controllerName: "Home");
}
<h2>
    حذف یک ردیف از اطلاعات به همراه پویانمایی محو شدن اطلاعات آن</h2>
<table>
    <tr>
```

```
        <th>
            عملیات
        </th>
        <th>
            عنوان
        </th>
    </tr>
    @foreach (var item in Model)
    {
        <tr>
            <td>
                <span id="row-@item.Id">حذف</span>
            </td>
            <td>
                @item.Title
            </td>
        </tr>
    }
</table>
@section JavaScript
{
    <script type="text/javascript">
        $(function () {
            $('span[id^="row"]').click(function () {
                var span = $(this);
                var postId = span.attr('id').replace('row-', '');
                var tableRow = span.parent().parent();
                $.ajax({
                    type: "POST",
                    url: '@postUrl',
                    data: JSON.stringify({ postId: postId }),
                    contentType: "application/json; charset=utf-8",
                    dataType: "json",
                    complete: function (xhr, status) {
                        var data = xhr.responseText;
                        if (xhr.status == 403) {
                            window.location = "/login";
                        }
                        else if (status === 'error' || !data || data == "nok") {
                            alert('خطایی رخ داده است');
                        }
                        else {
                            $(tableRow).fadeTo(600, 0, function () {
                                $(tableRow).remove();
                            });
                        }
                    }
                });
            });
        });
    </script>
}
```

کدهای View برنامه را در ادامه ملاحظه می‌کنید. اطلاعات مطالب دریافتی به صورت یک جدول در صفحه نمایش داده شده‌اند. در هر ردیف توسط یک span که با css تزئین گردیده است، یک دکمه حذف را تدارک دیده‌ایم. برای اینکه در حین کار با jQuery بتوانیم id هر ردیف را بدست بیاوریم، این id را در قسمتی از id این span اضافه شده قرار داده‌ایم.
در کدهای اسکریپتی صفحه، ابتدا کلیک بر روی کلیه spanهایی که id آن‌ها با row شروع می‌شود را مونیتور خواهیم کرد:

```
$('span[id^="row"]').click(function () {
```

سپس هر زمان که بر روی یکی از این spanها کلیک شد، می‌توان بر اساس span جاری، id و همچنین tableRow مرتبط را استخراج کرد:

```
var span = $(this);
var postId = span.attr('id').replace('row-', '');
var tableRow = span.parent().parent();
```

اکنون که به این اطلاعات دسترسی پیدا کرده‌ایم، تنها کافی است آن‌ها را توسط متد ajax به کنترلر برنامه برای پردازش نهایی ارسال نمائیم. همچنین در پایان کار عملیات، توسط متدهای fadeTo و remove ایی که ملاحظه می‌کنید، سبب حذف نمایشی یک

ردیف به همراه پویانمایی محو آن خواهیم شد.

دریافت کدها و پروژه کامل این قسمت
jQueryMvcSample06.zip

نظرات خوانندگان

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۹/۲۶ ۱۰:۵۹

در این روش، در صورتی که در طرف کنترلر از ValidateAntiForgeryToken استفاده شده باشه، این روش با خطا مواجه میشه درسته؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۶ ۱۱:۴۲

توکن را به view اضافه کنید (یکبار در بالای صفحه). در قسمت JSON.stringify مقدار آن‌را خوانده و ارسال کنید:

```
var token = $('[name=__RequestVerificationToken]').val();

$.ajax({
    // .....
    data: { __RequestVerificationToken: token, ..... },
    // .....
});
```

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۹/۲۶ ۱۳:۵۰

این روشم امتحان کردم قبلا اما خطای زیر دریافت میشه (توی فایرباگ چک کردم)

```
The required anti-forgery form field "__RequestVerificationToken" is not present
```

در حالی که در بالای ویو این دستور هست

```
@Html.AntiForgeryToken()
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۶ ۱۵:۵۹

با روش زیر امتحان کنید:

```
function addToken(data) {
   data.__RequestVerificationToken = $("input[name=__RequestVerificationToken]").val();
   return data;
}

$.ajax({
    // .....
    data: addToken({ postId: postId }), // اضافه کردن توکن
    dataType: "html", // نوع داده مهم است
    // .....
});
```

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۹/۲۶ ۱۷:۴۰

متشکرم از پاسخ شما
اینم امتحان کردم نشد

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۶ ۱۷:۵۷

من قبل از ارسال، امتحانش کرده بودم. جهت تکمیل بحث، پروژه آن پیوست شد:
jQueryMvcSample06_v2.zip

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۹/۲۶ ۱۸:۴۸

بله درست کار کرد
من این خط را حذف نکرده بودم

```
contentType: "application/json; charset=utf-8",
```

این خط را کامنت کردم درست شد. از صبر و شکیبایی شما متشکرم

نویسنده: صابر فتح الهی
تاریخ: ۱۳۹۲/۰۹/۲۶ ۲۲:۳۳

البته راه دیگری هم پیدا کردم.
به نظرم جالب اومد.
ابتدا یک اکشن فیلتر تعریف شده و در ان هدر درخواست خوانده شده و با محتوی کوکی چک می‌شود در صورتی که برابر باشند عمل انجام خواهد شد با این تفاوت باید توکن در زمان ارسال در هدر درخواست قرار گیرد مانند زیر:
کد اکشن فیلتر:

```
public class ValidateJsonAntiForgeryTokenAttribute : ActionFilterAttribute
    {
#region Methods (1)

// Public Methods (1)

        /// <summary>
        /// Called when [action executing].
        /// </summary>
        /// <param name="actionContext">The action context.</param>
        public void OnActionExecuting(HttpActionContext actionContext)
        {
            try
            {
                var cookieName = AntiForgeryConfig.CookieName;
                var headers = actionContext.Request.Headers;
                var cookie = headers
                    .GetCookies()
                    .Select(c => c[AntiForgeryConfig.CookieName])
                    .FirstOrDefault();
                var rvt = headers.GetValues("__RequestVerificationToken").FirstOrDefault();
                AntiForgery.Validate(cookie != null ? cookie.Value : null, rvt);
            }
            catch
            {
                actionContext.Response =
actionContext.Request.CreateErrorResponse(HttpStatusCode.Forbidden, "Unauthorized request.");
            }
        }

#endregion Methods
    }
```

سپس در اکشن دلخواه باید این اکشن فیلتر را بکار برد (در درخواست‌های json)

```
[HttpPost]
        //[ValidateAntiForgeryToken]
        [ValidateJsonAntiForgeryToken]
        [OutputCache(Location = OutputCacheLocation.None, NoStore = true)]
        public virtual ActionResult DeletePhoto(int? id)
        {
```

```
            string ret="error";
            if (id != null && id > 0)
            {
                bool result = _imageGalleryService.Value.Photos.Value.Delete(id.ToInt32());
                ret = result ? "ok" : "nok";
            }
            return Content(ret);
            //return Json(new { ret });
        }
```

سپس در طرف ویو به صورت زیر عمل شود

```
$.ajax({
                    ....
                type: "POST",
                url: '@Url',
                data: JSON.stringify({ id: Id }),
                contentType: "application/json; charset=utf-8",
                // این قسمت اضافه شود
                headers: { __RequestVerificationToken:
$("input[name=__RequestVerificationToken]").val() },
                dataType: "json",
                 ..........
```

امیدوارم برای دوستان هم کاربرد داشته باشد

نویسنده: ساشا
تاریخ: ۱۳۹۲/۰۹/۲۷ ۱۵:۴

برای Web Form هم به همین صورت است ؟ آیا این مثال برای وب فرم هم کاربرد دارد ؟ نحوه استفاده آن یکسان است ؟

نویسنده: ساشا
تاریخ: ۱۳۹۲/۰۹/۲۷ ۱۵:۵

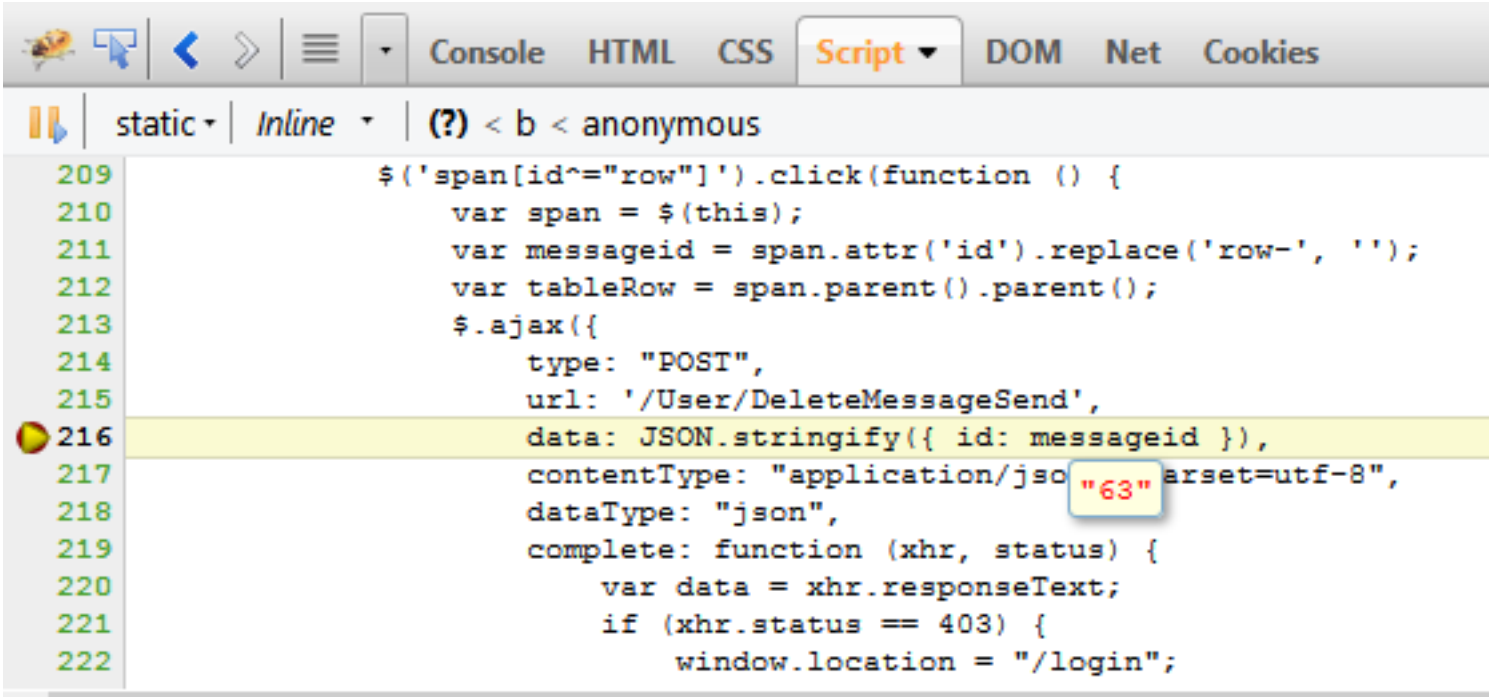
لطفا سورس پروژه رو بگذارید ، تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۲۷ ۱۵:۱۳

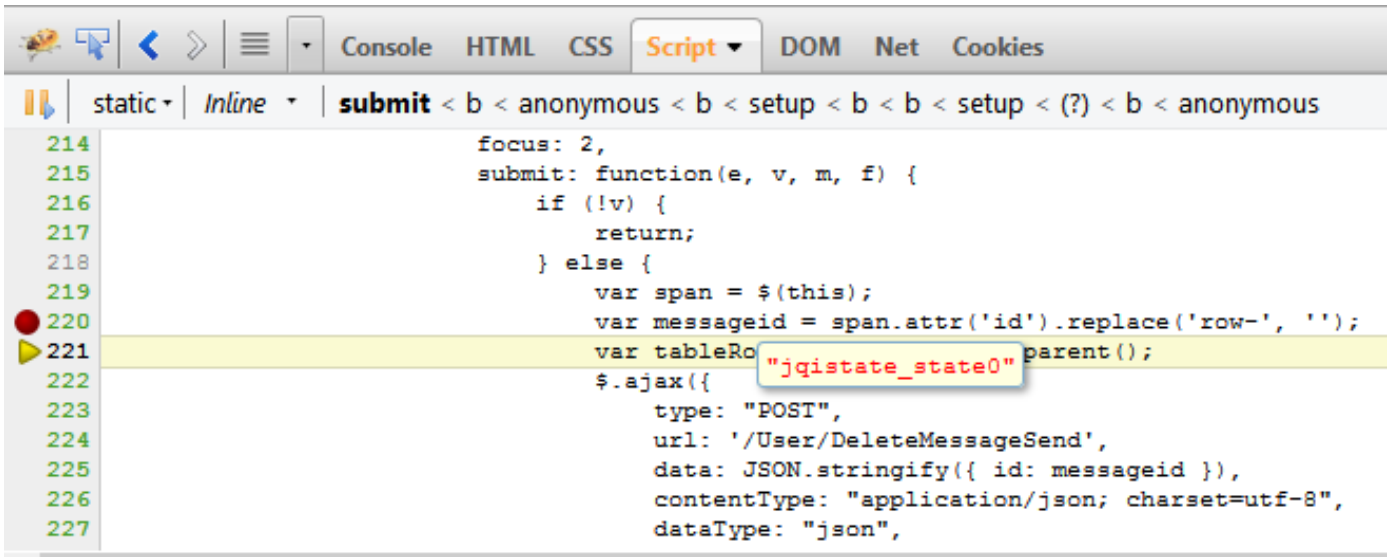
خیر (سمت سرور آن یکسان نیست). این دوره مربوط به MVC است؛ مثال آن هم مرتبط است به MVC. سورس پروژه هم در بحث پیوست شده (دو بار).

نویسنده: رضا منصوری
تاریخ: ۱۳۹۲/۱۱/۰۷ ۱۰:۱۶

سلام. ابتدا از مطلب خوبتون تشکر میکنم . واقعا بسیار جالب و کاربردیه .
من از این مثال استفاده کردم و درست کار میکنه ولی وقتی میخوام از کاربر تایید برای حذف بگیرم (با استفاده از jquery-impromptu) یه مشکلی پیش میاد .
حالت عادی که Id مربوط به پیغاممون درست انتخاب میشه.



ولی وقتی تایید میگیرم



موس روی messageid هستش و نوشته‌ی قرمز مقدار id هستش که قطعا درست نیست.

مسلما مشکل از این خطه که id رو انتخاب میکنه

```
var span = $(this);
var messageid = span.attr('id').replace('row-', '');
```

ولی نمیدونم چجوری id رو انتخاب کنم!

اینم کدها که تایید کاربرو بهش اضافه کردم

```
<script type="text/javascript">
        $(function() {
            $('span[id^="row"]').click(function() {
                $.prompt("آیا برای حذف اطلاعات موجود اطمینان دارید ؟",
                    {
                        title: 'Title',
                        buttons: { "بله": true, "خیر": false },
                        focus: 2,
                        submit: function(e, v, m, f) {
                            if (!v) {
                                return;// وقتی کاربر تایید نکرد
                            } else {// وقتی کاربر تایید کرد
                                var span = $(this);
                                var messageid = span.attr('id').replace('row-', '');
                                var tableRow = span.parent().parent();
                                $.ajax({
                                    type: "POST",
                                    url: '@postUrl',
                                    data: JSON.stringify({ id: messageid }),
                                    contentType: "application/json; charset=utf-8",
                                    dataType: "json",
                                    complete: function(xhr, status) {
                                        var data = xhr.responseText;
                                        if (xhr.status == 403) {
                                            window.location = "/login";
                                        } else if (status === 'error' || !data || data == "nok") {
                                            var noty = window.noty({ text: "خطایی رخ داده", type:
'error', layout: 'center', timeout: 2500 });
                                        } else {
                                            $(tableRow).fadeTo(600, 0, function() {
                                                $(tableRow).remove();
                                            });
                                            var noty = window.noty({ text: "پیغام حذف شد", type:
'information', layout: 'center', timeout: 2500 });
                                        }
                                    }
                                });

                            }

                        }
                    });

        });
        });
    </script>
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۰۷ ۱۴:۳۸

زمانیکه $.prompt نمایان می‌شود، this ذکر شده به قسمتی از طرحبندی خود این افزونه اشاره می‌کند و نه لایه زیرین آن که جدول ما است. بنابراین سه سطر مربوط به یافتن Id و سطر جاری را پیش از فراخوانی $.prompt قرار دهید.

عنوان: به روز رسانی غیرهمزمان قسمتی از صفحه به کمک jQuery در ASP.NET MVC
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۷ ۱۹:۵۳
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, jQuery, MVC, JavaScript

یک صفحه شلوغ و سنگین را در نظر بگیرید. برای مثال قرار است ابتدا مطلب خاصی در سایت نمایش یابد و سپس ادامه صفحه که شامل انبوهی از لیست نظرات مرتبط با آن مطلب است به صورت غیرهمزمان و Ajax ایی بدون توقف پردازش صفحه، در فرصتی مناسب از سرور دریافت و به صفحه اضافه گردد (به روز رسانی قسمتی از صفحه در فرصت مناسب). در این حالت چون نمایش اولیه صفحه سریع صورت می‌گیرد، کاربر نهایی آنچنان احساس کند بودن بازکردن صفحات سایت را نخواهد داشت. در ادامه نحوه پیاده سازی این روش را به کمک jQuery Ajax بررسی خواهیم کرد.

مدل و کنترلر برنامه

```
namespace jQueryMvcSample07.Models
{
    public class BlogPost
    {
        public int Id { set; get; }
        public string Title { set; get; }
        public string Body { set; get; }
    }
}
```

```
using System.Web.Mvc;
using System.Web.UI;
using jQueryMvcSample07.Models;
using jQueryMvcSample07.Security;

namespace jQueryMvcSample07.Controllers
{
    public class HomeController : Controller
    {
        [HttpGet]
        public ActionResult Index()
        {
            return View(); //نمایش یک منوی ساده در ابتدای کار
        }

        [HttpGet]
        public ActionResult ShowSynchronous()
        {
            var model = getModel();
            return View(model); //نمایش همزمان
        }

        private static BlogPost getModel()
        {
            //شبیه سازی یک عملیات طولانی
            System.Threading.Thread.Sleep(3000);
            var model = new BlogPost
            {
                Title = "عنوان ... ",
                Body = "مطلب... "
            };
            return model;
        }

        [HttpGet]
        public ActionResult ShowAsynchronous()
        {
            return View(); //نمایش ابتدایی صفحه
        }

        [HttpPost]
        [AjaxOnly]
        [OutputCache(Location = OutputCacheLocation.None, NoStore = true)]
        public ActionResult RenderAsynchronous()
        {
```

```
            //دریافت اطلاعات به صورت غیرهمزمان
            var model = getModel();
            return PartialView(viewName: "_Post", model: model);
        }
    }
}
```

مدل برنامه، بیانگر ساختار اطلاعات مطلبی است که قرار است نمایش یابد.
در کنترلر Home، ابتدا اکشن متد Index آن فراخوانی شده و در این حالت دو لینک زیر نمایش داده می‌شوند:

```
@{
    ViewBag.Title = "Index";
}
<h2>
    نمایش اطلاعات به صورت همزمان و غیرهمزمان</h2>
<ul>
    <li>
        @Html.ActionLink(linkText: "نمایش همزمان", actionName: "ShowSynchronous", controllerName:
"Home")
    </li>
    <li>
        @Html.ActionLink(linkText: "نمایش غیر همزمان", actionName: "ShowAsynchronous", controllerName:
"Home")
    </li>
</ul>
```

لینک اول، به اکشن متد ShowSynchronous اشاره می‌کند و لینک دوم به اکشن متد ShowAsynchronous.
در هر دو صفحه نهایتا از یک Partial View به نام _Post.cshtml برای نمایش اطلاعات استفاده خواهد شد:

```
@model jQueryMvcSample07.Models.BlogPost
<fieldset>
    <legend>@Model.Title</legend>
    @Model.Body
</fieldset>
```

زمانیکه کاربر بر روی لینک نمایش همزمان کلیک می‌کند، به صفحه زیر هدایت می‌شود:

```
@model jQueryMvcSample07.Models.BlogPost
@{
    ViewBag.Title = "ShowSynchronous";
}

<h2>نمایش همزمان</h2>
@{ Html.RenderPartial("_Post", Model); }
```

این صفحه، یک صفحه متداول است و اطلاعات آن دقیقا در زمان نمایش صفحه اخذ شده و چون در اینجا از یک Sleep عمدی برای تولید اطلاعات استفاده گردیده است، نمایش آن حداقل سه ثانیه طول خواهد کشید.
در حالتیکه کاربر بر روی لینک نمایش غیرهمزمان کلیک می‌کند، صفحه زیر را مشاهده خواهد کرد:

```
@{
    ViewBag.Title = "ShowAsynchronous";
    var loadInfoUrl = Url.Action(actionName: "RenderAsynchronous", controllerName: "Home");
}
<h2>
    نمایش غیر همزمان</h2>
<div id="info" align="center">
</div>
<div id="progress" align="center" style="display: none">
    <br />
    <img src="@Url.Content("~/Content/images/loadingAnimation.gif")" alt="loading..."  />
</div>
@section JavaScript
{
    <script type="text/javascript">
        $(function () {
            $("#progress").css("display", "block");
            $.ajax({
```

```
                type: "POST",
                url: '@loadInfoUrl',
                complete: function (xhr, status) {
                    var data = xhr.responseText;
                    if (xhr.status == 403) {
                        window.location = "/login";
                    }
                    else if (status === 'error' || !data || data == "nok") {
                        alert('خطایی رخ داده است');
                    }
                    else {
                        $("#progress").css("display", "none");
                        $("#info").html(data);
                    }
                }
            });
        });
    </script>
}
```

نمایش ابتدایی این صفحه بسیار سریع است. در ابتدای کار progress bar ایی فعال شده و سپس از طریق jQuery Ajax درخواست دریافت اطلاعات رندر شده اکشن متدی به نام RenderAsynchronous به سرور ارسال می‌شود. چون عملیات Ajax غیرهمزمان است، کاربر نیازی نیست تا رندر شدن کامل صفحه ابتدا صبر کند و سپس کل صفحه به او نمایش داده شود. در اینجا ابتدا صفحه به صورت کامل نمایان شده و سپس درخواستی Ajax ایی به سرور ارسال می‌گردد. در پایان عملیات، Partial View یاد شده رندر گردیده و در div ایی با id مساوی info نمایش داده می‌شود.
به این ترتیب می‌توان حس سریع بودن سایت را زمانیکه قسمتی از صفحه نیاز به زمان بیشتری برای نمایش اطلاعات دارد، به کاربر منتقل کرد.

دریافت پروژه کامل این قسمت
jQueryMvcSample07.zip

نظرات خوانندگان

نویسنده: رشوند
تاریخ: ۱۳۹۳/۰۶/۰۶ ۱۸:۲۲

با سلام
در قسمتی از پروژه کاملا همانند شما عمل کردم و حتی فایل پروژه را دانلود و نگاه انداختم کاملا شبیه هم بود, ولی متاسفانه در هنگام برگرداندن اطلاعات اکشن RenderAsynchronous , برنامه به _layout رفته, موارد انجا را هم (از جمله لینک ها, توضیحتان و ...) در خروجی (div info) اضافه می‌کند... اگر این اشکال متداول است لطفا راهنمایی بفرمایید که چه نکته ای را رعایت نکرده ام.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۶/۰۶ ۱۹:۶

- بررسی کنید آیا return PartialView دارید یا return View ؟ حالت return View فایل layout را هم لحاظ می‌کند.
- همچنین امکان نال تعریف کردن layout به صورت صریح هم وجود دارد:

```
@{
Layout = null;
}
```

عنوان: **ویژگی StringLength و اعمال maxlength به صورت خودکار در ASP.NET MVC**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۸ ۱۰:۲۴
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, jQuery, MVC, JavaScript

خاصیت نام را که توسط ویژگی StringLength مزین شده است درنظر بگیرید:

```
[StringLength(10, ErrorMessage = "حداکثر 10 حرف")]
public string Name { set; get; }
```

قصد داریم در سمت کاربر، فیلد متناظر را طوری تنظیم کنیم که واقعا کاربر نتواند بیش از 10 حرف را وارد کند:

```
$(function () {
            $("input[data-val-length-max]").each(function (i, e) {
                var input = $(e);
                var maxlength = input.attr("data-val-length-max");
                input.attr("maxlength", maxlength);
            });
        });
```

توضیحات این کدها به نحوه رندر ویژگی StringLength بر می‌گردد:

```
<input class="text-box single-line" data-val="true" data-val-length="حداکثر 10 حرف"
data-val-length-max="10" data-val-required="(*)" id="Name" name="Name" type="text" value="" />
```

توسط قطعه کد جی‌کوئری نوشته شده، ابتدا کلیه inputهایی که دارای data-val-length-max هستند را می‌یابیم و در این بین مقدار data-val-length-max را یافته و سپس خاصیت maxlength آن‌ها را به صورت پویا تنظیم می‌کنیم. به این ترتیب کاربر دیگر نمی‌تواند رشته‌ای دلخواه را در ابتدای کار وارد نماید و به طول تنظیم شده محدود می‌گردد.

نظرات خوانندگان

نویسنده: samin
تاریخ: ۱۳۹۲/۰۱/۱۴ ۱۰:۴۱

از زحمات بی دریغ شما ممنون.پاینده باشید