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:

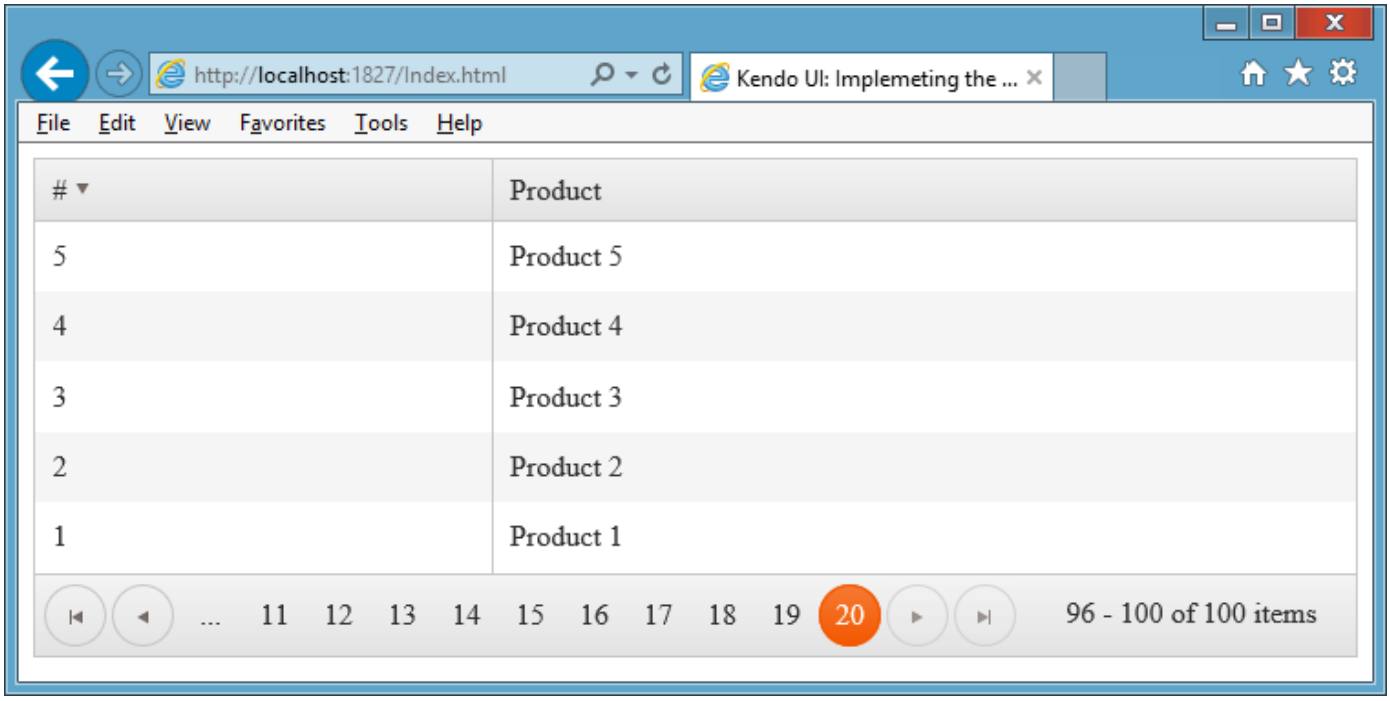
```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="utf-8" />
    <title>Kendo UI: Implemeting the Grid</title>

    <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

نظرات خوانندگان

نویسنده: ژوپیتر
تاریخ: ۱۳۹۳/۱۱/۰۹ ۱۰:۲۷

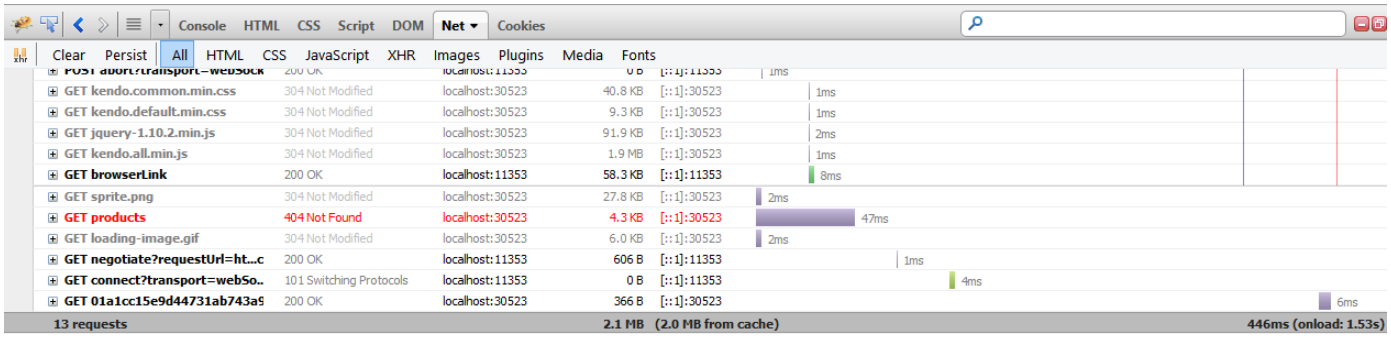
سلام
چرا زمان اجرا به جای نمایش اطلاعات گرید، پیام undefined داده می‌شود؟ بنده از MVC استفاده کردم و کاملا مطابق مقاله مسیریابی و ... را اعمال کردم.

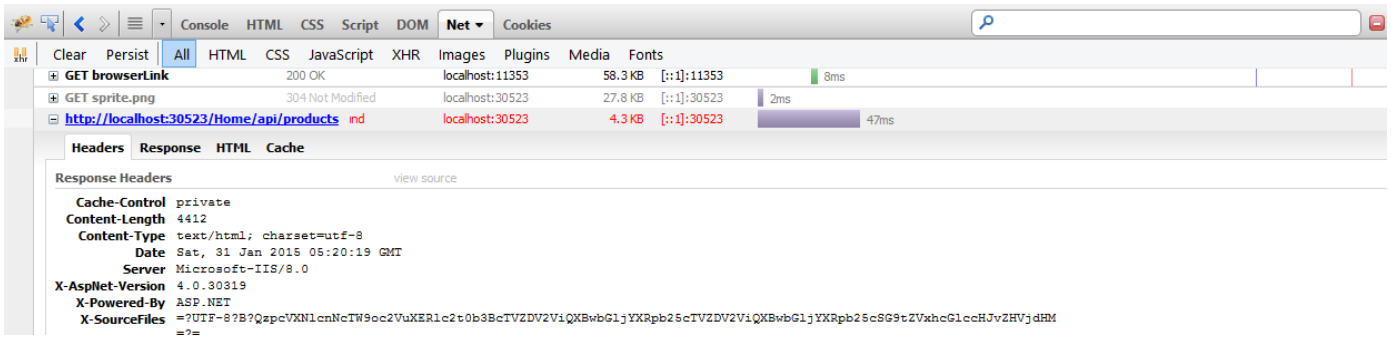
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۱/۰۹ ۱۱:۱۰

روی چه سطری پیام خطا دریافت کردید؟
حین کار با کتابخانه‌های جاوا اسکریپتی باید مدام کنسول developer مرورگر را باز نگه دارید تا بتوانید خطاها را بهتر بررسی و دیباگ کنید.

نویسنده: ژوپیتر
تاریخ: ۱۳۹۳/۱۱/۱۱ ۹:۳

خطا توسط error handler، گرفته می‌شود وقتی این بخش نیز حذف می‌شود مرورگر فقط منتظر دریافت اطلاعات از api/products می‌ماند و خطایی از کد گرفته نمی‌شود.





نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۱/۱۱ ۱۰:۳

خطای 404 به معنای یافتن نشدن مسیر تنظیمی "api/products" :url در برنامه شما است.

مطابق تصویر، مسیر /api/product **home** در حال جستجو است که به ریشه‌ی سایت اشاره نمی‌کند.
- آیا در ASP.NET MVC از یک اکشن متد برای بازگرداندن لیست جی‌سون محصولات استفاده کرده‌اید؟ اگر بله، از مطلب « نحوه صحیح تولید Url در ASP.NET MVC » کمک بگیرید؛ مثلا آدرس آن چنین شکلی را پیدا خواهد کرد:

```
@Url.Action("method_name", "Home")
```

- اگر وب API است در یک برنامه‌ی MVC، از روش زیر استفاده کنید:

```
'@Url.RouteUrl("DefaultApi", new { httproute = "", controller = "products" })'
```

و البته فرض بر این است که مسیریابی DefaultApi پیشتر در برنامه‌ی شما ثبت شده‌است:

```
routes.MapHttpRoute(
    name: "DefaultApi",
    routeTemplate: "api/{controller}/{id}",
    defaults: new { id = RouteParameter.Optional }
);
```

نویسنده: ژوپیتر
تاریخ: ۱۳۹۳/۱۱/۱۱ ۱۱:۸

- ممنون؛ از وب API استفاده شده بود که با راهنمایی شما حل شد. ولی استفاده از خروجی Json کنترلر به‌نظرم بهتر و ساده‌تر اومد. آیا تفاوتی محسوسی بین این دو روش وجود داره؟
- آیا امکان استفاده مستقیم اشیا Strongly Typed هم در توابع این کتابخانه وجود داره؟ (منظورم همون @model به صورت مستقیم یا با واسطه است).

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۱/۱۱ ۱۱:۲۰

- خیر. اگر هدف صرفا بازگشت جی‌سون است، تفاوت خاصی ندارند. فقط Web API به صورت پیش فرض از JSON.NET استفاده می‌کند؛ اطلاعات بیشتر و همچنین با وب فرم‌ها هم سازگار است.
- بله: استفاده ازExpressionها جهت ایجاد Strongly typed view در ASP.NET MVC