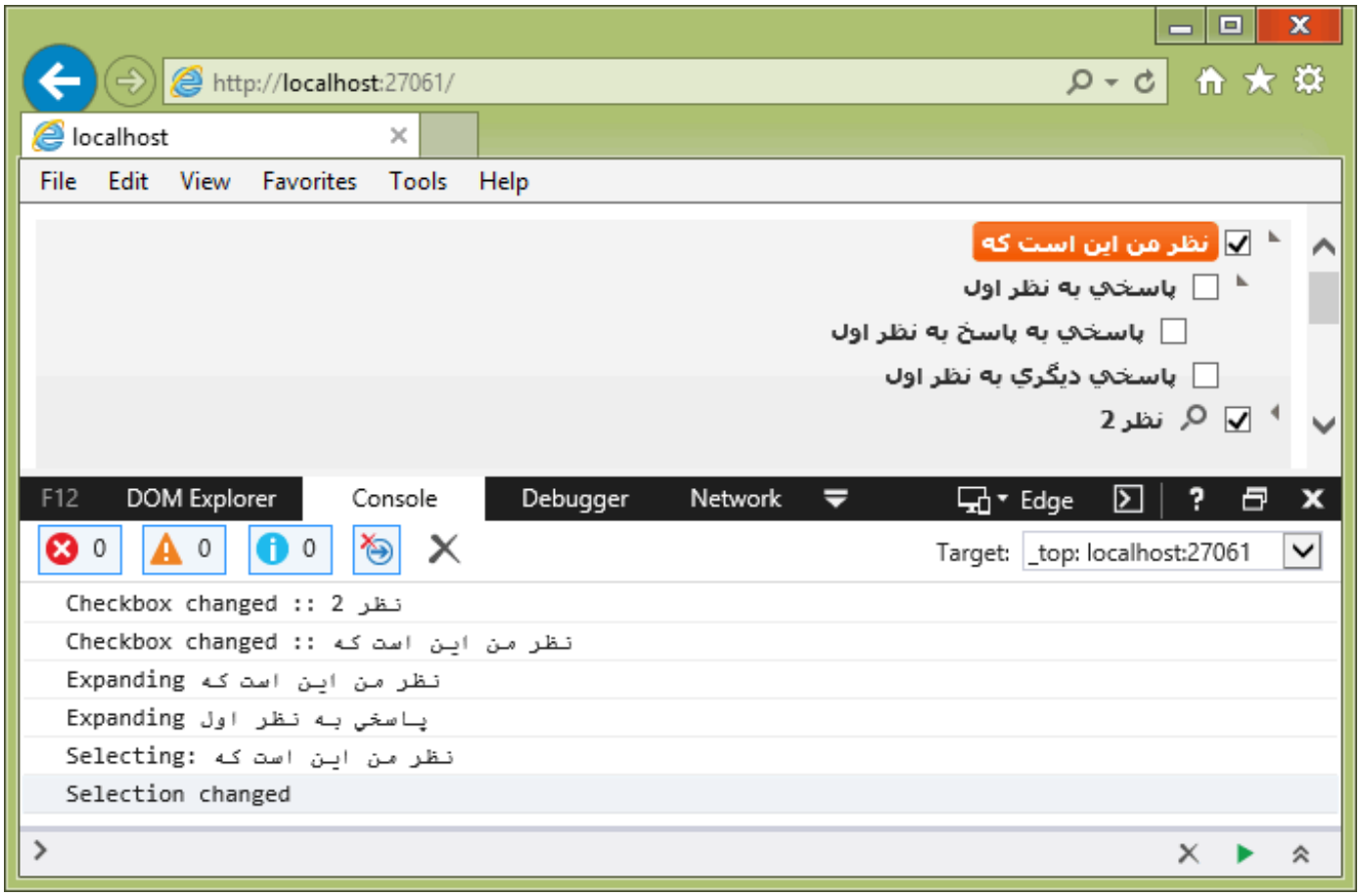
عنوان: **استفاده از Kendo UI TreeView به همراه یک منبع داده راه دور**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۱۵ ۲۱:۵۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery, Kendo UI, Tree

یکی دیگر از ویجت‌های Kendo UI، ویجت نمایش ساختارهای درختی است به نام TreeView. در ادامه قصد داریم با نحوه‌ی نمایش آن، به کمک اطلاعات JSON دریافتی از سرور آشنا شویم.



ساختار مورد نیاز یک Kendo UI Tree View

فرض کنید قصد دارید نظرات تو در توی مطلبی را توسط Kendo UI Tree View نمایش دهید. مدل خود ارجاع دهنده‌ی آن می‌تواند چنین شکلی را داشته باشد:

```
namespace KendoUI11.Models
{
    public class BlogComment
    {
        public int Id { set; get; }

        public string Body { set; get; }

        public int? ParentId { get; set; }

        // مخصوص کندو یو آی هستند
        public bool HasChildren { get; set; }
        public string imageUrl { get; set; }
    }
}
```

سه خاصیت اول این کلاس همواره در تمام کلاس‌های خود ارجاع دهنده حضور دارند؛ شماره ردیف، متن و شماره Id والد احتمالی.
چند خاصیت بعدی مانند HasChildren و imageUrl مخصوص Kendo UI هستند. از imageUrl اختیاری می‌توان جهت نمایش آیکنی در کنار یک آیتم استفاده کرد و HasChildren به این معنا است که آیا گره جاری دارای عناصر فرزندی می‌باشد یا خیر.

تهیه یک منبع داده نمونه

شکل ابتدای مطلب، از طریق منبع داده ذیل تهیه شده‌است:

```
using System.Collections.Generic;

namespace KendoUI11.Models
{
    /// <summary>
    /// منبع داده فرضی جهت سهولت دموی برنامه
    /// </summary>
    public static class BlogCommentsDataSource
    {
        private static readonly IList<BlogComment> _cachedItems;
        static BlogCommentsDataSource()
        {
            _cachedItems = createBlogCommentsDataSource();
        }

        public static IList<BlogComment> LatestComments
        {
            get { return _cachedItems; }
        }

        /// <summary>
        /// هدف صرفا تهیه یک منبع داده آزمایشی ساده تشکیل شده در حافظه است
        /// </summary>
        private static IList<BlogComment> createBlogCommentsDataSource()
        {
            var list = new List<BlogComment>();

            var comment1 = new BlogComment
            {
                Id = 1, Body = "نظر من این است که", HasChildren = true, ParentId = null
            };
            list.Add(comment1);

            var comment12 = new BlogComment
            {
                Id = 2, Body = "پاسخی به نظر اول", HasChildren = true, ParentId = 1
            };
            list.Add(comment12);

            var comment12A = new BlogComment
            {
                Id = 3, Body = "پاسخی دیگری به نظر اول", HasChildren = false, ParentId = 1
            };
            list.Add(comment12A);

            var comment121 = new BlogComment
            {
                Id = 4, Body = "پاسخی به پاسخ به نظر اول", HasChildren = false, ParentId = 2
            };
            list.Add(comment121);

            var comment2 = new BlogComment
            {
                Id = 5, Body = "2 نظر", HasChildren = true, ParentId = null, imageUrl=
"images/search.png"
            };
            list.Add(comment2);

            var comment21 = new BlogComment
            {
                Id = 6, Body = "2 پاسخ به نظر", HasChildren = false, ParentId = 5
            };
            list.Add(comment21);
```

```
            return list;
        }
    }
}
```

در اینجا نحوه‌ی مقدار دهی ParentId و HasChildren را جهت تو در تو سازی اطلاعات، مشاهده می‌کنید.
در این لیست دو رکورد، دارای ParentId مساوی null هستند. از این null بودن‌ها جهت کوئری گرفتن و نمایش ریشه‌های TreeView در ادامه استفاده خواهیم کرد.

بازگشت نظرات با فرمت JSON به سمت کلاینت

در ادامه یک کنترلر ASP.NET MVC را مشاهده می‌کنید که توسط اکشن متد GetBlogComments، رکوردهای مورد نظر را با فرمت JSON به سمت کلاینت ارسال می‌کند:

```
using System.Linq;
using System.Web.Mvc;
using KendoUI11.Models;

namespace KendoUI11.Controllers
{

    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View(); // shows the page.
        }

        [HttpGet]
        public ActionResult GetBlogComments(int? id)
        {
            if (id == null)
            {
                //دریافت ریشه‌ها
                return Json(
                    BlogCommentsDataSource.LatestComments
                        .Where(x => x.ParentId == null) // ریشه‌ها
                        .ToList(),
                    JsonRequestBehavior.AllowGet);
            }
            else
            {
                //دریافت فرزندهای یک ریشه
                return Json(
                    BlogCommentsDataSource.LatestComments
                              .Where(x => x.ParentId == id)
                              .ToList(),
                              JsonRequestBehavior.AllowGet);
            }
        }
    }
}
```

اگر از سمت Kendo UI، مقدار id تنظیم نشود، به معنای درخواست نمایش ریشه‌ها است. در این حالت رکوردها را بر اساس مواردی که دارای ParentId مساوی null هستند، فیلتر خواهیم کرد.
اگر مقدار id به سمت سرور ارسال شود، یعنی کاربر گره و نودی را گشوده‌است. بر این اساس، تمامی فرزندان این گره را یافته و بازگشت می‌دهیم.

کدهای سمت کاربر نمایش Kendo UI Tree View

برای کار با Kendo UI TreeView نیاز است از منبع داده خاصی به نام HierarchicalDataSource به نحو ذیل استفاده کنیم. در قسمت transport آن مشخص می‌کنیم که اطلاعات باید از چه آدرسی خوانده شوند که در اینجا به آدرس اکشن متد GetBlogComments اشاره می‌کند.

همچنین نیاز است مشخص کنیم کدامیک از خواص مدل بازگردانده شده، همان hasChildren است که در مثال فوق دقیقا به همین نام نیز تنظیم شده‌است.

```
<!-- نحوه‌ی راست به چپ سازی -->
<div class="k-rtl k-header demo-section">
    <div id="my-treeview"></div>
</div>

@section JavaScript
{
    <script type="text/javascript">
        $(function () {
            var dataSource = new kendo.data.HierarchicalDataSource({
                transport: {
                    read: {
                        url: "@Url.Action("GetBlogComments", "Home")",
                        dataType: "json",
                        contentType: 'application/json; charset=utf-8',
                        type: 'GET'
                    }
                },
                schema: {
                    model: {
                        id: "Id",
                        hasChildren: "HasChildren"
                    }
                }
            });

            $("#my-treeview").kendoTreeView({
                //استفاده از قالب در صورت نیاز
                template: kendo.template($("#treeview-template").html()),
                checkboxes: {
                    checkChildren: false
                },
                dataSource: dataSource,
                dataTextField: "Body",
                //رخدادها
                select: function (e) { console.log("Selecting: " + this.text(e.node)); },
                check: function (e) { console.log("Checkbox changed :: " + this.text(e.node)); },
                change: function (e) { console.log("Selection changed"); },
                collapse: function (e) { console.log("Collapsing " + this.text(e.node)); },
                expand: function (e) { console.log("Expanding " + this.text(e.node)); }
            });
        });
    </script>

    <script id="treeview-template" type="text/kendo-ui-template">
        <strong> #: item.Body # </strong>
    </script>

    <style scoped>
        .demo-section {
            width: 100%;
            height: 300px;
        }
    </style>
}
```

پس از تنظیم remote data source، اکنون نوبت به تعریف و تنظیم kendoTreeView است.

- در ابتدا به ازای هر ردیف این TreeView، از یک قالب استفاده شده‌است. تعریف این مورد اختیاری است. اگر نیاز به سفارشی سازی نحوه‌ی نمایش هر آیتم را داشتید، می‌توان از قالب‌ها استفاده کرد.
- قسمت checkboxes مشخص می‌کند که آیا نیاز است در کنار هر آیتم یک checkbox نیز نمایش داده شود یا خیر.
- dataSource را به HierarchicalDataSource تنظیم کرده‌ایم.
- dataTextField مشخص می‌کند که کدام فیلد دربرگیرنده‌ی متن هر آیتم TreeView است.
- تعدادی رخداد منتسب به TreeView نیز تنظیم شده‌اند که خروجی آن‌ها را در console تصویر ابتدای بحث مشاهده می‌کنید.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید.