عنوان: ذخیره TreeView ساخته شده توسط KendoUI در Asp.net MVC
نویسنده: حسین حسینی 128
تاریخ: ۱۳۹۲/۰۸/۱۳ ۱۲:۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, jQuery, Kendo UI

همانطور که از نمونه مثال‌های خود Kendo UI مشاهده میشود ، نحوه استفاده از TreeView آن به صورت زیر است :

```
<div>
@(Html.Kendo().TreeView()
        .Name("treeview")
        .TemplateId("treeview-template")
        .HtmlAttributes(new { @class = "demo-section" })
        .DragAndDrop(true)
        .BindTo(Model.Where(e=>e.ParentFolderID==null).OrderBy(e=>e.Order), mappings =>
        {
            mappings.For<DAL.Folder>(binding => binding
                    .ItemDataBound((item, folder) =>
                    {
                        item.Text = folder.FolderName;
                        item.SpriteCssClasses = "folder";
                        item.Expanded=true;
                        item.Id = folder.FolderID.ToString();
                    })
                    .Children(folder => folder.Folder1));
            mappings.For<DAL.Folder>(binding => binding
                    .ItemDataBound((item, folder) =>
                    {
                        item.Text = folder.FolderName;
                        item.SpriteCssClasses = " folder";
                        item.Expanded = true;
                        item.Id = folder.FolderID.ToString();
                    }));
        })
)
</div>
```

```
<style type="text/css" scoped>
    .demo-section {
        width: 200px;
    }

    #treeview  .k-sprite ,#treeview2 .k-sprite {
        background-image: url("@Url.Content("/Content/kendo/images/coloricons-sprite.png")");
    }
    .rootfolder { background-position: 0 0; }
    .folder { background-position: 0 -16px; }
    .pdf { background-position: 0 -32px; }
    .html { background-position: 0 -48px; }
    .image { background-position: 0 -64px; }
    .delete-link,.edit-link {
        width: 12px;
        height: 12px;
        overflow: hidden;
        display: inline-block;
        vertical-align: top;
        margin: 2px 0 0 3px;
        -webkit-border-radius: 5px;
        -mox-border-radius: 5px;
        border-radius: 5px;
    }
    .delete-link{
        background: transparent url("@Url.Content("/Content/kendo/images/close.png")") no-repeat 50%
50%;
    }
    .edit-link{
        background: transparent url("@Url.Content("/Content/kendo/images/edit.png")") no-repeat 50%
50%;
    }
</style>
```

استفاده از این TreeView ساده است ولی اگر احتیاج داشته باشیم که پس از drag&drop کردن گره‌ها آن را ذخیره کنیم چگونه باید عمل کنیم؟ این ریالسمت در خود Kendo تعبیه نشده است ، پس به صورت زیر عمل میکنیم :

پس از ساختن TreeView و اصلاح آن به شکل دلخواه لینک زیر را در ادامه اش می‌آوریم :

```
<a href="#" id="serialize">ذخیره</a>
```

سپس در تگ اسکریپت‌های خود این کد جاوا اسکریپت را برای serialize کردن تمام گره‌های TreeView مینویسیم :

```
$('#serialize').click(function () {
            serialized = serialize();
            window.location.href = "Folder/SaveMenu?serial=" + serialized + "!";
        });

        function serialize() {
            var tree = $("#treeview").data("kendoTreeView");
            var json = treeToJson(tree.dataSource.view());
            return JSON.stringify(json);
        }

        function treeToJson(nodes) {

            return $.map(nodes, function (n, i) {

                var result = { id: n.id};
                //var result = { text: n.text, id: n.id, expanded: n.expanded, checked: n.checked };
                if (n.hasChildren)
                    result.items = treeToJson(n.children.view());

                return result;
            });
        }
```

حال به تشریح کدها میپردازیم :

تابع serialize درخط اول تمام عناصر داخلی treeview را گرفته ، به صورت یک datasource قابل انقیاد درآورده وسپس datasource آن را به یک نمایش قابل تجزیه تبدیل میکند و به متد treeToJSON میفرستد.

```
var tree = $("#treeview").data("kendoTreeView");
var json = treeToJson(tree.dataSource.view());
```

تابع treeToJSON درخت را به عنوان یکسری گره گرفته و تمام عناصر آن را به فرمت json میبرد . (قسمتی که به صورت توضیحی درآمده میتواند برای بدست آوردن تمام اطلاعات گره از جمله متن و انتخاب شدن checkbox آن و غیره مورد استفاده قرار بگیرد) در ادامه این تابع اگر گره درحال استفاده فرزندی داشته باشد به صورت بازگشتی همین تابع برای آن فراخوانده میشود.

```
var result = { id: n.id};
 //var result = { text: n.text, id: n.id, expanded: n.expanded, checked: n.checked };
 if (n.hasChildren)
 result.items = treeToJson(n.children.view());
```

سپس اطلاعات برگشتی که در فرمت json هستند در خط آخر serialaize به رشته تبدیل میشوند و به رویداد کلیک که از آن فراخوانده شده بود بازمیگردند.

```
return JSON.stringify(json);
```

خط آخر رویداد نیز یک Action در Controller مورد نظر را هدف قرار میدهد و رشته بدست آمده را به آن ارسال میکند و صفحه redirect میشود.

```
window.location.href = "Folder/SaveMenu?serial=" + serialized + "!";
```

رشته ای که با عنوان serialize به کنترلر ارسال میشود مانند زیر است :

```
"[{\"id\":\"2\"},{\"id\":\"5\",\"items\":[{\"id\":\"3\"},{\"id\":\"6\"},{\"id\":\"7\"}]}]!"
```

این رشته مربوط به درختی به شکل زیر است :

همانطور که میبینید گره دوم که "پوشه چهارم45" نام دارد شامل سه فرزند است که در رشته داده شده با عنوان item شناخته شده است. حال باید این رشته با برنامه نویسی سی شارپ جداسازی کرد :

```
string serialized;
Dictionary<int, int> numbers = new Dictionary<int, int>();
```

```
public ActionResult SaveMenu(string serial)
        {
            var newfolders = new List<Folder>();
            serialized = serial;
            calculte_serialized(0);
            return RedirectToAction("Index");
        }
```

```
void calculte_serialized(int parent)
        {
            while (serialized.Length > 0)
            {
                var id_index=serialized.IndexOf("id");
                if (id_index == -1)
                {
                    return;
                }
                serialized = serialized.Substring(id_index + 5);
                var quote_index = serialized.IndexOf("\"");
                var id=serialized.Substring(0, quote_index);
                numbers.Add(int.Parse(id), parent);
                serialized = serialized.Substring(quote_index);
                var condition = serialized.Substring(0,3);
                switch (condition)
                {
                    case "\"},":
                        break;
                    case "\",\"":
                        calculte_serialized(int.Parse(id));
                        break;
                    case "\"}]":
                        return;
                        break;
                    default:
                        break;
                }
            }
        }
```

calculte_serialized با گرفتن 0 کار خود را شروع میکند یعنی از گره هایی که پدر ندارند و تمام idها را همراه با پدرشان در یک دیکشنری میریزد و هرکجا که به فرزندی برخورد به صورت بازگشتی فراخوانی میشود. پس از اجرای کامل آن ما درخت را در یک دیکشنری به صورت عنصرهای مجزا در اختیار داریم که میتوانیم در پایگاه داده ذخیره کنیم.

نظرات خوانندگان

نویسنده: امیر بختیاری
تاریخ: ۱۳۹۲/۱۱/۲۵ ۸:۳۱

با سلام و تشکر از شما
اگر tree دارای چک باکس باشد و بخواهیم نود هایی که چک خورده است را ذخیره کنیم چگونه عمل کنیم؟
قابلیت drag هم نداشت مهم نیست . یک tree ثابت از دیتا بیس پر می‌شود و بعد گزینه هایی که تیک دارد را در جدولی دیگر ذخیره می‌کنیم

عنوان: **یک تکنیک جالب در نحوه نام گذاری فیلدهای دیتابیس به منظور استفاده بهینه از فایل های T4 در MVC5**
نویسنده: صادق نجاتی
تاریخ: ۱۳۹۳/۰۲/۱۰ ۱۰:۴۰
آدرس: www.dotnettips.info
گروه‌ها: MVC, Kendo UI, T4

بدون شک دوستانی که با تکنولوژی محبوب ASP.NET MVC5 کار کرده اند این نکته را می‌دانند که اگر فایل‌های T4 که وظیفه Scaffolding را به عهده دارند به پروژه خود اضافه کنند می‌توانند نحوه تولید خودکار Controllerها و Viewهای متناظر را سفارشی کنند. مثلا می‌توان این فایل‌ها را طوری طراحی کرد که Controller و Viewهای تولیدی به طور اتوماتیک چند زبانه و یا Responsive تولید شوند (این موضوعات بحث اصلی مقاله نیستند) و اما بحث اصلی را با یک مثال آغاز می‌کنیم :
فرض کنید در دیتابیس خود یک Table دارید که قرار است اطلاعات یک Slider را در خود نگه دارد. این Table دارای یک فیلد از نوع nvarchar برای ذخیره آدرس تصویر ارسالی توسط کاربر است.
در حالت عادی اگر از روی مدل این Table اقدام به تولید خودکار Controller و View متناظر کنید، یک editor (تکست باکس) برای دریافت آدرس تصویر تولید خواهد شد که برنامه نویس یا طراح باید به طور دستی آن را (به طور مثال) با Kendo uploader جایگزین نماید. ما می‌خواهیم برای فیلدهایی که قرار است آدرس تصویر را در خود نگه دارد به طور اتوماتیک از Kendo uploader استفاده شود. راه حل چیست؟
بسیار ساده است. ابتدا باید در نظر داشت که هنگام طراحی Table در دیتا بیس فیلد مورد نظر را به این شکل نامگذاری کنید : ExampleIMGURL (نحوه نام گذاری دلخواه است) مقصود آن است که نام هر فیلدی که قرار است آدرس یک تصویر را در خود نگه دارد باید حاوی کلمه (IMGURL) باشد. مجددا ذکر می‌شود که نحوه نامگذاری اختیاری است. سپس فایل Create.t4 را باز کنید و کد :

```
@Html.EditorFor(model => model.<#= property.PropertyName #>)
```

را با کد زیر جایگزین کنید :

```
<#
if (GetAssociationName(property).Contains ("IMGURL")
{
#>
     @Html.Kendo().Upload().Name("<#= property.PropertyName #>")
<#
}
else
{
#>
     @Html.EditorFor(model => model.<#= property.PropertyName #>)
<#
}
#>
```

کد بالا چک می‌کند اگر نام فیلد مد نظر حاوی " IMGURL " باشد یک کندو آپلودر تولید کرده در غیر این صورت یک ادیتور ساده تولید می‌کند. البته این فقط یک مثال است و بدون شک دامنه استفاده از این تکنیک وسیع‌تر است.

اگر این مطلب مفید واقع شد با در نظر گرفتن نظرات ارسالی به تکنیک‌های آتی اشاره خواهد شد.

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۱۰ ۱۱:۳۰

قابلیت سفارشی سازی EditorFor در ASP.NET MVC پیش بینی شده‌است و با استفاده از UIHint قابل انتساب به خواص مدل مورد نظر است. البته این مورد برای حالت Code first یا حالتیکه از ViewModels استفاده کنید بیشتر کاربرد دارد.
یک مثال:
فایلی را به نام Upload.cshtml ، در مسیر Views/Shared/EditorTemplates با محتوای ذیل ایجاد کنید:

```
@model string
@Html.Kendo().Upload().Name("@ViewData.ModelMetadata.PropertyName")
```

سپس برای استفاده از آن فقط کافی است خاصیت مدنظر را با ویژگی UIHint مزین کنید:

```
[UIHint("Upload")]
public string ImageUrl {set;get;}
```

نویسنده: صادق نجاتی
تاریخ: ۱۳۹۳/۰۲/۲۹ ۱۱:۵

ضمن تشکر از آقای نصیری؛
بدون شک نقش UIHint در سفارشی سازی انکار ناپذیر است. ولی همانطور که گفته شد دامنه استفاده از این تکنیک وسیع‌تر است. مثلا حالتی را در نظر بگیرید که می‌خواهیم از طریق Scaffolding برای یک جدول بانک اطلاعاتی که یک فیلد آن آدرس یک تصویر را نگهداری می‌کند View ایجاد نماییم. خوب ما در صفحه Index می‌خواهیم تصویر مورد نظر با اندازه 100 * 100 پیکسل نمایش دهیم (چون قرار است لیستی از تصاویر نمایش داده شود باید در اندازه قابل نمایشی باشد) ولی در صفحه Details باید اندازه بزرگتری از تصویر را به نمایش بگذاریم. حال اگر از UIHint استفاده کنیم تنها یکی از موارد قبل (سفارشی سازی در لیست و جزئیات) محقق خواهد شد. اگر بخواهیم انجام این کارها را به صورت اتوماتیک به Scaffolding بسپاریم باید مطابق آنچه گفته شد ، فایل‌های T4 را (List.t4 و Details.t4) سفارشی سازی نماییم.

عنوان: استفاده ازExpressionها جهت ایجاد Strongly typed view در ASP.NET MVC
نویسنده: محسن موسوی
تاریخ: ۱۳۹۳/۰۴/۱۵ ۰:۰
آدرس: www.dotnettips.info
گروه‌ها: MVC, Reflection, Kendo UI, jqGrid

مدل زیر را در نظر بگیرید:

```
/// <summary>
    ///
    /// </summary>
    public class CompanyModel
    {
        /// <summary>
        /// Table Identity
        /// </summary>
        public int Id { get; set; }

        /// <summary>
        /// Company Name
        /// </summary>
        [DisplayName("نام شرکت")]
        public string CompanyName { get; set; }

        /// <summary>
        /// Company Abbreviation
        /// </summary>
        [DisplayName("نام اختصاری شرکت")]
        public string CompanyAbbr { get; set; }

    }
```

از View زیر جهت نمایش لیستی از شرکت‌ها متناظر با مدل جاری استفاده میشود:

```
@{
    const string viewTitle = "شرکت ها";
    ViewBag.Title = viewTitle;
    const string gridName = "companies-grid";
}
<div class="col-md-12">
    <div class="form-panel">
        <header>
            <div class="title">
                <i class="fa fa-book"></i>
                @viewTitle
            </div>
        </header>
        <div class="panel-body">
            <div id="@gridName">
            </div>
        </div>
    </div>
</div>
</div>
@section scripts
{
    <script type="text/javascript">
        $(document).ready(function () {
            $("#@gridName").kendoGrid({
                dataSource: {
                    type: "json",
                    transport: {
                        read: {
                            url: "@Html.Raw(Url.Action(MVC.Company.CompanyList()))",
                            type: "POST",
                            dataType: "json",
                            contentType: "application/json"
                        }
                    },
                    schema: {
                        data: "Data",
                        total: "Total",
                        errors: "Errors"
                        }
                    },
                    pageSize: 10,
```

```
                    serverPaging: true,
                    serverFiltering: true,
                    serverSorting: true
                },
                pageable: {
                    refresh: true
                },
                sortable: {
                    mode: "multiple",
                    allowUnsort: true
                },
                editable: false,
                filterable: false,
                scrollable: false,
                columns: [ {
                    field: "CompanyName",
                    title: "نام شرکت",
                    sortable: true,
                }, {
                    field: "CompanyAbbr",
                    title: "مخفف نام شرکت",
                    sortable: true
                }]
            });
        });
    </script>
}
```

مشکلی که در کد بالا وجود دارد این است که با تغییر نام هر یک از متغییر هایمان ، اطلاعات گرید در ستون مربوطه نمایش داده نمیشود.همچنین عناوین ستونها نیز از DisplayName مدل پیروی نمیکنند.توسط متدهای الحاقی زیر این مشکل برطرف شده است.

```
/// <summary>
    ///
    /// </summary>
    public static class PropertyExtensions
    {
        /// <summary>
        ///
        /// </summary>
        /// <typeparam name="T"></typeparam>
        /// <param name="expression"></param>
        /// <returns></returns>
        public static MemberInfo GetMember<T>(this Expression<Func<T, object>> expression)
        {
            var mbody = expression.Body as MemberExpression;

            if (mbody != null) return mbody.Member;
            //This will handle Nullable<T> properties.
            var ubody = expression.Body as UnaryExpression;
            if (ubody != null)
            {
                mbody = ubody.Operand as MemberExpression;
            }
            if (mbody == null)
            {
                throw new ArgumentException("Expression is not a MemberExpression", "expression");
            }
            return mbody.Member;
        }

        /// <summary>
        ///
        /// </summary>
        /// <typeparam name="T"></typeparam>
        /// <param name="expression"></param>
        /// <returns></returns>
        public static string PropertyName<T>(this Expression<Func<T, object>> expression)
        {
            return GetMember(expression).Name;
        }

        /// <summary>
        ///
        /// </summary>
```

```
        /// <typeparam name="T"></typeparam>
        /// <param name="expression"></param>
        /// <returns></returns>
        public static string PropertyDisplay<T>(this Expression<Func<T, object>> expression)
        {
            var propertyMember = GetMember(expression);
            var displayAttributes = propertyMember.GetCustomAttributes(typeof(DisplayNameAttribute),
true);
            return displayAttributes.Length == 1 ?
((DisplayNameAttribute)displayAttributes[0]).DisplayName : propertyMember.Name;
        }
    }
```

```
public static string PropertyName<T>(this Expression<Func<T, object>> expression)
```

جهت بدست آوردن نام متغییر هایمان استفاده مینماییم.

```
public static string PropertyDisplay<T>(this Expression<Func<T, object>> expression)
```

جهت بدست آوردن DisplayNameAttribute استفده میشود. درصورتیکه این DisplayNameAttribute یافت نشود نام متغییر بازگشت داده میشود.

بنابراین View مربوطه را اینگونه بازنویسی میکنیم:

```
@using Models
@{
    const string viewTitle = "شرکت ها";
    ViewBag.Title = viewTitle;
    const string gridName = "companies-grid";
}
<div class="col-md-12">
    <div class="form-panel">
        <header>
            <div class="title">
                <i class="fa fa-book"></i>
                @viewTitle
            </div>
        </header>
        <div class="panel-body">
            <div id="@gridName">
            </div>
        </div>
    </div>
</div>
</div>
@section scripts
{
    <script type="text/javascript">
        $(document).ready(function () {
            $("#@gridName").kendoGrid({
                dataSource: {
                    type: "json",
                    transport: {
                        read: {
                            url: "@Html.Raw(Url.Action(MVC.Company.CompanyList()))",
                            type: "POST",
                            dataType: "json",
                            contentType: "application/json"
                        }
                    },
                    schema: {
                        data: "Data",
                        total: "Total",
                        errors: "Errors"
                        }
```

```
                    },
                    pageSize: 10,
                    serverPaging: true,
                    serverFiltering: true,
                    serverSorting: true
                },
                pageable: {
                    refresh: true
                },
                sortable: {
                    mode: "multiple",
                    allowUnsort: true
                },
                editable: false,
                filterable: false,
                scrollable: false,
                columns: [ {
                    field: "@(PropertyExtensions.PropertyName<CompanyModel>(a => a.CompanyName))",
                    title: "@(PropertyExtensions.PropertyDisplay<CompanyModel>(a => a.CompanyName))",
                    sortable: true,
                }, {
                    field: "@(PropertyExtensions.PropertyName<CompanyModel>(a => a.CompanyAbbr))",
                    title: "@(PropertyExtensions.PropertyDisplay<CompanyModel>(a => a.CompanyAbbr))",
                    sortable: true
                }]
            });
        });
    </script>
}
```

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۱۶ ۱۲:۳۸

با تشکر از شما. حالت پیشرفته‌تر این مساله، کار با مدل‌های تو در تو هست. برای مثال:

```
public class CompanyModel
    {
        public int Id { get; set; }
        public string CompanyName { get; set; }
        public string CompanyAbbr { get; set; }

        public Product Product { set; get; }
    }

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

در اینجا اگر بخواهیم Product.Id را بررسی کنیم:

```
var data = PropertyExtensions.PropertyName<CompanyModel>(x => x.Product.Id);
```

فقط Id آن دریافت می‌شود.
راه حلی که از کدهای EF برای این مساله استخراج شده به صورت زیر است (نمونه‌اش متد Include تو در تو بر روی چند خاصیت):

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Linq.Expressions;

namespace PropertyExtensionsApp
{
    public class PropertyHelper : ExpressionVisitor
    {
        private Stack<string> _stack;
        public string GetNestedPropertyPath(Expression expression)
        {
            _stack = new Stack<string>();
            Visit(expression);
            return _stack.Aggregate((s1, s2) => s1 + "." + s2);
        }

        protected override Expression VisitMember(MemberExpression expression)
        {
            if (_stack != null)
                _stack.Push(expression.Member.Name);
            return base.VisitMember(expression);
        }

        public string GetNestedPropertyName<TEntity>(Expression<Func<TEntity, object>> expression)
        {
            return GetNestedPropertyPath(expression);
        }
    }
}
```

در این حالت خواهیم داشت:

```
var name = new PropertyHelper().GetNestedPropertyName<CompanyModel>(x => x.Product.Id);
```

که خروجی Product.Id را بر می‌گرداند.

نویسنده: محسن موسوی
تاریخ: ۱۳۹۳/۰۷/۲۶ ۱۸:۸

در نهایت این متد به این شکل اصلاح شود:

```
/// <summary>
        ///
        /// </summary>
        /// <typeparam name="T"></typeparam>
        /// <param name="expression"></param>
        /// <returns></returns>
        public static string PropertyName<T>(this Expression<Func<T, object>> expression)
        {
            return new PropertyHelper().GetNestedPropertyName(expression);
        }
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۱۰ ۲۳:۵۷

روش دیگری در اینجا: « Strongly-Typed ID References to Razor-Generated Fields »

نویسنده: محسن موسوی
تاریخ: ۱۳۹۳/۰۹/۱۱ ۱۴:۱

با تشکر
- نظر نویسنده مقاله تغییر کرده، بدلیل دوباره کاری انجام شده.(توضیحات بیشتر در کامنتهای مقاله ارجاعی)
- روش جاری وابسته به مدل ویو نیست و به همین دلیل محدودیت نداره، بطور مثال یک ویو شامل عملیات‌های اضافه و ویرایش و حذف و گرید لیست آنهاست.

عنوان: **معرفی Kendo UI**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۱۴ ۱:۳۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, Kendo UI

Kendo UI چیست؟

Kendo UI یک فریم ورک جاوا اسکریپتی ساخت برنامه‌های مدرن و تعاملی وب است و برای رسیدن به این مقصود، از HTML 5، CSS 3، JavaScript و jQuery کمک می‌گیرد.

امکانات فراهم شده توسط Kendo UI

1) انواع و اقسام ویجت‌ها: کنترل‌های وب تهیه شده برفراز jQuery
ویجت‌های آن در سه گروه کلی قرار می‌گیرند:
- گروه وب، مانند گرید، tree-view و غیره.
- گروه DataViz که جهت نمایش بصری اطلاعات و ترسیم انواع و اقسام نمودارها کاربرد دارد.
- گروه موبایل که با استفاده از فناوری adaptive rendering، در سیستم عامل‌های مختلف موبایل، مانند اندروید و آی او اس، ظاهری بومی و هماهنگ با آن‌ها را ارائه می‌دهد.

2) منبع داده سمت کاربر (Client side data source)
منبع داده سمت کاربر Kendo UI، از انواع و اقسام منابع داده محلی مانند آرایه‌های جاوا اسکریپتی تا منابع داده راه دور، مانند XML، JSON و JSONP، جهت نمایش اطلاعات و data binding پشتیبانی می‌کند. این منبع داده، مواردی مانند صفحه بندی، مرتب سازی اطلاعات و گروه بندی آن‌ها را نیز فراهم می‌کند. به علاوه با عملیات ثبت، ویرایش و حذف اطلاعات نیز هماهنگی کاملی را دارد.

3) به همراه یک فریم ورک MVVM توکار است
این فریم ورک MVVM مواردی مانند two way data binding و همچنین declarative binding را نیز پشتیبانی می‌کند.

4) امکان تعویض قالب
5) پویا نمایی، کشیدن و رها کردن
6) فریم ورک اعتبارسنجی

چرا Kendo UI؟

- مهم‌ترین مزیت کار با Kendo UI، فراهم آوردن تمام نیازهای توسعه‌ی یک برنامه‌ی مدرن وب، تنها در یک بسته است. به این ترتیب دیگر نیازی نیست تا گرید را از یک‌جا، tree-view را از جایی دیگر و کتابخانه‌های رسم نمودار را از منبعی ناهمگون با سایر عناصر برنامه دریافت و استفاده کنید؛ در اینجا تمام این‌ها در قالب یک بسته‌ی آماده برای شما فراهم شده‌است و همچنین با یکدیگر سازگاری کاملی دارند.
- تمام ویجت‌های آن برای نمایش سریع با کارآیی بالا طراحی شده‌اند.
- پشتیبانی خوب آن. این فریم ورک محصول شرکتی است که به صورت تخصصی کار تهیه کامپوننت‌های وب و دسکتاپ را انجام می‌دهد.

مرورگرهای پشتیبانی شده

یکی دیگر از مزایای مهم کار با Kendo UI پشتیبانی گسترده‌ی آن از اکثر مرورگرهای موجود است. این فریم ورک با مرورگرهای زیر سازگار است:
- IE 7 به بعد
- فایرفاکس 10 به بعد
- تمام نگارش‌های کروم
- اپرا 10 به بعد

- سفاری 4 به بعد

مجوز استفاده از Kendo UI

Kendo UI با سه مجوز ذیل ارائه می‌شود:

- 30 روزه آزمایشی رایگان
- تجاری
- سورس باز با مجوز Apache

پیشتر نسخه‌ی تجاری آن تحت مجوز GPL نیز در دسترس بود. اما اخیرا مجوز GPL آن حذف شده و به Apache تغییر یافته است. اما باید در نظر داشت که نسخه‌ی سورس باز آن شامل کنترل‌های مهمی مانند «گرید» نیست و این موارد تنها در نسخه‌ی تجاری آن لحاظ شده‌اند.

مثال‌های Kendo UI

پس از دریافت بسته‌ی کامل آن، پوشه‌هایی مانند styles، js و امثال آن قابل مشاهده هستند؛ به همراه پوشه‌ی examples آن که حداقل 86 پوشه‌ی دیگر در آن جهت ارائه مثال‌هایی از نحوه‌ی کاربرد المان‌های مختلف آن تدارک دیده شده‌اند.

نحوه‌ی افزودن Kendo UI به صفحه

از آنجائیکه Kendo UI یک فریم ورک جاوا اسکریپتی است، همانند سایر برنامه‌های وب، افزودن تعاریف فایل‌های css، js و تصاویر مرتبط با آن، برای شروع به کار کفایت می‌کند. برای این منظور ابتدا پوشه‌های js و styles بسته‌ی دریافتی آن‌را به برنامه‌ی خود اضافه کنید (این پوشه‌ها در فایل پیوست انتهای بحث موجود هستند).

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>

    <!--KendoUI: Web-->
    <link href="styles/kendo.common.min.css" rel="stylesheet" type="text/css" />
    <link href="styles/kendo.default.min.css" rel="stylesheet" type="text/css" />
    <script src="js/jquery.min.js" type="text/javascript"></script>
    <script src="js/kendo.web.min.js" type="text/javascript"></script>

    <!--KendoUI: DataViz-->
    <link href="styles/kendo.dataviz.min.css" rel="stylesheet" type="text/css" />
    <script src="js/kendo.dataviz.min.js" type="text/javascript"></script>

    <!--KendoUI: Mobile-->
    <link href="styles/kendo.mobile.all.min.css" rel="stylesheet" type="text/css" />
    <script src="js/kendo.mobile.min.js" type="text/javascript"></script>

    <script type="text/javascript">
        $(function() {
            $("#pickDate").kendoDatePicker();
        });
    </script>
</head>
<body>
    <span>
        Pick a date: <input id="pickDate" type="text"/>
    </span>
</body>
</html>
```

در اینجا یک مثال ساده‌ی استفاده از date picker کندو یو آی را ملاحظه می‌کنید. در قسمت head صفحه، نحوه‌ی ثبت سه گروه اسکریپت و شیوه نامه، مشخص شده‌اند. اگر نیاز به کامپونت‌های وب آن‌را دارید باید اجزایی مانند kendo.common.min.css، kendo.default.min.css، jquery.min.js و kendo.web.min.js به صفحه اضافه شوند. اگر نیاز به رسم نمودار هست، فایل‌ها kendo.dataviz.min.css و kendo.dataviz.min.js باید تعریف شوند و برای فعال سازی اجزای موبایل آن فایل‌های kendo.mobile.all.min.css و kendo.mobile.min.js نیاز است به صفحه پیوست شوند. در هر سه حالت ذکر jquery.min.js الزامی است.

دریافت سورس کامل این قسمت که حاوی فایل‌های اصلی kendoui.professional.2014.2.1008 نیز می‌باشد:

KendoUI01.7z

نظرات خوانندگان

نویسنده: محمد
تاریخ: ۱۳۹۳/۰۸/۱۴ ۱۴:۳۸

برای دریافت پکیج کامل Kendo از این آدرس استفاده کنید.

نویسنده: محمد رعیت پیشه
تاریخ: ۱۳۹۳/۰۸/۱۴ ۲۰:۵

آیا امکان استفاده از Razor Wrapper هم به صورت رایگان وجود دارد یا اینکه نیازمند تهیه بسته کامل می‌باشد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۱۴ ۲۰:۱۸

تجاری هست و توصیه هم نمی‌شود. چون نهایتا برای بسیاری از کارها باید به پشت صحنه‌ی این ویجت‌ها و امکانات مراجعه کنید؛ یعنی نیاز است مستقیما اسکریپت نویسی کنید و با ساختار واقعی آن‌ها آشنا باشید.

نویسنده: سعیدجلالی
تاریخ: ۱۳۹۳/۰۸/۱۷ ۸:۵۶

بله امکان استفاده از wrapper در نسخه asp.net.mvc.commercial وجود دارد
استفاده از اون هم خیلی ساده‌تر و خواناتر از جاوا اسکریپت هست. نظر آقای نصیری هم محترم است ولی در مواقع خواص میتونید همزمان هم از جاوااسکریپت استفاده کنید هم از wrapper یک نمونه رو در زیر با هم مقایسه میکنیم
با استفاده از جاوا اسکریپت

```
<input id="pickDate" type="text"/>

<script type="text/javascript">
  $(function() {
        $("#pickDate").kendoDatePicker();
  });
</script>
```

با استفاده از wrapper

```
@(Html.Kendo().DatePicker().Name("pickDate"))
```

در ضمن اینکه توی wrapper امکان استفاده از Intellisense و امکان تعریف ارتباط اغلب کامپوننت‌های وب به مدل با استفاده از forهای نمونه معادل کامپوننت فراهم شده است مانند wrapper زیر

```
@(Html.Kendo().DatePickerFor(m => m.HireDate).Name("pickDate1"))
```

نویسنده: Ara
تاریخ: ۱۳۹۳/۰۸/۱۷ ۱۹:۳۱

به دوستان پیشنهاد می‌شه این رو هم ببینید
استفاده از Kendo UI بهمراه AngularJs خیلی خوبه !
ما تو پروژه تجاری استفاده کردیم و خیلی راضی هستیم

عنوان: **بررسی ساختار ویجت‌های وب Kendo UI**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۱۴ ۱۴:۱۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, Kendo UI

ویجت‌های وب Kendo UI کدامند؟

ویجت‌های وب Kendo UI مجموعه‌ای از کنترل‌های سفارشی HTML 5 هستند که برفراز jQuery تهیه شده‌اند. این کنترل‌ها برای برنامه‌های وب و همچنین برنامه‌های دسکتاپ لمسی طراحی شده‌اند.
بهترین روش برای مشاهده‌ی این مجموعه، مراجعه به فایل examples\index.html پوشه‌ی اصلی Kendo UI است که لیست کاملی از این ویجت‌ها را به همراه مثال‌های مرتبط ارائه می‌دهد. تعدادی از اعضای این مجموعه شامل کنترل‌های ذیل هستند:

Window, TreeView, Tooltip, ToolBar, TimePicker, TabStrip, Splitter, Sortable, Slider, Gantt, Scheduler, ProgressBar, PanelBar, NumericTextBox, Notification, MultiSelect, Menu, MaskedTextBox, ListView, PivotGrid, Grid, Editor, DropDownList, DateTimePicker, DatePicker, ComboBox, ColorPicker, Calendar, Button, AutoComplete

نحوه‌ی استفاده کلی از ویجت‌های وب Kendo UI

با توجه به اینکه کنترل‌های Kendo UI مبتنی بر jQuery هستند، نحوه‌ی استفاده از آن‌ها، مشابه سایر افزونه‌های جی‌کوئری است.
ابتدا المانی به صفحه اضافه می‌شود:

```
<input id="pickDate" type="text"/>
```

سپس این المان را در رویداد document ready، به یکی از کنترل‌های Kendo UI مزین خواهیم کرد. برای مثال تزئین یک TextBox معمولی با یک Date Picker:

```
<script type="text/javascript">
  $(function() {
        $("#pickDate").kendoDatePicker();
  });
</script>
```

روش دیگری به نام declarative initialization نیز برای اعمال ویجت‌های وب Kendo UI قابل استفاده است که از ویژگی‌های data-* مرتبط با HTML 5 کمک می‌گیرد. برای نمونه، کدهای جاوا اسکریپتی فوق را می‌توان با ویژگی data-role ذیل جایگزین کرد:

```
<input id="dateOfBirth" type="text" data-role="datepicker" />
```

اگر در این حالت برنامه را اجرا کنید، تفاوتی را مشاهده نخواهید کرد.
برای فعال سازی حالت declarative initialization باید به دو نکته‌ی مهم دقت داشت:
الف) در مطلب معرفی Kendo UI اسکریپت‌های ذیل برای آماده سازی Kendo Ui معرفی شدند:

```
<!--KendoUI: Web-->
<link href="styles/kendo.common.min.css" rel="stylesheet" type="text/css" />
<link href="styles/kendo.default.min.css" rel="stylesheet" type="text/css" />
<script src="js/jquery.min.js" type="text/javascript"></script>
<script src="js/kendo.web.min.js" type="text/javascript"></script>

<!--KendoUI: DataViz-->
<link href="styles/kendo.dataviz.min.css" rel="stylesheet" type="text/css" />
<script src="js/kendo.dataviz.min.js" type="text/javascript"></script>

<!--KendoUI: Mobile-->
<link href="styles/kendo.mobile.all.min.css" rel="stylesheet" type="text/css" />
<script src="js/kendo.mobile.min.js" type="text/javascript"></script>
```

باید دقت داشت که در آن واحد نمی‌توان تمام این بسته‌ها را با هم بکار برد؛ چون برای مثال فایل‌های جداگانه ویجت‌های وب و موبایل با هم تداخل ایجاد می‌کنند . بجای اینکار بهتر است از فایل‌های kendo.all.min.js (که حاوی تمام اسکریپت‌های لازم است) و cssهای عنوان شده استفاده کرد:

```
<link href="styles/kendo.common.min.css" rel="stylesheet" type="text/css" />
<link href="styles/kendo.default.min.css" rel="stylesheet" type="text/css" />
<script src="js/jquery.min.js" type="text/javascript"></script>
<script src="js/kendo.all.min.js" type="text/javascript"></script>
```

ب) data-roleها توسط متد kendo.init فعال می‌شوند.

یک مثال کامل:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title></title>

    <link href="styles/kendo.common.min.css" rel="stylesheet" type="text/css" />
    <link href="styles/kendo.default.min.css" rel="stylesheet" type="text/css" />
    <script src="js/jquery.min.js" type="text/javascript"></script>
    <script src="js/kendo.all.min.js" type="text/javascript"></script>

    <script type="text/javascript">
        $(function () {
            $("#pickDate").kendoDatePicker();
        });

        $(function () {
            // initialize any widgets in the #container div
            kendo.init($("#container"));
        });
    </script>
</head>
<body>
    <span>
        Pick a date: <input id="pickDate" type="text" />
    </span>

    <div id="container">
        <input id="dateOfBirth" type="text" data-role="datepicker" />
        <div id="colors"
             data-role="colorpalette"
             data-columns="4"
             data-tile-size="{ width: 34, height: 19 }"></div>
    </div>
</body>
</html>
```

- در این مثال نحوه‌ی پیوست تمام فایل‌های لازم Kendo UI را به صورت یکجا ملاحظه می‌کنید که در ابتدای head صفحه ذکر شده‌اند.
- در اینجا pickDate به صورت معمولی فعال شده‌است.
- اما در قسمت kendo.init نام یک ناحیه یا نام یک کنترل را می‌توان ذکر کرد. برای مثال در اینجا کل ناحیه‌ی مشخص شده توسط یک div با id مساوی container به صورت یکجا با تمام کنترل‌های داخل آن فعال گردیده‌است.

بنابراین برای اعمال declarative initialization، یک ناحیه را توسط kendo.init مشخص کرده و سپس توسط data-roleها، نام ویجت وب مورد نظر را به صورت lower case مشخص می‌کنیم. همچنین فایل‌های اسکریپت مورد استفاده نیز نباید تداخلی داشته باشند.

تنظیمات ویجت‌های وب Kendo UI

تاکنون نمونه‌ی ساده‌ای از بکارگیری ویجت‌های وب Kendo UI را بررسی کردیم؛ اما این ویجت‌ها توسط تنظیمات پیش بینی شده برای آن‌ها بسیار قابل تنظیم و تغییر هستند. تنظیمات آن‌ها نیز بستگی به روش استفاده و آغاز آن‌ها دارد. برای مثال اگر این

ویجت‌ها را توسط کدهای جاوا اسکریپتی آغاز کرده‌اید، در همانجا توسط پارامترهای افزونه‌ی جی‌کوئری می‌توان تنظیمات مرتبط را اعمال کرد:

```
<script type="text/javascript">
  $(function () {
      $("#pickDate").kendoDatePicker({
             format: "yyyy/MM/dd"
        });
  });
</script>
```

که در اینجا توسط پارامتر format، نحوه‌ی دریافت تاریخ نهایی مشخص می‌شود.
در حالت declarative initialization، پارامتر format تبدیل به ویژگی data-format خواهد شد:

```
<input id="dateOfBirth" type="text"
          data-role="datepicker"
          data-format="yyyy/MM/dd" />
```

تنظیمات DataSource ویجت‌های وب

بسیاری از ویجت‌های وب Kendo UI با داده‌ها سر و کار دارند مانند Grid، Auto Complete، Combo box و غیره. این کنترل‌ها داده‌های خود را از طریق خاصیت DataSource دریافت می‌کنند. برای نمونه در اینجا یک combo box را در نظر بگیرید. در مثال اول، خاصیت dataSource کنترل ComboBox در همان افزونه‌ی جی‌کوئری تنظیم شده‌است:

```
<input id="colorPicker1" />
    <script type="text/javascript">
        $(document).ready(function () {
            $("#colorPicker1").kendoComboBox({
                dataSource: ["Blue", "Green", "Red", "Yellow"]
            });
        });
    </script>
```

و در مثال دوم، نحوه‌ی مقدار دهی ویژگی data-source را در حالت declarative initialization مشاهده می‌کنید. همانطور که عنوان شد، در این حالت ذکر متد kendo.init بر روی یک ناحیه و یا یک کنترل ویژه، جهت آغاز فعالیت آن ضروری است:

```
<input id="colorPicker2" data-role="combobox" data-source='["Blue", "Green", "Red", "Yellow"]' />
    <script type="text/javascript">
        $(document).ready(function () {
            kendo.init($("#colorPicker2"));
        });
    </script>
```

کار با رویدادهای ویجت‌های وب

نحوه‌ی کار با رویدادهای ویجت‌های وب نیز بر اساس نحوه‌ی آغاز آن‌ها متفاوت است. در مثال‌های ذیل، دو حالت متفاوت تنظیم رویداد change را توسط خواص افزونه‌ی جی‌کوئری:

```
<input id="colorPicker3" />
    <script type="text/javascript">
        function onColorChange(e) {
             alert('Color Change!');
        }

        $(document).ready(function () {
            $("#colorPicker3").kendoComboBox({
                dataSource: ["Blue", "Green", "Red", "Yellow"],
                change: onColorChange
```

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

و همچنین توسط ویژگی data-change مشاهده می‌کنید:

```
<input id="colorPicker4" data-role="combobox"
           data-source='["Blue", "Green", "Red", "Yellow"]'
           data-change="onColorChange" />

    <script type="text/javascript">
        function onColorChange(e) {
            alert('Color Change!');
        }

        $(document).ready(function () {
            kendo.init($("#colorPicker4"));
        });
    </script>
```

در هر دو حالت، انتخاب یک گزینه‌ی جدید combo box، سبب فراخوانی متد callback ایی به نام onColorChange می‌شود.

تغییر قالب ویجت‌های وب

Kendo UI همیشه یک جفت CSS را جهت تعیین قالب‌های ویجت‌های خود، مورد استفاده قرار می‌دهد. برای نمونه در مثال‌های فوق، kendo.common.min.css حاوی اطلاعات محل قرارگیری و اندازه‌ی ویجت‌ها است. شیوه نامه‌ی دوم همیشه به شکل kendo.[skin].min.css تعریف می‌شود که دارای اطلاعات رنگ و پس زمینه‌ی ویجت‌ها خواهد بود؛ مانند kendo.black.min.css، kendo.blueopal.min.css و امثال آن که در پوشه‌ی styles قابل مشاهده هستند.
همچنین باید دقت داشت که همیشه common باید پیش از skin ذکر شود؛ زیرا در تعدادی از حالات، شیوه نامه‌ی skin، اطلاعات common را بازنویسی می‌کند.
علاوه بر skinهای پیش فرض موجود در پوشه‌ی styles، امکان استفاده از یک theme builder آنلاین نیز وجود دارد: kendo-ui-themebuilder

نظرات خوانندگان

نویسنده: انصاری
تاریخ: ۱۳۹۳/۰۹/۰۱ ۸:۵۹

امکان تبدیل تاریخ (از میلادی به شمسی) در ویجت‌های وب مثل Scheduler , Calendar , dataPicker وجود دارد؟ من میخواستم از Scheduler استفاده کنم ولی با تاریخ و فرمت (ماه و روزهای هفته) فارسی، به نظر شما امکانش هست بدون دردسر (و به شکل استاندارد) این تقویم رو شمسی کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۰۱ ۱۰:۵

در اینجا بحث شده.

عنوان: **کار با Kendo UI DataSource**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۱۵ ۲۲:۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, ASP.NET Web API, jQuery, Kendo UI, json.net

Kendo UI DataSource جهت تامین داده‌های سمت کلاینت ویجت‌های مختلف KendoUI طراحی شده‌است و به عنوان یک اینترفیس استاندارد قابل استفاده توسط تمام کنترل‌های داده‌ای Kendo UI کاربرد دارد. Kendo UI DataSource امکان کار با منابع داده محلی، مانند اشیاء و آرایه‌های جاوا اسکریپتی و همچنین منابع تامین شده از راه دور، مانند JSONP، JSON و XML را دارد. به علاوه توسط آن می‌توان اعمال ثبت، ویرایش و حذف اطلاعات، به همراه صفحه بندی، گروه بندی و مرتب سازی داده‌ها را کنترل کرد.

استفاده از منابع داده محلی

در ادامه مثالی را از نحوه‌ی استفاده از یک منبع داده محلی جاوا اسکریپتی، مشاهده می‌کنید:

```
<script type="text/javascript">
        $(function () {
            var cars = [
                { "Year": 2000, "Make": "Hyundai", "Model": "Elantra" },
                { "Year": 2001, "Make": "Hyundai", "Model": "Sonata" },
                { "Year": 2002, "Make": "Toyota", "Model": "Corolla" },
                { "Year": 2003, "Make": "Toyota", "Model": "Yaris" },
                { "Year": 2004, "Make": "Honda", "Model": "CRV" },
                { "Year": 2005, "Make": "Honda", "Model": "Accord" },
                { "Year": 2000, "Make": "Honda", "Model": "Accord" },
                { "Year": 2002, "Make": "Kia", "Model": "Sedona" },
                { "Year": 2004, "Make": "Fiat", "Model": "One" },
                { "Year": 2005, "Make": "BMW", "Model": "M3" },
                { "Year": 2008, "Make": "BMW", "Model": "X5" }
            ];

            var carsDataSource = new kendo.data.DataSource({
                data: cars
            });

            carsDataSource.read();
            alert(carsDataSource.total());
        });
    </script>
```

در اینجا cars آرایه‌ای از اشیاء جاوا اسکریپتی بیانگر ساختار یک خودرو است. سپس برای معرفی آن به Kendo UI، کار با مقدار دهی خاصیت data مربوط به new kendo.data.DataSource شروع می‌شود.
ذکر new kendo.data.DataSource به تنهایی به معنای مقدار دهی اولیه است و در این حالت منبع داده مورد نظر، استفاده نخواهد شد. برای مثال اگر متد total آن‌را جهت یافتن تعداد عناصر موجود در آن فراخوانی کنید، صفر را بازگشت می‌دهد. برای شروع به کار با آن، نیاز است ابتدا متد read را بر روی این منبع داده مقدار دهی شده، فراخوانی کرد.

استفاده از منابع داده راه دور

در برنامه‌های کاربردی، عموما نیاز است تا منبع داده را از یک وب سرور تامین کرد. در اینجا نحوه‌ی خواندن اطلاعات JSON بازگشت داده شده از جستجوی توئیتر را مشاهده می‌کنید:

```
<script type="text/javascript">
        $(function () {
            var twitterDataSource = new kendo.data.DataSource({
                transport: {
                    read: {
                        url: "http://search.twitter.com/search.json",
                        dataType: "jsonp",
                        contentType: 'application/json; charset=utf-8',
                        type: 'GET',
                        data: { q: "#kendoui" }
                    },
                    schema: { data: "results" }
```

```
                },
                error: function (e) {
                    alert(e.errorThrown.stack);
                }
            });
        });
    </script>
```

در قسمت transport، جزئیات تبادل اطلاعات با سرور راه دور مشخص می‌شود؛ برای مثال url ارائه دهنده‌ی سرویس، dataType بیانگر نوع داده مورد انتظار و data کار مقدار دهی پارامتر مورد انتظار توسط سرویس توئیتر را انجام می‌دهد. در اینجا چون صرفا عملیات خواندن اطلاعات صورت می‌گیرد، خاصیت read مقدار دهی شده‌است.
در قسمت schema مشخص می‌کنیم که اطلاعات JSON بازگشت داده شده توسط توئیتر، در فیلد results آن قرار دارد.

کار با منابع داده OData

علاوه بر فرمت‌های یاد شده، Kendo UI DataSource امکان کار با اطلاعاتی از نوع OData را نیز دارا است که تنظیمات ابتدایی آن به صورت ذیل است:

```
<script type="text/javascript">
            var moviesDataSource = new kendo.data.DataSource({
                type: "odata",
                transport: {
                    read: "http://demos.kendoui.com/service/Northwind.svc/Orders"
                },
                error: function (e) {
                    alert(e.errorThrown.stack);
                }
            });
        });
    </script>
```

همانطور که ملاحظه می‌کنید، تنظیمات ابتدایی آن اندکی با حالت remote data پیشین متفاوت است. در اینجا ابتدا نوع داده‌ی بازگشتی مشخص می‌شود و در قسمت transport، خاصیت read آن، آدرس سرویس را دریافت می‌کند.

یک مثال: دریافت اطلاعات از ASP.NET Web API

یک پروژه‌ی جدید ASP.NET را آغاز کنید. تفاوتی نمی‌کند که Web forms باشد یا MVC؛ از این جهت که مباحث Web API در هر دو یکسان است.
سپس یک کنترلر جدید Web API را به نام ProductsController با محتوای زیر ایجاد کنید:

```
using System.Collections.Generic;
using System.Web.Http;

namespace KendoUI02
{
    public class Product
    {
        public int Id { set; get; }
        public string Name { set; get; }
    }

    public class ProductsController : ApiController
    {
        public IEnumerable<Product> Get()
        {
            var products = new List<Product>();
            for (var i = 1; i <= 100; i++)
            {
                products.Add(new Product { Id = i, Name = "Product " + i });
            }
            return products;
        }
    }
}
```

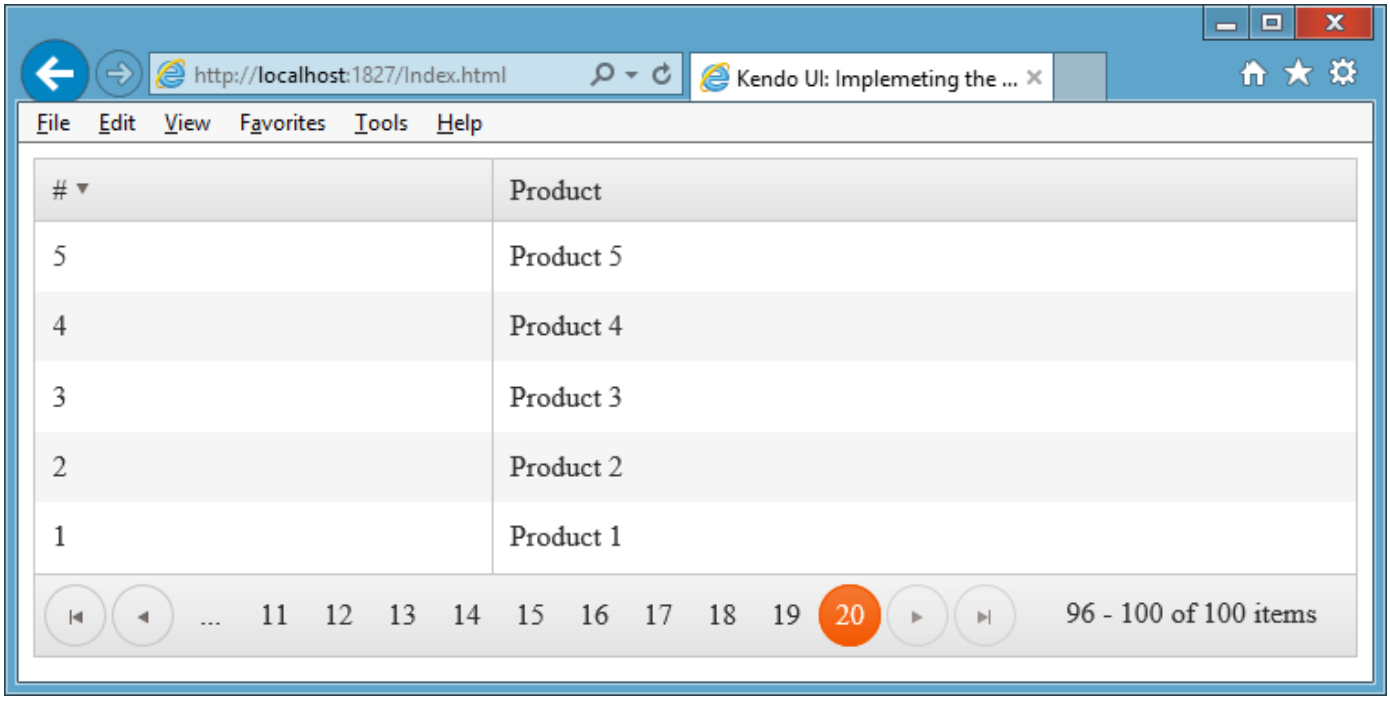
در این مثال، هدف صرفا ارائه یک خروجی ساده JSON از طرف سرور است.
در ادامه نیاز است تعریف مسیریابی ذیل نیز به فایل Global.asax.cs برنامه اضافه شود تا بتوان به آدرس api/products در سایت، دسترسی یافت:

```
using System;
using System.Web.Http;
using System.Web.Routing;

namespace KendoUI02
{
    public class Global : System.Web.HttpApplication
    {
        protected void Application_Start(object sender, EventArgs e)
        {
            RouteTable.Routes.MapHttpRoute(
               name: "DefaultApi",
               routeTemplate: "api/{controller}/{id}",
               defaults: new { id = RouteParameter.Optional }
               );
        }
    }
}
```

در ادامه فایلی را به نام Index.html (یا در یک View و یا یک فایل aspx دلخواه)، محتوای ذیل را اضافه کنید:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="utf-8" />
    <title>Kendo UI: Implemeting the Grid</title>

    <link href="styles/kendo.common.min.css" rel="stylesheet" type="text/css" />
    <link href="styles/kendo.default.min.css" rel="stylesheet" type="text/css" />
    <script src="js/jquery.min.js" type="text/javascript"></script>
    <script src="js/kendo.all.min.js" type="text/javascript"></script>
</head>
<body>

    <div id="report-grid"></div>
    <script type="text/javascript">
        $(function () {
            var productsDataSource = new kendo.data.DataSource({
                transport: {
                    read: {
                        url: "api/products",
                        dataType: "json",
                        contentType: 'application/json; charset=utf-8',
                        type: 'GET'
                    }
                },
                error: function (e) {
                    alert(e.errorThrown.stack);
                },
                pageSize: 5,
                sort: { field: "Id", dir: "desc" }
            });

            $("#report-grid").kendoGrid({
                dataSource: productsDataSource,
                autoBind: true,
                scrollable: false,
                pageable: true,
                sortable: true,
                columns: [
                    { field: "Id", title: "#" },
                    { field: "Name", title: "Product" }
                ]
            });
        });
    </script>
</body>
</html>
```

- ابتدا فایل‌های اسکریپت و CSS مورد نیاز Kendo UI اضافه شده‌اند.
- گرید صفحه، در محل div ایی با id مساوی report-grid تشکیل خواهد شد.
- سپس DataSource ایی که به آدرس api/products اشاره می‌کند، تعریف شده و در آخر productsDataSource را توسط یک kendoGrid نمایش داده‌ایم.
- نحوه‌ی تعریف productsDataSource، در قسمت استفاده از منابع داده راه دور ابتدای بحث توضیح داده شد. در اینجا فقط دو خاصیت pageSize و sort نیز به آن اضافه شده‌اند. این دو خاصیت بر روی نحوه‌ی نمایش گرید نهایی تاثیر گذار هستند. pageSize تعداد رکورد هر صفحه را مشخص می‌کند و sort نحوه‌ی مرتب سازی را بر اساس فیلد Id و در حالت نزولی قرار می‌دهد.
- در ادامه، ابتدایی‌ترین حالت کار با kendoGrid را ملاحظه می‌کنید.
- تنظیم dataSource و autoBind: true (حالت پیش فرض)، سبب خواهند شد تا به صورت خودکار، اطلاعات JSON از مسیر api/products خوانده شوند.
- سه خاصیت بعدی صفحه بندی و مرتب سازی خودکار ستون‌ها را فعال می‌کنند.
- در آخر هم دو ستون گرید، بر اساس نام‌های خواص کلاس Product تعریف شده‌اند.

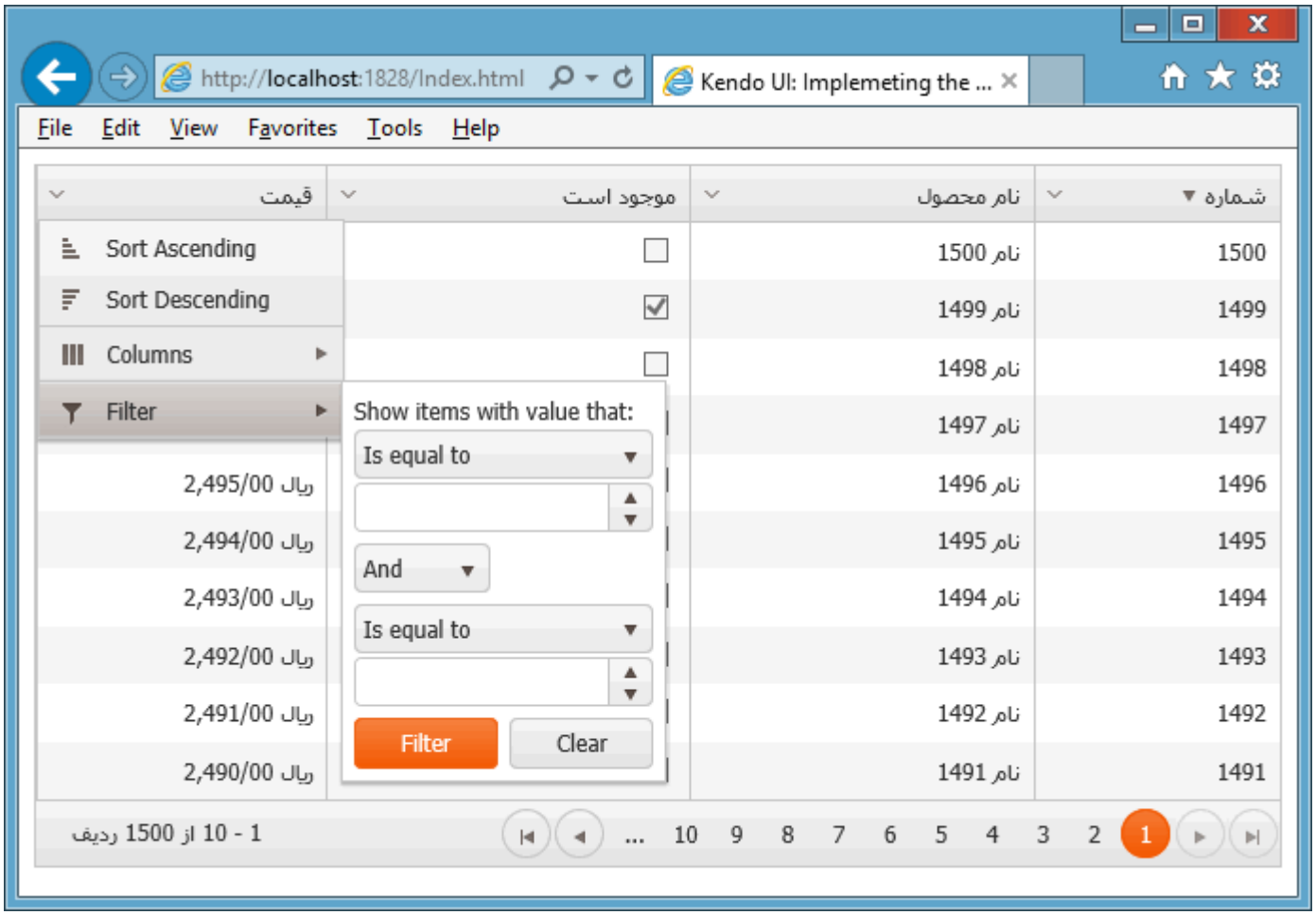


سورس کامل این قسمت را از اینجا می‌توانید دریافت کنید:

KendoUI02.zip

عنوان: **صفحه بندی، مرتب سازی و جستجوی پویای اطلاعات به کمک Kendo UI Grid**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۱۶ ۱۶:۳۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, ASP.NET Web API, jQuery, DynamicLINQ, Kendo UI, json.net

پس از آشنایی مقدماتی با Kendo UI DataSource ، اکنون می‌خواهیم از آن جهت صفحه بندی، مرتب سازی و جستجوی پویای سمت سرور استفاده کنیم. در مثال قبلی، هر چند صفحه بندی فعال بود، اما پس از دریافت تمام اطلاعات، این اعمال در سمت کاربر انجام و مدیریت می‌شد.



مدل برنامه

در اینجا قصد داریم لیستی را با ساختار کلاس Product در اختیار Kendo UI گرید قرار دهیم:

```
namespace KendoUI03.Models
{
    public class Product
    {
        public int Id { set; get; }
        public string Name { set; get; }
        public decimal Price { set; get; }
        public bool IsAvailable { set; get; }
    }
}
```

پیشنیاز تامین داده مخصوص Kendo UI Grid

برای ارائه اطلاعات مخصوص Kendo UI Grid، ابتدا باید درنظر داشت که این گرید، درخواست‌های صفحه بندی خود را با فرمت ذیل ارسال می‌کند. همانطور که مشاهده می‌کنید، صرفا یک کوئری استرینگ با فرمت JSON را دریافت خواهیم کرد:

```
/api/products?{"take":10,"skip":0,"page":1,"pageSize":10,"sort":[{"field":"Id","dir":"desc"}]}
```

سپس این گرید نیاز به سه فیلد، در خروجی JSON نهایی خواهد داشت:

```
{
"Data":
[
{"Id":1500,"Name":"1500 نام","Price":2499.0,"IsAvailable":false},
{"Id":1499,"Name":"1499 نام","Price":2498.0,"IsAvailable":true}
],
"Total":1500,
"Aggregates":null
}
```

فیلد Data که رکوردهای گرید را تامین می‌کنند. فیلد Total که بیانگر تعداد کل رکوردها است و Aggregates که برای گروه بندی بکار می‌رود.

می‌توان برای تمام این‌ها، کلاس و Parser تهیه کرد و یا ... پروژه‌ی سورس بازی به نام Kendo.DynamicLinq نیز چنین کاری را میسر می‌سازد که در ادامه از آن استفاده خواهیم کرد. برای نصب آن تنها کافی است دستور ذیل را صادر کنید:

```
PM> Install-Package Kendo.DynamicLinq
```

Kendo.DynamicLinq به صورت خودکار System.Linq.Dynamic را نیز نصب می‌کند که از آن جهت صفحه بندی پویا استفاده خواهد شد.

تامین کننده‌ی داده سمت سرور

همانند مطلب کار با Kendo UI DataSource ، یک ASP.NET Web API Controller جدید را به پروژه اضافه کنید و همچنین مسیریابی‌های مخصوص آن‌را به فایل global.asax.cs نیز اضافه نمائید.

```
using System.Linq;
using System.Net.Http;
using System.Web.Http;
using Kendo.DynamicLinq;
using KendoUI03.Models;
using Newtonsoft.Json;

namespace KendoUI03.Controllers
{
    public class ProductsController : ApiController
    {
        public DataSourceResult Get(HttpRequestMessage requestMessage)
        {
            var request = JsonConvert.DeserializeObject<DataSourceRequest>(
                requestMessage.RequestUri.ParseQueryString().GetKey(0)
            );

            var list = ProductDataSource.LatestProducts;
            return list.AsQueryable()
                       .ToDataSourceResult(request.Take, request.Skip, request.Sort, request.Filter);
        }
    }
}
```

تمام کدهای این کنترلر همین چند سطر فوق هستند. با توجه به ساختار کوئری استرینگی که در ابتدای بحث عنوان شد، نیاز است آن‌را توسط کتابخانه‌ی JSON.NET تبدیل به یک نمونه از DataSourceRequest نمائیم. این کلاس در Kendo.DynamicLinq تعریف شده‌است و حاوی اطلاعاتی مانند take و skip کوئری LINQ نهایی است.
ProductDataSource.LatestProducts صرفا یک لیست جنریک تهیه شده از کلاس Product است. در نهایت با استفاده از متد الحاقی جدید ToDataSourceResult ، به صورت خودکار مباحث صفحه بندی سمت سرور به همراه مرتب سازی اطلاعات، صورت گرفته و اطلاعات نهایی با فرمت DataSourceResult بازگشت داده می‌شود. DataSourceResult نیز در Kendo.DynamicLinq تعریف شده و سه فیلد یاد شده‌ی Data، Total و Aggregates را تولید می‌کند.

تا اینجا کارهای سمت سرور این مثال به پایان می‌رسد.

تهیه View نمایش اطلاعات ارسالی از سمت سرور

اعمال مباحث بومی سازی

```
<head>
    <meta charset="utf-8" />
    <meta http-equiv="Content-Language" content="fa" />
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

    <title>Kendo UI: Implemeting the Grid</title>

    <link href="styles/kendo.common.min.css" rel="stylesheet" type="text/css" />
    <!--شیوه نامه‌ی مخصوص راست به چپ سازی-->
    <link href="styles/kendo.rtl.min.css" rel="stylesheet" />
    <link href="styles/kendo.default.min.css" rel="stylesheet" type="text/css" />
    <script src="js/jquery.min.js" type="text/javascript"></script>
    <script src="js/kendo.all.min.js" type="text/javascript"></script>

    <!--محل سفارشی سازی پیام‌ها و مسایل بومی-->
    <script src="js/cultures/kendo.culture.fa-IR.js" type="text/javascript"></script>
    <script src="js/cultures/kendo.culture.fa.js" type="text/javascript"></script>
    <script src="js/messages/kendo.messages.en-US.js" type="text/javascript"></script>

    <style type="text/css">
        body {
            font-family: tahoma;
            font-size: 9pt;
        }
    </style>

    <script type="text/javascript">
        // جهت استفاده از فایل: kendo.culture.fa-IR.js
        kendo.culture("fa-IR");
    </script>
</head>
```

- در اینجا چند فایل js و css جدید اضافه شده‌اند. فایل kendo.rtl.min.css جهت تامین مباحث RTL توکار Kendo UI کاربرد دارد.
- سپس سه فایل kendo.culture.fa.js، kendo.culture.fa-IR.js و kendo.messages.en-US.js نیز اضافه شده‌اند. فایل‌های fa و fa-Ir آن هر چند به ظاهر برای ایران طراحی شده‌اند، اما نام ماه‌های موجود در آن عربی است که نیاز به ویرایش دارد. به همین جهت به سورس این فایل‌ها، جهت ویرایش نهایی نیاز خواهد بود که در پوشه‌ی src\js\cultures مجموعه‌ی اصلی Kendo UI موجود هستند (ر.ک. فایل پیوست).
- فایل kendo.messages.en-US.js حاوی تمام پیام‌های مرتبط با Kendo UI است. برای مثال«رکوردهای 10 تا 15 از 1000 ردیف» را در اینجا می‌توانید به فارسی ترجمه کنید.
- متد kendo.culture کار مشخص سازی فرهنگ بومی برنامه را به عهده دارد. برای مثال در اینجا به fa-IR تنظیم شده‌است. این مورد سبب خواهد شد تا از فایل kendo.culture.fa-IR.js استفاده گردد. اگر مقدار آن‌را به fa تنظیم کنید، از فایل kendo.culture.fa.js کمک گرفته خواهد شد.

راست به چپ سازی گرید

تنها کاری که برای راست به چپ سازی Kendo UI Grid باید صورت گیرد، محصور سازی div آن در یک div با کلاس مساوی k-

rtl است:

```
<div class="k-rtl">
        <div id="report-grid"></div>
    </div>
```

k-rtl و تنظیمات آن در فایل kendo.rtl.min.css قرار دارند که در ابتدای head صفحه تعریف شده‌است.

تامین داده و نمایش گرید

در ادامه کدهای کامل DataSource و Kendo UI Grid را ملاحظه می‌کنید:

```
<script type="text/javascript">
        $(function () {
            var productsDataSource = new kendo.data.DataSource({
                transport: {
                    read: {
                        url: "api/products",
                        dataType: "json",
                        contentType: 'application/json; charset=utf-8',
                        type: 'GET'
                    },
                    parameterMap: function (options) {
                        return kendo.stringify(options);
                    }
                },
                schema: {
                    data: "Data",
                    total: "Total",
                    model: {
                        fields: {
                            "Id": { type: "number" }, //تعیین نوع فیلد برای جستجوی پویا مهم است
                            "Name": { type: "string" },
                            "IsAvailable": { type: "boolean" },
                            "Price": { type: "number" }
                        }
                    }
                },
                error: function (e) {
                    alert(e.errorThrown);
                },
                pageSize: 10,
                sort: { field: "Id", dir: "desc" },
                serverPaging: true,
                serverFiltering: true,
                serverSorting: true
            });

            $("#report-grid").kendoGrid({
                dataSource: productsDataSource,
                autoBind: true,
                scrollable: false,
                pageable: true,
                sortable: true,
                filterable: true,
                reorderable: true,
                columnMenu: true,
                columns: [
                    { field: "Id", title: "شماره", width: "130px" },
                    { field: "Name", title: "نام محصول" },
                    {
                        field: "IsAvailable", title: "موجود است",
                        template: '<input type="checkbox" #= IsAvailable ? checked="checked" : "" #
disabled="disabled" ></input>'
                    },
                    { field: "Price", title: "قیمت", format: "{0:c}" }
                ]
            });
        });
    </script>
```

- با تعاریف مقدماتی Kendo UI DataSource پیشتر آشنا شده‌ایم و قسمت read آن جهت دریافت اطلاعات از سمت سرور کاربرد

دارد.
- در اینجا ذکر contentType الزامی است. زیرا ASP.NET Web API بر این اساس است که تصمیم می‌گیرد، خروجی را به صورت JSON ارائه دهد یا XML.
- با استفاده از parameterMap، سبب خواهیم شد تا پارامترهای ارسالی به سرور، با فرمت صحیحی تبدیل به JSON شده و بدون مشکل به سرور ارسال گردند.
- در قسمت schema باید نام فیلدهای موجود در DataSourceResult دقیقا مشخص شوند تا گرید بداند که data را باید از چه فیلدی استخراج کند و تعداد کل ردیف‌ها در کدام فیلد قرار گرفته‌است.
- نحوه‌ی تعریف model را نیز در اینجا ملاحظه می‌کنید. ذکر نوع فیلدها در اینجا بسیار مهم است و اگر قید نشوند، در حین جستجوی پویا به مشکل برخواهیم خورد. زیرا پیش فرض نوع تمام فیلدها string است و در این حالت نمی‌توان عدد 1 رشته‌ای را با یک فیلد از نوع int در سمت سرور مقایسه کرد.
- در اینجا serverFiltering، serverPaging و serverSorting نیز به true تنظیم شده‌اند. اگر این مقدار دهی‌ها صورت نگیرد، این اعمال در سمت کلاینت انجام خواهند شد.

پس از تعریف DataSource، تنها کافی است آن‌را به خاصیت dataSource یک kendoGrid نسبت دهیم.
- autoBind: true سبب می‌شود تا اطلاعات DataSource بدون نیاز به فراخوانی متد read آن به صورت خودکار دریافت شوند.
- با تنظیم scrollable: false، اعلام می‌کنیم که قرار است تمام رکوردها در معرض دید قرارگیرند و اسکرول پیدا نکنند.
- pageable: true صفحه بندی را فعال می‌کند. این مورد نیاز به تنظیم pageSize: 10 در قسمت DataSource نیز دارد.
- با sortable: true مرتب سازی ستون‌ها با کلیک بر روی سرستون‌ها فعال می‌گردد.
- filterable: true به معنای فعال شدن جستجوی خودکار بر روی فیلدها است. کتابخانه‌ی Kendo.DynamicLinq حاصل آن‌را در سمت سرور مدیریت می‌کند.
- reorderable: true سبب می‌شود تا کاربر بتواند محل قرارگیری ستون‌ها را تغییر دهد.
- ذکر columnMenu: true اختیاری است. اگر ذکر شود، امکان مخفی سازی انتخابی ستون‌ها نیز مسیر خواهد شد.
- در آخر ستون‌های گرید مشخص شده‌اند. با تعیین format: "{0:c}" سبب نمایش فیلدهای قیمت با سه رقم جدا کننده خواهیم شد. مقدار ریال آن از فایل فرهنگ جاری تنظیم شده دریافت می‌گردد. با استفاده از template تعریف شده نیز سبب نمایش فیلد bool به صورت یک checkbox خواهیم شد.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید:

KendoUI03.zip

نظرات خوانندگان

نویسنده: حمیدرضا کبیری
تاریخ: ۱۳۹۳/۰۸/۱۶ ۱۹:۳

آیا kendo UI کاملا از زبان فارسی پشتیبانی میکند ؟
برای calender آن ، به تقویم شمی گزینه ای موجود هست ؟
این گزینه با ورژن ۲۰۱۴/۱/۳۱۸ مطابقت دارد ، آیا با ورژنهای جدید مشکلی نخواهد داشت ؟

نویسنده: احمد رجبی
تاریخ: ۱۳۹۳/۰۸/۱۶ ۲۰:۱۵

میتوانید با اضافه کردن این اسکریپت تمامی قسمتهای kendo را به زبان فارسی ترجمه کنید.

نویسنده: سعیدجلالی
تاریخ: ۱۳۹۳/۰۸/۱۷ ۸:۴۴

با تشکر از مطلب مفید شما من از wrapper mvc مجموعه kendo استفاده میکنم
توی مطالب شما در مورد استفاده از Kendo.DynamicLinq صحبت شد خواستم بدونم آیا وقتی از wrapper هم استفاده میکنیم استفاده از این پکیج لازم هست؟

```
@(Html.Kendo().Grid(Model)
      .Name("gridKendo")
      .Columns(columns =>
      {
          columns.Bound(c => c.Name).Width(50);
          columns.Bound(c => c.Family).Width(100);
          columns.Bound(c => c.Tel);
      })
      .Pageable(pager => pager
          .Input(true)
          .Numeric(true)
          .Info(true)
          .PreviousNext(true)
          .Refresh(true)
          .PageSizes(true)
      )
      )
```

چون من با استفاده از telerik profiler وقتی درخواست رو بررسی مکنم توی دستور sql چنین دستوری رو در انتها مشاهده میکنم:
صفحه اول:

```
SELECT *
FROM (
    SELECT
    FROM table a
)
WHERE ROWNUM <= :TAKE
```

صفحات بعد:

```
SELECT *
FROM (
    SELECT
        a.*,
        ROWNUM OA_ROWNUM
    FROM (
        FROM table a
```

```
    ) a
    WHERE ROWNUM <= :TAKE
)
WHERE OA_ROWNUM > :SKIP
```

پایگاه داده اوراکل است.

نویسنده: سعیدجلالی
تاریخ: ۱۳۹۳/۰۸/۱۷ ۹:۱۲

امکان فارسی شدن تمام بخش‌ها وجود دارد.
تقویم هم فارسی شده است در این سایت برای نسخه‌های جدیدتر هم باید دوتا فایل جاوا اسکریپت all و mvc رو خودتون تغییر بدهید (با توجه به الگوی انجام شده در فایل فارسی شده فوق)
ولی برای تقویم زمانبدی scheduler من فارسی ندیده ام

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۱۷ ۹:۳۱

مطلب فوق نه وابستگی خاصی به وب فرم‌ها دارد و نه ASP.NET MVC. ویو آن یک فایل HTML ساده‌است و سمت سرور آن فقط یک کنترلر ASP.NET Web API که با تمام مشتقات ASP.NET سازگار است. در این حالت یک نفر می‌تواند ASP.NET نگارش خودش را خلق کند؛ بدون اینکه نگران جزئیات وب فرم‌ها باشد یا ASP.NET MVC. ضمنا دانش جاوا اسکریپتی آن هم قابل انتقال است؛ چون اساسا Kendo UI برای فناوری سمت سرور خاصی طراحی نشده‌است و حالت اصل آن با PHP، Java و امثال آن هم کار می‌کند.

نویسنده: میثم آقااحمدی
تاریخ: ۱۳۹۳/۰۸/۱۷ ۱۳:۱۷

در کنترلر این خط باعث بارگذاری تمامی داده‌ها می‌شود

```
var list = ProductDataSource.LatestProducts;
```

آیا راه حلی وجود دارد که دیتای به تعداد همان pagesize از پایگاه خوانده شود؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۱۷ ۱۳:۲۸

- این فقط یک مثال هست و منبع داده‌ای صرفا جهت دموی ساده‌ی برنامه. فقط برای اینکه با یک کلیک بتوانید برنامه را اجرا کنید و نیازی به برپایی و تنظیم بانک اطلاعاتی و امثال آن نداشته باشد.
- شما در کدها و کوئری‌های مثلا EF در اصل با یک سری IQueryable کار می‌کنید. همینجا باید متد الحاقی ToDataSourceResult را اعمال کنید تا نتیجه‌ی نهایی در حداقل بار تعداد رفت و برگشت و با کوئری مناسبی بر اساس پارامترهای دریافتی به صورت خودکار تولید شود. در انتهای کار بجای مثلا ToList بنویسید ToDataSourceResult.

نویسنده: امین
تاریخ: ۱۳۹۳/۰۸/۱۷ ۱۴:۴۱

سلام من در ویو خودم نمیتونم اطلاعاتم رو تو kendo.grid ببینم و برای من یک لیست استرینگ در ویو نمایش داده میشه و به این شکل در کنترلر و ویو کد نویسی کردم .

```
public class EFController : Controller    {
        //
        // GET: /EF/

        public ActionResult AjaxConnected([DataSourceRequest] DataSourceRequest request )
```

```
        {
            using (var dbef=new dbTestEntities())
            {
                IQueryable<Person> persons = dbef.People;
                DataSourceResult result = persons.ToDataSourceResult(request);
                return Json(result.Data,JsonRequestBehavior.AllowGet);
            }
        }


    }
```

و ویو

```
@{    ViewBag.Title = "AjaxConnected";
}


<h2>AjaxConnected</h2>
@(Html.Kendo().Grid<TelerikMvcApp2.Models.Person>(  )
      .Name("Grid")
      .DataSource(builder => builder
          .Ajax()
          .Read(operationBuilder => operationBuilder.Action("AjaxConnected", "EF"))


      )
      .Columns(factory =>
      {
          factory.Bound(person => person.personId);
          factory.Bound(person => person.Name);
          factory.Bound(person => person.LastName);
      })
      .Pageable()
      .Sortable())
```

و یک لیست استرینگ بهم در عمل خروجی میده و از خود قالب kendogrid خبری نیست . من اطلاعات رو به طور json پاس میدم و ajaxi میگیرم.
حالا قبلش همچین خطلایی داشتم که به allowget ایراد میگرفت ولی در کل با JsonRequestBehavior.AllowGet حل شد و حالا فقط یه لسیت بهم خروجی میده! و از ظاهر گرید خبری نیست. و اگر به جای json نوشته بشه view و با ویو return کنم ظاهر kendogrid رو دارم اما خروجی دارای مقداری نیست! اینم خروجی استرینگ من :(

```
[{"personId":1,"Name":"Amin","LastName":"Saadati"},{"personId":2,"Name":"Fariba","LastName":"Ghochani"},{"personId":4,"Name":"Milad","LastName":"Rahmani"},{"personId":5,"Name":"rima","LastName":"rad"},{"personId":6,"Name":"ali","LastName":"kiva"},{"personId":7,"Name":"sahel","LastName":"abasi"},{"personId":8,"Name":"medi","LastName":"ghaem"},{"personId":9,"Name":"mino","LastName":"kafash"},{"personId":10,"Name":"behzad","LastName":"tizro"},{"personId":11,"Name":"toti","LastName":"saadati"},{"personId":12,"Name":"parinaz","LastName":"karami"},{"personId":13,"Name":"sadegh","LastName":"hojati"},{"personId":14,"Name":"milad","LastName":"ebadipor"},{"personId":15,"Name":"farid","LastName":"riazi"},{"personId":16,"Name":"said","LastName":"abdoli"},{"personId":17,"Name":"behzad","LastName":"ariafar"},{"personId":18,"Name":"jamshid","LastName":"kotahi"}]
```

این سوال رو در چند سایت پرسیدم و به جوابی برایش نرسیدم. و نمیدونم ایراد کدهای نوشته شده ام کجاست!
متشکرم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۱۷ ۱۵:۲

- قصد پشتیبانی از wrapperهای آن‌را ندارم. لطفا خارج از موضوع سؤال نپرسید. اگر کسی دوست داشت در این زمینه مطلب منتشر کند، خوب. ولی من چنین قصدی ندارم.
- عرض کردم اگر از wrapperها استفاده کنید، به علت عدم درک زیر ساخت اصلی Kendo UI، قادر به دیباگ کار نخواهید بود.
- اگر متن را مطالعه کنید در قسمت «پیشنیاز تامین داده مخصوص Kendo UI Grid » دقیقا شکل نهایی خروجی JSON مورد نیاز ارائه شده‌است. این خروجی در سه فیلد data، total و aggregate قرار می‌گیرد. شما الان فقط قسمت data آن‌را بازگشت

داده‌اید؛ بجای اصل و کل آن. نام این سه فیلد هم مهم نیست؛ اما هر چیزی که تعیین می‌شوند، باید در قسمت data source در خاصیت schema آن مانند مثالی که در مطلب جاری آمده (در قسمت «تامین داده و نمایش گرید »)، دقیقا مشخص شوند، تا Kendo UI بداند که اطلاعات مختلف را باید از چه فیلدهایی از JSON خروجی دریافت کند.

عنوان: **استفاده از Kendo UI templates**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۱۸ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, Kendo UI

در مطلب « صفحه بندی، مرتب سازی و جستجوی پویای اطلاعات به کمک Kendo UI Grid » در انتهای بحث، ستون IsAvailable به صورت زیر تعریف شد:

```
columns: [
               {
                   field: "IsAvailable", title: "موجود است",
                   template: '<input type="checkbox" #= IsAvailable ? checked="checked" : "" #
disabled="disabled" ></input>'
                }
]
```

Templates، جزو یکی از پایه‌های Kendo UI Framework هستند و توسط آن‌ها می‌توان قطعات با استفاده‌ی مجدد HTML ایی را طراحی کرد که قابلیت یکی شدن با اطلاعات جاوا اسکریپتی را دارند.
همانطور که در این مثال نیز مشاهده می‌کنید، قالب‌های Kendo UI از syntax (#) Hash استفاده می‌کنند. در اینجا قسمت‌هایی از قالب که با علامت # محصور می‌شوند، در حین اجرا، با اطلاعات فراهم شده جایگزین خواهند شد.
برای رندر مقادیر ساده می‌توان از # =# استفاده کرد. از # :# برای رندر اطلاعات HTML-encoded کمک گرفته می‌شود و # # برای رندر کدهای جاوا اسکریپتی کاربرد دارد. از حالت HTML-encoded برای نمایش امن اطلاعات دریافتی از کاربران و جلوگیری از حملات XSS استفاده می‌شود.
اگر در این بین نیاز است # به صورت معمولی رندر شود، در حالت کدهای جاوا اسکریپتی به صورت #\\ و در HTML ساده به صورت #\ باید مشخص گردد.

مثالی از نحوه‌ی تعریف یک قالب Kendo UI

```
<!--دریافت اطلاعات از منبع محلی-->
    <script id="javascriptTemplate" type="text/x-kendo-template">
        <ul>
            # for (var i = 0; i < data.length; i++) { #
            <li>#= data[i] #</li>
            # } #
        </ul>
    </script>

    <div id="container1"></div>
    <script type="text/javascript">
        $(function () {
            var data = ['User 1', 'User 2', 'User 3'];
            var template = kendo.template($("#javascriptTemplate").html());
            var result = template(data); //Execute the template
            $("#container1").html(result); //Append the result
        });
    </script>
```

این قالب ابتدا در تگ script محصور می‌شود و سپس نوع آن مساوی text/x-kendo-template قرار می‌گیرد. در ادامه توسط یک حلقه‌ی جاوا اسکریپتی، عناصر آرایه‌ی فرضی data خوانده شده و با کمک Hash syntax در محل‌های مشخص شده قرار می‌گیرند.
در ادامه باید این قالب را رندر کرد. برای این منظور یک div با id مساوی container1 را جهت تعیین محل رندر نهایی اطلاعات مشخص می‌کنیم. سپس متد kendo.template بر اساس id قالب اسکریپتی تعریف شده، یک شیء قالب را تهیه کرده و سپس با ارسال آرایه‌ای به آن، سبب اجرای آن می‌شود. خروجی نهایی، یک قطعه کد HTML است که در محل container1 درج خواهد شد.
همانطور که ملاحظه می‌کنید، متد kendo.template، نهایتا یک رشته را دریافت می‌کند. بنابراین همینجا و به صورت inline نیز می‌توان یک قالب را تعریف کرد.

کار با منابع داده راه دور

فرض کنید مدل برنامه به صورت ذیل تعریف شده‌است:

```
namespace KendoUI04.Models
{
    public class Product
    {
        public int Id { set; get; }
        public string Name { set; get; }
        public decimal Price { set; get; }
        public bool IsAvailable { set; get; }
    }
}
```

و لیستی از آن توسط یک ASP.NET Web API کنترلر، به سمت کاربر ارسال می‌شود:

```
using System.Collections.Generic;
using System.Linq;
using System.Web.Http;
using KendoUI04.Models;

namespace KendoUI04.Controllers
{
    public class ProductsController : ApiController
    {
        public IEnumerable<Product> Get()
        {
            return ProductDataSource.LatestProducts.Take(10);
        }
    }
}
```

در سمت کاربر و در View برنامه خواهیم داشت:

```
<!--دریافت اطلاعات از سرور-->
    <div>
        <div id="container2"><ul></ul></div>
    </div>

    <script id="template1" type="text/x-kendo-template">
        <li> #=Id# - #:Name# - #=kendo.toString(Price, "c")#</li>
    </script>

    <script type="text/javascript">
        $(function () {
            var producatsTemplate1 = kendo.template($("#template1").html());

            var productsDataSource = new kendo.data.DataSource({
                transport: {
                    read: {
                        url: "api/products",
                        dataType: "json",
                        contentType: 'application/json; charset=utf-8',
                        type: 'GET'
                    }
                },
                error: function (e) {
                    alert(e.errorThrown);
                },
                change: function () {
                    $("#container2 > ul").html(kendo.render(producatsTemplate1, this.view()));
                }
            });
            productsDataSource.read();
        });
    </script>
```

ابتدا یک div با id مساوی container2 جهت تعیین محل نهایی رندر قالب template1 در صفحه تعریف می‌شود.
هرچند خروجی دریافتی از سرور نهایتا یک آرایه از اشیاء Product است، اما در template1 اثری از حلقه‌ی جاوا اسکریپتی مشاهده نمی‌شود. در اینجا چون از متد kendo.render استفاده می‌شود، نیازی به ذکر حلقه نیست و به صورت خودکار، به تعداد

عناصر آرایه دریافتی از سرور، قطعه HTML قالب را تکرار می‌کند.
در ادامه برای کار با سرور از یک Kendo UI DataSource استفاده شده‌است. قسمت transport/read آن، کار تعریف محل دریافت اطلاعات را از سرور مشخص می‌کند. رویدادگران change آن اطلاعات نهایی دریافتی را توسط متد view در اختیار متد kendo.render قرار می‌دهد. در نهایت، قطعه‌ی HTML رندر شده‌ی نهایی حاصل از اجرای قالب، در بین تگ‌های ul مربوط به container2 درج خواهد شد.
رویدادگران change زمانیکه data source، از اطلاعات راه دور و یا یک آرایه‌ی جاوا اسکریپتی پر می‌شود، فراخوانی خواهد شد. همچنین مباحث مرتب سازی اطلاعات، صفحه بندی و تغییر صفحه، افزودن، ویرایش و یا حذف اطلاعات نیز سبب فراخوانی آن می‌گردند. متد view ایی که در این مثال فراخوانی شد، صرفا در روال رویدادگردان change دارای اعتبار است و آخرین تغییرات اطلاعات و آیتم‌های موجود در data source را باز می‌گرداند.

یک نکته‌ی تکمیلی: فعال سازی intellisense کدهای جاوا اسکریپتی Kendo UI

اگر به پوشه‌ی اصلی مجموعه‌ی Kendo UI مراجعه کنید، یکی از آن‌ها vsdoc نام دارد که داخل آن فایل‌های min.intellisense.js و vsdoc.js مشهود هستند.
اگر از ویژوال استودیوهای قبل از 2012 استفاده می‌کنید، نیاز است فایل‌های vsdoc.js متناظری را به پروژه اضافه نمائید؛ دقیقا در کنار فایل‌های اصلی js موجود. اگر از ویژوال استودیوی 2012 و یا بالاتر استفاده می‌کنید باید از فایل‌های intellisense.js متناظر استفاده کنید. برای مثال اگر از kendo.all.min.js کمک می‌گیرید، فایل متناظر با آن kendo.all.min.intellisense.js خواهد بود.
بعد از اینکار نیاز است فایلی به نام _references.js را به پوشه‌ی اسکریپت‌های خود با این محتوا اضافه کنید (برای VS 2012 به بعد):

```
/// <reference path="jquery.min.js" />
/// <reference path="kendo.all.min.js" />
```

نکته‌ی مهم اینجا است که این فایل به صورت پیش فرض از مسیر /Scripts/_references.js ~ خوانده می‌شود. برای اضافه کردن مسیر دیگری مانند /js/_references.js ~ باید آن‌را به تنظیمات ذیل اضافه کنید:

```
Tools menu -> Options -> Text Editor -> JavaScript -> Intellisense -> References
```

گزینه‌ی Reference Group را به Implicit (Web) تغییر داده و سپس مسیر جدیدی را اضافه نمائید.

کدهای کامل این قسمت را از اینجا می‌توانید دریافت کنید:
KendoUI04.zip

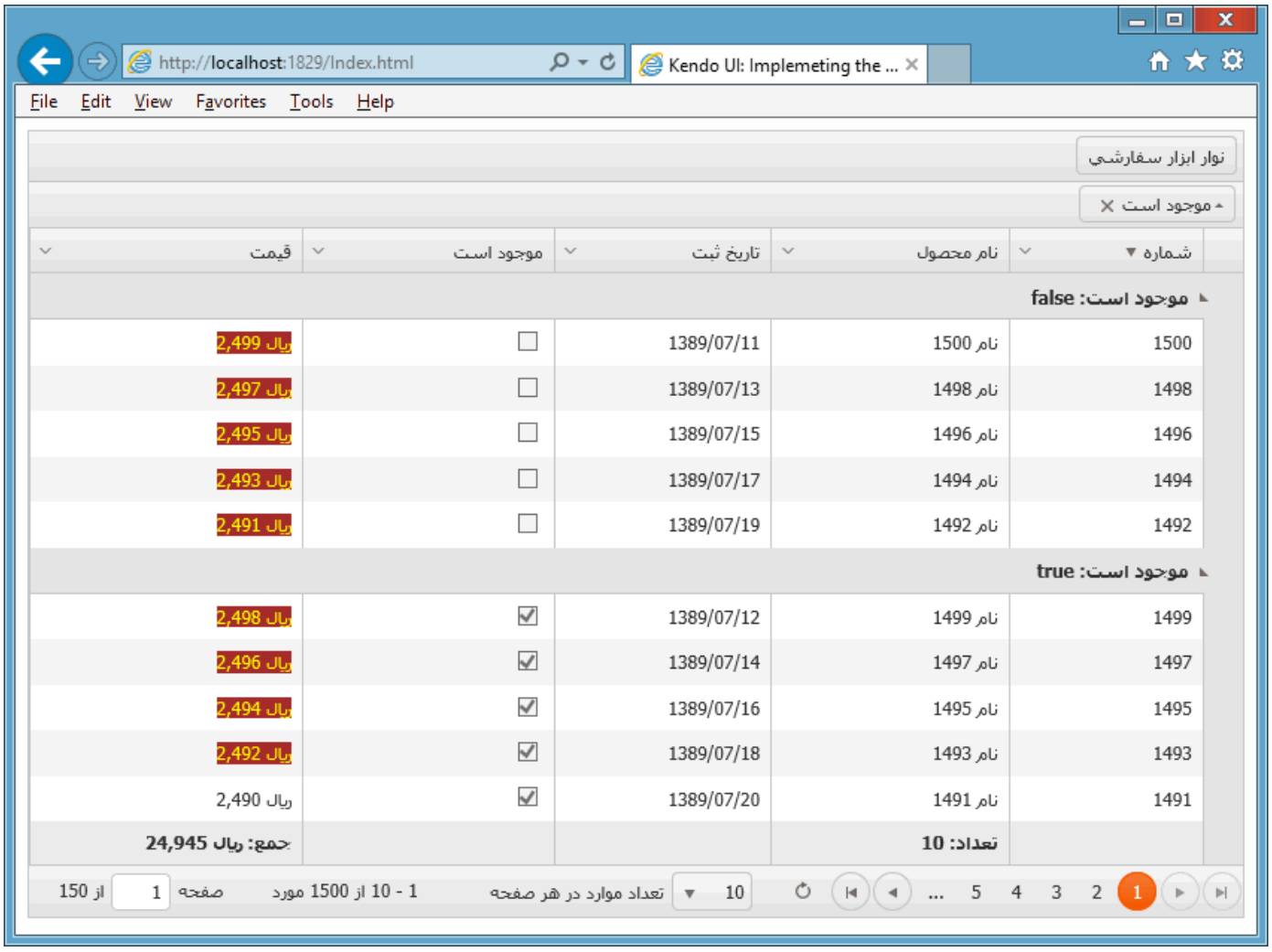
عنوان: **فرمت کردن اطلاعات نمایش داده شده به کمک Kendo UI Grid**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۱۹ ۱۳:۲۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, Kendo UI

پیشنیازهای بحث:

- « صفحه بندی، مرتب سازی و جستجوی پویای اطلاعات به کمک Kendo UI Grid »
- « استفاده از Kendo UI templates »

صورت مساله

می‌خواهیم به یک چنین تصویری برسیم؛ که دارای گروه بندی اطلاعات است، فرمت شرطی روی ستون قیمت آن اعمال شده و تاریخ نمایش داده شده در آن نیز شمسی است. همچنین برای مثال ستون قیمت آن دارای ته جمع صفحه بوده و به علاوه یک دکمه‌ی سفارشی به نوار ابزار آن اضافه شده‌است.



مباحث قسمت سمت سرور این مثال با مطلب « صفحه بندی، مرتب سازی و جستجوی پویای اطلاعات به کمک Kendo UI Grid » دقیقا یکی است. فقط یک خاصیت AddDate نیز در اینجا اضافه شده‌است.

تغییر نحوه‌ی نمایش pager

اگر به قسمت pager تصویر فوق دقت کنید، یک دکمه‌ی refresh، تعداد موارد هر صفحه و امکان وارد کردن دستی شماره صفحه، در آن پیش بینی شده‌است. این موارد را با تنظیمات ذیل می‌توان فعال کرد:

```
$("#report-grid").kendoGrid({
                // ...
                pageable: {
                    previousNext: true, // default true
                    numeric: true, // default true
                    buttonCount: 5, // default 10
                    refresh: true, // default false
                    input: true, // default false
                    pageSizes: true // default false
                },
```

بومی سازی پیغام‌های گرید

پیغام‌های فارسی را که در تصویر فوق مشاهده می‌کنید، حاصل پیوست فایل kendo.fa-IR.js هستند:

```
<!--https://github.com/loudenvier/kendo-global/blob/master/lang/kendo.fa-IR.js-->
<script src="js/messages/kendo.fa-IR.js" type="text/javascript"></script>
```

گروه بندی اطلاعات

برای گروه بندی اطلاعات در Kendo UI Grid دو قسمت باید تغییر کنند.
ابتدا باید فیلد پیش فرض گروه بندی در قسمت data source گرید تعریف شود:

```
var productsDataSource = new kendo.data.DataSource({
                // ...
                group: { field: "IsAvailable" },
                // ...
            });
```

همین تنظیم، گروه بندی را فعال خواهد کرد. اگر علاقمند باشید که به کاربران امکان تغییر دستی گروه بندی را بدهید، خاصیت groupable را نیز true کنید.

```
$("#report-grid").kendoGrid({
// ...
groupable: true, // allows the user to alter what field the grid is grouped by
// ...
```

در این حالت با کشیدن و رها کردن یک سرستون، به نوار ابزار مرتبط با گروه بندی، گروه بندی گرید بر اساس این فیلد انتخابی به صورت خودکار انجام می‌شود.

اضافه کردن ته جمع‌های ستون‌ها

این ته جمع‌ها که aggregate نام دارند باید در دو قسمت فعال شوند:

```
var productsDataSource = new kendo.data.DataSource({
                //...
                aggregate: [
                    { field: "Name", aggregate: "count" },
                    { field: "Price", aggregate: "sum" }
                ]
                //...
            });
```

ابتدا در قسمت data source مشخص می‌کنیم که چه تابع تجمعی قرار است به ازای یک فیلد خاص استفاده شود.
سپس این متدها را می‌توان مطابق فرمت hash syntax قالب‌های Kendo UI در قسمت footerTemplate هر ستون تعریف کرد:

```
$("#report-grid").kendoGrid({
                // ...
                columns: [
                    {
                        field: "Name", title: "نام محصول",
                        footerTemplate: "تعداد: #=count#"
                    },
                    {
                        field: "Price", title: "قیمت",
                        footerTemplate: "جمع: #=kendo.toString(sum,'c0')#"
                    }
                ]
                // ...
            });
```

فرمت شرطی اطلاعات

در ستون قیمت، می‌خواهیم اگر قیمتی بیش از 2490 بود، با پس زمینه‌ی قهوه‌ای و رنگ زرد نمایش داده شود. برای این منظور می‌توان یک قالب Kendo UI سفارشی را طراحی کرد:

```
<script type="text/x-kendo-template" id="priceTemplate">
        #if( Price > 2490 ) {#
        <span style="background:brown; color:yellow;">#=kendo.toString(Price,'c0')#</span>
        #} else {#
        #= kendo.toString(Price,'c0')#
        #}#
    </script>
```

سپس نحوه‌ی استفاده‌ی از آن به صورت ذیل خواهد بود:

```
$("#report-grid").kendoGrid({
                //...
                columns: [
                    {
                        field: "Price", title: "قیمت",
                        template: kendo.template($("#priceTemplate").html()),
                        footerTemplate: "جمع: #=kendo.toString(sum,'c0')#"
                    }
                ]
                //...
            });
```

توسط متد kendo.template امکان انتساب یک قالب سفارشی به خاصیت template یک ستون وجود دارد.

فرمت تاریخ میلادی به شمسی در حین نمایش

برای تبدیل سمت کلاینت تاریخ میلادی به شمسی از کتابخانه‌ی moment-jalaali.js کمک گرفته شده‌است:

```
<!--https://github.com/moment/moment/-->
<script src="js/cultures/moment.min.js" type="text/javascript"></script>
<!--https://github.com/jalaali/moment-jalaali-->
<script src="js/cultures/moment-jalaali.js" type="text/javascript"></script>
```

پس از آن تنها کافی است متد فرمت این کتابخانه را در قسمت template ستون تاریخ و توسط hash syntax قالب‌های Kendo UI بکار برد:

```
$("#report-grid").kendoGrid({
                //...
                columns: [
                    {
                        field: "AddDate", title: "تاریخ ثبت",
                        template: "#=moment(AddDate).format('jYYYY/jMM/jDD')#"
                    }
                ]
                //...
            });
```

اضافه کردن یک دکمه به نوار ابزار گرید

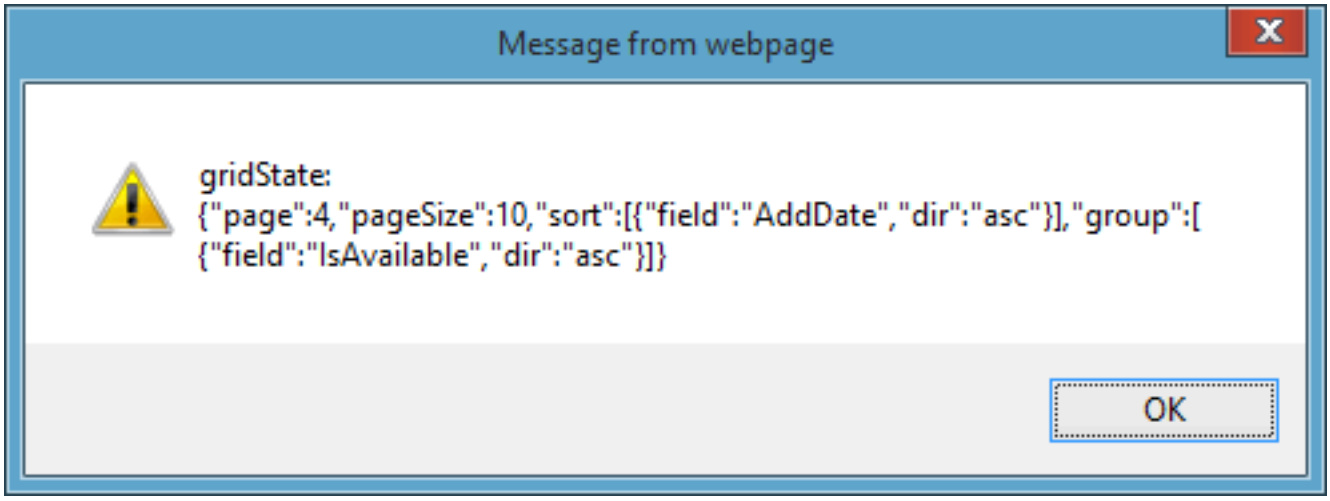
نوار ابزار Kendo UI Grid را نیز می‌توان توسط یک قالب سفارشی آن مقدار دهی کرد:

```
$("#report-grid").kendoGrid({
                // ...
                toolbar: [
                    { template: kendo.template($("#toolbarTemplate").html()) }
                ]
                // ...
            });
```

برای نمونه toolbarTemplate فوق را به نحو ذیل تعریف کرده‌ایم:

```
<script>
        // این اطلاعات برای تهیه خروجی سمت سرور مناسب هستند
        function getCurrentGridFilters() {
            var dataSource = $("#report-grid").data("kendoGrid").dataSource;
            var gridState = {
                page: dataSource.page(),
                pageSize: dataSource.pageSize(),
                sort: dataSource.sort(),
                group: dataSource.group(),
                filter: dataSource.filter()
            };
            return kendo.stringify(gridState);
        }
    </script>

    <script id="toolbarTemplate" type="text/x-kendo-template">
        <a class="k-button" href="\#" onclick="alert('gridState: ' + getCurrentGridFilters());">نوار
ابزار سفارشی</a>
    </script>
```

دکمه‌ی اضافه شده، وضعیت فیلتر data source متصل به گرید را بازگشت می‌دهد. برای مثال مشخص می‌کند که در چه صفحه‌ای با چه تعداد رکورد قرار داریم و همچنین وضعیت مرتب سازی، فیلتر و غیره چیست. از این اطلاعات می‌توان در سمت سرور برای تهیه‌ی خروجی‌های PDF یا اکسل استفاده کرد. وضعیت فیلتر اطلاعات مشخص است. بر همین مبنا کوئری گرفته و سپس می‌توان نتیجه‌ی آن‌را تبدیل به منبع داده تهیه خروجی مورد نظر کرد.



کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید:

KendoUI05.zip

نظرات خوانندگان

نویسنده: سروش
تاریخ: ۱۳۹۳/۰۸/۲۰ ۸:۵۱

با سلام هنگام Bulid پروژه با خطای زیر مواجه می‌شم

Error 1 Unable to locate 'C:\Users\Administrator\Desktop\KendoUI05\KendoUI05\.nuget\NuGet.exe' KendoUI05

دلیلش چیه ؟ البته Update-Package -Reinstall را انجام داده ام
متشکرم از شما

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۲۰ ۹:۵

- روی solution کلیک راست کنید و گزینه‌ی Enable NuGet Package Restore را انتخاب کنید.
- یا فایل NuGet.targets پوشه‌ی nuget. را باز کرده و دریافت خودکار nuget.exe را فعال کنید:

```
تغییر از
<DownloadNuGetExe Condition=" '$(DownloadNuGetExe)' == '' ">false</DownloadNuGetExe>
به
<DownloadNuGetExe Condition=" '$(DownloadNuGetExe)' == '' ">true</DownloadNuGetExe>
```

- و یا فایل nuget.exe را از این آدرس دریافت کنید و در پوشه‌ی nuget. کپی کنید.

نویسنده: سروش
تاریخ: ۱۳۹۳/۰۸/۲۰ ۹:۱۵

مرسی. فقط برای تغییر قالب KendoUi میشه کاری کرد برای ترکیب رنگهاش ؟ مثل JqueryUi Custom

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۲۰ ۹:۳۲

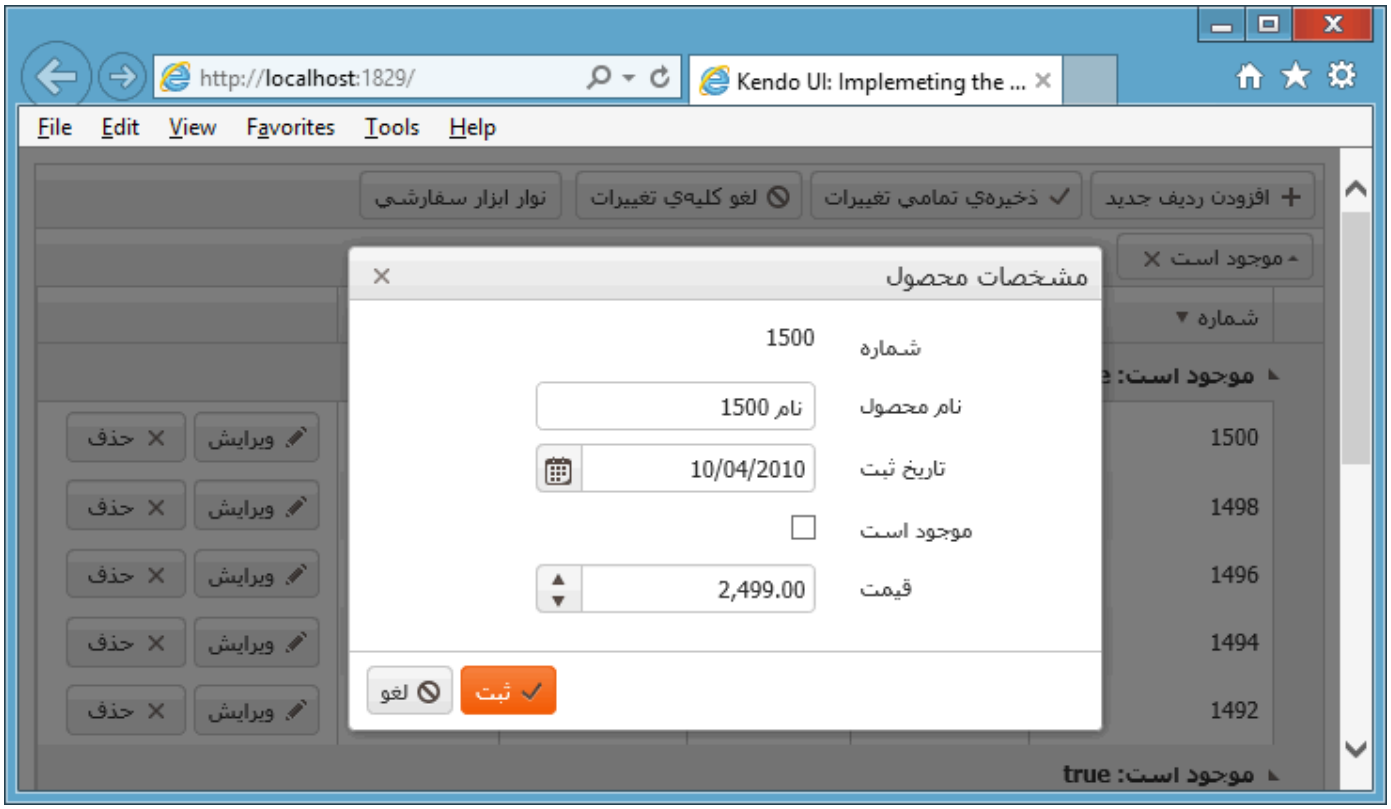
در انتهای مطلب «بررسی ساختار ویجت‌های وب Kendo UI » قسمت «تغییر قالب ویجت‌های وب»، توضیح داده شده.

عنوان: **فعال سازی عملیات CRUD در Kendo UI Grid**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۲۱ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, ASP.NET Web API, jQuery, Kendo UI

پیشنیاز بحث

- « فرمت کردن اطلاعات نمایش داده شده به کمک Kendo UI Grid »

Kendo UI Grid دارای امکانات ثبت، ویرایش و حذف توکاری است که در ادامه نحوه‌ی فعال سازی آن‌ها را بررسی خواهیم کرد. مثالی که در ادامه بررسی خواهد شد، در تکمیل مطلب « فرمت کردن اطلاعات نمایش داده شده به کمک Kendo UI Grid » است.



تنظیمات Data Source سمت کاربر

برای فعال سازی صفحه بندی سمت سرور ، با قسمت read منبع داده Kendo UI پیشتر آشنا شده بودیم. جهت فعال سازی قسمت‌های ثبت اطلاعات جدید (create)، به روز رسانی رکوردهای موجود (update) و حذف ردیفی مشخص (destroy) نیاز است تعاریف قسمت‌های متناظر را که هر کدام به آدرس مشخصی در سمت سرور اشاره می‌کنند، اضافه کنیم:

```
var productsDataSource = new kendo.data.DataSource({
                transport: {
                    read: {
                        url: "api/products",
                        dataType: "json",
                        contentType: 'application/json; charset=utf-8',
                        type: 'GET'
                    },
                    create: {
                        url: "api/products",
                        contentType: 'application/json; charset=utf-8',
                        type: "POST"
                    },
```

```
                    update: {
                        url: function (product) {
                            return "api/products/" + product.Id;
                        },
                        contentType: 'application/json; charset=utf-8',
                        type: "PUT"
                    },
                    destroy: {
                        url: function (product) {
                            return "api/products/" + product.Id;
                        },
                        contentType: 'application/json; charset=utf-8',
                        type: "DELETE"
                    },
                    //...
                },
                schema: {
                    //...
                    model: {
                        id: "Id", // define the model of the data source. Required for validation and
property types.
                        fields: {
                            "Id": { type: "number", editable: false }, //تعیین نوع فیلد برای جستجوی پویا
مهم است
                            "Name": { type: "string", validation: { required: true } },
                            "IsAvailable": { type: "boolean" },
                            "Price": { type: "number", validation: { required: true, min: 1 } },
                            "AddDate": { type: "date", validation: { required: true } }
                        }
                    }
                },
                batch: false, // enable batch editing - changes will be saved when the user clicks the
"Save changes" button
                //...
            });
```

- همانطور که ملاحظه می‌کنید، حالت‌های update و destroy بر اساس Id ردیف انتخابی کار می‌کنند. این Id را باید در قسمت model مربوط به اسکیمای تعریف شده، دقیقا مشخص کرد. عدم تعریف فیلد id، سبب خواهد شد تا عملیات update نیز در حالت create تفسیر شود.

- به علاوه در اینجا به ازای هر فیلد، مباحث اعتبارسنجی نیز اضافه شده‌اند؛ برای مثال فیلدهای اجباری با required: true مشخص گردیده‌اند.

- اگر فیلدی نباید ویرایش شود (مانند فیلد Id)، خاصیت editable آن‌را false کنید.

- در data source امکان تعریف خاصیتی به نام batch نیز وجود دارد. حالت پیش فرض آن false است. به این معنا که در حالت ویرایش، تغییرات هر ردیفی، یک درخواست مجزا را به سمت سرور سبب خواهد شد. اگر آن‌را true کنید، تغییرات تمام ردیف‌ها در طی یک درخواست به سمت سرور ارسال می‌شوند. در این حالت باید به خاطر داشت که پارامترهای سمت سرور، از حالت یک شیء مشخص باید به لیستی از آن‌ها تغییر یابند.

مدیریت سمت سرور ثبت، ویرایش و حذف اطلاعات

در حالت ثبت، متد Post، توسط آدرس مشخص شده در قسمت create منبع داده گرید، فراخوانی می‌گردد:

```
namespace KendoUI06.Controllers
{
    public class ProductsController : ApiController
    {
        public HttpResponseMessage Post(Product product)
        {
            if (!ModelState.IsValid)
                return Request.CreateResponse(HttpStatusCode.BadRequest);

            var id = 1;
            var lastItem = ProductDataSource.LatestProducts.LastOrDefault();
            if (lastItem != null)
            {
                id = lastItem.Id + 1;
            }
            product.Id = id;
            ProductDataSource.LatestProducts.Add(product);
```

```
            var response = Request.CreateResponse(HttpStatusCode.Created, product);
            response.Headers.Location = new Uri(Url.Link("DefaultApi", new { id = product.Id }));
            // گرید آی دی جدید را به این صورت دریافت می‌کند
            response.Content = new ObjectContent<DataSourceResult>(
                new DataSourceResult { Data = new[] { product } }, new JsonMediaTypeFormatter());
            return response;
        }
    }
}
```

نکته‌ی مهمی که در اینجا باید به آن دقت داشت، نحوه‌ی بازگشت Id رکورد جدید ثبت شده‌است. در این مثال، قسمت schema منبع داده سمت کاربر به نحو ذیل تعریف شده‌است:

```
var productsDataSource = new kendo.data.DataSource({
                //...
                schema: {
                    data: "Data",
                    total: "Total",
                }
                //...
            });
```

از این جهت که خروجی متد Get بازگرداننده‌ی اطلاعات صفحه بندی شده ، از نوع DataSourceResult است و این نوع، دارای خواصی مانند Data، Total و Aggergate است:

```
namespace KendoUI06.Controllers
{
    public class ProductsController : ApiController
    {
        public DataSourceResult Get(HttpRequestMessage requestMessage)
        {
            var request = JsonConvert.DeserializeObject<DataSourceRequest>(
                requestMessage.RequestUri.ParseQueryString().GetKey(0)
            );

            var list = ProductDataSource.LatestProducts;
            return list.AsQueryable()
                       .ToDataSourceResult(request.Take, request.Skip, request.Sort, request.Filter);
        }
    }
}
```

بنابراین در متد Post نیز باید بر این اساس، response.Content را از نوع لیستی از DataSourceResult تعریف کرد تا Kendo UI Grid بداند که Id رکورد جدید را باید از فیلد Data، همانند تنظیمات schema منبع داده خود، دریافت کند.

```
response.Content = new ObjectContent<DataSourceResult>(
                                new DataSourceResult { Data = new[] { product } }, new
JsonMediaTypeFormatter());
```

اگر این تنظیم صورت نگیرد، Id رکورد جدید را در گرید، مساوی صفر مشاهده خواهید کرد و عملا بدون استفاده خواهد شد؛ زیرا قابلیت ویرایش و حذف خود را از دست می‌دهد.

متدهای حذف و به روز رسانی سمت سرور نیز چنین امضایی را خواهند داشت:

```
namespace KendoUI06.Controllers
{
    public class ProductsController : ApiController
    {
        public HttpResponseMessage Delete(int id)
        {
            var item = ProductDataSource.LatestProducts.FirstOrDefault(x => x.Id == id);
            if (item == null)
                return Request.CreateResponse(HttpStatusCode.NotFound);
```

```
            ProductDataSource.LatestProducts.Remove(item);

            return Request.CreateResponse(HttpStatusCode.OK, item);
        }

        [HttpPut] // Add it to fix this error: The requested resource does not support http method
'PUT'
        public HttpResponseMessage Update(int id, Product product)
        {
            var item = ProductDataSource.LatestProducts
                                        .Select(
                                            (prod, index) =>
                                                new
                                                {
                                                    Item = prod,
                                                    Index = index
                                                })
                                        .FirstOrDefault(x => x.Item.Id == id);
            if (item == null)
                return Request.CreateResponse(HttpStatusCode.NotFound);


            if (!ModelState.IsValid || id != product.Id)
                return Request.CreateResponse(HttpStatusCode.BadRequest);

            ProductDataSource.LatestProducts[item.Index] = product;
            return Request.CreateResponse(HttpStatusCode.OK);
        }
    }
}
```

حالت Update از HTTP Verb خاصی به نام Put استفاده می‌کند و ممکن است در این بین خطای The requested resource does not support http method 'PUT' را دریافت کنید. برای رفع آن ابتدا بررسی کنید که آیا Web.config برنامه دارای تعاریف ExtensionlessUrlHandler هست یا خیر. همچنین مزین کردن این متد با ویژگی HttpPut، مشکل را برطرف می‌کند.

تنظیمات Kendo UI Grid جهت فعال سازی CRUD

در ادامه کلیه تغییرات مورد نیاز جهت فعال سازی CRUD را در Kendo UI، به همراه مباحث بومی سازی عبارات متناظر با دکمه‌ها و صفحات خودکار مرتبط، مشاهده می‌کنید:

```
$("#report-grid").kendoGrid({
                //....
                editable: {
                    confirmation: "آیا مایل به حذف ردیف انتخابی هستید؟",
                    destroy: true, // whether or not to delete item when button is clicked
                    mode: "popup", // options are "incell", "inline", and "popup"
                    //template: kendo.template($("#popupEditorTemplate").html()), // template to use
for pop-up editing
                    update: true, // switch item to edit mode when clicked?
                    window: {
                        title: "مشخصات محصول"   // Localization for Edit in the popup window
                    }
                },
                columns: [
                //....
                    {
                        command: [
                            { name: "edit", text: "ویرایش" },
                            { name: "destroy", text: "حذف" }
                        ],
                        title: " ", width: "160px"
                    }
                ],
                toolbar: [
                    { name: "create", text: "افزودن ردیف جدید" },
                    { name: "save", text: "ذخیره‌ی تمامی تغییرات" },
                    { name: "cancel", text: "لغو کلیه‌ی تغییرات" },
                    { template: kendo.template($("#toolbarTemplate").html()) }
                ],
                messages: {
                    editable: {
                        cancelDelete: "لغو",
                        confirmation: "آیا مایل به حذف این رکورد هستید؟",
```

```
                    confirmDelete: "حذف"
                },
                commands: {
                    create: "افزودن ردیف جدید",
                    cancel: "لغو کلیه‌ی تغییرات",
                    save: "ذخیره‌ی تمامی تغییرات",
                    destroy: "حذف",
                    edit: "ویرایش",
                    update: "ثبت",
                    canceledit: "لغو"
                }
            }
        });
```

- ساده‌ترین حالت CRUD در Kendo UI با مقدار دهی خاصیت editable آن به true آغاز می‌شود. در این حالت، ویرایش درون سلولی یا incell فعال خواهد شد که مباحث batching ابتدای بحث، فقط در این حالت کار می‌کند. زمانیکه incell editing فعال است، کاربر می‌تواند تمام ردیف‌ها را ویرایش کرده و در آخر کار بر روی دکمه‌ی «ذخیره‌ی تمامی تغییرات» موجود در نوار ابزار، کلیک کند. در سایر حالات، هر بار تنها یک ردیف را می‌توان ویرایش کرد.
- برای فعال سازی تولید صفحات خودکار ویرایش و افزودن ردیف‌ها، نیاز است خاصیت editable را به نحوی که ملاحظه می‌کنید، مقدار دهی کرد. خاصیت mode آن سه حالت incell (پیش فرض)، inline و popup را پشتیبانی می‌کند.
- اگر حالت‌های inline و یا popup را فعال کردید، در انتهای ستون‌های تعریف شده، نیاز است ستون ویژه‌ای به نام command را مطابق تعاریف فوق، تعریف کنید. در این حالت دو دکمه‌ی ویرایش و ثبت، فعال می‌شوند و اطلاعات خود را از تنظیمات data source گرید دریافت می‌کنند. دکمه‌ی ویرایش در حالت incell کاربردی ندارد (چون در این حالت کاربر با کلیک درون یک سلول می‌تواند آن‌را مانند برنامه‌ی اکسل ویرایش کند). اما دکمه‌ی حذف در هر سه حالت قابل استفاده است.
- به نوار ابزار گرید، سه دکمه‌ی افزودن ردیف‌های جدید، ذخیره‌ی تمامی تغییرات و لغو تغییرات صورت گرفته، اضافه شده‌اند. این دکمه‌ها استاندارد بوده و در اینجا نحوه‌ی بومی سازی پیام‌های مرتبط را نیز مشاهده می‌کنید. همانطور که عنوان شد، دکمه‌های «تمامی تغییرات» در حالت فعال سازی batching در منبع داده و استفاده از incell editing معنا پیدا می‌کند. در سایر حالات این دو دکمه کاربردی ندارند. اما دکمه‌ی افزودن ردیف‌های جدید در هر سه حالت کاربرد دارد و یکسان است.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید

KendoUI06.zip

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۲۱ ۹:۴۵

یک نکته‌ی تکمیلی

در مثال فوق از ASP.NET Web API استفاده شده‌است. اگر علاقمند به استفاده از WCF و یا حتی فایل‌های asmx قدیمی هم باشید، اینکار میسر است. مثال‌هایی را در این زمینه، در اینجا می‌توانید مشاهده کنید.

عنوان: **اعتبار سنجی ورودی‌های کاربر در Kendo UI**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۲۲ ۱۵:۱۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, Kendo UI

در مطلب « فعال سازی عملیات CRUD در Kendo UI Grid » با نحوه‌ی تعریف مقدماتی اعتبارسنجی فیلدهای تعریف شده، آشنا شدید:

```
fields: {
      "Price": { type: "number", validation: { required: true, min: 1 } }
}
```

در ادامه نگاهی خواهیم داشت به جزئیات تکمیلی امکانات اعتبارسنجی ورودی‌های کاربر در Kendo UI.

Kendo UI Validation و HTML 5

در HTML 5 امکان تعریف نوع‌های خاص کنترل‌های ورودی کاربر مانند email، url، number، range، date، search و color وجود دارد. برای مثال در اینجا اگر کاربر تاریخ غیرمعتبری را وارد کند، مرورگر پیام اعتبارسنجی متناظری را به او نمایش خواهد داد.
همچنین در HTML 5 امکان افزودن ویژگی required نیز به کنترل‌های ورودی پیش بینی شده‌است. اما باید درنظر داشت که مرورگرهای قدیمی از این امکانات پشتیبانی نمی‌کنند. در این حالت Kendo UI با تشویق استفاده از روش معرفی شده در HTML 5، با آن یکپارچه شده و همچنین این قابلیت‌های اعتبارسنجی HTML 5 را در مرورگرهای قدیمی نیز میسر می‌کند. Kendo UI Validation جزو نسخه‌ی سورس باز Kendo UI با مجوز Apache نیز می‌باشد.
نمونه‌ای از امکانات اعتبارسنجی توکار HTML 5 را در اینجا مشاهده می‌کنید:

```
<input type="text" name="firstName" required />
<input type="text" name="twitter" pattern="https?://(?:www\.)?twitter\.com/.+i" />
<input type="number" name="age" min="1" max="42" />
<input type="number" name="age" min="1" max="100" step="2" />
<input type="url" name="url" />
<input type="email" name="email" />
```

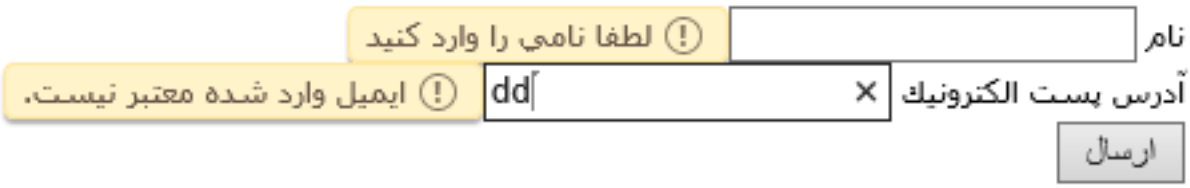
یکپارچه سازی اعتبارسنجی Kendo UI با اعتبارسنجی HTML 5

در اینجا یک فرم تشکیل شده با ساختار HTML 5 را ملاحظه می‌کنید. هر دو فیلد ورودی، با ویژگی استاندارد required مزین شده‌اند. همچنین توسط ویژگی type، ورودی دوم جهت دریافت آدرس ایمیل معرفی شده‌است.
چون فیلد دوم دارای دو اعتبارسنجی تعریف شده است، دارای دو ویژگی data-* برای تعریف پیام‌های اعتبارسنجی متناظر نیز می‌باشد. الگوی تعریف آن‌ها data-[rule]-msg است.

```
<div class="k-rtl">
        <form id="testView">
            <label for="firstName">نام</label>
            <input id="firstName"
                   name="firstName"
                   type="text"
                   class="k-textbox"
                   required
                   validationmessage="لطفا نامی را وارد کنید">
            <br>
            <label for="emailId">آدرس پست الکترونیک</label>
            <input id="emailId"
                   name="emailId"
                   type="email"
                   dir="ltr"
                   required
                   class="k-textbox"
                   data-required-msg="لطفا ایمیلی را وارد کنید."
                   data-email-msg="ایمیل وارد شده معتبر نیست.">
            <br>
```

```
            <input type="submit" class="k-button" value="ارسال">
        </form>
    </div>

    <script type="text/javascript">
        $(function () {
            $("form#testView").kendoValidator();
        });
    </script>
```

تنها کاری که جهت یکپارچه سازی امکانات اعتبارسنجی Kendo UI با اعتبارسنجی استاندارد HTML 5 باید انجام داد، فراخوانی متد kendoValidator بر روی ناحیه‌ی مشخص شده است.



تعیین محل نمایش پیام‌های اعتبارسنجی

پیام‌های اعتبارسنجی Kendo UI به صورت خودکار در کنار فیلد متناظر با آن نمایش داده می‌شوند. اما اگر نیاز به تعیین مکان دستی آن‌ها وجود داشت (جهت خوانایی بهتر) باید به نحو ذیل عمل کرد:

```
<input type="text" id="name" name="name" required>
     <span class="k-invalid-msg" data-for="name"></span>
```

در اینجا span با کلاس k-invalid-msg و ویژگی data-for که به name کنترل ورودی اشاره می‌کند، محل نمایش پیام اعتبارسنجی متناظر با فیلد name خواهد بود.

تعریف سراسری پیام‌های اعتبارسنجی

در مثال فوق، به ازای تک تک فیلدهای ورودی، پیام اعتبارسنجی متناظر با required وارد شد. می‌توان این پیام‌ها را حذف کرد و در قسمت messages متد kendoValidator قرار داد:

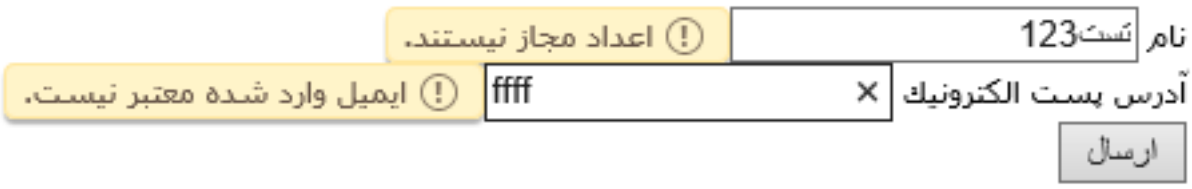
```
<script type="text/javascript">
        $(function () {
            $("form#testView").kendoValidator({
                messages: {
                    // {0} would be replaced with the input element's name
                    required: '{0} را تکمیل کنید.',
                    email: 'ایمیل وارد شده معتبر نیست.'
                }
            });
        });
    </script>
```

- به این صورت پیام‌های اعتبارسنجی required و email، به صورت یکسانی به تمام المان‌های دارای این ویژگی‌ها اعمال خواهند شد.
- در این پیام‌ها {0} با مقدار ویژگی name فیلد ورودی متناظر جایگزین می‌شود.
- اگر هم در markup و هم در تعاریف kendoValidator، پیام‌های اعتبارسنجی تعریف شوند، حق تقدم با تعاریف markup خواهد بود.

اعتبارسنجی سفارشی سمت کاربر

علاوه بر امکانات استاندارد HTML 5، امکان تعریف دستورهای اعتبارسنجی سفارشی نیز وجود دارد:

```
<script type="text/javascript">
        $(function () {
            $("form#testView").kendoValidator({
                rules: {
                    customRule1: function (input) {
                        if (!input.is("[id=firstName]"))
                            return true;

                        var re = /^[A-Za-z]+$/;
                        return re.test(input.val());
                    }
                   //, customRule1: ….
                },
                messages: {
                    // {0} would be replaced with the input element's name
                    required: '{0} را تکمیل کنید.',
                    email: 'ایمیل وارد شده معتبر نیست.',
                    customRule1: 'اعداد مجاز نیستند.'
                }
            });
        });
    </script>
```



- همانطور که ملاحظه می‌کنید، برای تعریف منطق اعتبارسنجی سفارشی، باید از خاصیت rules ورودی متد kendoValidator شروع کرد. در اینجا نام یک متد callback دلخواهی را وارد کرده و سپس بر اساس منطق اعتبارسنجی مورد نظر، باید true/false را بازگشت داد. برای نمونه در این مثال اگر کاربر در فیلد نام، عدد وارد کند، ورودی او مورد قبول واقع نخواهد شد.
- باید دقت داشت که اگر بررسی input.is صورت نگیرد، منطق تعریف شده به تمام کنترل‌های صفحه اعمال می‌شود.
- پیام متناظر با این دستور سفارشی جدید، در قسمت messages، دقیقا بر اساس نام callback method تعریف شده در قسمت rules باید تعریف شود.

فراخوانی دستی اعتبارسنجی یک فرم

در حالت پیش فرض، با کلیک بر روی دکمه‌ی ارسال، اعتبارسنجی کلیه عناصر فرم به صورت خودکار انجام می‌شود. اگر بخواهیم در این بین یک پیام سفارشی را نیز نمایش دهیم می‌توان به صورت زیر عمل کرد:

```
<script type="text/javascript">
        $(function () {
            $("form#testView").submit(function (event) {
                event.preventDefault();
                var validator = $("form#testView").data("kendoValidator");
                if (validator.validate()) {
                    alert("validated!");
                } else {
                    alert("There is invalid data in the form.");
                }
```

```
            });

            $("form#testView").kendoValidator();
        });
    </script>
```

در اینجا رخداد submit فرم بازنویسی شده و متد validate آن بر اساس kendoValidator تعریف شده، به صورت دستی فراخوانی می‌شود.

اعتبارسنجی سفارشی در DataSource

در تعریف فیلدهای مدل DataSource، امکان تعریف اعتبارسنجی‌های پیش فرضی مانند rquired، min، max و امثال آن وجود دارد که نمونه‌ای از آن‌را در بحث فعال سازی CRUD در Kendo UI Grid مشاهده کردید:

```
fields: {
   "serviceName": {
    type: "string",
    defaultValue: "Inspection",
    editable: true,
    nullable: false,
    validation: { /*...*/ }
   },
   // ...
}
```

برای تعریف اعتبارسنجی سفارشی در اینجا، همانند متد kendoValidator نیاز است یک یا چند callback متد سفارشی را طراحی کرد:

```
schema: {
            model: {
                        id: "ProductID",
                        fields: {
                                    ProductID: { editable: false, nullable: true },
                                    ProductName: {
                                        validation: {
                                            required: true,
                                            custom1: function (input) {
                                                if (input.is("[name='ProductName']") && input.val()
!= "") {
                                                    input.attr("data-custom1-msg", "نام محصول باید
با حرف بزرگ انگلیسی شروع شود");
                                                    return /^[A-Z]/.test(input.val());
                                                }

                                                return true;
                                            }
                                          // ,custom2: ...
                                        }
                                    },
                                    UnitPrice: { type: "number", validation: { required: true, min:
1} },
                                    Discontinued: { type: "boolean" },
                                    UnitsInStock: { type: "number", validation: { min: 0, required:
true} }
                                }
                            }
                        }
```

نام این متد که نهایتا true/false بر می‌گرداند، اختیاری است. نام کنترل جاری همان نام فیلد متناظر است (جهت محدود کردن بازه‌ی اعمال منطق اعتبارسنجی). برای مقدار دهی پیام اعتبارسنجی از متد input.attr و الگوی data-[validationRuleName]-msg استفاده می‌شود. ضمنا به هر تعداد لازم می‌توان در اینجا custom rule تعریف کرد.
متد input.val() مقدار کنترل جاری را بر می‌گرداند. برای دسترسی به مقدار سایر کنترل‌ها می‌توان از روش $("#fieldName").val() استفاده کرد.

نظرات خوانندگان

نویسنده: محمد رعیت پیشه
تاریخ: ۱۳۹۳/۰۸/۲۲ ۱۷:۱

ممنون مطلبتون.
آیا اعتبار سنجی در حالت Inline Editing درون گرید هم به همین شکل هست؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۲۲ ۱۸:۱۱

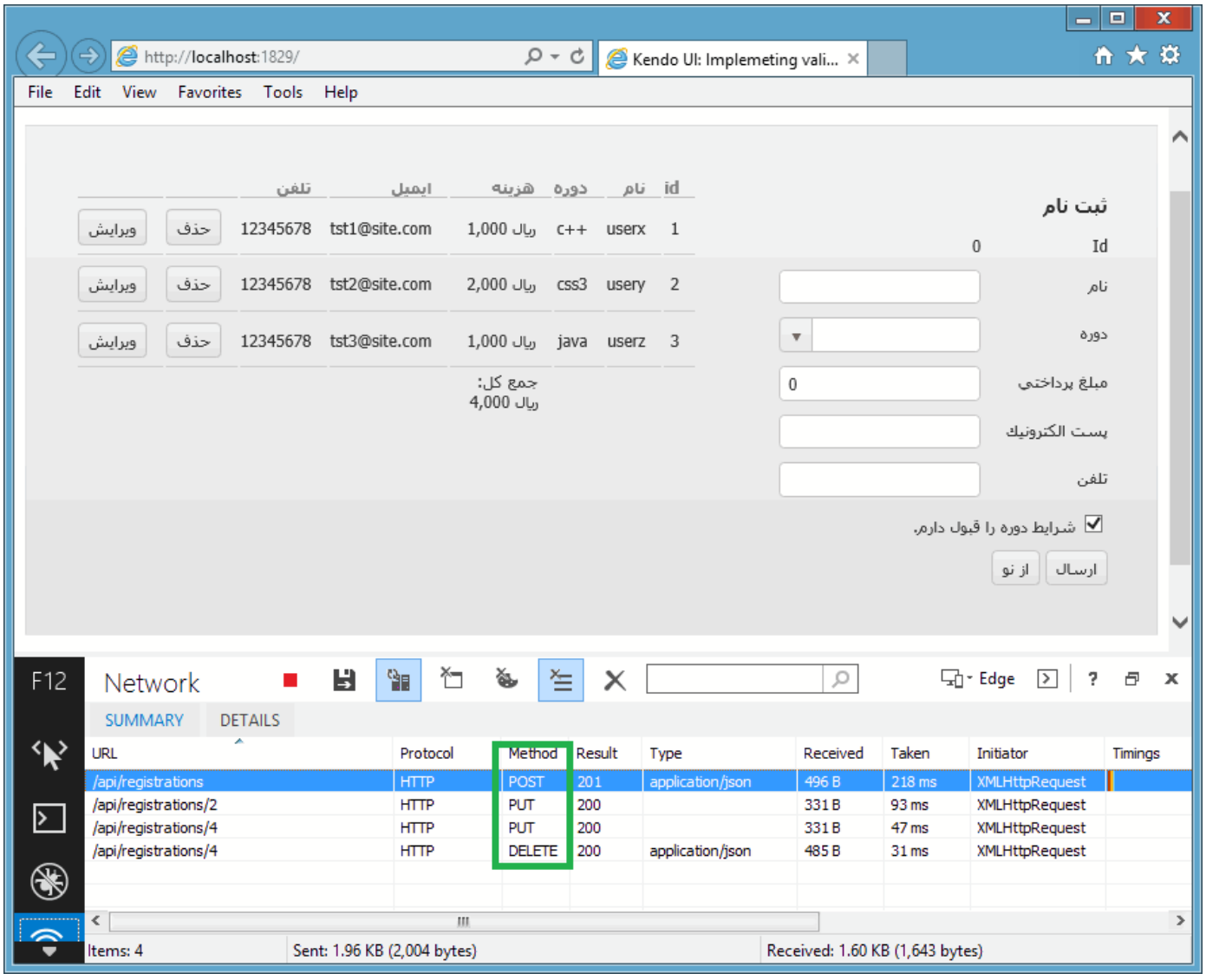
بله. اطلاعات اعتبارسنجی فیلدهای خودش را از data source دریافت می‌کند. مثال فعال سازی CRUD در Kendo UI Grid را اجرا کنید، این مورد در آن لحاظ شده.

عنوان: **Kendo UI MVVM**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۲۴ ۱۵:۵۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, MVVM, ASP.NET Web API, jQuery, KnockoutJS, Kendo UI

پیشنیازها

- « استفاده از Kendo UI templates »
- « اعتبار سنجی ورودی‌های کاربر در Kendo UI »
- « فعال سازی عملیات CRUD در Kendo UI Grid » جهت آشنایی با نحوه‌ی تعریف DataSource ایی که می‌تواند اطلاعات را ثبت، حذف و یا ویرایش کند.

در این مطلب قصد داریم به یک چنین صفحه‌ای برسیم که در آن در ابتدای نمایش، لیست ثبت نام‌های موجود، از سرور دریافت و توسط یک Kendo UI template نمایش داده می‌شود. سپس امکان ویرایش و حذف هر ردیف، وجود خواهد داشت، به همراه امکان افزودن ردیف‌های جدید. در این بین مدیریت نمایش لیست ثبت نام‌ها توسط امکانات binding توکار فریم ورک MVVM مخصوص Kendo UI صورت خواهد گرفت. همچنین کلیه اعمال مرتبط با هر ردیف نیز توسط data binding دو طرفه مدیریت خواهد شد.



Kendo UI MVVM

الگوی MVVM یا Model-View-ViewModel که برای اولین بار جهت کاربردهای WPF و Silverlight معرفی شد، برای ساده سازی اتصال تغییرات کنترل‌های برنامه به خواص ViewModel یک View کاربرد دارد. برای مثال با تغییر عنصر انتخابی یک DropDownList در یک View، بلافاصله خاصیت متصل به آن که در ViewModel برنامه تعریف شده‌است، مقدار دهی و به روز خواهد شد. هدف نهایی آن نیز جدا سازی منطق کدهای UI، از کدهای جاوا اسکریپتی سمت کاربر است. برای این منظور کتابخانه‌هایی مانند Knockout.js به صورت اختصاصی برای این کار تهیه شده‌اند؛ اما Kendo UI نیز جهت یکپارچگی هرچه تمامتر اجزای آن، دارای یک فریم ورک MVVM توکار نیز می‌باشد. طراحی آن نیز بسیار شبیه به Knockout.js است؛ اما با سازگاری 100 درصد با کل مجموعه.
پیاده سازی الگوی MVVM از 4 قسمت تشکیل می‌شود:
- Model که بیانگر خواص متناظر با اشیاء رابط کاربری است.
- View همان رابط کاربری است که به کاربر نمایش داده می‌شود.
- ViewModel واسطی است بین Model و View. کار آن انتقال داده‌ها و رویدادها از View به مدل است و در حالت binding دوطرفه، عکس آن نیز صحیح می‌باشد.
- Declarative data binding جهت رهایی برنامه نویس‌ها از نوشتن کدهای هماهنگ سازی اطلاعات المان‌های View و خواص ViewModel کاربرد دارد.

در ادامه این اجزا را با پیاده سازی مثالی که در ابتدای بحث مطرح شد، دنبال می‌کنیم.

تعریف Model و ViewModel

در سمت سرور، مدل ثبت نام برنامه چنین شکلی را دارد:

```
namespace KendoUI07.Models
{
    public class Registration
    {
        public int Id { set; get; }
        public string UserName { set; get; }
        public string CourseName { set; get; }
        public int Credit { set; get; }
        public string Email { set; get; }
        public string Tel { set; get; }
    }
}
```

در سمت کاربر، این مدل را به نحو ذیل می‌توان تعریف کرد:

```
<script type="text/javascript">
        $(function () {
            var model = kendo.data.Model.define({
                id: "Id",
                fields: {
                    Id: { type: 'number' }, // leave this set to 0 or undefined, so Kendo knows it is
new.
                    UserName: { type: 'string' },
                    CourseName: { type: 'string' },
                    Credit: { type: 'number' },
                    Email: { type: 'string' },
                    Tel: { type: 'string' }
                }
            });
        });
    </script>
```

و ViewModel برنامه در ساده‌ترین شکل آن اکنون چنین تعریفی را خواهد یافت:

```
<script type="text/javascript">
        $(function () {
            var viewModel = kendo.observable({
```

```
                accepted: false,
                course: new model()
            });
        });
    </script>
```

یک viewModel در Kendo UI به صورت یک observable object تعریف می‌شود که می‌تواند دارای تعدادی خاصیت و متد دلخواه باشد. هر خاصیت آن به یک عنصر HTML متصل خواهد شد. در اینجا این اتصال دو طرفه است؛ به این معنا که تغییرات UI به خواص viewModel و برعکس منتقل و منعکس می‌شوند.

اتصال ViewModel به View برنامه

تعریف فرم ثبت نام را در اینجا ملاحظه می‌کنید. فیلدهای مختلف آن بر اساس نکات اعتبارسنجی HTML 5 با ویژگی‌های خاص آن، مزین شده‌اند. جزئیات آن‌را در مطلب « اعتبار سنجی ورودی‌های کاربر در Kendo UI » پیشتر بررسی کرده‌ایم.
اگر به تعریف هر فیلد دقت کنید، ویژگی data-bind جدیدی را هم ملاحظه خواهید کرد:

```
<div id="coursesSection" class="k-rtl k-header">
        <div class="box-col">
            <form id="myForm" data-role="validator" novalidate="novalidate">
                <h3>ثبت نام</h3>
                <ul>
                    <li>
                        <label for="Id">Id</label>
                        <span id="Id" data-bind="text:course.Id"></span>
                    </li>
                    <li>
                        <label for="UserName">نام</label>
                        <input type="text" id="UserName" name="UserName" class="k-textbox"
                               data-bind="value:course.UserName"
                               required />
                    </li>
                    <li>
                        <label for="CourseName">دوره</label>
                        <input type="text" dir="ltr" id="CourseName" name="CourseName" required
                               data-bind="value:course.CourseName" />
                        <span class="k-invalid-msg" data-for="CourseName"></span>
                    </li>
                    <li>
                        <label for="Credit">مبلغ پرداختی</label>
                        <input id="Credit" name="Credit" type="number" min="1000" max="6000"
                               required data-max-msg="6000 و 1000 عددی بین" dir="ltr"
                               data-bind="value:course.Credit"
                               class="k-textbox k-input" />
                        <span class="k-invalid-msg" data-for="Credit"></span>
                    </li>
                    <li>
                        <label for="Email">پست الکترونیک</label>
                        <input type="email" id="Email" dir="ltr" name="Email"
                               data-bind="value:course.Email"
                               required class="k-textbox" />
                    </li>
                    <li>
                        <label for="Tel">تلفن</label>
                        <input type="tel" id="Tel" name="Tel" dir="ltr" pattern="\d{8}"
                               required class="k-textbox"
                               data-bind="value:course.Tel"
                               data-pattern-msg="8 رقم" />
                    </li>
                    <li>
                        <input type="checkbox" name="Accept"
                               data-bind="checked:accepted"
                               required />
                        شرایط دوره را قبول دارم.
                        <span class="k-invalid-msg" data-for="Accept"></span>
                    </li>
                    <li>
                        <button class="k-button"
                                data-bind="enabled: accepted, click: doSave"
                                type="submit">
                            ارسال
                        </button>
                        <button class="k-button" data-bind="click: resetModel">از نو</button>
```

```
                    </li>
                </ul>
                <span id="doneMsg"></span>
            </form>
        </div>
```

برای اتصال ViewModel تعریف شده به ناحیه‌ی مشخص شده با DIV ایی با Id مساوی coursesSection، می‌توان از متد kendo.bind استفاده کرد.

```
<script type="text/javascript">
        $(function () {
            var model = kendo.data.Model.define({
            // ...
            });

            var viewModel = kendo.observable({
            // ...
            });

            kendo.bind($("#coursesSection"), viewModel);
        });
    </script>
```

به این ترتیب Kendo UI به بر اساس تعریف data-bind یک فیلد، برای مثال تغییرات خواص course.UserName را به text box نام کاربر منتقل می‌کند و همچنین اگر کاربر اطلاعاتی را در این text box وارد کند، بلافاصله این تغییرات در خاصیت course.UserName منعکس خواهند شد.

```
<input type="text" id="UserName" name="UserName" class="k-textbox"
       data-bind="value:course.UserName"
       required />
```

بنابراین تا اینجا به صورت خلاصه، مدلی را توسط متد kendo.data.Model.define، معادل مدل سمت سرور خود ایجاد کردیم. سپس وهله‌ای از این مدل را به صورت یک خاصیت جدید دلخواهی در ViewModel تعریف شده توسط متد kendo.observable در معرض دید View برنامه قرار دادیم. در ادامه اتصال ViewModel و View، با فراخوانی متد kendo.bind انجام شد. اکنون برای دریافت تغییرات کنترل‌های برنامه، تنها کافی است ویژگی‌های data-bind ایی را به آن‌ها اضافه کنیم.
در ناحیه‌ی تعریف شده توسط متد kendo.bind، کلیه خواص ViewModel در دسترس هستند. برای مثال اگر به تعریف ViewModel دقت کنید، یک خاصیت دیگر به نام accepted با مقدار false نیز در آن تعریف شده‌است (این خاصیت چون صرفا کاربرد UI داشت، در model برنامه قرار نگرفت). از آن برای اتصال checkbox تعریف شده، به button ارسال اطلاعات، استفاده کرده‌ایم:

```
<input type="checkbox" name="Accept"
       data-bind="checked:accepted"
       required />

<button class="k-button"
        data-bind="enabled: accepted, click: doSave"
        type="submit">
       ارسال
</button>
```

برای مثال اگر کاربر این checkbox را انتخاب کند، مقدار خاصیت accepted، مساوی true خواهد شد. تغییر مقدار این خاصیت، توسط ViewModel بلافاصله در کل ناحیه coursesSection منتشر می‌شود. به همین جهت ویژگی enabled: accepted که به معنای مقید بودن فعال یا غیرفعال بودن دکمه بر اساس مقدار خاصیت accepted است، دکمه را فعال می‌کند، یا برعکس و برای انجام این عملیات نیازی نیست کدنویسی خاصی را انجام داد. در اینجا بین checkbox و button یک سیم کشی برقرار است.

ارسال داده‌های تغییر کرده‌ی ViewModel به سرور

تا اینجا 4 جزء اصلی الگوی MVVM که در ابتدای بحث عنوان شد، تکمیل شده‌اند. مدل اطلاعات فرم تعریف گردید. ViewModel ایی

که این خواص را به المان‌های فرم متصل می‌کند نیز در ادامه اضافه شده‌است. توسط ویژگی‌های data-bind کار Declarative data binding انجام می‌شود.

در ادامه نیاز است تغییرات ViewModel را به سرور، جهت ثبت، به روز رسانی و حذف نهایی منتقل کرد.

```
<script type="text/javascript">
        $(function () {
            var model = kendo.data.Model.define({
                //...
            });

            var dataSource = new kendo.data.DataSource({
                type: 'json',
                transport: {
                    read: {
                        url: "api/registrations",
                        dataType: "json",
                        contentType: 'application/json; charset=utf-8',
                        type: 'GET'
                    },
                    create: {
                        url: "api/registrations",
                        contentType: 'application/json; charset=utf-8',
                        type: "POST"
                    },
                    update: {
                        url: function (course) {
                            return "api/registrations/" + course.Id;
                        },
                        contentType: 'application/json; charset=utf-8',
                        type: "PUT"
                    },
                    destroy: {
                        url: function (course) {
                            return "api/registrations/" + course.Id;
                        },
                        contentType: 'application/json; charset=utf-8',
                        type: "DELETE"
                    },
                    parameterMap: function (data, type) {
                        // Convert to a JSON string.  Without this step your content will be form
encoded.
                        return JSON.stringify(data);
                    }
                },
                schema: {
                    model: model
                },
                error: function (e) {
                    alert(e.errorThrown);
                },
                change: function (e) {
                    // فراخوانی در زمان دریافت اطلاعات از سرور و یا تغییرات محلی
                    viewModel.set("coursesDataSourceRows", new
kendo.data.ObservableArray(this.view()));
                }
            });

            var viewModel = kendo.observable({
                //...
            });

            kendo.bind($("#coursesSection"), viewModel);
            dataSource.read(); // دریافت لیست موجود از سرور در آغاز کار
        });
    </script>
```

در اینجا تعریف DataSource کار با منبع داده راه دور ASP.NET Web API را مشاهده می‌کنید. تعاریف اصلی آن با تعاریف مطرح شده در مطلب « فعال سازی عملیات CRUD در Kendo UI Grid » یکی هستند. هر قسمت آن مانند read، create، update و destory به یکی از متدهای کنترلر ASP.NET Web API اشاره می‌کنند. حالت‌های update و destroy بر اساس Id ردیف انتخابی کار می‌کنند. این Id را باید در قسمت model مربوط به اسکیمای تعریف شده، دقیقا مشخص کرد. عدم تعریف فیلد id، سبب خواهد شد تا عملیات update نیز در حالت create تفسیر شود.

متصل کردن DataSource به ViewModel

تا اینجا DataSource ایی جهت کار با سرور تعریف شده‌است؛ اما مشخص نیست که اگر رکوردی اضافه شد، چگونه باید اطلاعات خودش را به روز کند. برای این منظور خواهیم داشت:

```
<script type="text/javascript">
        $(function () {
            $("#coursesSection").kendoValidator({
                // ...
            });

            var model = kendo.data.Model.define({
                // ...
            });

            var dataSource = new kendo.data.DataSource({
                // ...
            });

            var viewModel = kendo.observable({
                accepted: false,
                course: new model(),
                doSave: function (e) {
                    e.preventDefault();
                    console.log("this", this.course);
                    var validator = $("#coursesSection").data("kendoValidator");
                    if (validator.validate()) {
                        if (this.course.Id == 0) {
                            dataSource.add(this.course);
                        }
                        dataSource.sync(); // push to the server
                        this.set("course", new model()); // reset controls
                    }
                },
                resetModel: function (e) {
                    e.preventDefault();
                    this.set("course", new model());
                }
             });

            kendo.bind($("#coursesSection"), viewModel);
            dataSource.read(); // دریافت لیست موجود از سرور در آغاز کار
        });
    </script>
```

همانطور که در تعاریف تکمیلی viewModel مشاهده می‌کنید، اینبار دو متد جدید دلخواه doSave و resetModel را اضافه کرده‌ایم. در متد doSave، ابتدا بررسی می‌کنیم آیا اعتبارسنجی فرم با موفقیت انجام شده‌است یا خیر. اگر بله، توسط متد add منبع داده، اطلاعات فرم جاری را توسط شیء course که هم اکنون به تمامی فیلدهای آن متصل است، اضافه می‌کنیم. در اینجا بررسی شده‌است که آیا Id این اطلاعات صفر است یا خیر. از آنجائیکه از همین متد برای به روز رسانی نیز در ادامه استفاده خواهد شد، در حالت به روز رسانی، Id شیء ثبت شده، از طرف سرور دریافت می‌گردد. بنابراین غیر صفر بودن این Id به معنای عملیات به روز رسانی است و در این حالت نیازی نیست کار بیشتری را انجام داد؛ زیرا شیء متناظر با آن پیشتر به منبع داده اضافه شده‌است.
استفاده از متد add صرفا به معنای مطلع کردن منبع داده محلی از وجود رکوردی جدید است. برای ارسال این تغییرات به سرور، از متد sync آن می‌توان استفاده کرد. متد sync بر اساس متد add یک درخواست POST، بر اساس شیءایی که Id غیر صفر دارد، یک درخواست PUT و با فراخوانی متد remove بر روی منبع داده، یک درخواست DELETE را به سمت سرور ارسال می‌کند.
متد دلخواه resetModel سبب مقدار دهی مجدد شیء course با یک وهله‌ی جدید از شیء model می‌شود. همینقدر برای پاک کردن تمامی کنترل‌های صفحه کافی است.

تا اینجا دو متد جدید را در ViewModel برنامه تعریف کرده‌ایم. در مورد نحوه‌ی اتصال آن‌ها به View، به کدهای دو دکمه‌ی موجود در فرم دقت کنید:

```
<button class="k-button"
        data-bind="enabled: accepted, click: doSave"
        type="submit">
```

```
       ارسال
</button>
<button class="k-button" data-bind="click: resetModel">از نو</button>
```

این متدها نیز توسط ویژگی‌های data-bind به هر دکمه نسبت داده شده‌اند. به این ترتیب برای مثال با کلیک کاربر بر روی دکمه‌ی submit، متد doSave موجود در ViewModel فراخوانی می‌شود.

مدیریت سمت سرور ثبت، ویرایش و حذف اطلاعات

در حالت ثبت، متد Post توسط آدرس مشخص شده در قسمت create منبع داده، فراخوانی می‌گردد. نکته‌ی مهمی که در اینجا باید به آن دقت داشت، نحوه‌ی بازگشت Id رکورد جدید ثبت شده‌است. اگر این تنظیم صورت نگیرد، Id رکورد جدید را در لیست، مساوی صفر مشاهده خواهید کرد و منبع داده این رکورد را همواره به عنوان یک رکورد جدید، مجددا به سرور ارسال می‌کند.

```
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Http;
using System.Web.Http;
using KendoUI07.Models;

namespace KendoUI07.Controllers
{
    public class RegistrationsController : ApiController
    {
        public HttpResponseMessage Delete(int id)
        {
            var item = RegistrationsDataSource.LatestRegistrations.FirstOrDefault(x => x.Id == id);
            if (item == null)
                return Request.CreateResponse(HttpStatusCode.NotFound);

            RegistrationsDataSource.LatestRegistrations.Remove(item);
            return Request.CreateResponse(HttpStatusCode.OK, item);
        }

        public IEnumerable<Registration> Get()
        {
            return RegistrationsDataSource.LatestRegistrations;
        }

        public HttpResponseMessage Post(Registration registration)
        {
            if (!ModelState.IsValid)
                return Request.CreateResponse(HttpStatusCode.BadRequest);

            var id = 1;
            var lastItem = RegistrationsDataSource.LatestRegistrations.LastOrDefault();
            if (lastItem != null)
            {
                id = lastItem.Id + 1;
            }
            registration.Id = id;
            RegistrationsDataSource.LatestRegistrations.Add(registration);

            // ارسال آی دی مهم است تا از ارسال رکوردهای تکراری جلوگیری شود
            return Request.CreateResponse(HttpStatusCode.Created, registration);
        }

        [HttpPut] // Add it to fix this error: The requested resource does not support http method
'PUT'
        public HttpResponseMessage Update(int id, Registration registration)
        {
            var item = RegistrationsDataSource.LatestRegistrations
                                        .Select(
                                            (prod, index) =>
                                                new
                                                {
                                                    Item = prod,
                                                    Index = index
                                                })
                                        .FirstOrDefault(x => x.Item.Id == id);
            if (item == null)
```

```
                return Request.CreateResponse(HttpStatusCode.NotFound);


            if (!ModelState.IsValid || id != registration.Id)
                return Request.CreateResponse(HttpStatusCode.BadRequest);

            RegistrationsDataSource.LatestRegistrations[item.Index] = registration;
            return Request.CreateResponse(HttpStatusCode.OK);
        }
    }
}
```

در اینجا بیشتر امضای این متدها مهم هستند، تا منطق پیاده سازی شده در آن‌ها. همچنین بازگشت Id رکورد جدید، توسط متد Post نیز بسیار مهم است و سبب می‌شود تا DataSource بداند با فراخوانی متد sync آن، باید عملیات Post یا create انجام شود یا Put و update.

نمایش آنی اطلاعات ثبت شده در یک لیست

ردیف‌های اضافه شده به منبع داده را می‌توان بلافاصله در همان سمت کلاینت توسط Kendo UI Template که قابلیت کار با ViewModelها را دارد، نمایش داد:

```
<div id="coursesSection" class="k-rtl k-header">
        <div class="box-col">
            <form id="myForm" data-role="validator" novalidate="novalidate">
                        <!--فرم بحث شده در ابتدای مطلب-->
            </form>
        </div>
        <div id="results">
            <table class="metrotable">
                <thead>
                    <tr>
                        <th>Id</th>
                        <th>نام</th>
                        <th>دوره</th>
                        <th>هزینه</th>
                        <th>ایمیل</th>
                        <th>تلفن</th>
                        <th></th>
                        <th></th>
                    </tr>
                </thead>
                <tbody data-template="row-template" data-bind="source: coursesDataSourceRows"></tbody>
                <tfoot data-template="footer-template" data-bind="source: this"></tfoot>
            </table>
            <script id="row-template" type="text/x-kendo-template">
                <tr>
                    <td data-bind="text: Id"></td>
                    <td data-bind="text: UserName"></td>
                    <td dir="ltr" data-bind="text: CourseName"></td>
                    <td>
                        #: kendo.toString(get("Credit"), "c0") #
                    </td>
                    <td data-bind="text: Email"></td>
                    <td data-bind="text: Tel"></td>
                    <td><button class="k-button" data-bind="click: deleteCourse">حذف</button></td>
                    <td><button class="k-button" data-bind="click: editCourse">ویرایش</button></td>
                </tr>
            </script>
            <script id="footer-template" type="text/x-kendo-template">
                <tr>
                    <td colspan="3"></td>
                    <td>
                        جمع کل: #: kendo.toString(totalPrice(), "c0") #
                    </td>
                    <td colspan="2"></td>
                    <td></td>
                    <td></td>
                </tr>
            </script>
        </div>
    </div>
```

در ناحیه‌ی coursesSection که توسط متد kendo.bind به viewModel برنامه متصل شده‌است، یک جدول را برای نمایش ردیف‌های ثبت شده توسط کاربر اضافه کرده‌ایم. thead آن بیانگر سر ستون جدول است. قسمت tbody و tfoot این جدول توسط دو Kendo UI Template مقدار دهی شده‌اند. هر کدام نیز منبع داده‌اشان را از view model دریافت می‌کنند. در row- template معادل خواص شیء course را مشاهده می‌کنید. در footer-template متد totalPrice برای نمایش جمع ستون هزینه اضافه شده‌است. بنابراین مطابق این قسمت از View، به یک خاصیت جدید coursesDataSourceRows و سه متد deleteCourse، editCourse و totalPrice نیاز است:

```
<script type="text/javascript">
        $(function () {
            // ...
            var viewModel = kendo.observable({
                accepted: false,
                course: new model(),
                coursesDataSourceRows: new kendo.data.ObservableArray([]),
                doSave: function (e) {
                       // ...
                },
                resetModel: function (e) {
                      // ...
                },
                totalPrice: function () {
                    var sum = 0;
                    $.each(this.get("coursesDataSourceRows"), function (index, item) {
                        sum += item.Credit;
                    });
                    return sum;
                },
                deleteCourse: function (e) {
                    // the current data item is passed as the "data" field of the event argument
                    var course = e.data;
                    dataSource.remove(course);
                    dataSource.sync(); // push to the server
                },
                editCourse: function(e) {
                    // the current data item is passed as the "data" field of the event argument
                    var course = e.data;
                    this.set("course", course);
                }
            });

            kendo.bind($("#coursesSection"), viewModel);
            dataSource.read(); // دریافت لیست موجود از سرور در آغاز کار
        });
    </script>
```

نحوه‌ی اتصال خاصیت جدید coursesDataSourceRows که به عنوان منبع داده ردیف‌های row-template عمل می‌کند، به این صورت است:

- ابتدا خاصیت دلخواه coursesDataSourceRows به viewModel اضافه می‌شود تا در ناحیه‌ی coursesSection در دسترس قرار گیرد.

- سپس اگر به انتهای تعریف DataSource دقت کنید، داریم:

```
<script type="text/javascript">
        $(function () {
            var dataSource = new kendo.data.DataSource({
                //...
                change: function (e) {
                    // فراخوانی در زمان دریافت اطلاعات از سرور و یا تغییرات محلی
                    viewModel.set("coursesDataSourceRows", new
kendo.data.ObservableArray(this.view()));
                }
            });
        });
    </script>
```

متد change آن، هر زمانیکه اطلاعاتی در منبع داده تغییر کنند یا اطلاعاتی به سمت سرور ارسال یا دریافت گردد، فراخوانی می‌شود. در همینجا فرصت خواهیم داشت تا خاصیت coursesDataSourceRows را جهت نمایش اطلاعات موجود در منبع داده،

مقدار دهی کنیم. همین مقدار دهی ساده سبب اجرای row-template برای تولید ردیف‌های جدول می‌شود. استفاده از new kendo.data.ObservableArray سبب خواهد شد تا اگر اطلاعاتی در فرم برنامه تغییر کند، این اطلاعات بلافاصله در لیست گزارش برنامه نیز منعکس گردد.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید:

KendoUI07.zip

عنوان: یکپارچه سازی سیستم اعتبارسنجی ASP.NET MVC با Kendo UI validator
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۲۷ ۱۱:۵۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, MVC, jQuery, Kendo UI

روش پیش فرض اعتبارسنجی برنامه‌های ASP.NET MVC، استفاده از دو افزونه‌ی jquery.validate و jquery.validate.unobtrusive است.

```
<script src="~/Scripts/jquery.validate.min.js" type="text/javascript"></script>
    <script src="~/Scripts/jquery.validate.unobtrusive.min.js" type="text/javascript"></script>
```

کار اصلی اعتبارسنجی، توسط افزونه‌ی jquery.validate انجام می‌شود و فایل jquery.validate.unobtrusive صرفا یک وفق دهنده و مترجم ویژگی‌های خاص ASP.NET MVC به jquery.validate است.

عدم سازگاری پیش فرض jquery.validate با بعضی از ویجت‌های Kendo UI

در حالت استفاده از Kendo UI، این سیستم هنوز هم کار می‌کند؛ اما با یک مشکل. اگر برای مثال از kendoComboBox استفاده کنید، اعتبارسنجی‌های تعریف شده در برنامه، توسط jquery.validate دیده نخواهند شد. برای مثال فرض کنید یک چنین مدلی در اختیار View برنامه قرار گرفته است:

```
public class OrderDetailViewModel
    {
        [StringLength(15)]
        [Required]
        public string Destination { get; set; }
    }
```

با این View که در آن به فیلد Destination، یک kendoComboBox متصل شده‌است:

```
@model Mvc4TestViewModel.Models.OrderDetailViewModel

@using (Ajax.BeginForm(actionName: "Index", controllerName: "Home",
                       ajaxOptions:new AjaxOptions(),
                       htmlAttributes: new { id = "Form1", name ="Form1" }, routeValues: new { }
                       ))
{
    @Html.AntiForgeryToken()
    @Html.ValidationSummary(true)

    <fieldset>
        <legend>OrderDetail</legend>
        <div class="editor-label">
            @Html.LabelFor(model => model.Destination)
        </div>
        <div class="editor-field">
            @Html.TextBoxFor(model => model.Destination, new { @class = "k-textbox" })
            @Html.ValidationMessageFor(model => model.Destination)
        </div>

        <p>
            <button class="k-button" type="submit" title="Sumbit">
                Sumbit
            </button>
        </p>
    </fieldset>
}

@section JavaScript
{
    <script type="text/javascript">
        $(function () {
            $("#Destination").kendoComboBox({
                dataSource: [
                    "loc 1",
                    "loc 2"
                ]
```

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

اگر برنامه را اجرا کنید و بر روی دکمه‌ی submit کلیک نمائید، ویژگی Required عمل نخواهد کرد و عملا در سمت کاربر اعتبارسنجی رخ نمی‌دهد.

```
<div class="editor-field">
  <span class="k-widget k-combobox k-header k-textbox" style="">
    <span tabindex="-1" class="k-dropdown-wrap k-state-default" unselectable="on">
      <input name="Destination_input" tabindex="0" class="k-input k-textbox" role="combobox" aria-
      busy="false" aria-disabled="false" aria-expanded="false" aria-readonly="false" aria-
      activedescendant="Destination_option_selected" aria-owns="Destination_listbox" style="width: 100%;"
      aria-autocomplete="list" type="text" autocomplete="off"></input>
      <span tabindex="-1" class="k-select" unselectable="on">…</span>
    </span>
    <input name="Destination" class="k-textbox input-validation-error" id="Destination" aria-
    disabled="false" aria-invalid="true" aria-readonly="false" aria-required="true" aria-
    describedby="Destination-error" style="display: none;" type="text" data-val-required="The Destination
    field is required." data-val="true" data-val-length-max="15" data-val-length="The field Destination must
    be a string with a maximum length of 15." data-role="combobox" value=""></input>
  </span>
  <span class="field-validation-error" data-valmsg-replace="true" data-valmsg-for="Destination">…</span>
</div>
```

همانطور که در تصویر مشاهده می‌کنید، با اتصال kendoComboBox به یک فیلد، این فیلد در حالت مخفی قرار می‌گیرد و ویجت کندو یو آی بجای آن نمایش داده خواهد شد. در این حالت چون در فایل jquery.validate.js چنین تنظیمی وجود دارد:

```
$.extend( $.validator, {
    defaults: {
     //…
     ignore: ":hidden",
```

به صورت پیش فرض از اعتبارسنجی فیلدهای مخفی صرفنظر می‌شود.
راه حل آن نیز ساده‌است. تنها باید خاصیت ignore را بازنویسی کرد و تغییر داد:

```
<script type="text/javascript">
        $(function () {
            var form = $('#Form1');
            form.data('validator').settings.ignore = ''; // default is ":hidden".
        });
    </script>
```

در اینجا صرفا خاصیت ignore فرم یک، جهت درنظر گرفتن فیلدهای مخفی تغییر کرده‌است. اگر می‌خواهید این تنظیم را به تمام فرم‌ها اعمال کنید، می‌توان از دستور ذیل استفاده کرد:

```
<script type="text/javascript">
    $.validator.setDefaults({
        ignore: ""
    });
</script>
```

یکپارچه کردن سیستم اعتبارسنجی Kendo UI با سیستم اعتبارسنجی ASP.NET MVC

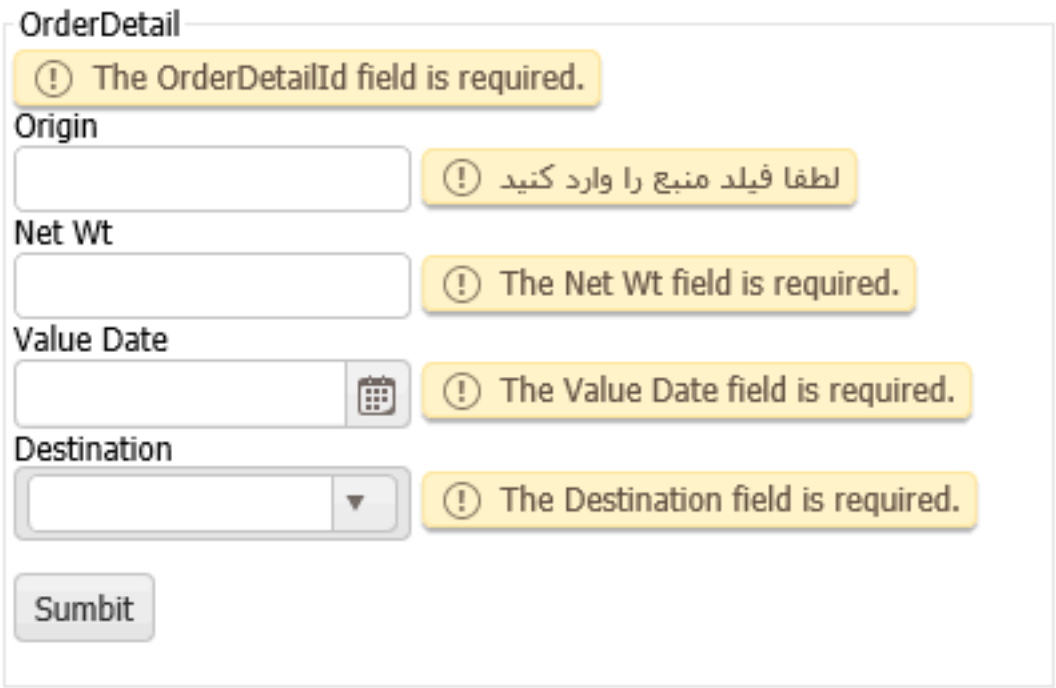
در مطلب « اعتبار سنجی ورودی‌های کاربر در Kendo UI » با زیرساخت اعتبارسنجی Kendo UI آشنا شدید. برای اینکه بتوان این سیستم را با ASP.NET MVC یکپارچه کرد، نیاز است دو کار صورت گیرد:
الف) تعریف فایل kendo.aspnetmvc.js به صفحه اضافه شود:

```
<script src="~/Scripts/kendo.aspnetmvc.js" type="text/javascript"></script>
```

ب) همانند قبل، متد kendoValidator بر روی فرم فراخوانی شود تا سیستم اعتبارسنجی Kendo UI در این ناحیه فعال گردد:

```
<script type="text/javascript">
        $(function () {
            $("form").kendoValidator();
        });
    </script>
```

پس از آن خواهیم داشت:



فایل kendo.aspnetmvc.js که در بسته‌ی مخصوص Kendo UI تهیه شده برای ASP.NET MVC موجود است (در پوشه‌ی js آن)، عملکردی مشابه فایل jquery.validate.unobtrusive مایکروسافت دارد. کار آن وفق دادن و ترجمه‌ی اعتبارسنجی unobtrusive به روش Kendo UI است.

این فایل را از اینجا می‌توانید دریافت کنید:
kendo.mvc.zip

البته باید دقت داشت که در حال حاضر فقط ویژگی‌های ذیل از ASP.NET MVC توسط kendo.aspnetmvc.js پشتیبانی می‌شوند :

```
Required
```

```
StringLength
Range
RegularExpression
```

برای تکمیل آن می‌توان از یک پروژه‌ی سورس باز به نام Moon.Validation for KendoUI Validator استفاده کرد. برای مثال remote validation مخصوص Kendo UI را اضافه کرده‌است .

نظرات خوانندگان

نویسنده: امیر
تاریخ: ۱۳۹۳/۰۸/۲۸ ۱۳:۰

سئوالی برای من پیش اومد امکان استفاده از مدل mvc داخل کنترل‌های جاوااسکریپت کندو هست ؟ مثلا اگه combobox انتخاب بشه آیدیش در مدل اصلی انتخاب بشه ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۸/۲۸ ۱۳:۴۶

« استفاده ازExpressionها جهت ایجاد Strongly typed view در ASP.NET MVC »

عنوان: **بررسی ویجت Kendo UI File Upload**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۰۳ ۱۱:۳۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, Kendo UI

Kendo UI به همراه یک ویجت وب مخصوص ارسال فایل‌ها به سرور نیز هست. این ویجت قابلیت ارسال چندین فایل با هم را به صورت Ajax ایی دارا است و همچنین کاربران می‌توانند فایل‌ها را با کشیدن و رها کردن بر روی آن، به لیست فایل‌های قابل ارسال اضافه کنند.
ارسال فایل Ajax ایی آن توسط HTML5 File API صورت می‌گیرد که در تمام مرورگرهای جدید پشتیبانی خوبی از آن وجود دارد. در مرورگرهای قدیمی‌تر، به صورت خودکار همان حالت متداول ارسال همزمان فایل‌ها را فعال می‌کند (یا همان post back معمولی).

فعال سازی مقدماتی kendoUpload

ابتدایی‌ترین حالت کار با kendoUpload، فعال سازی حالت post back معمولی است؛ به شرح زیر:

```
<form method="post" action="submit" enctype="multipart/form-data">
  <div>
    <input name="files" id="files" type="file" />
    <input type="submit" value="Submit" class="k-button" />
  </div>
</form>
<script>
  $(document).ready(function() {
     $("#files").kendoUpload();
  });
</script>
```

در این حالت صرفا input با نوع file، با ظاهری سازگار با سایر کنترل‌های Kendo UI به نظر می‌رسد و عملیات ارسال فایل، همانند قبل به همراه یک post back است. این روش برای حالتی مفید است که بخواهید یک فایل را به همراه سایر عناصر فرم در طی یک مرحله به سمت سرور ارسال کنید.

فعال سازی حالت ارسال فایل Ajax ایی kendoUpload

برای فعال سازی ارسال Ajax ایی فایل‌ها در Kendo UI نیاز است خاصیت async آن‌را به نحو ذیل مقدار دهی کرد:

```
<script type="text/javascript">
        $(function () {
            $("#files").kendoUpload({
                name: "files",
                async: { // async configuration
                    saveUrl: "@Url.Action("Save", "Home")", // the url to save a file is '/save'
                    removeUrl: "@Url.Action("Remove", "Home")", // the url to remove a file is
'/remove'
                    autoUpload: false, // automatically upload files once selected
                    removeVerb: 'POST'
                },
                multiple: true,
                showFileList: true
            });
        });
    </script>
```

در اینجا دو آدرس ذخیره سازی فایل‌ها و همچنین حذف آن‌ها را مشاهده می‌کنید. امضای این دو اکشن متد در ASP.NET MVC به صورت ذیل هستند:

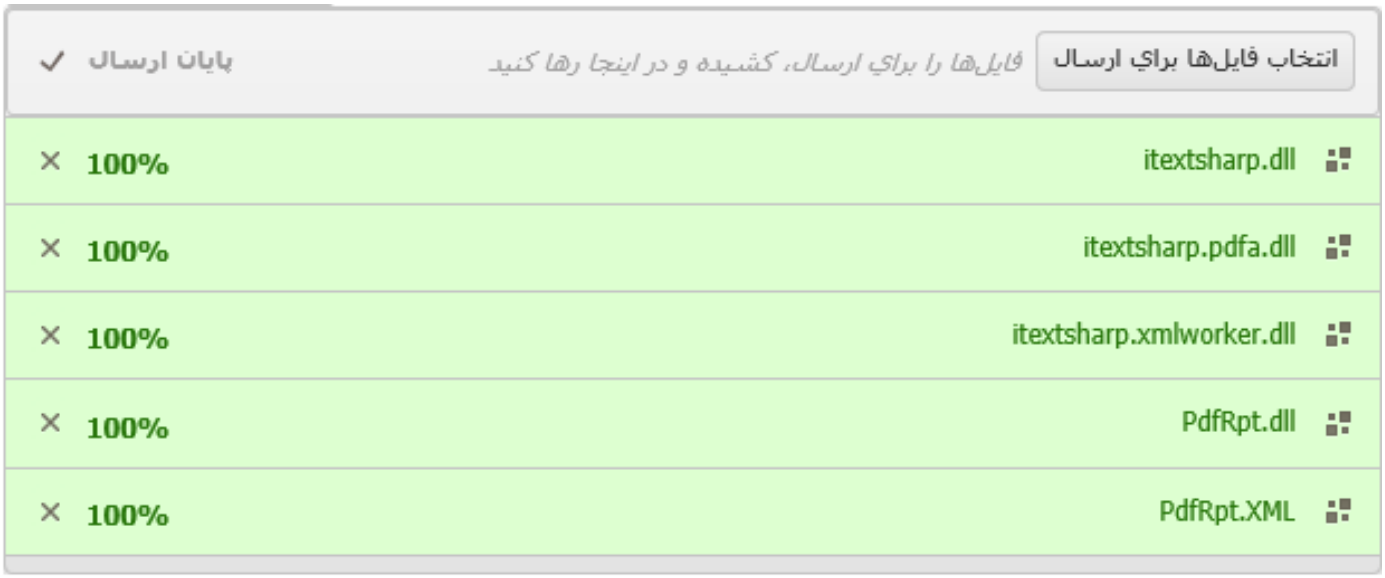
```
[HttpPost]
        public ActionResult Save(IEnumerable<HttpPostedFileBase> files)
        {
```

```
            if (files != null)
            {
                // ...
                // Process the files and save them
                // ...
            }

            // Return an empty string to signify success
            return Content("");
        }

        [HttpPost]
        public ContentResult Remove(string[] fileNames)
        {
            if (fileNames != null)
            {
                foreach (var fullName in fileNames)
                {
                    // ...
                    // delete the files
                    // ...
                }
            }

            // Return an empty string to signify success
            return Content("");
        }
```

در هر دو حالت، لیستی از فایل‌ها توسط kendoUpload به سمت سرور ارسال می‌شوند. در حالت Save، محتوای این فایل‌ها جهت ذخیره سازی بر روی سرور در دسترس خواهد بود. در حالت Remove، صرفا نام این فایل‌ها برای حذف از سرور، توسط کاربر ارسال می‌شوند.
دو دکمه‌ی حذف با کارکردهای متفاوت در ویجت kendoUpload وجود دارند. در ابتدای کار، پیش از ارسال فایل‌ها به سرور:

کلیک بر روی دکمه‌ی حذف در این حالت، صرفا فایلی را از لیست سمت کاربر حذف می‌کند.

پس از ارسال فایل‌ها به سرور:



اما پس از پایان عملیات ارسال، اگر کاربر بر روی دکمه‌ی حذف کلیک کند، توسط آدرس مشخص شده توسط خاصیت removeUrl، نام فایل‌های مورد نظر، برای حذف از سرور ارسال می‌شوند.

چند نکته‌ی تکمیلی

- تنظیم خاصیت autoUpload به true سبب می‌شود تا پس از انتخاب فایل‌ها توسط کاربر، بلافاصله و به صورت خودکار عملیات ارسال فایل‌ها به سرور آغاز شوند. اگر به false تنظیم شود، دکمه‌ی ارسال فایل‌ها در پایین لیست نمایش داده خواهد شد.
- شاید علاقمند باشید تا removeVerb را به DELETE تغییر دهید؛ بجای POST. به همین منظور می‌توان خاصیت removeVerb در اینجا مقدار دهی کرد.
- با تنظیم خاصیت multiple به true، کاربر قادر خواهد شد تا توسط صفحه‌ی دیالوگ انتخاب فایل‌ها، قابلیت انتخاب بیش از یک فایل را داشته باشد.
- showFileList نمایش لیست فایل‌ها را سبب می‌شود.

تعیین پسوند فایل‌های صفحه‌ی انتخاب فایل‌ها

هنگامیکه کاربر بر روی دکمه‌ی انتخاب فایل‌ها برای ارسال کلیک می‌کند، در صفحه‌ی دیالوگ باز شده می‌توان پسوندهای پیش فرض مجاز را نیز تعیین کرد.
برای این منظور تنها کافی است ویژگی accept را به input از نوع فایل اضافه کرد. چند مثال در این مورد:

```
<!-- Content Type with wildcard.  All Images -->
<input type="file" id="demoFile" title="Select file" accept="image/*" />

<!-- List of file extensions -->
<input type="file" id="demoFile" title="Select file" accept=".jpg,.png,.gif" />

<!-- Any combination of the above -->
<input type="file" id="demoFile" title="Select file" accept="audio/*,application/pdf,.png" />
```

نمایش متن کشیدن و رها کردن، بومی سازی برچسب‌ها و نمایش راست به چپ

همانطور که در تصاویر فوق ملاحظه می‌کنید، نمایش این ویجت راست به چپ و پیام‌های آن نیز ترجمه شده‌اند.
برای راست به چپ سازی آن مانند قبل تنها کافی است input مرتبط، در یک div با کلاس k-rtl محصور شود:

```
<div class="k-rtl k-header">
            <input name="files" id="files" type="file"  />
        </div>
```

برای بومی سازی پیام‌های آن می‌توان مانند مثال ذیل، خاصیت localization را مقدار دهی کرد:

```
<script type="text/javascript">
        $(function () {
            $("#files").kendoUpload({
                name: "files",
                async: {
                 //...
                },
                //...
                localization: {
                    select: 'انتخاب فایل‌ها برای ارسال',
                    remove: 'حذف فایل',
                    retry: 'سعی مجدد',
                    headerStatusUploading: 'در حال ارسال فایل‌ها',
                    headerStatusUploaded: 'پایان ارسال',
                    cancel: "لغو",
                    uploadSelectedFiles: "ارسال فایل‌ها",
                    dropFilesHere: "فایل‌ها را برای ارسال، کشیده و در اینجا رها کنید",
                    statusUploading: "در حال ارسال",
                    statusUploaded: "ارسال شد",
                    statusWarning: "اخطار",
                    statusFailed: "خطا در ارسال"
                }
            });
        });
    </script>
```

به علاوه متن dropFilesHere به صورت پیش فرض نامرئی است. برای نمایش آن نیاز است CSS موجود را بازنویسی کرد تا em مرتبط مرئی شود:

```
<style type="text/css">
div.k-dropzone {
    border: 1px solid #c5c5c5; /* For Default; Different for each theme */
}

div.k-dropzone em {
    visibility: visible;
}
</style>
```

تغییر قالب نمایش لیست فایل‌ها

لیست فایل‌ها در ویجت kendoUpload دارای یک قالب پیش فرض است که امکان بازنویسی کامل آن وجود دارد. ابتدا نیاز است یک kendo-template را بر این منظور تدارک دید:

```
<script id="fileListTemplate" type="text/x-kendo-template">
        <li class='k-file'>
            <span class='k-progress'></span>
            <span class='k-icon'></span>
            <span class='k-filename' title='#=name#'>#=name# (#=size# bytes)</span>
            <strong class='k-upload-status'></strong>
        </li>
    </script>
```

و سپس برای استفاده از آن خواهیم داشت:

```
<script type="text/javascript">
        $(function () {
            $("#files").kendoUpload({
```

```
                name: "files",
                async: {
                // ...
                },
                // ...
                template: kendo.template($('#fileListTemplate').html()),
                // ...
            });
        });
    </script>
```

در این قالب، مقدار size هر فایل نیز در کنار نام آن نمایش داده می‌شود.

رخدادهای ارسال فایل‌ها

افزونه‌ی kendoUpload در حالت ارسال Ajax ایی فایل‌ها، رخدادهایی مانند شروع به ارسال، موفقیت، پایان، درصد ارسال فایل‌ها و امثال آن‌را نیز به همراه دارد که لیست کامل آن‌ها را در ذیل مشاهده می‌کنید:

```
<script type="text/javascript">
        $(function () {
            $("#files").kendoUpload({
                name: "files",
                async: { // async configuration
                //...
                },
                //...
                localization: {
                },
                cancel: function () {
                    console.log('Cancel Event.');
                },
                complete: function () {
                    console.log('Complete Event.');
                },
                error: function () {
                    console.log('Error uploading file.');
                },
                progress: function (e) {
                    console.log('Uploading file ' + e.percentComplete);
                },
                remove: function () {
                    console.log('File removed.');
                },
                select: function () {
                    console.log('File selected.');
                },
                success: function () {
                    console.log('Upload successful.');
                },
                upload: function (e) {
                    console.log('Upload started.');
                }
            });
        });
    </script>
```

ارسال متادیتای اضافی به همراه فایل‌های ارسالی

فرض کنید می‌خواهید به همراه فایل‌های ارسالی به سرور، پارامتر codeId را نیز ارسال کنید. برای این منظور باید خاصیت e.data رویداد upload را به نحو ذیل مقدار دهی کرد:

```
<script type="text/javascript">
        $(function () {
            $("#files").kendoUpload({
                name: "files",
                async: {
                //...
```

```
                },
                //...
                localization: {
                },
                upload: function (e) {
                    console.log('Upload started.');
                    // Sending metadata to the save action
                    e.data = {
                        codeId: "1234567",
                        param2: 12
                        //, ...
                    };
                }
            });
        });
    </script>
```

سپس در سمت سرور، امضای متد Save بر اساس پارامترهای تعریف شده در سمت کاربر، به نحو ذیل تغییر می‌کند:

```
[HttpPost]
  public ActionResult Save(IEnumerable<HttpPostedFileBase> files, string codeId)
```

فعال سازی ارسال batch

اگر در متد Save سمت سرور یک break point قرار دهید، مشاهده خواهید کرد که به ازای هر فایل موجود در لیست در سمت کاربر، یکبار متد Save فراخوانی می‌شود و عملا متد Save، لیستی از فایل‌ها را در طی یک فراخوانی دریافت نمی‌کند. برای فعال سازی این قابلیت تنها کافی است خاصیت batch را به true تنظیم کنیم:

```
<script type="text/javascript">
        $(function () {
            $("#files").kendoUpload({
                name: "files",
                async: {
                    // ....
                    batch: true
                },
            });
        });
    </script>
```

به این ترتیب دیگر لیست فایل‌ها به صورت مجزا در سمت کاربر نمایش داده نمی‌شود و تمام آن‌ها با یک کاما از هم جدا خواهند شد. همچنین دیگر شاهد نمایش درصد پیشرفت تکی فایل‌ها نیز نخواهیم بود و اینبار درصد پیشرفت کل batch گزارش می‌شود.
در یک چنین حالتی باید دقت داشت که تنظیم maxRequestLength در web.config برنامه الزامی است؛ زیرا به صورت پیش فرض محدودیت 4 مگابایتی ارسال فایل‌ها توسط ASP.NET اعمال می‌شود:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
  <system.web>
    <!-- The request length is in kilobytes, execution timeout is in seconds  -->
    <httpRuntime maxRequestLength="10240" executionTimeout="120" />
  </system.web>

  <system.webServer>
    <security>
      <requestFiltering>
        <!-- The content length is in bytes  -->
        <requestLimits maxAllowedContentLength="10485760"/>
      </requestFiltering>
    </security>
  </system.webServer>
</configuration>
```