عنوان: **صفحه بندی و مرتب سازی خودکار اطلاعات به کمک jqGrid در ASP.NET MVC**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۱۱ ۱۲:۴۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, jqGrid

jqGrid یکی از افزونه‌های بسیار محبوب jQuery جهت نمایش جدول مانند اطلاعات، در سمت کلاینت است. توانمندی‌های آن صرفا به نمایش ستون‌ها و ردیف‌ها خلاصه نمی‌شود. قابلیت‌هایی مانند صفحه بندی، مرتب سازی، جستجو، ویرایش توکار، تولید خودکار صفحات افزودن رکوردها، اعتبارسنجی داده‌ها، گروه بندی، نمایش درختی و غیره را نیز به همراه دارد. همچنین به صورت توکار پشتیبانی از راست به چپ را نیز لحاظ کرده‌است.
مجوز استفاده از فایل‌های جاوا اسکریپتی آن MIT است؛ به این معنا که در هر نوع پروژه‌ای قابل استفاده است. مجوز استفاده از کامپوننت‌های سمت سرور آن که برای نمونه جهت ASP.NET MVC یک سری HTML Helper را تدارک دیده‌اند، تجاری می‌باشد. در ادامه قصد داریم صرفا از فایل‌های JS عمومی آن استفاده کنیم.

دریافت jqGrid

برای دریافت jqGrid می‌توانید به مخزن کد آن، در آدرس https://github.com/tonytomov/jqGrid/releases و یا از طریق NuGet اقدام کنید:

```
PM> Install-Package Trirand.jqGrid
```

استفاده از NuGet بیشتر توصیه می‌شود، زیرا به صورت خودکار وابستگی‌های jQuery و همچنین jQuery UI آن‌را نیز به همراه داشته و نصب خواهد کرد.
از jQuery UI برای تولید صفحات جستجوی بر روی رکوردها و همچنین تولید خودکار صفحات ویرایش و یا افزودن رکوردها استفاده می‌کند. به علاوه آیکن‌ها، قالب و رنگ خود را نیز از jQuery UI دریافت می‌کند. بنابراین اگر قصد تغییر قالب آن‌را داشتید تنها کافی است یک قالب استاندارد دیگر jQuery UI را مورد استفاده قرار دهید.

تنظیمات اولیه فایل Layout سایت

پس از دریافت بسته‌ی نیوگت jqGrid، نیاز است فایل‌های مورد نیاز اصلی آن‌را به شکل زیر به فایل layout پروژه اضافه کرد:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - My ASP.NET Application</title>

    <link href="~/Content/themes/base/jquery.ui.all.css" rel="stylesheet" />
    <link href="~/Content/jquery.jqGrid/ui.jqgrid.css" rel="stylesheet" />
    <link href="~/Content/Site.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <div>
        @RenderBody()
    </div>

    <script src="~/Scripts/jquery-1.7.2.min.js"></script>
    <script src="~/Scripts/jquery-ui-1.8.11.min.js"></script>
    <script src="~/Scripts/i18n/grid.locale-fa.js"></script>
    <script src="~/Scripts/jquery.jqGrid.min.js"></script>

    @RenderSection("Scripts", required: false)
</body>
</html>
```

فایل jquery.ui.all.css شامل تمامی فایل‌های CSS مرتبط با jQuery UI است و نیازی نیست تا سایر فایل‌های آن‌را لحاظ کرد.
این گرید به همراه فایل زبان فارسی grid.locale-fa.js نیز می‌باشد که در کدهای فوق پیوست شده‌است. البته اگر فرصت

کردید نیاز است کمی ترجمه‌های آن بهبود پیدا کنند.

تنظیمات ثانویه site.css

```
.ui-widget {
    font-family: Tahoma !important;
    font-size: 9pt !important;
}

/*how to move jQuery dialog close (X) button from right to left*/
.ui-jqgrid .ui-jqgrid-caption-rtl {
    text-align: center !important;
}

.ui-dialog .ui-dialog-titlebar-close {
    left: .3em !important;
}

.ui-dialog .ui-dialog-title {
    margin: .1em 0 .1em .8em !important;
    direction: rtl !important;
    float: right !important;
}
```

احتمالا تنظیمات قلم‌های jQuery UI و یا jqGrid مدنظر شما نیستند و نیاز به تعویض دارند. در اینجا نحوه‌ی بازنویسی آن‌ها را ملاحظه می‌کنید.
همچنین محل قرار گیری دکمه‌ی بسته شدن دیالوگ‌ها و راست به چپ کردن عناوین آن‌ها نیز در اینجا قید شده‌اند.

مدل برنامه

در ادامه قصد داریم لیستی از محصولات را با ساختار ذیل، توسط jqGrid نمایش دهیم:

```
namespace jqGrid01.Models
{
    public class Product
    {
        public int Id { set; get; }
        public string Name { set; get; }
        public decimal Price { set; get; }
        public bool IsAvailable { set; get; }
    }
}
```

ساختار داده‌ای مورد نیاز توسط jqGrid

jqGrid مستقل است از فناوری سمت سرور. بنابراین هر چند در عنوان بحث ASP.NET MVC ذکر شده‌است، اما از ASP.NET MVC صرفا جهت بازگرداندن خروجی JSON استفاده خواهیم کرد و این مورد در هر فناوری سمت سرور دیگری نیز می‌تواند انجام شود.

```
using System.Collections.Generic;

namespace jqGrid01.Models
{
    public class JqGridData
    {
        public int Total { get; set; }

        public int Page { get; set; }

        public int Records { get; set; }

        public IList<JqGridRowData> Rows { get; set; }

        public object UserData { get; set; }
```

```
    }

    public class JqGridRowData
    {
        public int Id { set; get; }
        public IList<string> RowCells { set; get; }
    }
}
```

خروجی JSON مدنظر توسط jqGrid، یک چنین ساختاری را باید داشته باشد.
Total، نمایانگر تعداد صفحات اطلاعات است. عدد Page، شماره صفحه‌ی جاری است. عدد Records، تعداد کل رکوردهای گزارش را مشخص می‌کند. ساختار ردیف‌های آن نیز تشکیل شده‌است از یک Id به همراه سلول‌هایی که باید با فرمت string، بازگشت داده شوند.
UserData اختیاری است. برای مثال اگر خواستید جمع کل صفحه را در ذیل گرید نمایش دهید، می‌توانید یک anonymous object را در اینجا مقدار دهی کنید. خاصیت‌های آن دقیقا باید با نام خاصیت‌های ستون‌های متناظر، یکی باشند. برای مثال اگر می‌خواهید عددی را در ستون Id، در فوتر گرید نمایش دهید، باید نام خاصیت را Id ذکر کنید.

کدهای سمت کلاینت گرید

در اینجا کدهای کامل سمت کلاینت گرید را ملاحظه می‌کنید:

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

<div dir="rtl" align="center">
    <div id="rsperror"></div>
    <table id="list" cellpadding="0" cellspacing="0"></table>
    <div id="pager" style="text-align:center;"></div>
</div>

@section Scripts
{
    <script type="text/javascript">
        $(document).ready(function () {
            $('#list').jqGrid({
                caption: "آزمایش اول",
                //url from wich data should be requested
                url: '@Url.Action("GetProducts","Home")',
                //type of data
                datatype: 'json',
                jsonReader: {
                    root: "Rows",
                    page: "Page",
                    total: "Total",
                    records: "Records",
                    repeatitems: true,
                    userdata: "UserData",
                    id: "Id",
                    cell: "RowCells"
                },
                //url access method type
                mtype: 'GET',
                //columns names
                colNames: ['شماره', 'نام محصول', 'موجود است', 'قیمت'],
                //columns model
                colModel: [
                { name: 'Id', index: 'Id', align: 'right', width: 50, sorttype: "number" },
                { name: 'Name', index: 'Name', align: 'right', width: 300 },
                { name: 'IsAvailable', index: 'IsAvailable', align: 'center', width: 100, formatter:
'checkbox' },
                { name: 'Price', index: 'Price', align: 'center', width: 100, sorttype: "number" }
                ],
                //pager for grid
                pager: $('#pager'),
                //number of rows per page
                rowNum: 10,
                rowList: [10, 20, 50, 100],
                //initial sorting column
                sortname: 'Id',
```

```
                //initial sorting direction
                sortorder: 'asc',
                //we want to display total records count
                viewrecords: true,
                altRows: true,
                shrinkToFit: true,
                width: 'auto',
                height: 'auto',
                hidegrid: false,
                direction: "rtl",
                gridview: true,
                rownumbers: true,
                footerrow: true,
                userDataOnFooter: true,
                loadComplete: function() {
                    //change alternate rows color
                    $("tr.jqgrow:odd").css("background", "#E0E0E0");
                },
                loadError: function(xhr, st, err) {
                     jQuery("#rsperror").html("Type: " + st + "; Response: " + xhr.status + " " +
xhr.statusText);
                }
                //, loadonce: true
            })
            .jqGrid('navGrid', "#pager",
            {
                edit: false, add: false, del: false, search: false,
                refresh: true
            })
            .jqGrid('navButtonAdd', '#pager',
            {
                caption: "تنظیم نمایش ستون‌ها", title: "Reorder Columns",
                onClickButton: function() {
                     jQuery("#list").jqGrid('columnChooser');
                }
            });
        });
    </script>
}
```

- برای نمایش این گرید، به یک جدول و یک div نیاز است. از جدول با id مساوی list جهت نمایش رکوردهای برنامه استفاده می‌شود. از div با id مساوی pager برای نمایش اطلاعات صفحه بندی و نوار ابزار پایین گرید کمک گرفته خواهد شد.
Div سومی با id مساوی rsperror نیز تعریف شده‌است که از آن جهت نمایش خطاهای بازگشت داده شده از سرور استفاده کرده‌ایم.
- در ادامه نحوه‌ی فراخوانی افزونه‌ی jqGrid را بر روی جدول list ملاحظه می‌کنید.
- خاصیت caption، عنوان نمایش داده شده در بالای گرید را مقدار دهی می‌کند:

آزمایش اول

	شماره	نام محصول	موجود است	قیمت
1	1	نام 1	☑	1000
2	2	نام 2	☐	1001
3	3	نام 3	☑	1002
4	4	نام 4	☐	1003
5	5	نام 5	☑	1004
6	6	نام 6	☐	1005
7	7	نام 7	☑	1006
8	8	نام 8	☐	1007
9	9	نام 9	☑	1008
10	10	نام 10	☐	1009
		جمع صفحه		10045

نمایش 1 - 10 از 500 صفحه 1 از 50 10 تنظیم نمایش ستون‌ها

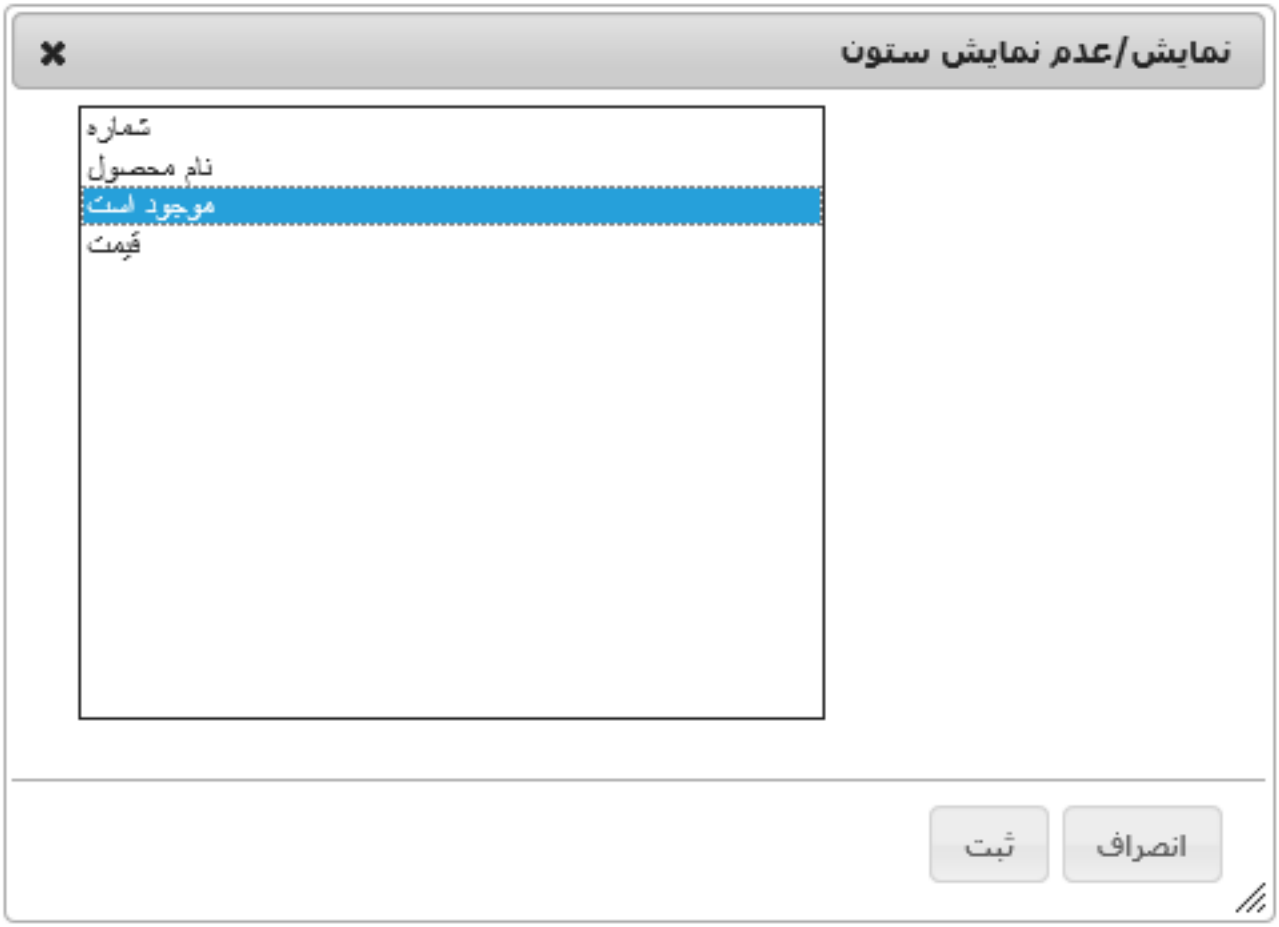
- خاصیت url، به آدرسی اشاره می‌کند که قرار است ساختار JqGridData ایی را که پیشتر در مورد آن بحث کردیم، با فرمت JSON بازگشت دهد. در اینجا برای مثال به یک اکشن متد کنترلری در یک پروژه‌ی ASP.NET MVC اشاره می‌کند.
- datatype را برابر json قرار داده‌ایم. از نوع xml نیز پشتیبانی می‌کند.
- شیء jsonReader را از این جهت مقدار دهی کرده‌ایم تا بتوانیم شیء JqGridData را با اصول نامگذاری دات نت، هماهنگ کنیم. برای درک بهتر این موضوع، فایل jquery.jqGrid.src.js را باز کنید و در آن به دنبال تعریف jsonReader بگردید. به یک چنین مقادیر پیش فرضی خواهید رسید:

```
ts.p.jsonReader = $.extend(true,{
root: "rows",
page: "page",
total: "total",
records: "records",
repeatitems: true,
cell: "cell",
id: "id",
userdata: "userdata",
subgrid: {root:"rows", repeatitems: true, cell:"cell"}
},ts.p.jsonReader);
```

برای مثال سلول‌ها را با نام cell دریافت می‌کند که در شیء JqGridData به RowCells تغییر نام یافته‌است. برای اینکه این تغییر نام‌ها توسط jqGrid پردازش شوند، تنها کافی است jsonReader را مطابق تعاریفی که ملاحظه می‌کنید، مقدار دهی کرد.
- در ادامه mtype به GET تنظیم شده‌است. در اینجا مشخص می‌کنیم که عملیات Ajax ایی دریافت اطلاعات از سرور توسط GET انجام شود یا برای مثال توسط POST.
- خاصیت colNames، معرف نام ستون‌های گرید است. برای اینکه این نام‌ها از راست به چپ نمایش داده شوند، باید خاصیت direction به rtl تنظیم شود.
- colModel آرایه‌ای است که تعاریف ستون‌ها را در بر دارد. مقدار name آن باید یک نام منحصربفرد باشد. از این نام در حین جستجو یا ویرایش اطلاعات استفاده می‌شود. مقدار index نامی است که جهت مرتب سازی اطلاعات، به سرور ارسال می‌شود. تنظیم sorttype در اینجا مشخص می‌کند که آیا به صورت پیش فرض، ستون جاری رشته‌ای مرتب شود یا اینکه برای مثال عددی پردازش گردد. مقادیر مجاز آن text (مقدار پیش فرض)، date، integer، int ، numeric، currency، number، float و datetime

هستند.

- در ستون IsAvailable، مقدار formatter نیز تنظیم شده‌است. در اینجا توسط formatter، نوع bool دریافتی با یک checkbox نمایش داده خواهد شد.
- خاصیت pager به id متناظری در صفحه اشاره می‌کند.
- توسط rowNum مشخص می‌کنیم که در هر صفحه چه تعداد رکورد باید نمایش داده شوند.
- تعداد رکوردهای نمایش داده شده را می‌توان توسط rowList پویا کرد. در اینجا آرایه‌ای را ملاحظه می‌کنید که توسط اعداد آن، کاربر امکان انتخاب صفحاتی مثلا 100 ردیفه را نیز پیدا می‌کند. rowList به صورت یک dropdown در کنار عناصر راهبری صفحه در فوتر گرید ظاهر می‌شود.
- خاصیت sortname، نحوه‌ی مرتب سازی اولیه گرید را مشخص می‌کند.
- خاصیت sortorder، جهت مرتب سازی اولیه‌ی گردید را تنظیم می‌کند.
- viewrecords: تعداد رکوردها را در نوار ابزار پایین گرید نمایش می‌دهد.
- altRows: سبب می‌شود رنگ متن ردیف‌ها یک در میان متفاوت باشد.
- shrinkToFit: به معنای تنظیم خودکار اندازه‌ی سلول‌ها بر اساس اندازه‌ی داده‌ای است که دریافت می‌کنند.
- width: عرض گرید، که در اینجا به auto تنظیم شده‌است.
- height: طول گرید، که در اینجا به auto جهت محاسبه‌ی خودکار، تنظیم شده‌است.
- gridview: برای بالا بردن سرعت نمایشی به true تنظیم شده‌است. در این حالت کل ردیف یکباره درج می‌شود. اگر از subgird یا حالت نمایش درختی استفاده شود، باید این خاصیت را false کرد.
- rownumbers: ستون سمت راست شماره ردیف‌های خودکار را نمایش می‌دهد.
- footerrow: سبب نمایش ردیف فوتر می‌شود.
- userDataOnFooter: سبب خواهد شد تا خاصیت UserData مقدار دهی شده، در ردیف فوتر ظاهر شود.
- loadComplete : یک callback است که زمان پایان بارگذاری صفحه‌ی جاری را مشخص می‌کند. در اینجا با استفاده از jQuery سبب شده‌ایم تا رنگ پس زمینه‌ی ردیف‌ها یک در میان تغییر کند.
- loadError: اگر از سمت سرور خطایی صادر شود، در این callback قابل دریافت خواهد بود.
- در ادامه توسط فراخوانی متد jqGrid با پارامتر navGrid، در ناحیه pager سبب نمایش دکمه refresh شده‌ایم. این دکمه سبب بارگذاری مجدد اطلاعات گردید از سرور می‌شود.
- همچنین به کمک متد jqGrid با پارامتر navButtonAdd در ناحیه pager، سبب نمایش دکمه‌ای که صفحه‌ی انتخاب ستون‌ها را ظاهر می‌کند، خواهیم شد.



پیشنیاز کدهای سمت سرور jqGrid

اگر به تنظیمات گرید دقت کرده باشید، خاصیت index ستون‌ها، نامی است که به سرور، جهت اطلاع رسانی در مورد فیلتر اطلاعات و مرتب سازی مجدد آن‌ها ارسال می‌گردد. این نام، بر اساس کلیک کاربر بر روی ستون‌های موجود، هر بار می‌توان متفاوت باشد. بنابراین بجای if و else نوشتن‌های طولانی جهت مرتب سازی اطلاعات، می‌توان از کتابخانه‌ی معروفی به نام dynamic LINQ استفاده کرد.

```
PM> Install-Package DynamicQuery
```

به این ترتیب می‌توان قسمت orderby را به صورت پویا و با رشته‌ای دریافتی، مقدار دهی کرد.

کدهای سمت سرور بازگشت اطلاعات به فرمت JSON

در کدهای سمت کلاینت، به اکشن متد GetProducts اشاره شده بود. تعاریف کامل آن‌را در ذیل مشاهده می‌کنید:

```
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Web.Mvc;
using jqGrid01.Models;
using jqGrid01.Extensions; // for dynamic OrderBy

namespace jqGrid01.Controllers
{
    public class HomeController : Controller
```

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

        public ActionResult GetProducts(string sidx, string sord, int page, int rows)
        {
            var list = ProductDataSource.LatestProducts;

            var pageIndex = page - 1;
            var pageSize = rows;
            var totalRecords = list.Count;
            var totalPages = (int)Math.Ceiling(totalRecords / (float)pageSize);

            var products = list.AsQueryable()
                               .OrderBy(sidx + " " + sord)
                               .Skip(pageIndex * pageSize)
                               .Take(pageSize)
                               .ToList();

            var jqGridData = new JqGridData
            {
                UserData = new // نمایش در فوتر
                {
                    Name = "جمع صفحه",
                    Price = products.Sum(x => x.Price)
                },
                Total = totalPages,
                Page = page,
                Records = totalRecords,
                Rows = (products.Select(product => new JqGridRowData
                                                   {
                                                       Id = product.Id,
                                                       RowCells = new List<string>
                                                       {

product.Id.ToString(CultureInfo.InvariantCulture),
                                                           product.Name,
                                                           product.IsAvailable.ToString(),

product.Price.ToString(CultureInfo.InvariantCulture)
                                                       }
                                                   })).ToList()
            };
            return Json(jqGridData, JsonRequestBehavior.AllowGet);
        }
    }
}
```

- سطر ProductDataSource.LatestProducts چیزی نیست بجز لیست جنریکی از محصولات.
- امضای متد GetProducts نیز مهم است. دقیقا همین پارامترها با همین نام‌ها از طرف jqGrid به سرور ارسال می‌شوند که توسط آن‌ها ستون مرتب سازی، جهت مرتب سازی، صفحه‌ی جاری و تعداد ردیفی که باید بازگشت داده شوند، قابل دریافت است.
- در این کدها دو قسمت مهم وجود دارند:

الف) متد OrderBy نوشته شده، به صورت پویا عمل می‌کند و از کتابخانه‌ی Dynamic LINQ مایکروسافت بهره می‌برد.

به علاوه توسط Take و Skip کار صفحه بندی و بازگشت تنها بازه‌ای از اطلاعات مورد نیاز، انجام می‌شود.

ب) لیست جنریک محصولات، در نهایت باید با فرمت JqGridData به صورت JSON بازگشت داده شود. نحوه‌ی این Projection را در اینجا می‌توانید ملاحظه کنید.

هر ردیف این لیست، باید تبدیل شود به ردیفی از جنس JqGridRowData، تا توسط jqGrid قابل پردازش گردد.

- توسط مقدار دهی UserData، برچسبی را در ذیل ستون Name و مقداری را در ذیل ستون Price نمایش خواهیم داد.

برای مطالعه‌ی بیشتر

بهترین راهنمای جزئیات این Grid، مستندات آنلاین آن هستند:

http://www.trirand.com/jqgridwiki/doku.php?id=wiki:jqgriddocs

همچنین این مستندات را با فرمت PDF نیز می‌توانید مطالعه کنید:

http://www.trirand.com/blog/jqgrid/downloads/jqgriddocs.pdf

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید

jqGrid01.zip

نظرات خوانندگان

نویسنده: امانی فرد
تاریخ: ۱۳۹۳/۰۴/۱۱ ۱۳:۳۳

امکان دارد شکل گرید را بصورت گرید بوت استرپ نمایش داد ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۱۱ ۱۴:۱۰

همانطور که در بحث عنوان شد، قالب این گرید از jQuery UI تامین می‌شود. بنابراین هر نوع قالب نوشته شده‌ای برای jQuery UI با آن سازگار است؛ مانند:

- jQuery UI Bootstrap
- jqGrid.bootstrap

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۳/۰۴/۱۱ ۱۹:۲۱

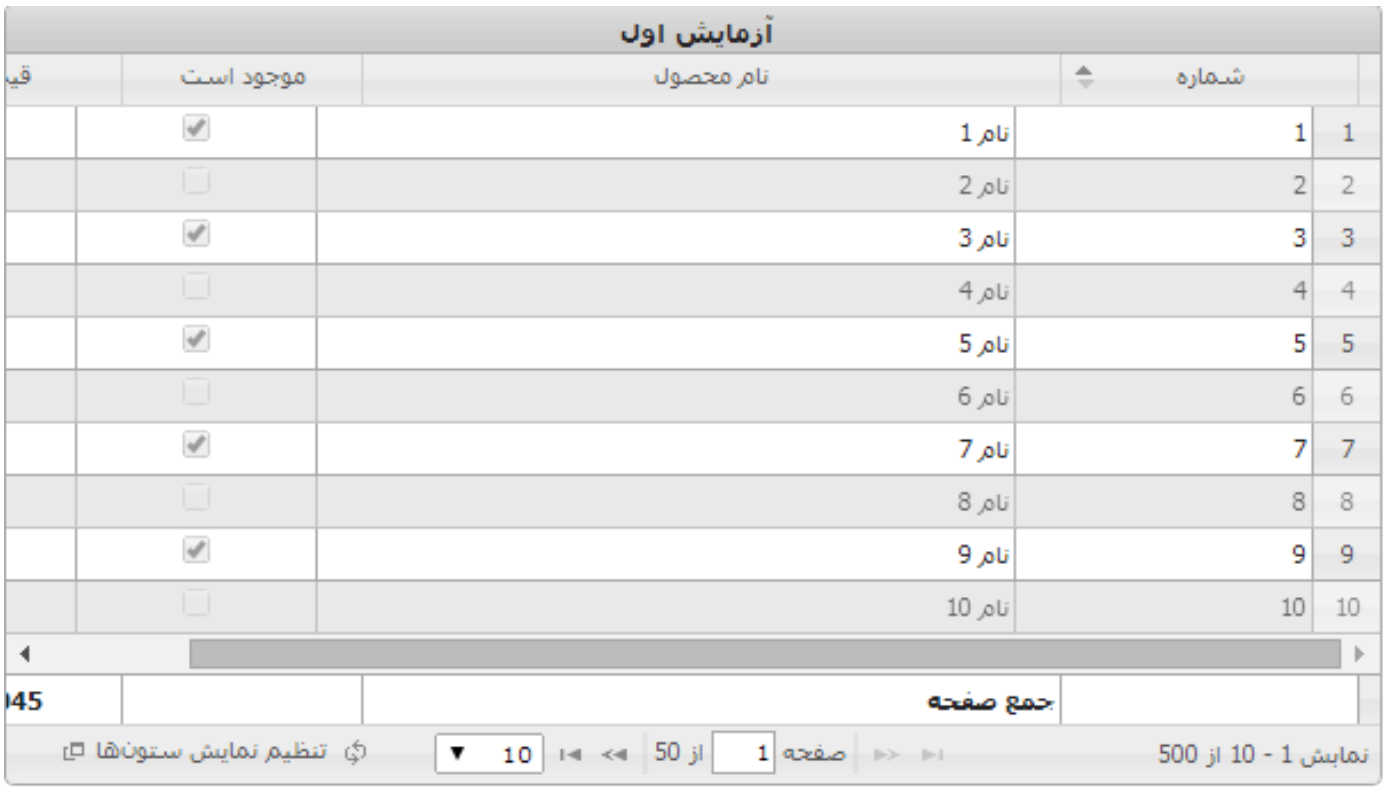
متاسفانه در حالت rtl تغییر اندازه‌ی ستون‌ها با باگ همراه هست.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۱۱ ۲۰:۶

چه مشکلی دقیقا؟

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۳/۰۴/۱۱ ۲۰:۳۰

مثلا در اینجا سایز ستون Id را بیشتر کردم، ردیف‌های متناظرش باهاش هماهنگ نیستند.(با موس اندازه‌ی ستون را تغییر دهیم)



نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۱۱ ۲۰:۵۱

درسته. البته فقط با کروم 35 اینطوری هست. با IE 11 و فایرفاکس 30 تست کردم مشکلی نبود.

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۳/۰۴/۱۱ ۲۳:۴۳

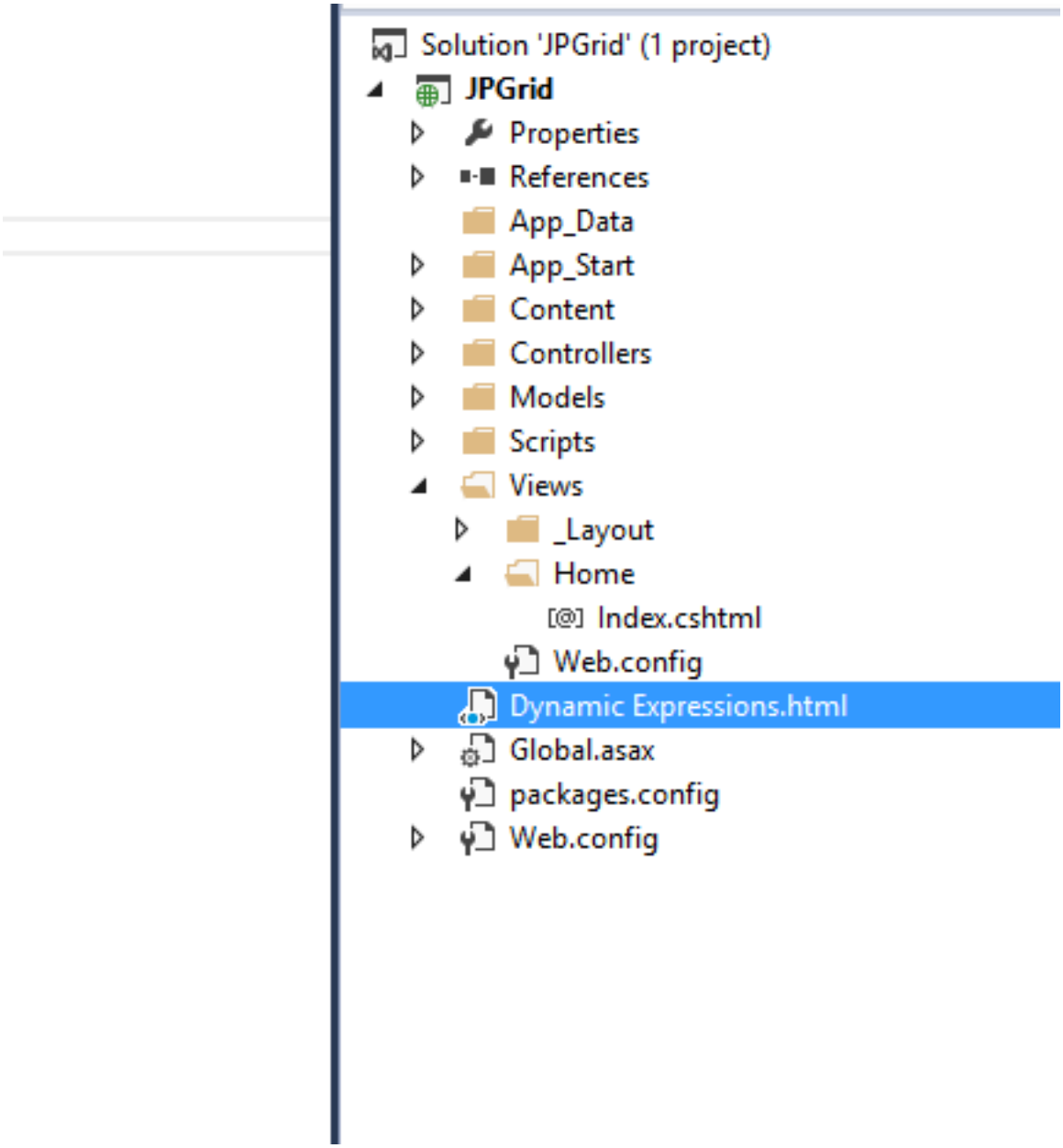
خیلی عالی. خیلی وقته که دنبال یک دیتا گرید می‌گردم تا مشکلی نداشته باشه. برای AngularJS یک Datagrid هست به نام ng-grid که خیلی خوب و کامل هست ولی rtl را خوب ساپورت نمی‌کنه. نسخه‌ی rtl هم که یه نفر براش نوشته باگهای زیادی داره مثل همین تغییر سایز ستون ها.
در کل سوالم این هست که باگ تغییر سایز ستون jqGrid از کروم هست یا خود jqGrid؟ چون می‌خوام از ng-grid کوچ کنم به jq-grid و ببینم ارزش این مهاجرت را داره یا نه.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۱۲ ۰:۱۹

به نظر کروم در هر نگارشی نحوه‌ی تشخیص آن فرق می‌کند و این مساله مشکلاتی را به همراه داشته. بحثی در این مورد .

نویسنده: محمد دلیری
تاریخ: ۱۳۹۳/۰۴/۱۶ ۲۲:۱۳

وقتی این کتاب خانه (Install-Package DynamicQuery) رو اضافه کردم پوشه‌ی به نام Extensions و کلاسی به نام LinqExtensions.cs نساخت فقط چیزی که در تصویر زیر میبینید ساخته ؟ چکار باید بکنم ؟



نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۱۶ ۲۲:۴۵

این بسته، کامپایل شده‌ی آن فایل را تحت عنوان Dynamic.dll به لیست ارجاعات پروژه اضافه می‌کند. فایل Dynamic Expressions.html هم راهنمای آن است. به عبارتی همه چیز آماده است؛ از فضای نام جدید System.Linq.Dynamic آن استفاده کنید.

نویسنده: daniyal
تاریخ: ۱۳۹۳/۰۴/۲۱ ۱۵:۲۷

سلام
آیا می‌شه ستون هایی که عدم نمایش آنها انتخاب شده اند را از طریق یک پارامتر در getproducts گرفت
اگر می‌شه چه طوری ؟

نویسنده: وحید نصیری

تاریخ: ۱۳۹۳/۰۴/۲۱ ۱۸:۵۶

columnChooser رویداد done هم دارد که از آن می‌شود برای آنالیز ستون‌ها استفاده کرد:

```
.jqGrid('navButtonAdd', '#pager',
            {
                caption: "تنظیم نمایش ستون‌ها", title: "Reorder Columns",
                onClickButton: function () {
                    jQuery("#list").jqGrid('columnChooser', {
                        done: function (perm) {
                            if (perm) {
                                $("#list").jqGrid("remapColumns", perm, true, false);
                            }

                            var colModel = $("#list").jqGrid('getGridParam', 'colModel');
                            var hiddenColumns = new Array();
                            for (var i = 0; i < colModel.length; i++) {
                                if (colModel[i].hidden) {
                                    hiddenColumns.push(colModel[i].name);
                                }
                            }

                            $.ajax({
                                type: "POST",
                                url: "@Url.Action("HiddenColumns","Home")",
                                dataType: "json",
                                traditional: true,
                                data: { hiddenColumns: hiddenColumns }
                            });
                        }
                    });
                }
            })
```

متد getGridParam با پارامتر colModel، آرایه‌ای از خواص ستون‌ها را بر می‌گرداند.

```
var colModel = $("#list").jqGrid('getGridParam', 'colModel');
```

در این خواص اگر hidden مساوی true بود، یعنی مخفی شده‌است. در نهایت از این‌ها می‌شود یک آرایه را تشکیل داد و به سرور ارسال کرد:

```
public ActionResult HiddenColumns(string[] hiddenColumns)
        {
            //todo: save it in the DB or cookies or session ....

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

امضای اکشن متدی است که لیست نام ستون‌های مخفی را دریافت می‌کند.

نویسنده: daniyal
تاریخ: ۱۳۹۳/۰۴/۲۲ ۹:۳۰

خیلی مچکرم؛ من می‌خواستم بدونم آیا می‌تونم بدون فرستادن لیست ستون‌های پنهان شده به یک اکشن دیگر و در نتیجه ذخیره کردن آن بر روی کوکی و یا دیتابیس و یا روش‌های دیگر مستقیم در اکشن GetProducts بگیرتش، اگر بخوام شفاف‌تر بگم ، من می‌خوام وقتی کاربر دکمه خروجی پی دی اف رو زد ستون‌های گرید رو کاربر قبلش تنظیم کرده باشه و در پی دی اف مورد نظر ستون‌های حذف شده رو دیگر نبینه همچنین با در نظر گرفتن موارد فیلتر شده در گرید .

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۲۲ ۱۲:۲

برای ارسال پارامترهای دلخواه به سرور از خاصیت postData استفاده کنید:

```
function getHiddenColumnsList() {
            var colModel = $("#list").jqGrid('getGridParam', 'colModel');
            var hiddenColumns = new Array();

            if (!colModel)
                return hiddenColumns;

            for (var i = 0; i < colModel.length; i++) {
                if (colModel[i].hidden) {
                    hiddenColumns.push(colModel[i].name) ;
                }
            }
            return hiddenColumns;
        }

      $(document).ready(function () {
            $('#list').jqGrid({
                    // ...
                    postData: { 'hiddenColumns': function() { return getHiddenColumnsList(); } }
                    // ...
                });
        });
```

سمت سرور در اکشن متد GetProducts، خاصیت جدید hiddenColumns به صورت یک رشته که عناصر آن با کاما از هم جدا شده‌اند، قابل دریافت و آنالیز است.

و برای گزارش‌گیری با Pdf Report در تعریف ستون‌ها (مثلا ستون Id):

```
column.IsVisible(hiddenColumns.Split(',').All(col => col != "Id"));
```

یک نکته: ذکر function در postData ضروری است؛ وگرنه فقط یکبار محاسبه می‌شود.

نویسنده: alireza
تاریخ: ۱۳۹۳/۰۵/۱۰ ۱۵:۳۲

سلام خسته نباشید. فایل‌های پروژه‌هاتون را که دانلود میکنم، وقتی اجرا میکنم ارور زیر را میدهد:

```
Could not load file or assembly 'System.Web.Mvc, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35' or one of its dependencies. The system cannot find the file specified.
```

دلیلش چیه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۱۰ ۱۷:۳۳

- ابتدا به اینترنت وصل شوید.
- سپس در خط فرمان پاورشل نیوگت (^ و ^) دستور زیر را اجرا کنید:

```
PM> update-package -reinstall
```

به این صورت بسته‌های MVC 5 آن به صورت صحیح به پروژه اضافه می‌شوند.

نویسنده: alireza
تاریخ: ۱۳۹۳/۰۵/۱۱ ۱۱:۳

با mvc5 باید باشه ؟ من mvc 4 دارم

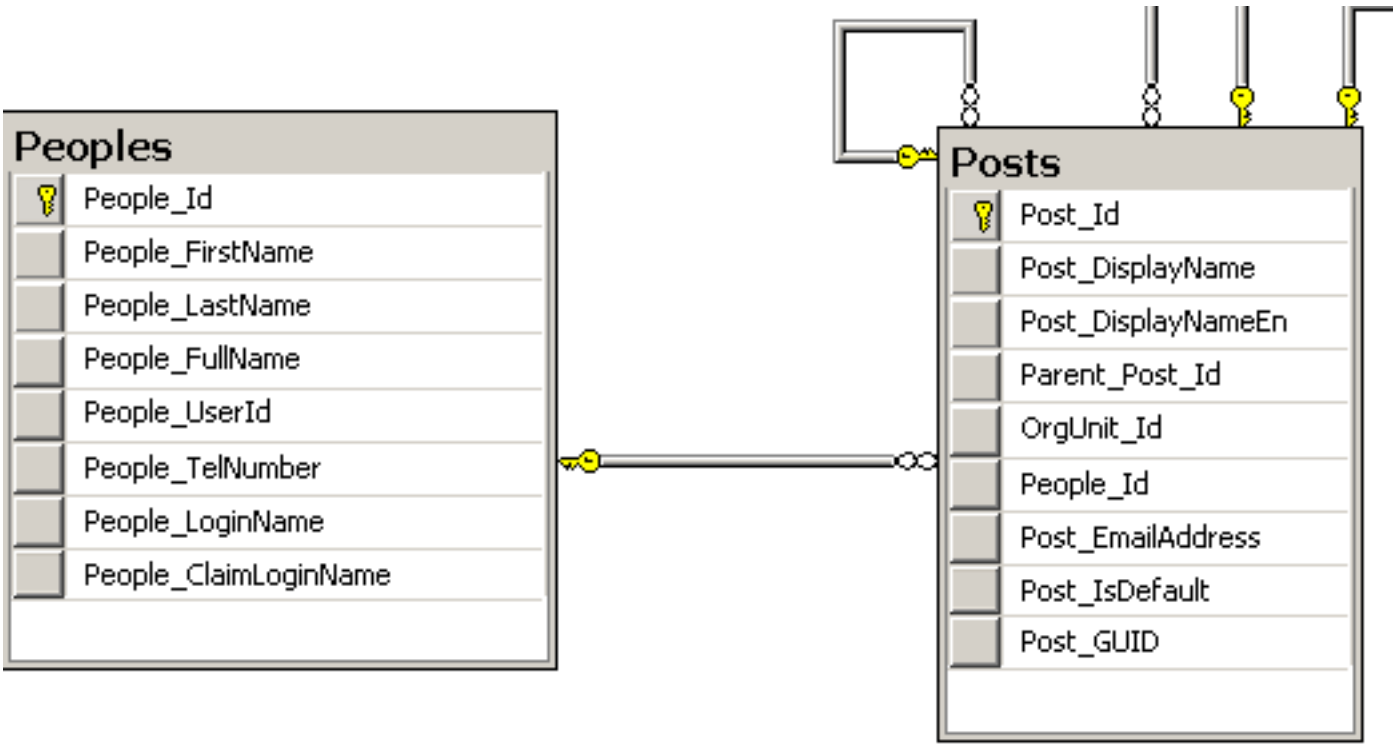
نویسنده: وحید نصیری

تاریخ: ۱۳۹۳/۰۵/۱۱ ۱۱:۲۱

خیر. یک پروژه خالی MVC4 ایجاد کنید. بعد پوشه‌ها و فایل‌های این پروژه را در آن اضافه کنید (منهای فایل‌های کانفیگ)؛ کار می‌کند. jqGrid هیچگونه وابستگی به فناوری‌های سمت سرور ندارد. با وب فرم‌ها قابل استفاده است. با PHP هم قابل استفاده است. در اینجا از MVC فقط برای بازگشت اطلاعات مورد نیاز آن به فرمت JSON استفاده شده‌است. در این مثال خاص، کاربرد ویژه‌ی دیگری ندارد. می‌شد بجای آن از Web API هم استفاده کرد.

نویسنده: ناصر پورعلی
تاریخ: ۱۳۹۳/۰۵/۱۴ ۱۴:۲۲

سلام؛ از مثالی که زدید دارم استفاده میکنم، در کلاس ReflectionHelper متد FindFieldType موقعی که از جداول خود ارجاع دهنده استفاده میکنیم(در اینجا جدول پست)، خطا stackoverflow می‌گیرم، کجاش رو باید چک کنیم که فراخوانی بی نهایت اتفاق نیفته؟ مگه فیلد dumplevel همچین کاری انجام نمیده؟
data model م به این صورته :



نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۱۴ ۱۵:۶

«مگه فیلد dumplevel همچین کاری انجام نمیده؟»
بله. dump level از stack overflow جلوگیری می‌کند. اما جای دیگری است که stack overflow واقعی رخ می‌دهد:
« بررسی خطای Circular References در ASP.NET MVC Json Serialization »

نویسنده: بهاره
تاریخ: ۱۳۹۳/۰۵/۲۰ ۱۶:۰

سلام
اگه بخوام نام ستون‌ها رو از روی مدل یا ویو بگ بخونم و دستی ننویسم باید چیکار کنم. من از مدل Database First استفاده

می‌کنم. چطور میتونم اینکار رو انجام بدم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۲۰ ۱۷:۱۴

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

نویسنده: احسان شیروان
تاریخ: ۱۳۹۳/۰۶/۱۲ ۲۱:۷

با سلام و خسته نباشید من یه مشکلی داشتم می‌خواستم در صورت امکان راهنمایی نمایید :
توی یه صفحه می‌خوام لود شدن داده‌های این گرید با رویداد کلیک یه دکمه اتفاق بیفته یه مقدار سرچ هم زدم که تقریبا همشون می‌گفتن توی رویداد کلیک بنویسم :

```
jQuery("#list").trigger("reloadGrid");
```

اما با این هم جواب نداد و اصلا ActionMethod فراخوانی نمیشه ممنون میشم راهنمایی نمایید

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۶/۱۲ ۲۱:۲۴

reloadGrid برای حالتی است که Grid در صفحه نمایش داده شده و موجود است. برای نمایش یک گرید با کلیک بر روی یک دکمه، کل کدهای داخل document.ready مثال فوق را داخل یک متد جداگانه قرار دهید و سپس آن‌را مستقلا فراخوانی کنید.
document.ready یعنی به محض آماده شدن DOM این اطلاعات را اجرا کن.

نویسنده: احسان شیروان
تاریخ: ۱۳۹۳/۰۶/۱۲ ۲۱:۳۹

راستش اولش هم همین کارو کردم جواب نگرفتم تستش هم کردم مطمئنم که رویداد کلیک و تابع مذکور فراخوانی میشه اما متد سرور فراخوانی نمیشه

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۶/۱۲ ۲۲:۳۸

پس از آزمایش مشکلی مشاهده نشد.
همانطور که عنوان شد، کل اطلاعات داخل document ready به داخل یک متد مجزا منتقل شد:

```
function loadGrid() {
            $('#list').jqGrid({
                caption: "آزمایش اول",
                // .....
            });
        }
```

و سپس فراخوانی آن توسط یک دکمه:

```
<button onclick="loadGrid()">Load Grid</button>
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۶/۲۰ ۱۳:۹

یک نکته‌ی تکمیلی

نسخه‌ی جدید و زنده‌ی dynamic LINQ در اینجا نگهداری می‌شود.

```
PM> Install-Package System.Linq.Dynamic.Library
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۷/۰۲ ۱۴:۲۱

برای رفع این مشکل فقط کافی است .ui-jqgrid .ui-jqgrid-hbox-rtl را در فایل css آن یافته و تبدیل کنید به:

```
.ui-jqgrid .ui-jqgrid-hbox-rtl {float: right; padding-left: 0;}
```

عنوان: **فرمت کردن اطلاعات نمایش داده شده به کمک jqGrid در ASP.NET MVC**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۱۲ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, jqGrid

پیشنیاز این بحث مطالعه‌ی مطلب « صفحه بندی و مرتب سازی خودکار اطلاعات به کمک jqGrid در ASP.NET MVC » است و در اینجا جهت کوتاه شدن بحث، صرفا به تغییرات مورد نیاز جهت اعمال بر روی مثال اول اکتفاء خواهد شد.

صورت مساله

می‌خواهیم اطلاعات نمایش داده شده در گرید را به نحوی فرمت کنیم که
الف) اگر id ردیف مساوی 5 بود، رنگ و پس زمینه‌ی آن تغییر کند.
ب) نام محصول، به جزئیات آن لینک شود و این اطلاعات توسط یک jQuery UI Dialog نمایش داده شود.
ج) عدد قیمت با سه رقم جدا کننده همراه باشد.

تکمیل مدل برنامه

مدل قسمت اول صرفا یک محصول بود. مدل قسمت جاری، اطلاعات تولید/تامین کننده آن‌را توسط کلاس Supplier نیز به همراه دارد:

```
namespace jqGrid02.Models
{
    public class Product
    {
        public int Id { set; get; }
        public string Name { set; get; }
```

```
        public decimal Price { set; get; }
        public Supplier Supplier { set; get; }
    }

    public class Supplier
    {
public int Id { set; get; }
        public string CompanyName { set; get; }
        public string Address { set; get; }
        public string PostalCode { set; get; }
        public string City { set; get; }
        public string Country { set; get; }
        public string Phone { set; get; }
        public string HomePage { set; get; }
    }
}
```

کدهای سمت سرور

کدهای سمت سرور مانند متد GetProducts به همراه صفحه بندی و مرتب سازی پویای آن دقیقا مانند قسمت قبل است.
در اینجا فقط یک اکشن متد جدید جهت بازگشت اطلاعات تولید کننده‌ای مشخص با فرمت JSON، اضافه شده‌است:

```
public ActionResult GetGetSupplierData(int id)
        {
            var list = ProductDataSource.LatestProducts;
            var product = list.FirstOrDefault(x => x.Id == id);
            if (product == null)
                return Json(null, JsonRequestBehavior.AllowGet);

            return Json(new
                        {
                            product.Supplier.CompanyName,
                            product.Supplier.Address,
                            product.Supplier.PostalCode,
                            product.Supplier.City,
                            product.Supplier.Country,
                            product.Supplier.Phone,
                            product.Supplier.HomePage
                        }, JsonRequestBehavior.AllowGet);
        }
```

کدهای سمت کلاینت

صفحه دیالوگی که قرار است اطلاعات تولید کننده را نمایش دهد، یک چنین ساختاری دارد:

```
<div dir="rtl" id="supplierDialog">
    <span id="CompanyName"></span><br /><br />
    <span id="Address"></span><br />
    <span id="PostalCode"></span>, <span id="City"></span><br />
    <span id="Country"></span><br /><br />
    <span id="Phone"></span><br />
    <span id="HomePage"></span>
</div>
```

و تغییرات گرید برنامه به شرح زیر است:

```
<script type="text/javascript">
        function showSupplierDialog(linkElement, supplierId) {
            //request json data
            $.getJSON('@Url.Action("GetGetSupplierData","Home")', { id: supplierId }, function (data) {
                //set values in dialog
                for (var property in data) {
                    if (data.hasOwnProperty(property)) {
                        $('#' + property).text(data[property]);
                    }
                }

                //get link position
                var linkPosition = $(linkElement).offset();
```

```
                $('#supplierDialog').dialog('option', 'position', [linkPosition.left,
linkPosition.top]);
                //open dialog
                $('#supplierDialog').dialog('open');
            });
        }

        $(document).ready(function () {

            $('#supplierDialog').dialog({
                 autoOpen: false, bgiframe: true, resizable: false, title: 'تولید کننده'
            });

            $('#list').jqGrid({
                // .... مانند قبل
                colNames: ['قیمت' ,'نام محصول' ,'شماره'],
                //columns model
                colModel: [
                    {
                        name: 'Id', index: 'Id', align: 'right', width: 20,
                        formatter: function (cellvalue, options, rowObject) {
                            var cellValueInt = parseInt(cellvalue);
                            if (cellValueInt == 5) {
                                return "<span style='background: brown; color: yellow'>" + cellvalue +
"</span>";
                            }
                            return cellvalue;
                        }
                    },
                    {
                        name: 'Name', index: 'Name', align: 'right', width: 300,
                        formatter: function (cellvalue, options, rowObject) {
                            return "<a href='#' onclick='showSupplierDialog(this, " + rowObject[0] +
");'>" + cellvalue + "</a>";
                        }
                    },
                    {
                        name: 'Price', index: 'Price', align: 'center', width: 50,
                        formatter: 'currency',
                        formatoptions:
                        {
                            decimalSeparator: '.', thousandsSeparator: ',', decimalPlaces: 2, prefix:
'$'
                        }
                    }
                ],
                // .... مانند قبل
            });
        });
    </script>
```

- همانطور که ملاحظه می‌کنید، توسط خاصیت formatter می‌توان عناصر در حال نمایش را فرمت کرد و بر روی نحوه‌ی نمایش نهایی آن‌ها تاثیرگذار بود.

در حالت ستون Id، از یک formatter سفارشی استفاده شده‌است. در اینجا این فرمت کننده به صورت یک callback عمل کرده و پیش از رندر نهایی اطلاعات، مقدار سلول جاری را توسط cellvalue در اختیار ما قرار می‌دهد. در این بین هر نوع فرمتی را که نیاز است می‌توان اعمال کرد و سپس یک رشته را بازگشت می‌دهیم. این رشته در سلول جاری درج خواهد شد.

- اگر مانند ستون Name، نیاز به مقادیر سایر سلول‌ها نیز وجود داشت، می‌توان از آرایه‌ی rowObject استفاده کرد. برای مثال در این حالت، یک لینک که کلیک بر روی آن سبب فراخوانی تابع showSupplierDialog می‌شود، در سلول‌های ستون Name درج خواهند شد. اولین rowObject که در اینجا مورد استفاده است، به ستون اول یا همان Id محصول اشاره می‌کند.

- در ستون Price از یک سری formatter از پیش تعریف شده استفاده شده‌است. نمونه‌ای از آن را در قسمت اول در ستون نمایش وضعیت موجود بودن محصول با تنظیم formatter: checkbox مشاهده کرده‌اید. در اینجا از یک formatter توکار دیگر به نام currency برای کار با مقادیر پولی استفاده شده‌است به همراه تنظیمات خاص آن.

- متد showSupplierDialog طوری تنظیم شده‌است که پس از دریافت Id یک محصول، آن‌را به سرور ارسال کرده و مشخصات تولید کننده‌ی آن‌را با فرمت JSON دریافت می‌کند. سپس در حلقه‌ای که مشاهده می‌کنید، خواص شیء جاوا اسکریپتی دریافتی استخراج و به spanهای supplierDialog انتساب داده می‌شوند. جهت سهولت کار، Id این spanها دقیقا مساوی Id خواص شیء دریافتی از سرور، درنظر گرفته شده‌اند.

- در مورد راست به چپ نمایش داده شدن عنوان دیالوگ، تغییرات CSS ایی لازم است که در قسمت اول بیان شدند.

برای مطالعه بیشتر

لیست کامل فرمت کننده‌های توکار

فرمت کننده‌های سفارشی

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید

jqGrid02.zip

عنوان: **فعال سازی و پردازش جستجوی پویای jqGrid در ASP.NET MVC**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۱۳ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, jqGrid

پیشنیاز این بحث مطالعه‌ی مطلب « صفحه بندی و مرتب سازی خودکار اطلاعات به کمک jqGrid در ASP.NET MVC » است و در اینجا جهت کوتاه شدن بحث، صرفا به تغییرات مورد نیاز جهت اعمال بر روی مثال اول اکتفاء خواهد شد.

تغییرات مورد نیاز سمت کلاینت جهت فعال سازی جستجو در jqGrid

در سمت کلاینت، در حین تعریف ستون‌ها، ابتدا باید توسط مقدار دهی خاصیت search، ستون‌های مشارکت کننده‌ی در حین جستجو را مشخص کرد:

```
colModel: [
                    {
                        name: 'Name', index: 'Name', align: 'right', width: 200,
                        search: true, stype: 'text', searchoptions: { sopt: searchOptions }
                    },
                    {
                        name: 'Supplier.Id', index: 'Supplier.Id', align: 'right', width: 200,
                        search: true, stype: 'select', edittype: 'select', searchOperators: true,
                        searchoptions:
                        {
                            sopt: searchOptions, dataUrl: '@Url.Action("SuppliersSelect","Home")'
                        }
                    }
                ],
```

- برای نمونه در اینجا search: true، جهت دو ستون نام محصول و نام تولید کننده، تنظیم شده‌اند.
- stype، روش مقایسه‌ی مقادیر را مشخص می‌کند. مقدار پیش فرض آن text است و مقادیری مانند int، integer، float، number، numeric، date و datetime را می‌پذیرد.
- searchoptions برای تنظیم جزئیات نحوه‌ی جستجوی بر روی فیلدها بکار می‌رود. توسط sopt می‌توان آرایه‌ای با مقادیر ذیل را مقدار دهی کرد:

```
var searchOptions = ['eq', 'ne', 'lt', 'le', 'gt', 'ge', 'bw', 'bn', 'in', 'ni', 'ew', 'en', 'cn',
'nc'];
```

eq به معنای مساوی است، ne، مساوی نیست، lt کمتر و به همین ترتیب.
البته باید دقت داشت که آرایه فوق کاملترین حالت ممکن است و ضرورتی ندارد تمام حالات را برای یک فیلد تعریف کرد. چون برای مثال جستجو cn یا contains برای مقادیر رشته‌ای معنا دارد و نه سایر حالات.
- searchoptions گزینه‌های دیگری را نیز می‌تواند شامل شود. برای مثال در حین نمایش جستجوی داخل ردیفی یا صفحه‌ی دیالوگ مخصوص آن، قصد داریم فیلد نام تولید کننده را توسط یک drop down نمایش دهیم و نه یک text box پیش فرض. برای این منظور dataUrl مقدار دهی شده‌است.
SuppliersSelect آن به اکشن متد ذیل اشاره می‌کند که لیست تولید کنندگان را با فرمت لیستی از SelectListItemها به یک partial view تولید کننده‌ی دراپ داون ارسال می‌کند:

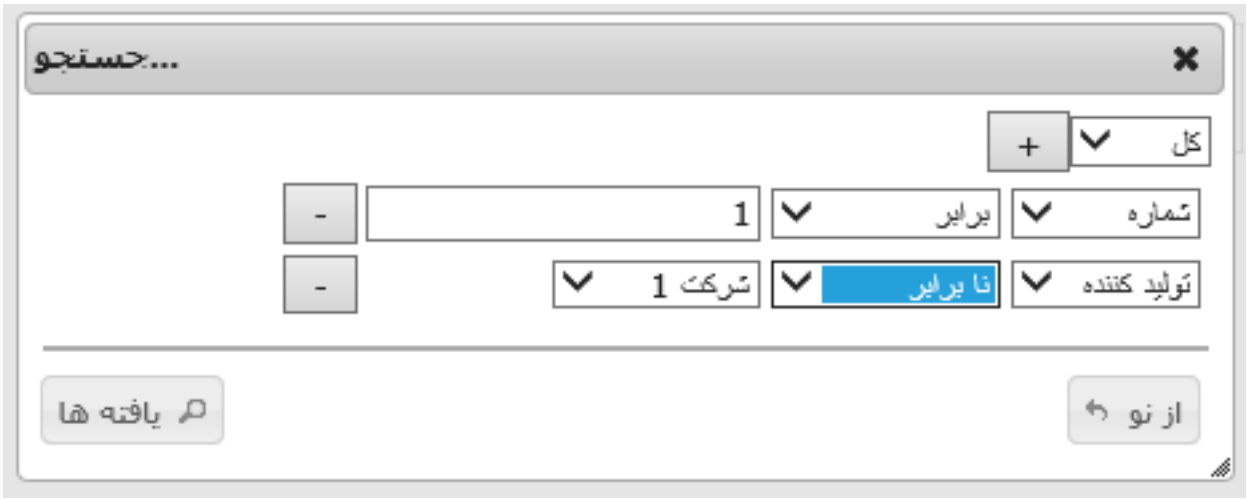
```
public ActionResult SuppliersSelect()
        {
            var list = ProductDataSource.LatestProducts;
            var suppliers = list.Select(x => new SelectListItem
            {
                Text = x.Supplier.CompanyName,
                Value = x.Supplier.Id.ToString(CultureInfo.InvariantCulture)
            }).ToList();
            return PartialView("_SelectPartial", suppliers);
        }
```

و محتوای فایل _SelectPartial نیز به صورت ذیل است:

```
@model IList<SelectListItem>
@Html.DropDownList("srch", Model)
```

در این حالت با نمایش صفحه‌ی جستجو و انتخاب فیلد نام تولید کننده، به صورت خودکار یک dorp down پویا نمایش داده خواهد شد.

- اگر دقت کرده باشید، نام فیلد تولید کننده با Supplier.Id مقدار دهی شده‌است. علت اینجا است که در زمان استفاده از drop down، مقدار Id آیتم انتخابی، به سرور ارسال می‌شود. به همین جهت کار کردن پویا با Supplier.Id ساده‌تر خواهد بود.



- برای نمایش دیالوگ و یا نوار ابزار توکار جستجو، می‌توان دکمه‌هایی را به فوتر گرید اضافه کرد:


```
$(document).ready(function () {
            $('#list').jqGrid({
              // ... مانند قبل و توضیحات فوق
            })
            .jqGrid('navGrid', '#pager',
                { add: false, edit: false, del: false },
                {},  // default settings for edit
                {},  // default settings for add
                {},  // delete instead that del:false we need this
                {
                    // search options
                    multipleSearch: true,
                    closeOnEscape: true,
                    closeAfterSearch: true,
                    ignoreCase: true
                })
            .jqGrid('navButtonAdd', "#pager", {
                caption: "نوار ابزار جستجو", title: "Search Toolbar", buttonicon: 'ui-icon-search',
                onClickButton: function () {
                    toolbarSearching();
                }
            });
        });


        function toolbarSearching() {
            $('#list').filterToolbar({
                groupOp: 'OR',
                defaultSearch: "cn",
                autosearch: true,
                searchOnEnter: true,
                searchOperators: true, // فعال سازی منوی اپراتورها
                stringResult : true // وجود این سطر سبب می‌شود تا اپراتورها به سرور ارسال شوند
            });
        };

        function singleSearching() {
            $('#list').searchGrid({
                closeAfterSearch: true
            });
        };

        function advancedSearching() {
            $('#list').searchGrid({
                multipleSearch: true,
                closeAfterSearch: true
            });
        };
```

در اینجا نکات ذیل قابل توجه هستند:

- با استفاده از متد jqGrid و پارامتر navGrid، در ناحیه‌ی pager گرید، تنظیمات جستجو را فعال کرده‌ایم.

multipleSearch به معنای امکان جستجوی همزمان بر روی بیش از یک فیلد است. closeOnEscape سبب بسته شدن صفحه‌ی دیالوگ جستجو با فشردن دکمه‌ی ESC می‌شود. اگر closeAfterSearch به true تنظیم نشود، صفحه‌ی دیالوگ جستجو پس از جستجو، در صفحه باقی مانده و بسته نخواهد شد.

- این دکمه‌ی جستجو، جزو موارد توکار jqGrid است. اگر قصد داشته باشیم یک دکمه‌ی سفارشی دیگر را نیز اضافه کنیم، مجددا از متد jqGrid با پارامتر navButtonAdd در ناحیه‌ی pager استفاده خواهیم کرد. کلیک بر روی آن سبب اجرای متد

toolbarSearching می‌شود.

در اینجا حداقل سه نوع جستجو را می‌توان فعال کرد:
- filterToolbar که سبب نمایش نوار ابزار جستجو، دقیقا بالای ستون‌های جدول می‌شود.
- searchGrid که صفحه‌ی دیالوگ مستقلی را جهت جستجو به صورت پویا تولید می‌کند. اگر خاصیت multipleSearch آن به true تنظیم نشود، این جستجو هربار تنها بر روی یک فیلد قابل انجام خواهد بود و برعکس.
- در حالت جستجوی نوار ابزاری، اگر خواص searchOperators و stringResult به true تنظیم شوند، مانند تصویر ذیل، به ازای هر ستون می‌توان از عملگرهای مختلفی استفاده کرد. در غیراینصورت جستجوی انجام شده بر اساس groupOp و defaultSearch پیش فرض انجام می‌شود. یعنی And یا Or تمام موارد تنها در حالت مثلا contains یا تساوی و امثال آن و نه حالت پیشرفته‌ی انتخاب عملگرها توسط کاربر.

یک نکته

اگر می‌خواهید صفحه‌ی جستجو در وسط صفحه ظاهر شود، می‌توانید از تنظیمات CSS ذیل استفاده کنید:

```
/* align center search popup in jqgrid */
.ui-jqdialog {
    display: none;
    width: 300px;
    position: absolute;
    padding: .2em;
    font-size: 11px;
    overflow: visible;
    left: 30% !important;
    top: 40% !important;
}
```

پردازش سمت سرور جستجوی پویای jqGrid

کدهای سمت سرور، با کدهای استفاده از dynamic LINQ مایکروسافت یکی است. با این تفاوت که اینبار قسمت where این کوئری نیز پویا می‌باشد. پیشتر قسمت order by را پویا پردازش کرده بودیم. برای ساخت where پویا که در dynamic LINQ به خوبی

پشتیبانی می‌شود، باید ابتدا ساختار اطلاعات ارسالی به سرور را آنالیز کنیم:

```
//single field search
//_search=true&nd=1403935889318&rows=10&page=1&sidx=Id&sord=asc&searchField=Id&searchString=4444&search
Oper=eq&filters=

//multi-field search
//_search=true&nd=1403935941367&rows=10&page=1&sidx=Id&sord=asc&filters=%7B%22groupOp%22%3A%22AND%22%2C
%22rules%22%3A%5B%7B%22field%22%3A%22Id%22%2C%22op%22%3A%22eq%22%2C%22data%22%3A%2244%22%7D%2C%7B%22fie
ld%22%3A%22SupplierID%22%2C%22op%22%3A%22eq%22%2C%22data%22%3A%221%22%7D%5D%7D&searchField=&searchStrin
g=&searchOper=
// filters ->
{"groupOp":"AND","rules":[{"field":"All","op":"cn","data":"fffff"},{"field":"Price","op":"bn","data":"f
fff"}]}

//toolbar search
//_search=true&nd=1403935593036&rows=10&page=1&sidx=Id&sord=asc&Id=2&Name=333&SupplierID=1&CategoryID=1
&Price=44
```

در اینجا ساختار ارسالی به سرور را در سه حالت مختلف جستجوی پویای jqGrid، ملاحظه می‌کنید:

- در تمام این حالات پارامتر _search مساوی true است (تفاوت آن با درخواست اطلاعات معمولی).
- در حالت جستجوی نوار ابزاری، اگر گزینه‌های searchOperators و stringResult به true تنظیم نشوند، حالت toolbar search فوق را شاهد خواهیم بود. در غیراینصورت به حالت multi-field search سوئیچ می‌شود.
- در حالت جستجوی تک فیلدی توسط صفحه دیالوگ جستجوی jqGrid، فیلد در حال جستجو توسط searchField و مقدار آن توسط searchString به سرور ارسال شده‌اند. مابقی پارامترها نال هستند.
- در حالت جستجوی چند فیلدی توسط صفحه دیالوگ جستجوی jqGrid، اینبار filters مقدار دهی شده‌است و سایر پارامترها نال هستند. مقدار filters ارسالی، در حقیقت یک شیء JSON است با ساختار کلی ذیل:

```
{ "groupOp": "AND",
               "groups" : [
                 { "groupOp": "OR",
                     "rules": [
                         { "field": "name", "op": "eq", "data": "England" },
                         { "field": "id", "op": "le", "data": "5"}
                      ]
                 }
               ],
               "rules": [
                 { "field": "name", "op": "eq", "data": "Romania" },
                 { "field": "id", "op": "le", "data": "1"}
               ]
        }
```

که می‌توان چنین ساختاری را برای آن متصور شد:

```
public class SearchFilter
    {
        public string groupOp { set; get; }
        public List<SearchGroup> groups { set; get; }
        public List<SearchRule> rules { set; get; }
    }

    public class SearchRule
    {
        public string field { set; get; }
        public string op { set; get; }
        public string data { set; get; }

        public override string ToString()
        {
            return string.Format("'{0}' {1} '{2}'", field, op, data);
        }
    }

    public class SearchGroup
    {
        public string groupOp { set; get; }
        public List<SearchRule> rules { set; get; }
    }
```

در اینجا AND و Or کلی مشخص می‌شود، به همراه فیلدهای ارسالی به سرور، عملگرهای اعمالی بر روی آن‌ها و مقادیر مرتبط.
اگر این موارد را کنار هم قرار دهیم، به متدی عمومی ApplyFilter با امضای ذیل خواهیم رسید.

```
public IQueryable<T> ApplyFilter<T>(IQueryable<T> query, bool _search, string searchField, string
searchString,
string searchOper, string filters, NameValueCollection form)
```

کدهای کامل آن‌را به علت طولانی بودن پردازش سه حالت ذکر شده‌ی فوق، از پروژه‌ی پیوست می‌توانید دریافت کنید.
پس از آن، تغییراتی که در کدهای متد GetProducts باید اعمال شوند به صورت ذیل است:

```
[HttpPost]
        public ActionResult GetProducts(string sidx, string sord, int page, int rows,
                                        bool _search, string searchField, string searchString,
                                        string searchOper, string filters)
        {
            var list = ProductDataSource.LatestProducts;

            var pageIndex = page - 1;
            var pageSize = rows;
            var totalRecords = list.Count;
            var totalPages = (int)Math.Ceiling(totalRecords / (float)pageSize);

            var productsQuery = list.AsQueryable();

            productsQuery = new JqGridSearch().ApplyFilter(productsQuery, _search, searchField,
searchString,
                                                           searchOper, filters, this.Request.Form);
            var productsList = productsQuery.OrderBy(sidx + " " + sord)
                                            .Skip(pageIndex * pageSize)
                                            .Take(pageSize)
                                            .ToList();

            var productsData = new JqGridData
            {
                Total = totalPages,
                Page = page,
                Records = totalRecords,
                Rows = (productsList.Select(product => new JqGridRowData
                {
                    Id = product.Id,
                    RowCells = new List<string>
                                {
                                      product.Id.ToString(CultureInfo.InvariantCulture),
                                      product.Name,
                                      product.Supplier.CompanyName,
                                      product.Category.Name,
                                      product.Price.ToString(CultureInfo.InvariantCulture)
                                 }
                })).ToArray()
            };
            return Json(productsData, JsonRequestBehavior.AllowGet);
        }
```

- ابتدا چند پارامتر اضافه‌تر به امضای متد اضافه شده‌اند، تا فیلدهای جستجو را نیز دریافت کنند.
- نوع متد به HttpPost تغییر کرده‌است. این مورد برای ارسال اطلاعات حجیم جستجوها به سرور ضروری است و بهتر است از حالت Get استفاده نشود.
این حالت در سمت کلاینت نیز باید تنظیم شود:

```
$('#list').jqGrid({
//url access method type
mtype: 'POST',
```

- سایر سطرها مانند قبل است؛ فقط یک سطر ذیل جهت اعمال where پویا به عبارت LINQ ساخته شده، اضافه شده‌است:

```
productsQuery = new JqGridSearch().ApplyFilter(productsQuery, _search, searchField, searchString,
  searchOper, filters, this.Request.Form);
```

برای مطالعه بیشتر

جستجوی تک فیلدی

جستجوی نوار ابزاری

جستجوی چند فیلدی

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید

jqGrid03.zip

نظرات خوانندگان

نویسنده: شهربانو مشهدی
تاریخ: ۱۳۹۳/۰۴/۲۷ ۱۵:۲

سلام؛ در حالت جستجو نوار ابزاری با زدن دوباره دکمه جستجو نوار ابزاری ، نوار ابزار حذف نمی‌شود. چه طور می‌شه با زدن دوباره دکمه نوار ابزار ، نوار ابزار از روی صفحه حذف شود.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۲۷ ۱۵:۲۴

با استفاده از متد toggleToolbar می‌شود نوار ابزار نمایش داده شده را مخفی یا مجددا نمایش داد:

```
<button onclick="$('#list')[0].toggleToolbar()">نمایش/حذف تولبار</button>
```

نویسنده: شهربانو مشهدی
تاریخ: ۱۳۹۳/۰۴/۲۷ ۱۷:۴۴

با سپاس؛ من کدتون رو به شکل زیر تبدیل کردم ولی برای بار اول باید دوبار بر روی دکمه نوار ابزار جستجو کلیک کرد آیا راه بهتری هم هست که همان بار اول درست کار کند؟ خیلی مچکرم

```
.jqGrid('navButtonAdd', "#pager", {
                caption: "نوار ابزار جستجو", title: "Search Toolbar", buttonicon: 'ui-icon-search',
                onClickButton: function () {
                    toolbarSearching();
                    $('#list')[0].toggleToolbar();
                }
            });
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۲۷ ۱۷:۵۳

- در کدهای شما ابتدا نوار ابزار نمایش داده می‌شود، سپس toggle تا خاموش شود.
+ این‌ها یک سری مثال هستند برای نمایش نحوه‌ی فعال سازی جداگانه‌ی این قابلیت‌ها از هم.
اگر می‌خواهید از ابتدای کار نوار ابزار جستجو نمایش داده شود، متد مربوطه را در انتهای کدهای jqGrid ذکر کنید:

```
$("#list").jqGrid({
        //...
    }).filterToolbar(options);
```

متدهای دیگر را هم به همین نحو «زنجیر وار» می‌توان ذکر کرد.
- سورس این گرید در فایل jquery.jqGrid.src.js قابل بررسی است. toggleToolbar را در آن جستجو کنید و از کدهای آن جهت یافتن tr.ui-search-toolbar و مخفی یا آشکار کردن آن ایده بگیرید.

نویسنده: شهربانو مشهدی
تاریخ: ۱۳۹۳/۰۴/۲۷ ۲۰:۱۰

با سپاس؛ خوشبختانه کد رو به شکل زیر تغییر دادم و نتیجه گرفتم:

```
$("#list").jqGrid({
        //...
    }).jqGrid('filterToolbar', { stringResult: true, searchOnEnter: true, autosearch: true,
searchOperators: true, groupOp: 'OR', defaultSearch: 'cn' })
            .jqGrid('navButtonAdd', "#pager", {
                caption: "نوار ابزار جستجو", title: "Search Toolbar", buttonicon: 'ui-icon-search',
                onClickButton: function () {
                    $("#list")[0].toggleToolbar();
```

```
            }
        });
        $("#list")[0].toggleToolbar();
```

نویسنده: محسن تقی پور
تاریخ: ۱۳۹۳/۰۵/۱۱ ۲:۳۷

با سلام؛ موقع لیست کردن دسته بندی‌ها در هنگام جستجو (در کمبو باکس) به ازای هر رکورد یک دسته بندی نمایش داده میشه از این کد استفاده کردم ولی نشد چکار باید بکنم تا درست نشون بده ؟

```
public ActionResult SuppliersSelect()
        {
            var list = BlNews.Select().Distinct().ToList();
            var suppliers = list.Select(x => new SelectListItem
            {
                Text = x.Admin.UserName,
                Value = x.Admin.Code.ToString(CultureInfo.InvariantCulture)
            }).ToList();
            return PartialView("_SelectPartial", suppliers);
        }
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۱۱ ۹:۳۴

« پیدا کردن آیتم‌های تکراری در یک لیست به کمک LINQ »

عنوان: **فعال سازی و پردازش صفحات پویای افزودن، ویرایش و حذف رکوردهای jqGrid در ASP.NET MVC**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۱۴ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, jqGrid

پیشنیاز این بحث مطالعه‌ی مطلب « صفحه بندی و مرتب سازی خودکار اطلاعات به کمک jqGrid در ASP.NET MVC » است و در اینجا جهت کوتاه شدن بحث، صرفا به تغییرات مورد نیاز جهت اعمال بر روی مثال اول اکتفاء خواهد شد.

تغییرات مورد نیاز جهت فعال سازی ویرایش، حذف و افزودن رکوردهای jqGrid

می‌خواهیم در بدو نمایش گرید، یک ستون خاص دارای دکمه‌های ویرایش و حذف ظاهر شوند:

برای اینکار تنها کافی است در انتهای ستون‌های تعریف شده، یک ستون خاص را با formatter مساوی actions ایجاد کنیم:

```
colModel: [
                    {
                        // سایر ستون‌ها
                        name: 'myac', width: 80, fixed: true, sortable: false,
                        resize: false, formatter: 'actions',
                        formatoptions: {
                            keys: true
                        }
                    }
                ],
```

برای اینکه دکمه‌های ویرایش و حذف ردیف‌های آن عمل کنند:

نیاز است تعاریف سایر ستون‌هایی را که باید قابلیت ویرایش داشته باشند، به نحو ذیل تغییر دهیم:

```
colModel: [
                    {
                        name: 'Id', index: 'Id', align: 'right', width: 70,
                        editable: false
                    },
                    {
                        name: 'Name', index: 'Name', align: 'right', width: 100,
                        editable: true, edittype: 'text',
                        editoptions: {
                            maxlength: 40
                        },
                        editrules: {
                            required: true
                        }
                    },
                    {
                        name: 'Supplier.Id', index: 'Supplier.Id', align: 'right', width: 110,
                        editable: true, edittype: 'select',
                        editoptions: {
                            dataUrl: '@Url.Action("SuppliersSelect","Home")'
                        },
                        editrules: {
                            required: true
                        }
                    },
                    {
                        name: 'Category.Id', index: 'Category.Id', align: 'right', width: 110,
                        editable: true, edittype: 'select',
                        editoptions: {
                            dataUrl: '@Url.Action("CategoriesSelect","Home")'
                        },
                        editrules: {
                            required: true
                        }
                    },
                    {
                        name: 'Price', index: 'Price', align: 'center', width: 100,
                        formatter: 'currency',
                        formatoptions:
                        {
                            decimalSeparator: '.',
                            thousandsSeparator: ',',
                            decimalPlaces: 2,
                            prefix: '$'
                        },
                        editable: true, edittype: 'text',
                        editrules: {
                            required: true,
                            number: true,
                            minValue: 0
                        }
                    },
                    {
                        name: 'myac', width: 80, fixed: true, sortable: false,
                        resize: false, formatter: 'actions',
                        formatoptions: {
                            keys: true
                        }
                    }
                ],
```

- در اینجا هر ستونی که دارای خاصیت editable مساوی true است، قابلیت ویرایش پیدا می‌کند.
- edittype آن بیانگر کنترلی است که باید حین ویرایش آن سلول خاص ظاهر شود. برای مثال اگر text باشد، یک text box و

اگر مانند حالت Supplier.Id مساوی select تعریف شود، یک drop down را ظاهر خواهد کرد. برای مقدار دهی این drop down می‌توان editoptions و سپس dataUrl آن‌را مقدار دهی نمود.

```
public ActionResult SuppliersSelect()
        {
            var list = ProductDataSource.LatestProducts;
            var suppliers = list.Select(x => new SelectListItem
            {
                Text = x.Supplier.CompanyName,
                Value = x.Supplier.Id.ToString(CultureInfo.InvariantCulture)
            }).ToList();
            return PartialView("_SelectPartial", suppliers);
        }
```

در مثال فوق، این dataUrl به اکشن متد SuppliersSelect اشاره می‌کند که نهایتا لیستی از تولید کننده‌ها را توسط partial view ذیل بازگشت می‌دهد:

```
@model IList<SelectListItem>

@Html.DropDownList("srch", Model)
```

در کل مقادیر قابل تنظیم در اینجا شامل text، textarea، select، checkbox، password، button، image، file و custom هستند.
- خاصیت editrules، برای مباحث اعتبارسنجی اطلاعات ورودی توسط کاربر پیش بینی شده‌است. برای مثال اگر required: true در آن تنظیم شود، کاربر مجبور به تکمیل این سلول خاص خواهد بود. در اینجا خواصی مانند number و integer از نوع bool، خاصیت‌های minValue و maxValue از نوع عددی، email, url, date, time از نوع bool و custom قابل تنظیم است (مثال‌های حالت custom را در منابع انتهای بحث می‌توانید مطالعه کنید).
- پس از اینکه مشخص شدند کدامیک از ستون‌ها باید قابلیت ویرایش داشته باشند، مسیری که باید اطلاعات نهایی را به سرور ارسال کند، توسط خاصیت editurl مشخص می‌شود:

```
$('#list').jqGrid({
   caption: "آزمایش چهارم",
   //url from wich data should be requested
   url: '@Url.Action("GetProducts","Home")',
   //url for edit operation
   editurl: '@Url.Action("EditProduct","Home")',
```

اکشن متد متناظر با این آدرس یک چنین شکلی را می‌تواند داشته باشد:

```
[HttpPost]
        public ActionResult EditProduct(Product postData)
        {
            //todo: Edit product based on postData

            return Json(true);
        }
```

- تعاریف مسیرهای ارسال اطلاعات Add و Delete، در قسمت تنظیمات navGrid باید ذکر شوند:

```
$('#list').navGrid(
                '#pager',
                //enabling buttons
                { add: true, del: true, edit: false, search: false },
                //edit options
                {},
                //add options
                { width: 'auto', url: '@Url.Action("AddProduct","Home")' },
                //delete options
                { url: '@Url.Action("DeleteProduct","Home")' }
                );
```

امضای این اکشن متدها نیز بسیار شبیه به اکشن متد ویرایش است:

```
[HttpPost]
        public ActionResult DeleteProduct(string id)
        {
            //todo: Delete product
            return Json(true);
        }

        [HttpPost]
        public ActionResult AddProduct(Product postData)
        {
            //todo: Add product to repository
            return Json(true);
        }
```

- حالت ویرایش و حذفی که تا اینجا بررسی شد (ستون actions)، جزو خواص توکار این گرید است. اگر بخواهیم آن‌ها را دستی فعال کنیم (جهت اطلاعات عمومی) می‌توان از فراخوانی متد ذیل نیز کمک گرفت:

```
var lastSel;
        function inlineEdit() {
            $('input[name=rdEditApproach]').attr('disabled', true);
            $('#list').navGrid(
                '#pager',
                //enabling buttons
                { add: true, del: true, edit: false, search: false },
                //edit options
                {},
                //add options
                { width: 'auto', url: '@Url.Action("AddProduct","Home")' },
                //delete options
                { url: '@Url.Action("DeleteProduct","Home")' }
                );
            //add onSelectRow event to support inline edit
            $('#list').setGridParam({
                onSelectRow: function (id) {
                    if (id && id != lastSel) {
                        //save changes in row
                        $('#list').saveRow(lastSel, false);
                        lastSel = id;
                    }
                    //trigger inline edit for row
                    $('#list').editRow(id, true);
                }
            });
        };
```

در اینجا ابتدا همان تنظیمات مسیرهای Add و Delete انجام شده‌است. سپس با فراخوانی دستی متد editRow در زمان کلیک بر روی یک ردیف، همان کاری را که ستون actions در جهت فعال سازی خودکار حالت ویرایش سلول‌ها انجام می‌دهد، می‌توان شبیه سازی کرد. متد saveRow نیز کار ارسال اطلاعات تغییر کرده را به سرور انجام می‌دهد.

- برای فعال سازی خودکار فرم‌های افزودن رکوردها و یا ویرایش ردیف‌های موجود می‌توان از فراخوانی متد formEdit ذیل کمک گرفت:

```
function formEdit() {
            $('input[name=rdEditApproach]').attr('disabled', true);
            $('#list').navGrid(
                '#pager',
                //enabling buttons
                { add: true, del: true, edit: true, search: false },
                //edit option
                {
                    width: 'auto', checkOnUpdate: true, checkOnSubmit: true,
                    beforeShowForm: function (form) {
                        centerDialog(form, $('#list'));
                    }
                },
                //add options
                {
                    width: 'auto', url: '@Url.Action("AddProduct","Home")',
                    reloadAfterSubmit: false, checkOnUpdate: true, checkOnSubmit: true,
                    beforeShowForm: function (form) {
                        centerDialog(form, $('#list'));
                    }
```

```
                },
                //delete options
                {
                    url: '@Url.Action("DeleteProduct","Home")', reloadAfterSubmit: false
                })
            .jqGrid('navButtonAdd', "#pager", {
                caption: "حذف ردیف‌های انتخابی", title: "Delete Toolbar", buttonicon: 'ui-icon ui-icon-
trash',
                onClickButton: function () {
                    var idsList = jQuery("#list").jqGrid('getGridParam', 'selarrrow');
                    alert(idsList);
                    //jQuery("#list").jqGrid('delGridRow',idsList,{reloadAfterSubmit:false});
                }
            });
        };

        function centerDialog(form, grid) {
            var dlgDiv = $("#editmod" + grid[0].id);
            var parentDiv = dlgDiv.parent(); // div#gbox_list
            var dlgWidth = dlgDiv.width();
            var parentWidth = parentDiv.width();
            var dlgHeight = dlgDiv.height();
            var parentHeight = parentDiv.height();
            var parentTop = parentDiv.offset().top;
            var parentLeft = parentDiv.offset().left;
            dlgDiv[0].style.top =  Math.round(  parentTop  + (parentHeight-dlgHeight)/2  ) + "px";
            dlgDiv[0].style.left = Math.round(  parentLeft + (parentWidth-dlgWidth  )/2 )  + "px";
        }
```

ابتدای تنظیمات آن، شاهد add: true, del: true, edit: true هستید. این مورد سبب می‌شود تا در فوتر گرید، سه دکمه‌ی افزودن، ویرایش و حذف ردیف‌ها ظاهر شوند:

با کلیک بر روی دکمه‌ی افزودن ردیف جدید، صفحه‌ی ذیل به صورت خودکار تولید می‌شود:

و با کلیک بر روی دکمه‌ی ویرایش ردیفی انتخاب شده، صفحه‌ی ویرایش آن ردیف به همراه مقادیر سلول‌های آن ظاهر خواهند شد:

تنظیمات قسمت‌های Add و Delete ویرایش توسط فرم‌ها، با حالت ویرایش داخل ردیفی آنچنان تفاوتی ندارد. فقط در اینجا پیش از نمایش فرم، از متد centerDialog برای نمایش صفحات افزودن و ویرایش رکوردها در وسط صفحه، استفاده شده‌است. توسط checkOnUpdate: true, checkOnSubmit: true سبب خواهیم شد تا اگر کاربر مقادیر موجود فرمی را تغییر داده‌است و سعی در بستن فرم، بدون ذخیره سازی اطلاعات کند، پیغام هشدار دهنده‌ای به او نمایش داده شود که آیا می‌خواهید تغییرات را ذخیره کنید یا خیر؟

- در انتهای متد formEdit، به کمک متد jqGrid و پارامتر navButtonAdd یک دکمه‌ی سفارشی را نیز اضافه کرده‌ایم. اگر به ستون پس از شماره‌های خودکار ردیف‌ها، در سمت راست گرید دقت کنید، یک سری chekbox قابل مشاهده هستند. برای فعال سازی خودکار آن‌ها کافی است خاصیت multiselect گرید به true تنظیم شود. اکنون برای دسترسی به این ستون‌های انتخاب شده، می‌توان از متد jqGrid به همراه پارامترهای getGridParam و selarrrow استفاده کرد. خروجی آن، لیست idهای ستون‌ها است.

برای مطالعه بیشتر

Common Editing Properties
Inline Editing
Form Editing
Cell Editing

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید
jqGrid04.zip

نظرات خوانندگان

نویسنده: بهمن خلفی
تاریخ: ۱۳۹۳/۰۴/۱۴ ۱۲:۳۳

با تشکر بسیار جهت ارائه این مطلب اما یک نکته اینکه با وجود ارائه KENDO GRID بصورت open source :

آیا این گرید هم مثل kendo grid در صفحه بندی هنگام استفاده بعنوان partial مشکل دارد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۱۴ ۱۲:۴۹

- کل Kendo UI سورس باز هست. اما مجوز عمومی استفاده از آن GPL است. یعنی *باید* کل کارتان را سورس باز کنید یا مجوز آن‌را بخرید.
- اخیرا یک نسخه‌ی سبک‌تر از Kendo UI با مجوز BSD ارائه شده که Grid آن‌را ندارد (به عمد).
بنابراین از این لحاظ، مجوز jqGrid بهتر است. مجوز عمومی آن MIT است و در هر نوع پروژه‌ای قابل استفاده‌است. مجوز تجاری هم دارد برای حالتیکه بخواهید کامپونت‌های ASP.NET آن‌را بخرید که ... نیازی نیست (^ و ^).

```
3. Can be used in proprietary works
The license policy allow you to use this piece of code even inside commercial (not open source)
projects. So you can use this software without giving away your own (precious?) source code.
```

سایر مسایل خارج از بحث جاری است.

نویسنده: محمد رضا خزائی
تاریخ: ۱۳۹۳/۰۵/۱۰ ۹:۲۲

سلام.خسته نباشید.
اگه بخواهیم توی ویرایش به صورت Dialog به جای DropDownList از JQuery AutoComplete استفاده کنیم باید چه تغییری بدیم؟
مرسی

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۱۰ ۹:۴۱

« سفارشی سازی عناصر صفحات پویای افزودن و ویرایش رکوردهای jqGrid در ASP.NET MVC »

نویسنده: ابوالفضل رجب پور
تاریخ: ۱۳۹۳/۰۵/۲۰ ۱۳:۴۸

در کد بالا، برای حالت ویرایش یک سلول در نوع select از mvc controller استفاده شده. من کد بالا رو برای webApi میخوام تنظیم کنم. رشته html رو تولید میکنم و پاس میدم به خروجی، اما خطا میده.
کد رو ببینید.

```
[HttpGet]
public string SelectAllJqTree()
        {
            var result = _kpiTypeService.SelectAll().Select(e => new
            {
                value = e.Id,
                text = e.Name,
            });

            var select = "<select>{0}</select>";
            var option = "<option value='{0}' >{1}</option>";
```

```
            var options = "";

            foreach (var item in result)
            {
                options += string.Format(option, item.value, item.text);
            }

            return  string.Format(select, options).Trim();
}
```

و در کد js ستون "kpiType" رو هم اینجوری تنظیم کردم:

```
colModel: [
                    {
                        name: 'KpiTypeID', index: 'KpiTypeID', align: 'right', width: 300,
                        editable: true, edittype: 'select',
                        editoptions: {
                            dataUrl: '/api/KPIType/SelectAllJqTree'
                        },
                        editrules: {
                            required: true
                        }
                    }
              ]
```

خروجی این رو با جی کوئری که با صفحه اضافه میکنم، یک کنترل صحیح رو تولید میکنه.

```
$.ajax({
                url: "/api/KPIType/SelectAllJqTree",
                context: document.body
            }).done(function (data) {
                $("#selection").html(data);
            });
```

اما اینجا بهم پیام میده که 'Uncaught Error: Syntax error, unrecognized expression: "<select><option value='1 >عملکردی</option><option value='2' >بودجه ای</option>

چه نکته ای وجود داره؟ ایا مربوط به webApi هست؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۲۰ ۱۴:۲۱

- زمانیکه از یک اکشن متد، خروجی HTML دریافت می‌کنید، Content-Type آن مساوی text/html است. در حالت Web Api این مورد application/json یا حالات دیگر می‌تواند باشد (جهت دیباگ بهتر ، برگه‌ی network فایرباگ را در این دو حالات با هم مقایسه کنید. بررسی کنید Response ارسالی چه محتوایی و چه Content-type ایی دارد).
- ضمنا نیازی نیست اطلاعات select را در سمت سرور تولید کنید. امکان دریافت JSON از سرور و تبدیل آن به فرمت مورد نظر در سمت کلاینت هم پیش بینی شده‌است:

```
editoptions: { dataUrl: '...url to get json....',
               buildSelect: function (response) {
                    var data = typeof response === "string" ?  $.parseJSON(response.responseText) :
response,
                    var s = "<select>";
                    s += '<option value="0">--No Manager--</option>';
                    $.each(data, function () {
                          s += '<option value="' + this.EmployeeId + '">' + this.EmployeeName +
'</option>';
                    });
               return s + "</select>";
              }
}
```

در این حالت dataUrl شیء JSON مدنظر را از سرور دریافت می‌کند (آرایه‌ای از EmployeeId و EmployeeName ها). در رویدادگردان سمت کاربر buildSelect ، این مورد دریافت و پردازش می‌شود.

نویسنده: ابوالفضل رجب پور
تاریخ: ۱۳۹۳/۰۵/۲۱ ۱۵:۱۵

تشکر از پاسختون.
با تنظیم نوع خروجی به json کار کرد

```
[HttpGet]
public IHttpActionResult SelectAllFake()
{
            var result = new List<KpiTypeView>
            {
                new KpiTypeView() {KpiTypeID = 1, KpiTypeName = "افزایشی"},
                new KpiTypeView() {KpiTypeID = 2, KpiTypeName = "کاهشی"}
            };

            return Json(result);
}
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۲۱ ۱۵:۳۴

البته در Web API اگر می‌خواهید همیشه خروجی JSON بگیرید می‌شود به نحو زیر هم عمل کرد:

```
configuration.Formatters.Clear();
configuration.Formatters.Add(new JsonMediaTypeFormatter());
```

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);
        }
```

عنوان: **سفارشی سازی عناصر صفحات پویای افزودن و ویرایش رکوردهای jqGrid در ASP.NET MVC**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۱۵ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, jqGrid

پیشنیاز این بحث مطالعه‌ی مطالب « صفحه بندی و مرتب سازی خودکار اطلاعات به کمک jqGrid در ASP.NET MVC » و « فعال سازی و پردازش صفحات پویای افزودن، ویرایش و حذف رکوردهای jqGrid در ASP.NET MVC » است و در اینجا جهت کوتاه شدن بحث، صرفا به تغییرات مورد نیاز جهت اعمال بر روی مثال‌ها اکتفاء خواهد شد.

صورت مساله

```
public class Product
    {
        public int Id { set; get; }
        public DateTime AddDate { set; get; }
        public string Name { set; get; }
        public decimal Price { set; get; }
    }
```

در اینجا تعریف محصول، شامل خاصیت‌های تاریخ ثبت، نام و قیمت آن است.
می‌خواهیم زمانیکه فرم‌های پویای ویرایش یا افزودن رکوردها ظاهر شدند، در حین تکمیل نام، یک auto complete ظاهر شود:

در حین ورود تاریخ، یک date picker شمسی جهت سهولت ورود اطلاعات نمایش داده شود:

همچنین در قسمت ورود مبلغ و قیمت، به صورت خودکار حرف سه رقم جدا کننده هزارها، نمایش داده شوند تا کاربران در حین ورود مبالغ بالا دچار اشتباه نشوند.

پیشنیازها

- برای نمایش auto complete از همان امکانات توکار jQuery UI که به همراه jqGrid عرضه می‌شوند، استفاده خواهیم کرد.
- برای نمایش date picker شمسی از مطلب « PersianDatePicker یک DatePicker شمسی به زبان JavaScript که از تاریخ سرور استفاده می‌کند » کمک خواهیم گرفت.
- جهت اعمال خودکار حرف سه رقم جدا کننده هزارها از افزونه‌ی Price Format جی‌کوئری استفاده می‌کنیم.

تعریف و الحاق این پیشنیازها، فایل layout برنامه را به شکل زیر تغییر خواهد داد:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - My ASP.NET Application</title>

    <link href="~/Content/themes/base/jquery.ui.all.css" rel="stylesheet" />
    <link href="~/Content/jquery.jqGrid/ui.jqgrid.css" rel="stylesheet" />
    <link href="~/Content/PersianDatePicker.css" rel="stylesheet" />
    <link href="~/Content/Site.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <div>
        @RenderBody()
    </div>

    <script src="~/Scripts/jquery-1.7.2.min.js"></script>
    <script src="~/Scripts/jquery-ui-1.8.11.min.js"></script>
    <script src="~/Scripts/i18n/grid.locale-fa.js"></script>
    <script src="~/Scripts/jquery.jqGrid.min.js"></script>
    <script src="~/Scripts/PersianDatePicker.js"></script>
    <script src="~/Scripts/jquery.price_format.2.0.js"></script>

    @RenderSection("Scripts", required: false)
```

```
</body>
</html>
```

تغییرات مورد نیاز سمت کلاینت، جهت اعمال افزونه‌های جی‌کوئری و سفارشی سازی عناصر دریافت اطلاعات

الف) نمایش auto complete در حین ورود نام محصولات

```
colModel: [
                    {
                        name: 'Name', index: 'Name', align: 'right', width: 100,
                        editable: true, edittype: 'text',
                        editoptions: {
                            maxlength: 40,
                            dataInit: function (elem) {
                                // http://jqueryui.com/autocomplete/
                                $(elem).autocomplete({
                                    source: '@Url.Action("GetProductNames","Home")',
                                    minLength: 2,
                                    select: function (event, ui) {
                                        $(elem).val(ui.item.value);
                                        $(elem).trigger('change');
                                    }
                                });
                            }
                        },
                        editrules: {
                            required: true
                        }
                    }
    ],
```

برای اعمال هر نوع افزونه‌ی جی‌کوئری به عناصر فرم‌های خودکار ورود اطلاعات در jqGrid، تنها کافی است که رویداد dataInit یک ستون را بازنویسی کنیم. در اینجا توسط elem، المان جاری را در اختیار خواهیم داشت. سپس از این المان جهت اعمال افزونه‌ای دلخواه استفاده می‌کنیم. برای مثال در اینجا از متد autocomplete استفاده شده‌است که جزئی از jQuery UI استاندارد است.

برای پردازش سمت سرور آن و مقدار دهی url آن، یک چنین اکشن متدی را می‌توان تدارک دید:

```
public ActionResult GetProductNames(string term)
        {
            var list = ProductDataSource.LatestProducts
                .Where(x => x.Name.StartsWith(term))
                .Select(x => x.Name)
                .Take(10)
                .ToArray();
            return Json(list, JsonRequestBehavior.AllowGet);
        }
```

مقدار term، عبارتی است که کاربر وارد کرده است. توسط متد StartsWith، کلیه نام‌هایی را که با این عبارت شروع می‌شوند (البته 10 مورد از آن‌ها را) بازگشت می‌دهیم.

ب) نمایش date picker شمسی در حین ورود تاریخ

```
colModel: [
                    {
                        name: 'AddDate', index: 'AddDate', align: 'center', width: 100,
                        editable: true, edittype: 'text',
                        editoptions: {
                            maxlength: 10,
                            // http://www.dotnettips.info/post/1382
                            onclick: "PersianDatePicker.Show(this,'@today');"
                        },
                        editrules: {
                            required: true
                        }
```

```
                }
            ],
```

Date picker مورد استفاده ، وابستگی خاصی به jQuery ندارد. مطابق مستندات آن باید در رویدادگردان onclick، این تقویم شمسی را فعال کرد. بنابراین در قسمت onclick دقیقا این مورد را اعمال می‌کنیم.

```
@{
ViewBag.Title = "Index";
var today = DateTime.Now.ToPersianDate();
}
```

مقدار today آن در ابتدای View به نحو فوق تعریف شده‌است. کدهای کامل متد کمکی ToPersianDate در پروژه‌ی پیوست موجود است.

ج) اعمال حروف سه رقم جدا کننده هزارها در حین ورود قیمت

```
colModel: [
                {
                    name: 'Price', index: 'Price', align: 'center', width: 100,
                    formatter: 'currency',
                    formatoptions:
                    {
                        decimalSeparator: '.',
                        thousandsSeparator: ',',
                        decimalPlaces: 2,
                        prefix: '$'
                    },
                    editable: true, edittype: 'text',
                    editoptions: {
                        dir: 'ltr',
                        dataInit: function (elem) {
                            // http://jquerypriceformat.com/
                            $(elem).priceFormat({
                                prefix: '',
                                thousandsSeparator: ',',
                                clearPrefix: true,
                                centsSeparator: '',
                                centsLimit: 0
                            });
                        }
                    },
                    editrules: {
                        required: true,
                        minValue: 0
                    }
                }
            ],
```

افزونه‌ی price format نیز یک افزونه‌ی جی‌کوئری است. بنابراین دقیقا مانند حالت auto complete آن‌را در dataInit فعال سازی می‌کنیم و همچنین یک سری تنظیم ابتدایی مانند مشخص سازی thousandsSeparator آن‌را مقدار دهی خواهیم کرد.

یک نکته

همین تعاریف را دقیقا به فرم‌های جستجو نیز می‌توان اعمال کرد. در اینجا برای حالات ویرایش و افزودن رکوردها، editoptions مقدار دهی شده‌است؛ در مورد فرم‌های جستجو باید searchoptions و برای مثال dataInit آن‌را مقدار دهی کرد.

مشکل مهم!

با تنظیمات فوق، قسمت UI بدون مشکل کار می‌کند. اما اگر در سمت سرور، مقادیر دریافتی را بررسی کنیم، نه تاریخ و نه قیمت، قابل دریافت نیستند. زیرا تاریخ ارسالی به سرور شمسی است و مدل برنامه DateTime میلادی می‌باشد. همچنین به دلیل وجود

حروف سه رقم جدا کننده هزارها، عبارت دریافتی قابل تبدیل به عدد نیستند و مقدار دریافتی صفر خواهد بود.
برای رفع این مشکلات، نیاز به تغییر model binder توکار ASP.NET MVC است. برای تاریخ‌ها از کلاس PersianDateModelBinder می‌توان استفاده کرد. برای اعداد decimal از کلاس ذیل:

```
using System;
using System.Globalization;
using System.Threading;
using System.Web.Mvc;

namespace jqGrid05.CustomModelBinders
{
    /// <summary>
    /// How to register it in the Application_Start method of Global.asax.cs
    /// ModelBinders.Binders.Add(typeof(decimal), new DecimalBinder());
    /// </summary>
    public class DecimalBinder : DefaultModelBinder
    {
        public override object BindModel(ControllerContext controllerContext, ModelBindingContext
bindingContext)
        {
            if (bindingContext.ModelType == typeof(decimal) || bindingContext.ModelType ==
typeof(decimal?))
            {
                return bindDecimal(bindingContext);
            }
            return base.BindModel(controllerContext, bindingContext);
        }

        private static object bindDecimal(ModelBindingContext bindingContext)
        {
            var valueProviderResult = bindingContext.ValueProvider.GetValue(bindingContext.ModelName);
            if (valueProviderResult == null)
                return null;

            bindingContext.ModelState.SetModelValue(bindingContext.ModelName, valueProviderResult);
            decimal value;
            var valueAsString = valueProviderResult.AttemptedValue == null ?
                                        null : valueProviderResult.AttemptedValue.Trim();
            if (string.IsNullOrEmpty(valueAsString))
                return null;

            if (!decimal.TryParse(valueAsString, NumberStyles.Any, Thread.CurrentThread.CurrentCulture,
out value))
            {
                const string error ="عدد وارد شده معتبر نیست";
                var ex = new InvalidOperationException(error, new Exception(error, new
FormatException(error)));
                bindingContext.ModelState.AddModelError(bindingContext.ModelName, ex);
                return null;
            }
            return value;
        }
    }
}
```

در اینجا عبارت ارسالی به سرور به صورت یک رشته دریافت شده و سپس تبدیل به یک عدد decaimal می‌شود. در آخر به سیستم model binding بازگشت داده خواهد شد. به این ترتیب دیگر مشکلی با پردازش حروف سه رقم جدا کننده هزارها نخواهد بود.

برای ثبت و معرفی این کلاس‌ها باید به نحو ذیل در فایل global.asax.cs برنامه عمل کرد:

```
using System;
using System.Web.Mvc;
using System.Web.Routing;
using jqGrid05.CustomModelBinders;

namespace jqGrid05
{
    public class MvcApplication : System.Web.HttpApplication
    {
        protected void Application_Start()
        {
            AreaRegistration.RegisterAllAreas();
```

```
            RouteConfig.RegisterRoutes(RouteTable.Routes);
            ModelBinders.Binders.Add(typeof(DateTime), new PersianDateModelBinder());
            ModelBinders.Binders.Add(typeof(decimal), new DecimalBinder());
        }
    }
}
```

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید

jqGrid05.zip

نظرات خوانندگان

نویسنده: رضا
تاریخ: ۱۳۹۳/۰۴/۱۸ ۱۱:۴۶

با تشکر اگر نیاز باشه که یک button به صورت custom مثلا برای Redirect کردن به صفحه دیگه اضافه کنیم باید به چه صورت عمل کنیم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۱۸ ۱۱:۵۴

- از formatterها برای سفارشی سازی و تغییر اطلاعات نهایی نمایش داده شده در یک سلول استفاده کنید.
- از callback ایی به نام beforeShowForm برای اضافه کردن عناصر سفارشی به فرم‌های ویرایش و افزودن رکوردها می‌شود استفاده کرد:

```
$.extend($.jgrid.edit, {
    bSubmit: "Save and Close",
    bCancel: "Cancel",
    width: 370,
    recreateForm: true,
    beforeShowForm: function () {
        $('<a href="#">Save and New<span class="ui-icon ui-icon-disk"></span></a>')
            .click(function() {
                alert("click!");
            }).addClass("fm-button ui-state-default ui-corner-all fm-button-icon-left")
              .prependTo("#Act_Buttons>td.EditButton");
    }
});
```

نویسنده: داود
تاریخ: ۱۳۹۳/۰۴/۱۸ ۱۴:۳۶

باسلام؛ اگر بخواهیم در مودال مورد نظر، 10 قلم اطلاعاتی را بگیریم ولی در گرید 5 تای آن را نشان بدهیم چه باید کرد؟ ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۱۸ ۱۵:۱۱

از تنظیمات hidden ستون‌ها استفاده کنید:

```
colModel:[
    {
       name:'providerUserId',
       index:'providerUserId',
       width:100,
       editable:true,
       editrules:{
           required:true,
           edithidden:true
       },
       hidden:true,
       editoptions:{
          dataInit: function(element) {
             $(element).attr("readonly", "readonly");
          }
       }
},
//...
]
```

در اینجا فیلد فوق در گرید نمایش داده نمی‌شود (hidden:true)، در حین ویرایش نمایان خواهد شد (edithidden:true)، همچنین در قسمت dataInit (البته در صورت نیاز)، به صورت readonly تنظیم شده‌است.

نویسنده: Masoud.Bahrami
تاریخ: ۱۳۹۳/۰۴/۱۹ ۱۳:۲

با سلام و تشکر فراوان
لطفا در مورد امکان افزودن و البته ویرایش Inline که بسیار مهم هستند توضیح بدین. همچنین اگر امکان Export هم داشته باشه که افزونه‌ی خیلی خوب است
بنده در حال حاضر از افزونه ی jTable استفاده کنم. هر چند مجبور شدم فایل‌های js اون رو خیلی تغییر بدم. ولی افزونه‌ی خوبی است.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۱۹ ۱۳:۱۲

مطالب مرتبط به هم سایت جاری را از طریق برچسب‌ها و گروه‌بندی‌های ذیل مطالب بهتر می‌توانید دنبال کنید.
- ویرایش inline
- تهیه خروجی
+
لیست مجموعه مثال‌ها و قابلیت‌های تکمیلی jqGrid (اگر قصد مقایسه دارید)

نویسنده: صالح
تاریخ: ۱۳۹۳/۰۴/۲۰ ۱۸:۵۵

من از گرید شما استفاده کردم ولی متأسفانه وقتی تعداد سطرها زیاد باشه و صفحه اسکرول پیدا کنه، برای حذف و یا تغییرات عناصر انتهایی، فرم تغییرات در قسمت بالای صفحه ظاهر میشه که نمود چندان جالبی نداره
برای رفع این ایراد چه میشه کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۲۰ ۲۰:۳۸

در مطلب فعال سازی این فرم‌ها ، متد centerDialog و نحوه‌ی اعمال آن به رخداد beforeShowForm عنوان شده. ایده‌ای هست در مورد نحوه‌ی مشخص سازی محل نمایش این فرم‌ها در صفحه.

نویسنده: Masoud Bahrami
تاریخ: ۱۳۹۳/۰۴/۲۳ ۱۴:۲۲

با تشکر
منظور امکان افزودن بود. یعنی وقتی دکمه افزودن رو میزنه یک سطر جدید واشه توگرید همونجا بنویسه مشکل اصلی من با jTable هم همینه و این یک مورد خیلی مهمه برا کاربرا که نخواد برا افزودن فرم مودال واشه براش.
مثلا من صفحه‌ی MVC دارم که کاربر محصولات و خدماتشو وارد می‌کنه و مدیریت. موردی که هست ممکن کالاهای هر صاحب شغلی حداقل حدود 1000 و به بالا است. و برای ورود اولی خیلی اذیت میشه و ممکن نیست. اگر این افروزنه هم داشته باشه و معرفی کنید ممنون میشم
در هر صورت بازم ازتون تشکر می‌کنم

نویسنده: rahele
تاریخ: ۱۳۹۳/۰۴/۲۸ ۱۴:۳۲

با سلام؛ چطور می‌توانیم به تابع refresh گرید دسترسی داشته باشم تا بعد از انجام عملیات‌های مختلف , آن را فراخوانی کنم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۲۸ ۱۴:۳۸

```
$("#list").trigger("reloadGrid");
```

ولی در عمل نیازی به این‌کار به صورت دستی نیست؛ چون اگر به مطلب « فعال سازی و پردازش صفحات پویای افزودن، ویرایش و حذف رکوردهای jqGrid در ASP.NET MVC » دقت کنید، تنظیمات false : **reloadAfterSubmit** است که می‌شود آن‌ها را true کرد تا به صورت خودکار به همین نتیجه رسید.

نویسنده: rahele
تاریخ: ۱۳۹۳/۰۴/۲۸ ۱۸:۴۴

با سلام و تشکر
من در گرید عملیات‌های اضافه کردن و ویرایش کردن و پاک کردن را به صورت inline انجام میدهم در صورتیکه راهنمایی‌های شما جهت کار با pop up می باشد. محبت بفرمایید برای عملیات‌های inline راهنمایی نمایید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۳۰ ۱۱:۲۸

مراجعه کنید به مطلب « فعال سازی و پردازش Inline Add در jqGrid »

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۳۰ ۱۲:۳۵

مراجعه کنید به مطلب « فعال سازی و پردازش Inline Add در jqGrid »

عنوان: **تهیه خروجی PDF و اکسل از حاصل جستجوی پویای jqGrid به کمک PDF Report**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۱۶ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, PdfReport, jqGrid

پیشنیازها

- صفحه بندی و مرتب سازی خودکار اطلاعات به کمک jqGrid در ASP.NET MVC
- فعال سازی و پردازش جستجوی پویای jqGrid در ASP.NET MVC
- سفارشی سازی عناصر صفحات پویای افزودن و ویرایش رکوردهای jqGrid در ASP.NET MVC
- آشنایی با کتابخانه‌ی PDF Report

اضافه کردن دکمه‌ی خروجی به jqGrid

برای تهیه خروجی از jqGrid نیاز است بدانیم، اکنون در چه صفحه‌ای از اطلاعات قرار داریم؟ بر روی چه ستونی، مرتب سازی صورت گرفته‌است؟ بر روی کدام فیلدها با چه مقادیری جستجو انجام شده‌است؟ تا ... بتوانیم بر این مبنا، منبع داده‌ی موجود را فیلتر کرده و لیست نهایی را تبدیل به گزارش کنیم. گزارشی که دقیقا با اطلاعاتی که کاربر در صفحه مشاهده می‌کند، تطابق داشته باشد.

خوشبختانه تمام این سؤالات توسط متد توکار excelExport در سمت سرور قابل دریافت است:

```
@section Scripts
{
    <script type="text/javascript">
    $(document).ready(function () {
        $('#list').jqGrid({
            caption: "آزمایش ششم",
                // مانند قبل
            }).navGrid(
                // مانند قبل
                }).jqGrid('navButtonAdd', '#pager', {
                    caption: "", buttonicon: "ui-icon-print", title: "خروجی پی دی اف",
                    onClickButton: function () {
                        $("#list").jqGrid('excelExport', { url: '@Url.Action("GetProducts",
"Home")' });
                    }
                });
        });
    </script>
}
```

در اینجا توسط متد navButtonAdd یک دکمه‌ی جدید را اضافه کرده‌ایم که کلیک بر روی آن سبب فراخوانی متد excelExport و ارسال اطلاعات گزارش به url تنظیم شده‌است. باید دقت داشت که این اطلاعات از طریق Http Get به سرور ارسال می‌شوند و دقیقا اجزای آن همان اجزای جستجوی پویای jqGrid است:

```
public ActionResult GetProducts(string sidx, string sord, int page, int rows,
                                 bool _search, string searchField, string searchString,
                                 string searchOper, string filters, string oper)
```

با این تفاوت که یک oper نیز به مجموعه‌ی پارامترهای ارسالی به سرور اضافه شده‌است. این oper در اینجا با excel مقدار دهی می‌شود.
البته چون تعداد این پارامترها بیش از اندازه شده‌است، بهتر است آن‌ها را تبدیل به یک کلاس کرد:

```
namespace jqGrid06.Models
{
    public class JqGridRequest
    {
        public string sidx { set; get; }
        public string sord { set; get; }
        public int page { set; get; }
        public int rows { set; get; }
        public bool _search { set; get; }
        public string searchField { set; get; }
        public string searchString { set; get; }
        public string searchOper { set; get; }
        public string filters { set; get; }
        public string oper { set; get; }
    }
}
```

و متد جستجوی پویا را به نحو ذیل بازنویسی نمود:

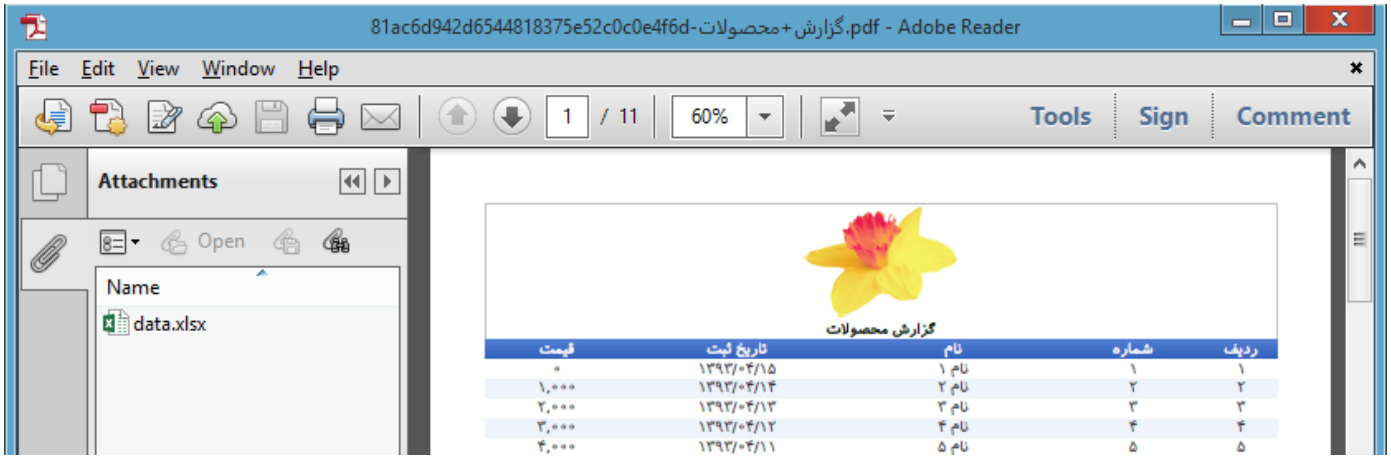
```
public ActionResult GetProducts(JqGridRequest request)
        {
            var list = ProductDataSource.LatestProducts;

            var pageIndex = request.page - 1;
            var pageSize = request.rows;
            var totalRecords = list.Count;
            var totalPages = (int)Math.Ceiling(totalRecords / (float)pageSize);

            var productsQuery = list.AsQueryable();

            productsQuery = new JqGridSearch().ApplyFilter(productsQuery, request, this.Request.Form);
            productsQuery = productsQuery.OrderBy(request.sidx + " " + request.sord);

            if (string.IsNullOrWhiteSpace(request.oper))
            {
                productsQuery = productsQuery
                                    .Skip(pageIndex * pageSize)
                                    .Take(pageSize);
            }
            else if (request.oper == "excel")
            {
                productsQuery = productsQuery
                                    .Skip(pageIndex * pageSize);
            }

            var productsList = productsQuery.ToList();

            if (!string.IsNullOrWhiteSpace(request.oper) && request.oper == "excel")
            {
                new ProductsPdfReport().CreatePdfReport(productsList);
            }

            var productsData = new JqGridData
            {
                Total = totalPages,
                Page = request.page,
                Records = totalRecords,
                Rows = (productsList.Select(product => new JqGridRowData
                {
                    Id = product.Id,
                    RowCells = new List<string>
                    {
                        product.Id.ToString(CultureInfo.InvariantCulture),
                        product.Name,
                        product.AddDate.ToPersianDate(),
                        product.Price.ToString(CultureInfo.InvariantCulture)
                    }
                })).ToArray()
            };
```

```
            return Json(productsData, JsonRequestBehavior.AllowGet);
        }
```

توضیحات:

اکثر قسمت‌های این متد با متدی که در مطلب « فعال سازی و پردازش جستجوی پویای jqGrid در ASP.NET MVC » مشاهده کردید یکی است؛ برای مثال order by آن با استفاده از کتابخانه‌ی Dynamic LINQ به صورت پویا عمل می‌کند و متد ApplyFilter، کار تهیه where پویا را انجام می‌دهد.

فقط در اینجا بررسی و پردازش پارامتر oper نیز اضافه شده‌است. اگر این پارامتر مقدار دهی شده باشد، یعنی نیاز است کل اطلاعات را واکشی کرد؛ زیرا می‌خواهیم گزارش گیری کنیم و نه اینکه صرفا اطلاعات یک صفحه را به کاربر بازگشت دهیم. همچنین در اینجا List نهایی فیلتر شده به یک گزارش Pdf Report ارسال می‌شود. این گزارش چون نهایتا اطلاعات را در مرورگر کاربر Flush می‌کند، کار به اجرای سایر قسمت‌ها نخواهد رسید و همینجا گزارش نهایی تهیه می‌شود.



کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید

jqGrid06.7z

نظرات خوانندگان

نویسنده: سروش
تاریخ: ۱۳۹۳/۰۴/۱۷ ۹:۳۸

با سلام من هنگام اجرای پروژه با خطای زیر روربرو میشم

```
Could not load file or assembly 'System.Web.Mvc, Version=4.0.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35' or one of its dependencies. The system cannot find the file specified.
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۱۷ ۹:۵۴

- ابتدا به اینترنت وصل شوید.
- سپس در خط فرمان پاورشل نیوگت دستور زیر را اجرا کنید:

```
PM> update-package -reinstall
```

به این صورت بسته‌های MVC 5 آن به صورت صحیح به پروژه اضافه می‌شوند.

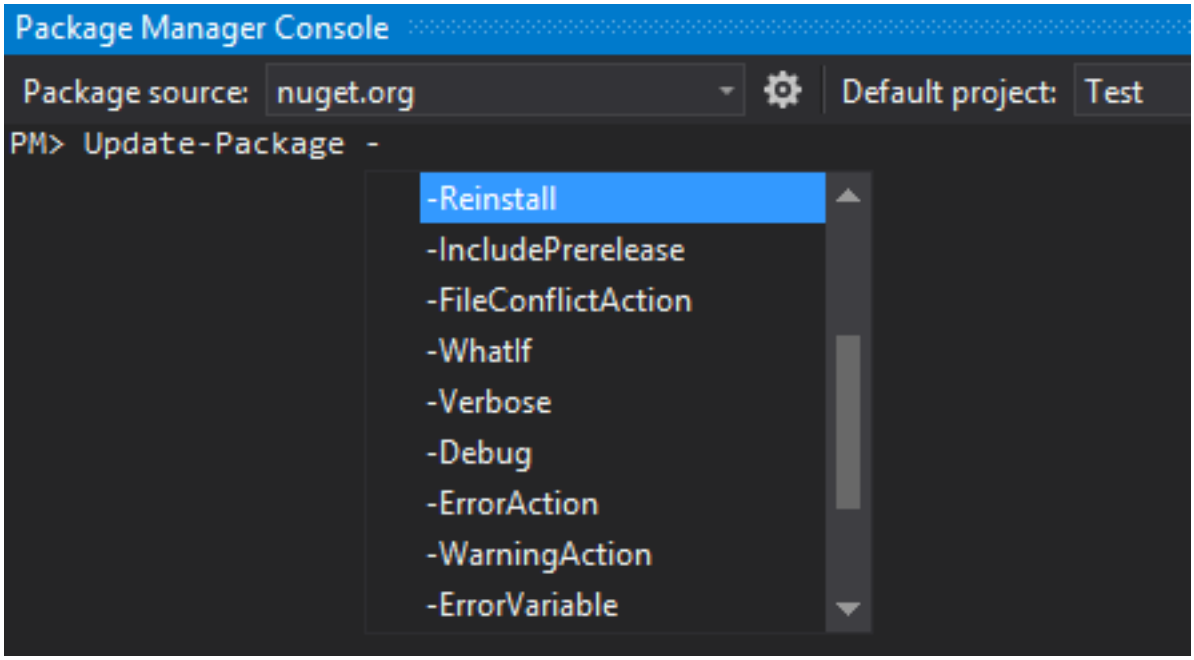
نویسنده: سروش
تاریخ: ۱۳۹۳/۰۴/۱۷ ۱۷:۵۷

متاسفانه با تایپ فرمان بالا پیغام زیر صادر می‌گردد

```
Update-Package : A parameter cannot be found that matches parameter name 'reinstall'.
At line:1 char:26
+ update-package -reinstall <<<<
    + CategoryInfo          : InvalidArgument: (:) [Update-Package], ParameterBindingException
    + FullyQualifiedErrorId : NamedParameterNotFound,NuGet.PowerShell.Commands.UpdatePackageCommand
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۱۷ ۱۹:۰

دریافت آخرین نگارش نیوگت از اینجا ^ و ^



نویسنده: سروش
تاریخ: ۱۳۹۳/۰۴/۱۸ ۱۱:۴۶

با سلام و ادب

در پروژه خروجی Excel بوسیله EPPlus موجود نبود میشه راهنمایی کنید

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۱۸ ۱۱:۵۲

اگر به تصویر آخر دقت کنید، خروجی اکسل کتابخانه‌ی Pdf Report در قسمت پیوست‌های فایل PDF تولیدی قرار می‌گیرد.

نویسنده: داوود
تاریخ: ۱۳۹۳/۰۴/۲۷ ۱۵:۲۷

با سلام
به نظر فایل zip این مطلب دچار مشکل است بعد از دانلود نتوانستم آن را اکسترکت کنم
درضمن پسوند عجیبی دارد (jqGrid06.7z)
ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۲۷ ۱۷:۱۵

از برنامه‌ی 7zip یا winrar استفاده کنید.

نویسنده: جواد وکیلی
تاریخ: ۱۳۹۳/۰۵/۱۴ ۱۵:۷

یه مشکلی که این گرید با راست به چپ دارد نمایش اشتباه تعداد رکوردها هنگامی که از هزار بیشتر می‌شود

تصویر بالا تعداد هزار و پانصد می‌باشد .

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۱۴ ۱۵:۵۱

پیش فرض‌های آن قابل سفارشی سازی است:

```
$.jgrid.formatter.integer.thousandsSeparator = ',';
$.jgrid.formatter.number.thousandsSeparator = ',';
$.jgrid.formatter.currency.thousandsSeparator = ',';
```

این سطرها را پیش از تعریف گرید قرار دهید.

عنوان: **آپلود فایل توسط فرم‌های پویای jqGrid**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۲۴ ۱۱:۳۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, jqGrid

پیشنیازها

Ajax.BeginForm و ارسال فایل به سرور در ASP.NET MVC
فعال سازی و پردازش صفحات پویای افزودن، ویرایش و حذف رکوردهای jqGrid در ASP.NET MVC
فرمت کردن اطلاعات نمایش داده شده به کمک jqGrid در ASP.NET MVC
استفاده ازExpressionها جهت ایجاد Strongly typed view در ASP.NET MVC

فرم‌های پویای jqGrid نیز به صورت Ajax ایی به سرور ارسال می‌شوند و اگر نوع عناصر تشکیل دهنده‌ی آن‌ها file تعیین شوند، قادر به ارسال این فایل‌ها به سرور نخواهند بود. در ادامه نحوه‌ی یکپارچه سازی افزونه‌ی AjaxFileUpload را با فرم‌های jqGrid بررسی خواهیم کرد.

تغییرات فایل Layout برنامه

در اینجا دو فایل جدید ajaxfileupload.js و jquery.blockUI.js به مجموعه‌ی فایل‌های تعریف شده اضافه شده‌اند:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - My ASP.NET Application</title>

    <link href="~/Content/themes/base/jquery.ui.all.css" rel="stylesheet" />
    <link href="~/Content/jquery.jqGrid/ui.jqgrid.css" rel="stylesheet" />
    <link href="~/Content/Site.css" rel="stylesheet" type="text/css" />
</head>
<body>
    <div>
        @RenderBody()
    </div>

    <script src="~/Scripts/jquery-1.7.2.min.js"></script>
    <script src="~/Scripts/jquery-ui-1.8.11.min.js"></script>
    <script src="~/Scripts/i18n/grid.locale-fa.js"></script>
    <script src="~/Scripts/jquery.jqGrid.src.js"></script>
    <script src="~/Scripts/ajaxfileupload.js"></script>
    <script src="~/Scripts/jquery.blockUI.js"></script>

    @RenderSection("Scripts", required: false)
</body>
</html>
```

از فایل jquery.blockUI.js برای نمایش صفحه‌ی منتظر بمانید تا فایل آپلود شود، استفاده خواهیم کرد.

```
PM> Install-Package jQuery.BlockUI
```

نکته‌ای در مورد واکنشگرا کردن jqGrid

اگر می‌خواهید عرض jqGrid به تغییرات اندازه‌ی مرورگر پاسخ دهد، تنها کافی است تغییرات ذیل را اعمال کنید:

```
<div dir="rtl" id="grid1" style="width:100%;" align="center">
    <div id="rsperror"></div>
    <table id="list" cellpadding="0" cellspacing="0"></table>
    <div id="pager" style="text-align:center;"></div>
</div>
```

```
    <script type="text/javascript">
        $(document).ready(function () {

            // Responsive jqGrid
            $(window).bind('resize', function () {
                var targetContainer = "#grid1";
                var targetGrid = "#list";

                $(targetGrid).setGridWidth($(targetContainer).width() - 2, true);
            }).trigger('resize');


            $('#list').jqGrid({
                caption: "آزمایش هفتم",
                /// .....
            }).navGrid(
                /// .....
            ).jqGrid('gridResize', { minWidth: 400, minHeight: 150 });
        });
    </script>
```

در اینجا به تغییرات resize صفحه گوش فرا داده شده و سپس به کمک متد توکار setGridWidth، به صورت پویا اندازه‌ی عرض jqGrid تغییر خواهد کرد.
همچنین اگر می‌خواهید کاربر بتواند اندازه‌ی گرید را دستی تغییر دهد، به انتهای تعاریف گرید، تعریف متد gridResize را نیز اضافه کنید.

نحوه‌ی تعریف ستونی که قرار است فایل آپلود کند

```
colModel: [
                    {
                        name: '@(StronglyTyped.PropertyName<Product>(x=>x.ImageName))',
                        index: '@(StronglyTyped.PropertyName<Product>(x => x.ImageName))',
                        align: 'center', width: 220,
                        editable: true,
                        edittype: 'file',
                        formatter: function (cellvalue, options, rowObject) {
                            return "<img src='/images/" + cellvalue + "?rnd=" + new Date().getTime() +
"' />";
                        },
                        unformat: function (cellvalue, options, cell) {
                            return $('img', cell).attr('src').replace('/images/', '');
                        }
                    }
                ],
```

edittype ستونی که قرار است فایل آپلود کند، باید به file تنظیم شود. همچنین چون در اینجا این فایل آپلودی، تصویر یک محصول است، از formatter برای تبدیل مسیر فایل به تصویر و از unformat برای بازگشت این مسیر به مقدر اصلی آن استفاده خواهیم کرد. از unformat برای حالت ویرایش اطلاعات استفاده می‌شود. از formatter برای تغییر اطلاعات دریافتی از سرور به فرمت دلخواهی در سمت کلاینت می‌توان کمک گرفت.
Rnd اضافه شده به انتهای آدرس تصویر، جهت جلوگیری از کش شدن آن تعریف شده‌است.

کتابخانه‌ی JqGridHelper

در قسمت‌های قبل مطالب بررسی jqGrid یک سری کلاس مانند JqGridData برای بازگشت اطلاعات مخصوص jqGrid و یا JqGridRequest برای دریافت پارامترهای ارسالی توسط آن به سرور، تهیه کردیم؛ به همراه کلاس‌هایی مانند جستجو و مرتب سازی پویای اطلاعات.
اگر این کلاس‌ها را از پروژه‌ها و مثال‌های ارائه شده خارج کنیم، می‌توان به کتابخانه‌ی JqGridHelper رسید که فایل‌های آن در پروژه‌ی پیوست موجود هستند.

همچنین در این پروژه، کلاسی به نام StronglyTyped با متد PropertyName جهت دریافت نام رشته‌ای یک خاصیت تعریف شده‌است. گاهی از اوقات این تنها چیزی است که کدهای سمت کلاینت، جهت سازگار شدن با Refactoring و Strongly typed تعریف شدن نیاز دارند و نه ... محصور کننده‌هایی طویل و عریض که هیچگاه نمی‌توانند تمام قابلیت‌های یک کتابخانه‌ی غنی جاوا اسکریپتی را به همراه داشته باشند.
با کمی جستجو، برای jqGrid نیز می‌توانید از این دست محصور کننده‌هارا پیدا کنید اما ... هیچکدام کامل نیستند و دست آخر مجبور خواهید شد در بسیاری از موارد مستقیما JavaScript نویسی کنید.

یکپارچه سازی افزونه‌ی AjaxFileUpload با فرم‌های jqGrid

پس از این مقدمات، ستون ویژه‌ی actions که inline edit را فعال می‌کند، چنین تعریفی را پیدا خواهد کرد:

```
colModel: [
                    {
                        name: 'myac', width: 80, fixed: true, sortable: false,
                        resize: false, formatter: 'actions',
                        formatoptions: {
                            keys: true,
                            afterSave: function (rowid, response) {
                                doInlineUpload(response, rowid);
                            },
                            delbutton: true,
                            delOptions: {
                                url: "@Url.Action("DeleteProduct","Home")"
                            }
                        }
                    }
                ],
```

در اینجا afterSave اضافه شده‌است تا کار ارسال فایل به سرور را در حالت ویرایش inline فعال کند.
و ویژگی‌های قسمت‌های add، edit و delete فرم‌های پویای jqGrid باید به نحو ذیل تغییر کنند:

```
$('#list').jqGrid({
                caption: "آزمایش هفتم",
                // ....
            }).navGrid(
                '#pager',
                //enabling buttons
                { add: true, del: true, edit: true, search: false },
                //edit option
                {
                    width: 'auto',
                    reloadAfterSubmit: true, checkOnUpdate: true, checkOnSubmit: true,
                    beforeShowForm: function (form) {
                        centerDialog(form, $('#list'));
                    },
                    afterSubmit: doFormUpload,
                    closeAfterEdit: true
                },
                //add options
                {
                    width: 'auto', url: '@Url.Action("AddProduct","Home")',
                    reloadAfterSubmit: true, checkOnUpdate: true, checkOnSubmit: true,
                    beforeShowForm: function (form) {
                        centerDialog(form, $('#list'));
                    },
                    afterSubmit: doFormUpload,
                    closeAfterAdd: true
                },
                //delete options
                {
                    url: '@Url.Action("DeleteProduct","Home")',
                    reloadAfterSubmit: true
                }).jqGrid('gridResize', { minWidth: 400, minHeight: 150 });
```

با اکثر این تنظیمات در مطلب « فعال سازی و پردازش صفحات پویای افزودن، ویرایش و حذف رکوردهای jqGrid در ASP.NET MVC » آشنا شده‌اید. تنها قسمت جدید آن شامل رویدادگردان afterSubmit است. در اینجا است که افزونه‌ی AjaxFileUpload

فعال شده و سپس اطلاعات المان فایل را به سرور ارسال می‌کند.
افزونه‌ی AjaxFileUpload پس از ارسال اطلاعات عناصر غیر فایلی فرم، باید فعال شود. به همین جهت است که از رویداد afterSubmit در حالت نمایش فرم‌های پویا و رویداد afterSave در حالت ویرایش inline استفاده کرده‌ایم.
در ادامه تعاریف متدهای doInlineUpload و doUpload بکار گرفته شده در رویداد afterSubmit را مشاهده می‌کنید:

```
function doInlineUpload(response, rowId) {
            return doUpload(response, null, rowId);
        }

        function doFormUpload(response, postdata) {
            return doUpload(response, postdata, null);
        }

        function doUpload(response, postdata, rowId) {
            // دریافت خروجی متد ثبت اطلاعات از سرور
            // و استفاده از آی دی رکورد ثبت شده برای انتساب فایل آپلودی به آن رکورد
            var result = $.parseJSON(response.responseText);
            if (result.success === false)
                return [false, "عملیات ثبت موفقیت آمیز نبود", result.id];

            var fileElementId = '@(StronglyTyped.PropertyName<Product>(x=>x.ImageName))';
            if (rowId) {
                fileElementId = rowId + "_" + fileElementId;
            }

            var val = $("#" + fileElementId).val();
            if (val == '' || val === undefined) {
                // فایلی انتخاب نشده
                return [false, "لطفا فایلی را انتخاب کنید", result.id];
            }

            $('#grid1').block({ message: '<h4>در حال ارسال فایل به سرور</h4>' });
            $.ajaxFileUpload({
                url: "@Url.Action("UploadFiles", "Home")", // مسیری که باید فایل به آن ارسال شود
                secureuri: false,
                fileElementId: fileElementId, // آی دی المان ورودی فایل
                dataType: 'json',
                data: { id: result.id }, // اطلاعات اضافی در صورت نیاز
                complete: function () {
                    $('#grid1').unblock();
                },
                success: function (data, status) {
                    $("#list").trigger("reloadGrid");
                },
                error: function (data, status, e) {
                    alert(e);
                }
            });

            return [true, "با تشکر!", result.id];
        }
```

امضای رویدادگردان‌های afterSubmit و afterSave یکی نیست. به همین جهت دو متد اضافی به جای یک متد doUpload مورد استفاده قرار گرفته‌اند.
متد doUpload توسط پارامتر response، اطلاعات بازگشتی پس از ذخیره سازی متداول اطلاعات فرم را دریافت می‌کند. برای مثال ابتدا اطلاعات معمولی یک محصول در بانک اطلاعاتی ذخیره شده و سپس id آن به همراه یک خاصیت به نام success از طرف سرور بازگشت داده می‌شوند.
اگر success مساوی true بود، ادامه‌ی کار آپلود فایل انجام خواهد شد. در اینجا ابتدا بررسی می‌شود که آیا فایلی از طرف کاربر انتخاب شده‌است یا خیر؟ اگر خیر، یک پیام اعتبارسنجی سفارشی به او نمایش داده خواهد شد.
خروجی متد doUpload حتما باید به شکل یک آرایه سه عضوی باشد. عضو اول آن true و false است؛ به معنای موفقیت یا عدم موفقیت عملیات. عضو دوم پیام اعتبارسنجی سفارشی است و عضو سوم، Id ردیف.
در ادامه افزونه‌ی jQuery.BlockUI فعال می‌شود تا ارسال فایل به سرور را به کاربر گوشزد کند.
سپس فراخوانی متداول افزونه‌ی ajaxFileUpload را مشاهده می‌کنید. تنها نکته‌ی مهم آن فراخوانی متد reloadGrid در حالت success است. به این ترتیب گرید را وادار می‌کنیم تا اطلاعات ذخیره شده در سمت سرور را دریافت کرده و سپس تصویر را به نحو صحیحی نمایش دهد.

کدهای سمت سرور آپلود فایل

```
[HttpPost]
        public ActionResult AddProduct(Product postData)
        {
            // ...
            return Json(new { id = postData.Id, success = true }, JsonRequestBehavior.AllowGet);
        }

        [HttpPost]
        public ActionResult EditProduct(Product postData)
        {
            // ...
            return Json(new { id = postData.Id, success = true }, JsonRequestBehavior.AllowGet);
        }


        // todo: change `imageName` according to the form's file element name
        [HttpPost]
        public ActionResult UploadFiles(HttpPostedFileBase imageName, int id)
        {
            // ....
            return Json(new { FileName = product.ImageName }, "text/html",
JsonRequestBehavior.AllowGet);
        }
```

در اینجا تنها نکته‌ی مهم، خروجی‌های JSON این متدها هستند.
در حالت‌های Add و Edit، نیاز است id رکورد ثبت شده بازگشت داده شود. این id در سمت کلاینت توسط پارامتر response دریافت می‌شود. از آن در افزونه‌ی ارسال فایل به سرور استفاده خواهیم کرد. اگر به متد UploadFiles دقت کنید، این id را دریافت می‌کند. بنابراین می‌توان یک ربط منطقی را بین فایل ارسالی و رکورد متناظر با آن برقرار کرد.
Content type مقدار بازگشتی از متد UploadFiles حتما باید text/html باشد (افزونه‌ی ارسال فایل‌ها، اینگونه کار می‌کند).

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید:

jqGrid07.zip

نظرات خوانندگان

نویسنده: daniyal
تاریخ: ۱۳۹۳/۰۵/۰۴ ۲۲:۱۱

سلام
strongly typeها فقط نام پروپرتی‌ها را بر می‌گرداند اگر بخواهیم مقدار یکی از data annotationها مثل display برگرداند و یا اگر بخواهیم از validationهای data annotationها استفاده کنیم باید چه کار کنیم .ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۰۵ ۱:۱۰

- به پیشنیازهای بحث مراجعه کنید؛ مورد آخر .
- مستقل هست از مباحث اعتبارسنجی سمت کاربر تعریف شده توسط data annotationها. از افزونه‌ی jQuery Validator استفاده نمی‌کند. سیستم خاص خودش را دارد. فقط از اعتبارسنجی سمت سرور حاصل data annotationها می‌شود در آن استفاده کرد .

عنوان: **اعتبارسنجی سفارشی سمت کاربر و سمت سرور در jqGrid**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۲۷ ۱۲:۰
آدرس: www.dotnettips.info
گروه‌ها: MVC, jqGrid

همانطور که در مطلب « فعال سازی و پردازش صفحات پویای افزودن، ویرایش و حذف رکوردهای jqGrid در ASP.NET MVC » نیز ذکر شد، خاصیت editrules یک ستون، برای مباحث اعتبارسنجی اطلاعات ورودی توسط کاربر پیش بینی شده‌است. برای مثال اگر required: true در آن تنظیم شود، کاربر مجبور به تکمیل این سلول خاص خواهد بود. در اینجا خواصی مانند number و integer از نوع bool هستند، minValue و maxValue از نوع عددی، email, url, date, time از نوع bool و custom قابل تنظیم است. در ادامه نحوه‌ی اعمال اعتبارسنجی‌های سفارشی سمت سرور و همچنین سمت کلاینت را بررسی خواهیم کرد.

مدل برنامه و نیازمندی‌های اعتبارسنجی آن

```
namespace jqGrid08.Models
{
    public class User
    {
        public int Id { set; get; }

        public string Name { set; get; }

        public string Email { set; get; }

        public string Password { set; get; }

        public string SiteUrl { set; get; }
    }
}
```

مدل کاربر فوق را در نظر بگیرید. در حین ورود اطلاعات نیاز است:
- نام کاربر به صورت اجباری وارد شود و همچنین بین 3 تا 40 حرف باشد.
- همچنین نام کاربر نباید بر اساس اطلاعات موجود در بانک اطلاعاتی، تکراری وارد شود.
- ورود ایمیل شخص اجباری است؛ به علاوه فرمت آن نیز باید با یک ایمیل واقعی تطابق داشته باشد.
- ایمیل وارد شده‌ی یک کاربر جدید نیز نباید تکراری بوده و پیشتر توسط کاربر دیگری وارد شده باشد.
- ورود کلمه‌ی عبور در حالت ثبت اطلاعات اجباری است؛ اما در حالت ویرایش اطلاعات خیر (از کلمه‌ی عبور موجود در این حالت استفاده خواهد شد).
- ورود آدرس سایت کاربر اجباری بوده و همچنین فرمت آدرس وارد شده نیز باید معتبر باشد.

اعتبار سنجی سمت سرور و سمت کلاینت نام کاربر

```
colModel: [
                    {
                        name: '@(StronglyTyped.PropertyName<User>(x => x.Name))',
                        index: '@(StronglyTyped.PropertyName<User>(x => x.Name))',
                        align: 'right', width: 150,
                        editable: true, edittype: 'text',
                        editoptions: {
                            maxlength: 40
                        },
                        editrules: {
                            required: true,
                            custom: true,
                            custom_func: function (value, colname) {
                                if (!value)
                                    return [false, "لطفا نامی را وارد کنید"];

                                if (value.length < 3 || value.length > 40)
                                    return [false, colname + " باید بین 3 تا 40 حرف باشد"];
                                else
```

```
                                    return [true, ""];
                                }
                            }
                        },
                    ],
```

با تنظیم required: true، کار تنظیم ورود اجباری نام کاربر به پایان می‌رسد. اما نیاز است این نام بین 3 تا 40 حرف باشد. بنابراین نیاز است سمت کاربر بتوان اطلاعات وارد شده توسط کاربر را دریافت کرده و سپس طول آن‌را بررسی نمود. این‌کار، توسط مقدار دهی خاصیت custom به true و سپس تعریف متدی برای custom_func قابل انجام است.
خروجی این متد یک آرایه دو عضوی است. اگر عضو اول آن true باشد، یعنی اعتبارسنجی موفقیت آمیز بوده‌است؛ اگر خیر، عضو دوم آرایه، پیامی است که به کاربر نمایش داده خواهد شد.
تا اینجا کار اعتبارسنجی سمت کاربر به پایان می‌رسد. اما نیاز است در سمت سرور نیز بررسی شود که آیا نام وارد شده تکراری است یا خیر. برای این منظور تنها کافی است رویداد afterSubmit حالت‌های Add و Edit را بررسی کنیم:

```
$('#list').jqGrid({
                // ...
            }).navGrid(
                '#pager',
                //enabling buttons
                { add: true, del: true, edit: true, search: false },
                //edit option
                {
                    afterSubmit: showServerSideErrors
                },
                //add options
                {
                    afterSubmit: showServerSideErrors
                },
                //delete options
                {
                });
        });

        function showServerSideErrors(response, postdata) {
            var result = $.parseJSON(response.responseText);
            if (result.success === false) {
                //نمایش خطای اعتبار سنجی سمت سرور پس از ویرایش یا افزودن
                //و همچنین جلوگیری از ثبت نهایی فرم
                return [false, result.message, result.id];
            }

            return [true, "", result.id];
        }
```

شیء response، حاوی اطلاعات بازگشت داده شده از طرف سرور است. برای مثال یک چنین خروجی JSON ایی را در حالت‌های شکست اعتبارسنجی بازگشت می‌دهیم:

```
[HttpPost]
        public ActionResult AddUser(User postData)
        {
            //todo: Add user to repository
            if (postData == null)
                return Json(new { success = false, message = "اطلاعات دریافتی خالی است" },
JsonRequestBehavior.AllowGet);

            if (_usersInMemoryDataSource.Any(
                    user => user.Name.Equals(postData.Name,
StringComparison.InvariantCultureIgnoreCase)))
            {
                return Json(new { success = false, message = "نام کاربر تکراری است" },
JsonRequestBehavior.AllowGet);
            }

            if (_usersInMemoryDataSource.Any(
                    user => user.Email.Equals(postData.Email,
StringComparison.InvariantCultureIgnoreCase)))
            {
                return Json(new { success = false, message = "آدرس ایمیل کاربر تکراری است" },
JsonRequestBehavior.AllowGet);
            }
```

```
            postData.Id = _usersInMemoryDataSource.LastOrDefault() == null ? 1 :
_usersInMemoryDataSource.Last().Id + 1;
            _usersInMemoryDataSource.Add(postData);

            return Json(new { id = postData.Id, success = true }, JsonRequestBehavior.AllowGet);
        }
```

در سمت کلاینت در روال رویدادگردان afterSubmit می‌توان با آنالیز response و سپس استخراج فیلدهای JSON آن، وضعیت success و همچنین پیام‌های بازگشت داده شده را بررسی کرد.
خروجی روال رویدادگردان afterSubmit نیز بسیار شبیه است به حالت اعتبارسنجی سفارشی یک ستون. اگر عضو اول آرایه بازگشت داده شده توسط آن false باشد، یعنی اعتبارسنجی سمت سرور، با شکست مواجه شده و در این حالت از عضو دوم آرایه برای نمایش پیام خطای بازگشت داده شده از طرف سرور استفاده خواهد شد.

اعتبار سنجی ایمیل کاربر

```
colModel: [
                    {
                        name: '@(StronglyTyped.PropertyName<User>(x => x.Email))',
                        index: '@(StronglyTyped.PropertyName<User>(x => x.Email))',
                        align: 'center', width: 150,
                        editable: true, edittype: 'text',
                        editoptions: {
                            maxlength: 250,
                            dir: 'ltr'
                        },
                        editrules: {
                            required: true,
                            email: true
                        },
                        formatter: 'email'
                    },
                ],
```

با تنظیم required: true، کار تنظیم ورود اجباری ایمیل کاربر به پایان می‌رسد. همچنین با تنظیم email: true، به صورت خودکار فرمت ایمیل وارد شده نیز بررسی می‌شود.
مطابق نیازمندی‌های اعتبارسنجی پروژه، ایمیل وارد شده نیز نباید تکراری باشد. این مورد نیز توسط خروجی روال رویدادگردان afterSubmit که پیشتر توضیح داده شده، مدیریت می‌شود.

اعتبار سنجی کلمه عبور کاربر

```
colModel: [
                    {
                        name: '@(StronglyTyped.PropertyName<User>(x => x.Password))',
                        index: '@(StronglyTyped.PropertyName<User>(x => x.Password))',
                        align: 'center', width: 70,
                        editable: true, edittype: 'password',
                        editoptions: {
                            maxlength: 10,
                            dir: 'ltr'
                        },
                        editrules: {
                            //required: true ---> در این حالت خاص قابل استفاده نیست
                            //در حالت ویرایش رکورد، ورود کلمه عبور اختیاری است
                            //در حالت افزودن رکورد، ورود کلمه عبور اجباری است
                        }
                    },
                ],
```

حالت بررسی اعتبارسنجی کلمه‌ی عبور در اینجا، حالت ویژه‌ای است. نیاز است در حالت ثبت اطلاعات اجباری باشد اما در حالت

ویرایش خیر. بنابراین نمی‌توان از خاصیت required: true استفاده کرد؛ چون به هر دو حالت ویرایش و ثبت اطلاعات به صورت یکسان اعمال می‌شود.
برای این منظور تنها کافی است از روال رویدادگردان beforeSubmit استفاده کرد:

```
$('#list').jqGrid({
                // ...
            }).navGrid(
                '#pager',
                //enabling buttons
                { add: true, del: true, edit: true, search: false },
                //edit option
                {
                    /*, beforeSubmit: function (posdata, obj) {
                        //در حالت ویرایش رکورد، ورود کلمه عبور اختیاری است
                        return [true, ""];
                    }*/
                },
                //add options
                {
                    beforeSubmit: function (postdata, obj) {
                        //در حالت افزودن رکورد، ورود کلمه عبور اجباری است
                        if (postdata.Password == null || postdata.Password == "" || postdata.Password
== undefined)
                            return [false, "لطفا کلمه عبور را وارد کنید"];
                        return [true, ""];
                    }
                },
                //delete options
                {
                });
        });
```

چون می‌خواهیم تنها حالت Add را تحت کنترل قرار دهیم، رویدادگردان beforeSubmit آن‌را مقدار دهی کرده‌ایم. توسط postdata کلیه اطلاعات قابل ارسال به سرور به صورت یک شیء جاوا اسکریپتی یا JSON در اختیار ما است. سپس با بررسی برای مثال postdata.Password می‌توان در مورد مقدار کلمه‌ی عبور تصمیم گیری کرد. در اینجا نیز خروجی متد باید یک آرایه دو عضوی باشد تا در صورت false بودن اولین عضو آن، پیام سفارشی اعتبارسنجی خاصی را بتوان به کاربر نمایش داد.

اعتبار سنجی آدرس سایت کاربر

```
colModel: [
                    {
                        name: '@(StronglyTyped.PropertyName<User>(x => x.SiteUrl))',
                        index: '@(StronglyTyped.PropertyName<User>(x => x.SiteUrl))',
                        align: 'center', width: 150,
                        editable: true, edittype: 'text',
                        editoptions: {
                            maxlength: 1000,
                            dir: 'ltr'
                        },
                        editrules: {
                            required: true,
                            url: true
                        },
                        formatter: function (cellvalue, options, rowObject) {
                            return "<a href='" + cellvalue + "' >" + cellvalue + "</a>";
                        },
                        unformat: function (cellvalue, options, cell) {
                            return $('a', cell).attr('href');
                        }
                    },
                ],
```

با تنظیم required: true، کار تنظیم ورود اجباری آدرس سایت کاربر به پایان می‌رسد. همچنین با تنظیم url: true، به صورت خودکار فرمت URL وارد شده نیز بررسی می‌شود.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید

jqGrid08.zip

نظرات خوانندگان

نویسنده: داوود
تاریخ: ۱۳۹۳/۰۴/۲۷ ۱۵:۳۲

باسلام
بعداز دانلود فایل zip واجرای آن تمام فونت‌های فارسی به شکل ناخوانا و یک فرمت عجیب ظاهر میشوند
تمام مثال‌های دوره jqgrid را دانلود کردم که از شماره 4 به بعد با این مشکل برخورد کردم
پیشاپیش از راهنمایی هاتون متشکرم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۲۷ ۱۷:۱۹

- فرمت فایل‌ها اگر 1256 است (بر اساس تنظیمات جاری سیستم)، از منوی File گزینه‌ی Advanced save options آن‌را بر روی Utf-8 with signature قرار دهید.
- در ابتدای فایل layout برنامه در قسمت هدر، این چند سطر را اضافه کنید:

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

نویسنده: محسن عباس آبادی
تاریخ: ۱۳۹۳/۰۴/۳۰ ۱۱:۳۱

با سلام در مورد مطلب مفیدتون
اگر بخواهیم سطوح دسترسی رو روی این گرید تعریف کنیم به چه صورت می‌تونیم عمل کنیم منظورم به یک کاربر اجازه ثبت بدیم و یا فقط بتونه اطلاعات را ببینه ولی نتواند ویرایش کند ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۳۰ ۱۱:۴۰

در این گرید، تعریف ستون‌ها در حقیقت یک آرایه است. از ترکیب Razor سمت سرور و JavaScript سمت کاربر به صورت زیر برای اعمال نقشی خاص استفاده کنید:

```
var colMdl = [];
colMdl.push({ name: 'id', index: 'id', hidden: true });

@if (User.IsInRole("myCustomRole")) {
<text>
colMdl.push(تعریف ستون اکشن در اینجا اضافه شود);
</text>
}
```

از این ایده‌ی ترکیبی، برای تمام قسمت‌های آن نیز می‌شود استفاده کرد.

نویسنده: محسن عباس آبادی
تاریخ: ۱۳۹۳/۰۵/۰۱ ۱۰:۹

با سلام؛ امکان دارد درقالب یک مثال کامل توضیح بدهید

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۰۱ ۱۰:۲۲

جهت مباحث تکمیلی اعتبارسنجی کاربران به این مطالب مراجعه کنید:

- اعتبار سنجی کاربران در ASP.NET MVC
- مدیریت سفارشی سطوح دسترسی کاربران در MVC
- Iris Membership برای احراز هویت کاربران در ASP.NET MVC به صورت پویا
- ASP.NET Identity

عنوان: **فعال سازی و پردازش Inline Add در jqGrid**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۳۰ ۸:۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, jqGrid

پیشنیازها

فعال سازی و پردازش صفحات پویای افزودن، ویرایش و حذف رکوردهای jqGrid در ASP.NET MVC
اعتبارسنجی سفارشی سمت کاربر و سمت سرور در jqGrid

پیشتر با نحوه‌ی فعال سازی صفحات پویای افزودن، ویرایش و حذف رکوردهای jqGrid آشنا شدیم. اما ... شاید علاقمند نباشید که اصلا از این صفحات استفاده کنید. شاید به نظر شما با کلیک بر روی دکمه‌ی + افزودن یک رکورد جدید، بهتر باشد داخل خود گرید، یک سطر خالی جدید باز شده تا بتوان آن‌را پر کرد. شاید این نحو کار کردن با گرید، از دید عده‌ای طبیعی‌تر باشد نسبت به حالت نمایش صفحات popup افزودن و یا ویرایش رکوردها. در ادامه این مورد را بررسی خواهیم کرد.

فعال سازی افزودن، ویرایش و حذف Inline

فعال سازی ویرایش و حذف Inline را پیشتر نیز بررسی کرده بودیم. تنها کافی است یک ستون جدید را با 'formatter:' actions' تعریف کنیم. به صورت خودکار، دکمه‌ی ویرایش، حذف، ذخیره سازی و لغو Inline ظاهر می‌شوند و همچنین بدون نیاز به کدنویسی بیشتری کار می‌کنند.
اما در کدهای ذیل اندکی این ستون را سفارشی سازی کرده‌ایم. در قسمت formatter آن، دکمه‌های edit و delete یک سطر جدید توکار اضافه شده را حذف کرده‌ایم. زیرا در این حالت خاص، وجود این دکمه‌ها ضروری نیستند. بهتر است در این حالت دکمه‌های save و cancel ظاهر شوند:

```
$('#list').jqGrid({
                caption: "آزمایش نهم",
                // ....
                colModel: [
                    {
                        name: 'myac', width: 80, fixed: true, sortable: false,
                        resize: false,
                        //formatter: 'actions',
                        formatter: function (cellvalue, options, rowObject) {
                            if (cellvalue === undefined && options.rowId === "_empty") {
                                // در حالت نمایش ردیف توکار جدید دکمه‌های ویرایش و حذف معنی ندارند
                                options.colModel.formatoptions.editbutton = false;
                                options.colModel.formatoptions.delbutton = false;
                            }
                            return $.fn.fmatter.actions(cellvalue, options, rowObject);
                        },
                        formatoptions: {
                            keys: true,
                            afterSave: function (rowid, response) {
                            },
                            delbutton: true,
                            delOptions: {
                                url: "@Url.Action("DeleteUser", "Home")"
                            }
                        }
                    }
                ],
                //...
            }).navGrid(
                '#pager',
//...
)
                .jqGrid('gridResize', { minWidth: 400, minHeight: 150 })
                .jqGrid('inlineNav', '#pager',
                {
                    edit: true, add: true, save: true, cancel: true,
                    edittext: "ویرایش", addtext: "جدید", savetext: "ذخیره", canceltext: "لغو",
                    addParams: {
                        // اگر می‌خواهید ردیف‌های جدید در ابتدا ظاهر شوند، این سطر را حذف کنید
                        position: "last", //ردیف‌های جدید در آخر ظاهر می‌شوند
                        rowID: '_empty',
                        useDefValues: true,
```

```
                    addRowParams: getInlineNavParams(true)
                },
                editParams: getInlineNavParams(false)
            });
```

قسمتی که کار فعال سازی Inline Add را انجام می‌دهد، تعریف مرتبط با inlineNav است که به انتهای تعاریف متداول گرید اضافه شده‌است.

در اینجا 4 دکمه‌ی ویرایش، جدید، ذخیره و لغو، در نوار pager پایین گرید ظاهر خواهند شد (سمت چپ؛ سمت راست همان دکمه‌های نمایش فرم‌های پویا هستند).

سپس باید دو قسمت مهم addParams و editParams آن‌را مقدار دهی کرد.

در قسمت addParams، مشخص می‌کنیم که ID ردیف اضافه شده، مساوی کلمه‌ی _empty باشد. اگر به کدهای formatter ستون action دقت کنید، از این ID برای تشخیص افزوده شدن یک ردیف جدید استفاده شده‌است.

position در اینجا به معنای محل افزوده شدن یک ردیف خالی است. مقدار پیش فرض آن first است؛ یعنی همیشه در اولین ردیف گرید، این ردیف جدید اضافه می‌شود. در اینجا به last تنظیم شده‌است تا در پایین گرید و پس از رکوردهای موجود، نمایش داده شود.

useDefValues سبب استفاده از مقادیر پیش فرض تعریف شده در ستون‌های گرید در حین افزوده شدن یک ردیف جدید می‌گردد.

addRowParams و editParams هر دو ساختار تقریبا یکسانی دارند که به نحو ذیل تعریف می‌شوند:

```
function getInlineNavParams(isAdd) {
            return {
                // استفاده از آدرس‌های مختلف برای حالات ویرایش و ثبت اطلاعات جدید
                url: isAdd ? '@Url.Action("AddUser", "Home")' : '@Url.Action("EditUser","Home")',
                key: true,
                restoreAfterError: false, // این مورد سبب می‌شود تا اعتبارسنجی سمت سرور قابل اعمال
شود
                oneditfunc: function (rowId) {
                    // نمایش دکمه‌های ذخیره و لغو داخل همان سطر
                    $("#jSaveButton_" + rowId).show();
                    $("#jCancelButton_" + rowId).show();
                },
                successfunc: function () {
                    var $self = $(this);
                    setTimeout(function () {
                        $self.trigger("reloadGrid"); // دریافت کلید اصلی ردیف از سرور
                    }, 50);
                },
                errorfunc: function (rowid, response, stat) {
                    if (stat != 'error') // this.Response.StatusCode == 200
                        return;

                    var result = $.parseJSON(response.responseText);
                    if (result.success === false) {
                        //نمایش خطای اعتبار سنجی سمت سرور پس از ویرایش یا افزودن
                        $.jgrid.info_dialog($.jgrid.errors.errcap,
                            '<div class="ui-state-error">' + result.message + '</div>',
                            $.jgrid.edit.bClose,
```

```
                            { buttonalign: 'center' });
                    }
                }
            };
        }
```

در ابتدای کار مشخص می‌کنیم که آدرس‌های ذخیره سازی اطلاعات در سمت سرور برای حالت‌های Add و Edit کدام‌اند.
تنظیم restoreAfterError به false بسیار مهم است. اگر در سمت سرور خطای اعتبارسنجی گزارش شود و restoreAfterError مساوی true باشد (مقدار پیش فرض)، کاربر مجبور خواهد شد اطلاعات را دوباره وارد کند.
در روال رویدادگران oneditfunc دکمه‌ی save و cancel ردیف را که مخفی هستند، ظاهر می‌کنیم (مکمل formatter ستون action است).
در قسمت successfunc، پس از پایان موفقیت آمیز کار، متد reloadGrid را فراخوانی می‌کنیم. اینکار سبب می‌شود تا Id واقعی رکورد، از سمت سرور دریافت شود. از این Id برای ویرایش و همچنین حذف، استفاده خواهد شد. علت استفاده از setTimeout در اینجا این است که اندکی به DOM فرصت داده شود تا کارش به پایان برسد.
در قسمت errorfunc خطاهای اعتبارسنجی سفارشی سمت سرور را می‌توان دریافت و سپس توسط متد توکار info_dialog به کاربر نمایش داد.

یک نکته‌ی مهم در مورد ارسال خطاهای اعتبارسنجی از سمت سرور در حالت Inline Add

```
if (_usersInMemoryDataSource.Any(
                    user => user.Name.Equals(postData.Name,
StringComparison.InvariantCultureIgnoreCase)))
            {
                this.Response.StatusCode = 500; //این مورد برای افزودن داخل ردیف‌های گرید لازم است
                return Json(new { success = false, message = "نام کاربر تکراری است" },
JsonRequestBehavior.AllowGet);
            }
```

روال رویداد گردان errorfunc، اگر مقدار StatusCode بازگشتی از سمت سرور مساوی 200 باشد (حالت عادی و موفقیت آمیز)، مقدار stat مساوی error را باز نمی‌گرداند. به همین جهت است که در کدهای فوق، مقدار دهی this.Response.StatusCode را به 500 مشاهده می‌کنید. هر عددی غیر از 200 در اینجا به error تفسیر می‌شود. همچنین اگر این StatusCode سمت سرور تنظیم نشود، گرید فرض را بر موفقیت آمیز بودن عملیات گذاشته و successfunc را فراخوانی می‌کند.

مدیریت StatusCodeهای غیر از 200 در حالت کار با فرم‌های jqGrid

اگر هر دو حالت Inline Add و فرم‌های پویا را فعال کرده‌اید، بازگشت StatusCode = 500 سبب می‌شود تا دیگر نتوان خطاهای سفارشی سمت سرور را در بالای فرم‌ها به کاربر نمایش داد و در این حالت تنها یک internal server error را مشاهده خواهند کرد. برای رفع این مشکل فقط کافی است روال رویدادگران errorTextFormat را مدیریت کرد:

```
$('#list').jqGrid({
                caption: "آزمایش نهم",
                //.........
            }).navGrid(
                '#pager',
                //enabling buttons
                { add: true, del: true, edit: true, search: false },
                //edit option
                {
                    //.........
                    errorTextFormat: serverErrorTextFormat
                },
                //add options
                {
                    //.........
                    errorTextFormat: serverErrorTextFormat
                },
                //delete options
                {
```

```
                //.........
            })
            .jqGrid('gridResize', { minWidth: 400, minHeight: 150 })
            .jqGrid('inlineNav', '#pager',
            {
                //.........
            });

        function serverErrorTextFormat (response) {
            // در حالتیکه وضعیت خروجی از سرور 200 نیست فراخوانی می‌شود
            var result = $.parseJSON(response.responseText);
            if (result.success === false) {
                return result.message;
            }
            return "لطفا ورودی‌های وارد شده را بررسی کنید";
        }
```

errorTextFormat تنها در حالتیکه StatusCode بازگشتی از طرف سرور مساوی 200 نیست، فراخوانی می‌شود. در اینجا می‌توان response دریافتی را آنالیز و سپس پیام خطای سفارشی آن‌را جهت نمایش در فرم‌های پویای گرید، بازگشت داد.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید:

jqGrid09.zip

عنوان: **گروه بندی اطلاعات در jqGrid**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۰۴ ۰:۲۰
آدرس: www.dotnettips.info
گروه‌ها: MVC, jqGrid

فرض کنید لیستی از مطالب را به فرمت ذیل در اختیار داریم:

```
namespace jqGrid10.Models
{
    public class Post
    {
        public int Id { set; get; }
        public string Title { set; get; }
        public string CategoryName { set; get; }
        public int NumberOfViews { set; get; }
    }
}
```

می‌خواهیم آن‌ها را با شرایط ذیل گروه بندی کنیم:
- گروه بندی بر روی ستون CategoryName انجام شود.
- ستونی که بر روی آن گروه بندی انجام می‌شود، نمایش داده نشود.
- در ابتدای نمایش گروه‌ها، تمام آن‌ها به صورت جمع شده و Collapsed نمایش داده شوند.
- پس از نمایش گروه‌ها، اولین گروه به صورت خودکار باز شود.
- تعداد ردیف هر گروه به عنوان گروه اضافه شود.
- جمع کل ستون تعداد بار مشاهدات هر گروه قابل محاسبه شود.
- جمع کل هر گروه در زمانیکه هر گروه نیز بسته‌است نمایش داده شود.
- رنگ ردیف جمع کل قابل تنظیم باشد.

آزمایش دهم

شماره	عنوان	تعداد بار مشاهده
گروه 1 - 5 ردیف		
1	عنوان 1	230
17	عنوان 17	180
37	عنوان 37	926
42	عنوان 42	322
48	عنوان 48	849
خلاصه		**جمع مشاهدات: 2507**
گروه 2 - 5 ردیف		
خلاصه		**جمع مشاهدات: 2951**
گروه 3 - 5 ردیف		
خلاصه		**جمع مشاهدات: 2461**
گروه 4 - 5 ردیف		
خلاصه		**جمع مشاهدات: 3498**

نمایش 1 - 20 از 50 صفحه 1 از 3 20

فعال سازی گروه بندی در jqGrid

فعال سازی گروه بندی در jqGrid به سادگی افزودن تعاریف ذیل است:

```
$('#list').jqGrid({
                caption: "آزمایش دهم",
                //...
                grouping: true,
                groupingView: {
                    groupField: ['CategoryName'],
                    groupOrder: ['asc'],
                    groupText : ['<b>{0} - {1} ردیف</b>'],
                    groupDataSorted: true,
                    groupColumnShow: false,
                    groupCollapse: true,
                    groupSummary: [true],
                    showSummaryOnHide: true
                }
            });
```

grouping: true سبب می‌شود تا گروه بندی فعال شود. سپس نیاز است ستون یا ستون‌هایی را که قرار است بر روی آن‌ها گروه بندی صورت گیرد، معرفی کنیم. اینکار توسط خواص شیء groupingView انجام می‌شوند.
groupField بیانگر آرایه‌ای از ستون‌هایی است که قرار است بر روی آن‌ها گروه بندی صورت گیرد.
groupOrder آرایه‌ای اختیاری از مقادیر asc یا desc است که متناظر هستند با نحوه‌ی مرتب سازی پیش فرض ستون‌های معرفی شده در آرایه groupField.
groupText آرایه‌ای اختیاری از عناوین گروه بندی‌های انجام شده‌است. اگر ذکر شود، {0} آن با نام گروه و {1} آن با تعداد عناصر گروه جایگزین می‌شود.
تنظیم groupDataSorted سبب خواهد شد تا نام ستونی که بر روی آن گروه بندی صورت می‌گیرد، به سرور ارسال شود (توسط پارامتر sidx). به این ترتیب در سمت سرور می‌توان اطلاعات را به صورت پویا مرتب سازی کرده و بازگشت داد.
با تنظیم groupColumnShow به false سبب خواهیم شد تا ستون‌های معرفی شده در قسمت groupField نمایش داده نشوند.
با تنظیم groupCollapse به true، در ابتدای نمایش گروه‌ها، ردیف‌های آن‌ها نمایش داده نخواهند شد و در حالت جمع شده قرار می‌گیرند.
groupSummary به معنای فعال سازی نمایش ردیف محاسبه‌ی summary مانند sum، min، max و امثال آن بر روی یک گروه است.
اگر مقدار showSummaryOnHide مساوی true باشد، ردیف محاسبه‌ی summary حتی در حالت groupCollapse: true نمایش داده خواهد شد.

فعال سازی محاسبه‌ی جمع ستون تعداد بار مشاهدات مطالب

برای فعال سازی نهایی محاسبه‌ی جمع ستون تعداد بار مشاهدات، علاوه بر تنظیم groupSummary به true، نیاز است در همان ستون مشخص کنیم که این محاسبات چگونه باید انجام شوند:

```
colModel: [
// ........
                    {
                        name: '@(StronglyTyped.PropertyName<Post>(x => x.Title))',
                        index: '@(StronglyTyped.PropertyName<Post>(x => x.Title))',
                        align: 'right',
                        width: 150,
                        summaryTpl: '<div style="text-align: left;">خلاصه </div>',
                        summaryType: function (val, name, record) {
                            return "";
                        }
                    },
// ........
                    {
                        name: '@(StronglyTyped.PropertyName<Post>(x => x.NumberOfViews))',
                        index: '@(StronglyTyped.PropertyName<Post>(x => x.NumberOfViews))',
                        align: 'center',
                        width: 70,
                        summaryType: 'sum', summaryTpl: '<b>0} :جمع مشاهدات}</b>'
                    }
                ],
```

خاصیت summaryType نوع متد محاسبه‌ی summary مانند sum را مشخص می‌کند. خاصیت summaryTpl اختیاری است. اگر ذکر شود سبب فرمت مقدار محاسبه شده‌ی نهایی می‌گردد. در اینجا {0} به مقدار نهایی که قرار است در این سلول درج شود، اشاره می‌کند.
summaryType مانند ستون عنوان، سفارشی شده نیز می‌توان باشد. در ردیف summary و ستون عنوان تنها می‌خواهیم یک مقدار ثابت را نمایش دهیم، به همین جهت summaryType آن به یک مقدار خالی تنظیم شده‌است.

تغییر رنگ ردیف خلاصه عملیات هر گروه به همراه گشودن خودکار اولین گروه

گروه بندی به همراه یک سری متد توکار نیز هست. برای مثال اگر متد groupingToggle را بر روی Id هر گروه فراخوانی کنیم، می‌توان سبب باز یا بسته شده آن گروه شد. متدهای دیگری مانند groupingGroupBy برای گروه بندی پویا و groupingRemove برای حذف گروه بندی نیز وجود دارند:

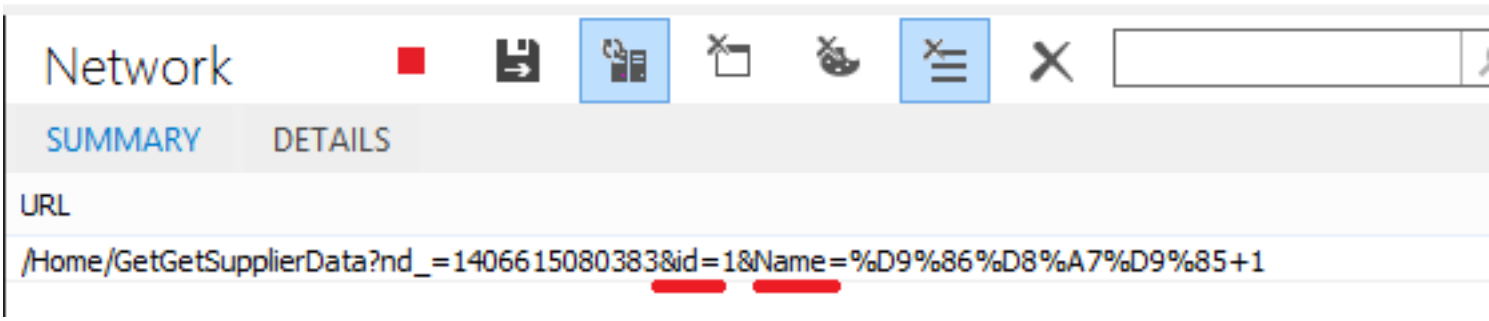
```
$('#list').jqGrid({
                caption: "آزمایش دهم",
                //.........
                loadComplete: function() {
                    //.........
                    $('#list').jqGrid('groupingToggle', 'list' + 'ghead_0_0');
                    $("tr.jqfoot td").css({
                        "background": "#2f4f4f",
                        "color": "#FFF"
                    });
                },
            });
```

بهترین محل اعمال رنگ به ردیف‌های summary، در روال رویدادگردان loadComplete گرید است.

مثال کامل این قسمت را از اینجا می‌توانید دریافت کنید:
jqGrid10.zip

عنوان: **نمایش Subgrid در jqGrid**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۰۷ ۱۱:۲۰
آدرس: www.dotnettips.info
گروه‌ها: MVC, jqGrid

اگر به مثال « فرمت کردن اطلاعات نمایش داده شده در jqGrid » دقت کنید، لینکی را جهت نمایش یک popup جزئیات رکورد انتخاب شده قرار دادیم. شاید طراحی بهتر به این صورت باشد که یک دکمه‌ی + در کنار ردیف قرار دهیم. با کلیک کاربر بر روی این دکمه، جزئیات این ردیف، از سرور دریافت شده و به صورت یک زیر گرید نمایش داده شود. در ادامه همان مثال را با همان ساختار داده‌ای و کدهای سمت سرور، جهت کار با subgrids بازنویسی خواهیم کرد.



فعال سازی Subgrid

در اینجا مواردی را که باید جهت فعال سازی subgrid به تعاریف اولیه‌ی jqGrid اضافه کرد، مشاهده می‌کنید:

```
$('#list').jqGrid({
                caption: "آزمایش یازدهم",
                // ...
                jsonReader: {
                // ...
                    subgrid: { root: "Rows", repeatitems: true, cell: "RowCells" }
                },
                // ...
                subGrid: true,
                subGridModel: [{
                    name: ['شرکت', 'آدرس', 'کد پستی', 'شهر', 'کشور', 'تلفن', 'وب سایت'],
                    width: [100, 100, 100, 100, 100, 100, 100],
                    align: ['center', 'center', 'center', 'center', 'center', 'center', 'center'],
                    params: ['@(StronglyTyped.PropertyName<Product>(x=>x.Name))']
                }],
                subGridOptions:{
                    reloadOnExpand : false //load only once
                },
                subGridUrl: '@Url.Action("GetGetSupplierData", "Home")'
            });
```

چون قصد داریم در سمت سرور، از همان ساختار JqGridData خودمان برای بازگشت اطلاعات subgrid استفاده کنیم، نیاز است خاصیت subgrid مربوط به jsonReader را اندکی ویرایش کنیم تا jqGrid بداند که بجای cell قرار است RowCells دریافت کند.
با تنظیم subGrid: true نمایش ستون + داری که در تصویر فوق مشخص است، انجام می‌شود.
subGridModel بیانگر ساختار اطلاعاتی است که قرار است نمایش داده شوند.
آرایه name، نام سر ستون‌ها را مشخص می‌کند.
آرایه width، عرض ستون‌های زیرگرید را مقدار دهی خواهد کرد.
آرایه align محل و سمت قرارگیری هر یک از مقادیر سلول‌ها را تعیین می‌کند.
آرایه params اختیاری است. زمانیکه کاربر بر روی یک + ستون subgrid، برای باز شدن این زیرگرید کلیک می‌کند، صرفا Id ردیف به سرور ارسال می‌شود. اگر در این بین می‌خواهید، خاصیت خاصی از گرید اصلی نیز به سرور ارسال شود، آرایه params را مقدار دهی کنید. برای نمونه در اینجا Name ردیف انتخاب شده نیز به ارسال ارسال خواهد شد (برگه شبکه شکل فوق).
subGridOptions یک سری تنظیمات اضافه را به همراه دارد. اگر می‌خواهید اطلاعات زیرگرید فقط یکبار بارگذاری شود و با هربار کلیک کاربر از سرور دریافت نگردد، خاصیت reloadOnExpand آن‌را false کنید.
subGridUrlآدرسی که تامین کننده اطلاعات JSON زیرگرید می‌باشد.

در این حالت، کدهای سمت سرور بازگشت اطلاعات زیر گرید به شکل زیر می‌باشد:

```
public ActionResult GetGetSupplierData(int id, string name)
        {
            var list = ProductDataSource.LatestProducts;
            var products = list.Where(x => x.Id == id).ToList();
            if (!products.Any())
                return Json(null, JsonRequestBehavior.AllowGet);

            var productsData = new JqGridData
            {
                Rows = (products.Select(product => new JqGridRowData
                {
                    Id = product.Id,
                    RowCells = new List<string>
                                {
                                     product.Supplier.CompanyName,
                                     product.Supplier.Address,
                                     product.Supplier.PostalCode,
                                     product.Supplier.City,
                                     product.Supplier.Country,
                                     product.Supplier.Phone,
                                     product.Supplier.HomePage
                                 }
                })).ToList()
            };
            return Json(productsData, JsonRequestBehavior.AllowGet);
        }
```

همانطور که ملاحظه می‌کنید، حالت ساده شده‌ی JqGridData بازگشت داده می‌شود. زیرا در حالت نمایش زیرگرید، مباحث مرتب سازی اطلاعات و همچنین paging فعال نیستند و نیازی به اطلاعات آن‌ها نیست.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید

jqGrid11.zip

نظرات خوانندگان

نویسنده: daniyal
تاریخ: ۱۳۹۳/۰۵/۰۷ ۱۴:۲۱

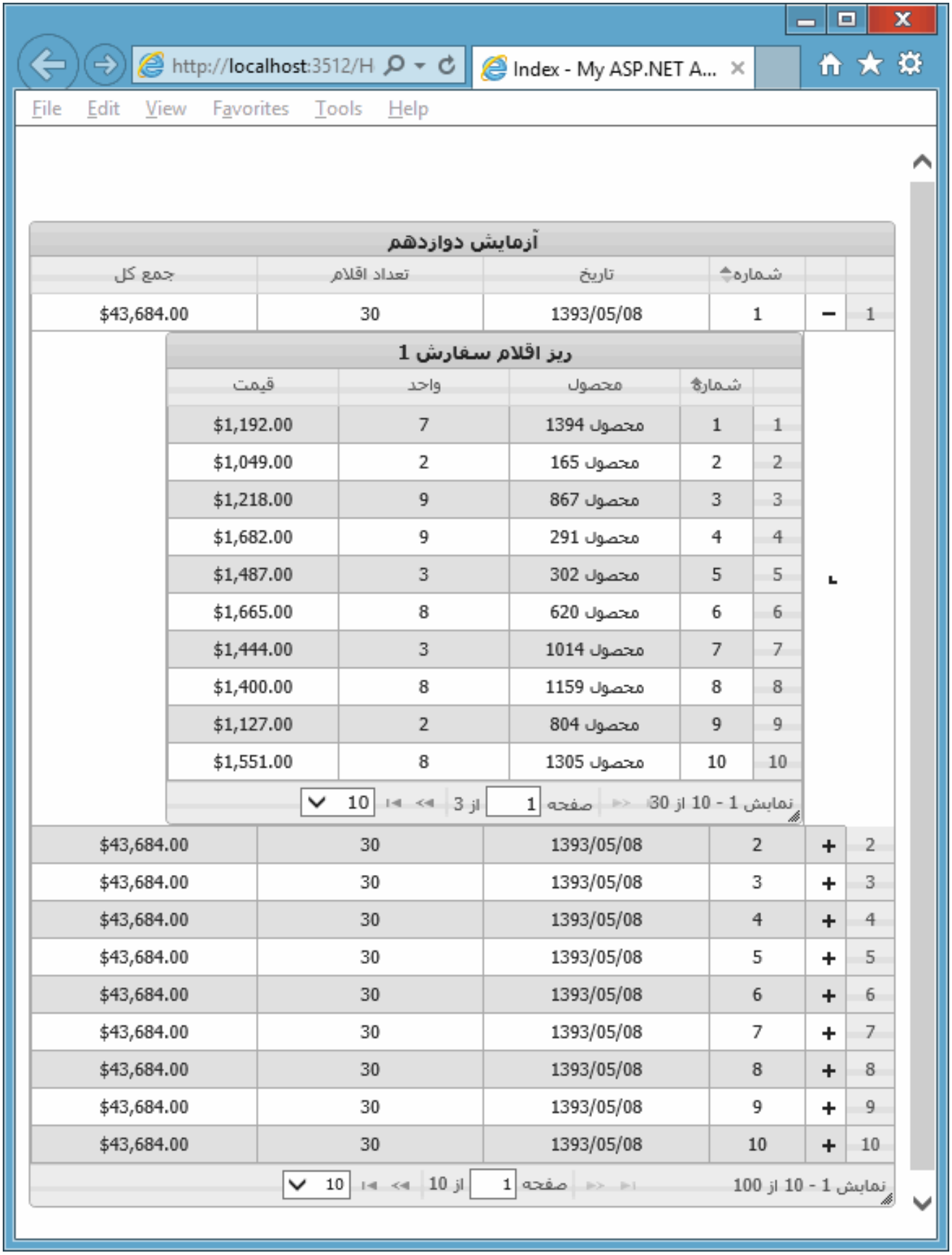
سلام
ممنون از مقاله خوبتون ، سوالی که دارم اینه که اگر بخواهیم گریدی چند سطحی داشته باشیم باید چه کار کنیم به عنوان مثال اگر بخواهیم گریدی به شکل درختی داشته باشیم و دکمه به علاوه در تمام سطح‌ها وجود داشته باشد نه فقط در یک سطح همون طوری که در مثال شما موجود می‌باشد.ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۰۸ ۱۲:۵۵

« ایجاد زیر گریدهای چند سطحی در jqGrid »

عنوان: **ایجاد زیر گریدهای چند سطحی در jqGrid**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۰۸ ۱۲:۵۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, jqGrid

همانطور که در مطلب ایجاد زیرگریدها در jqGrid مشاهده کردید، هرچند این قابلیت برای نمایش لیست ساده‌ای از عناصر مفید است اما ... امکانات آنچنانی را به همراه ندارد. برای مثال صفحه بندی، جستجو، سفارشی سازی عناصر و غیره را به همراه ندارد. اگر علاقمند باشید که این امکانات را نیز اضافه کنید، می‌توان این زیرگرید را با یک گرید کامل jqGrid نیز جایگزین کرد. همچنین اگر نیاز بود، این گرید جدید چون یک jqGrid کامل است، باز هم می‌توان یک سطح دیگر را به آن افزود و الی آخر.


http://localhost:3512/H
Index - My ASP.NET A...
File Edit View Favorites Tools Help
آزمایش دوازدهم
شماره
تاریخ
تعداد اقلام
جمع کل
1 1393/05/08 30 $43,684.00
ریز اقلام سفارش 1
شماره
محصول
واحد
قیمت
1 محصول 1394 7 $1,192.00
2 محصول 165 2 $1,049.00
3 محصول 867 9 $1,218.00
4 محصول 291 9 $1,682.00
5 محصول 302 3 $1,487.00
6 محصول 620 8 $1,665.00
7 محصول 1014 3 $1,444.00
8 محصول 1159 8 $1,400.00
9 محصول 804 2 $1,127.00
10 محصول 1305 8 $1,551.00
نمایش 1 - 10 از 30
صفحه 1 از 3
10
2 1393/05/08 30 $43,684.00
3 1393/05/08 30 $43,684.00
4 1393/05/08 30 $43,684.00
5 1393/05/08 30 $43,684.00
6 1393/05/08 30 $43,684.00
7 1393/05/08 30 $43,684.00
8 1393/05/08 30 $43,684.00
9 1393/05/08 30 $43,684.00
10 1393/05/08 30 $43,684.00
نمایش 1 - 10 از 100
صفحه 1 از 10
10

جایگزین کردن یک Subgrid با یک jqGrid کامل

خلاصه‌ی عملیات جایگزینی یک Subgrid را توسط یک jqGrid کامل، در ذیل مشاهده می‌کنید:

```
$('#list').jqGrid({
                caption: "آزمایش دوازدهم",
                //.....مانند قبل
                subGrid: true,
                subGridRowExpanded: grid1RowExpanded
            });

        function grid1RowExpanded(subGridId, rowId) {
            var subgridTableId = subGridId + "_t";
            var pagerId = "p_" + subgridTableId;
            var container = 'g_' + subGridId;
            $("#" + subGridId).html('<div dir="rtl" id="' + container + '" style="width:100%; height:
100%">' +
                '<table id="' + subgridTableId + '" class="scroll"></table><div id="'
                + pagerId + '" class="scroll"></div>');

            var url = '@Url.Action("GetOrderDetails", "Home", routeValues: new { id = "js-id" })'
                      .replace("js-id", encodeURIComponent(rowId)); // تزریق اطلاعات سمت کاربر به
خروجی سمت سرور

            $("#" + subgridTableId).jqGrid({
                caption: "ریز اقلام سفارش " + rowId,
                autoencode: true, //security - anti-XSS
                url: url,
                //.....مانند قبل
            });
        }
```

همانند نمایش subgridهای معمولی، ابتدا subGrid: true باید اضافه شود تا ستونی با ردیف‌های + دار، ظاهر شود. اینبار توسط روال رویدادگردان subGridRowExpanded، کنترل نمایش subgrid را در دست گرفته و آن‌را با یک jqGrid جایگزین می‌کنیم.
امضای متد gridlRowExpanded، شامل id یک div است که گرید جدید در آن قرار خواهد گرفت، به همراه Id ردیفی که اطلاعات زیرگرید آن نیاز است از سرور واکشی شود.
بر مبنای subGridId، مانند قبل، یک جدول و یک div را برای نمایش jgGrid و pager آن به صفحه به صورت پویا اضافه می‌کنیم.
سپس تعاریف jqGrid آن مانند قبل است و نکته‌ی خاصی ندارد. بدیهی است گرید جدید نیز می‌تواند در صورت نیاز یک subgrid دیگر داشته باشد.
در اینجا تنها نکته‌ی مهم آن نحوه‌ی ارسال اطلاعات rowId به سرور است. اکشن متدی که قرار است اطلاعات زیر گرید را تامین کند، یک چنین امضایی دارد:

```
public ActionResult GetOrderDetails(int id, JqGridRequest request)
```

بنابراین نیاز است که به نحوی rowId را به آن ارسال کرد. مشکل اینجا است که Url.Action یک کد سمت سرور است و rowId یک متغیر سمت کاربر. نمی‌توان این متغیر را در کدهای Razor مستقیما قرار داد. اما می‌توان یک محل جایگزینی را در کدهای سمت سرور پیش بینی کرد. مثلا js-id. زمانیکه این رشته در صفحه رندر می‌شود، به صورت معمول و به کمک متد replace جاوا اسکریپت، js-id آن‌را با rowId جایگزین می‌کنیم. به این ترتیب امکان تزریق اطلاعات سمت کاربر به خروجی سمت سرور Razor میسر می‌شود.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید:

jqGrid12.zip

عنوان: **نمایش ساختارهای درختی توسط jqGrid**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۰۹ ۱۵:۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, jqGrid

jqGrid از نمایش دو ساختار درختی Nested Set model و Adjacency model پشتیبانی می‌کند. توضیحات تکمیلی و پایه‌ای را در مورد این دو روش مدل سازی اطلاعات، در مطلب « SQL Antipattern #2 » می‌توانید مطالعه کنید.
در اینجا روش Adjacency model را به علت بیشتر مرسوم بودن آن و شباهت بسیار زیاد آن به « مدل‌های خود ارجاع دهنده » بررسی خواهیم کرد.

مدل داده‌ای Adjacency

در حالت ساختار درختی از نوع مجاورت، علاوه بر خواص اصلی یک کلاس، سه خاصیت دیگر نیز باید تعریف شوند:

```
using System;

namespace jqGrid13.Models
{
    public class BlogComment
    {
        // Other properties
        public int Id { set; get; }
        public string Body { set; get; }
        public DateTime AddDateTime { set; get; }

        // for treeGridModel: 'adjacency'
        public int? ParentId { get; set; }
        public bool IsNotExpandable { get; set; }
        public bool IsExpanded { get; set; }
    }
}
```

ParentId که سبب تولید یک مدل خود ارجاع دهنده می‌شود.
IsNotExpandable به این معنا است که نود جاری آیا قرار است باز شود و فرزندی دارد یا خیر؟ اگر فرزندی ندارد باید مساوی True قرار گیرد.
IsExpanded حالت پیش فرض باز بودن یا نبودن یک نود را مشخص می‌کند.

نحوه‌ی بازگشت اطلاعات درختی از سمت سرور

در نگارش فعلی jqGrid، در حالت نمایش درختی، مباحث صفحه بندی و مرتب سازی غیرفعال هستند و کدهای مرتبط با آن که در اینجا ذکر شده‌اند، فعلا تاثیری ندارند (البته با کمی تغییر در کدهای آن، می‌توان این قابلیت را هم فعال کرد. اطلاعات بیشتر).
نکته‌ی مهم treeGrid، سه پارامتر دیگر هستند که از سمت کلاینت به سرور ارسال می‌شوند:

```
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Web.Mvc;
using jqGrid13.Models;
using JqGridHelper.DynamicSearch; // for dynamic OrderBy
using JqGridHelper.Models;
using JqGridHelper.Utils;

namespace jqGrid13.Controllers
{
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            return View();
        }
```

```
        public ActionResult GetComments(JqGridRequest request, int? nodeid, int? parentid, int?
n_level)
        {
            var list = BlogCommentsDataSource.LatestBlogComments;

            در این حالت خاص فعلا در نگارش جای جی‌کیوگرید صفحه بندی کار نمی‌کند و فعال نیست و //
محاسبات ذیل اهمیتی ندارند
            var pageIndex = request.page - 1;
            var pageSize = request.rows;
            var totalRecords = list.Count;
            var totalPages = (int)Math.Ceiling(totalRecords / (float)pageSize);

            var productsQuery = list.AsQueryable();

            if (nodeid == null)
            {
                productsQuery = productsQuery.Where(x => x.ParentId == null);
            }
            else
            {
                productsQuery = productsQuery.Where(x => x.ParentId == nodeid.Value);
            }

            var products = productsQuery.OrderBy(request.sidx + " " + request.sord)
                                        .Skip(pageIndex * pageSize)
                                        .Take(pageSize)
                                        .ToList();


            var newLevel = n_level == null ? 0 : n_level.Value + 1;
            var productsData = new JqGridData
            {
                Total = totalPages,
                Page = request.page,
                Records = totalRecords,
                Rows = (products.Select(comment => new JqGridRowData
                {
                    Id = comment.Id,
                    RowCells = new List<object>
                               {
                                   comment.Id,
                                   comment.Body,
                                   comment.AddDateTime.ToPersianDate(),
                                   // اطلاعات خاص نمایش درختی به ترتیب
           newLevel,
           comment.ParentId == null ? "" :
comment.ParentId.Value.ToString(CultureInfo.InvariantCulture),
           comment.IsNotExpandable,
           comment.IsExpanded
                               }
                })).ToList()
            };
            return Json(productsData, JsonRequestBehavior.AllowGet);
        }
    }
}
```

nodeid اگر نال بود، یعنی کل اطلاعات ریشه‌ها (مواردی که parentId مساوی نال دارند)، باید واکشی شوند. اگر nodeid مقدار داشت، یعنی فرزند نود جاری قرار است بازگشت داده شود.

n_level مقدار جلو رفتگی نمایش اطلاعات یک نود را مشخص می‌کند. در اینجا چون با کلیک بر روی هر نود، فرزند آن از سرور واکشی می‌شود و lazy loading برقرار است، بازگشت مقدار n_level دریافتی از کلاینت به علاوه یک، کافی است. اگر نیاز است تمام نودها باز شده نمایش داده شوند، این مورد را باید به صورت دستی محاسبه کرده و در مدل BlogComment پیش بینی کنید.

در نهایت آرایه‌ای از خواص مدنظر به همراه 4 خاصیت ساختار درختی باید به ترتیب بازگشت داده شوند.

آزمایش سیزدهم			
	شماره	نظر	تاریخ
1	1	نظر ریشه 1	1393/05/09
1	2	پاسخ 1 به ریشه 1	1393/05/09
1	3	پاسخ 1 به پاسخ 2	1393/05/09
2	4	پاسخ 2 به ریشه 1	1393/05/09
1	5	پاسخ 1 به پاسخ 4	1393/05/09
3	6	پاسخ 3 به ریشه 1	1393/05/09
1	7	پاسخ 1 به پاسخ 6	1393/05/09
4	8	پاسخ 4 به ریشه 1	1393/05/09
1	9	پاسخ 1 به پاسخ 8	1393/05/09
5	10	پاسخ 5 به ریشه 1	1393/05/09
1	11	پاسخ 1 به پاسخ 10	1393/05/09
2	12	نظر ریشه 12	1393/05/09
3	23	نظر ریشه 23	1393/05/09
4	34	نظر ریشه 34	1393/05/09
5	45	نظر ریشه 45	1393/05/09
6	56	نظر ریشه 56	1393/05/09
7	67	نظر ریشه 67	1393/05/09

فعال سازی سمت کاربر treeGrid

برای فعال سازی سمت کاربر نمایش درختی اطلاعات، باید سه خاصیت ذیل تنظیم شوند:

```
$('#list').jqGrid({
                caption: "آزمایش سیزدهم",
                // .... مانند قبل
                treeGrid: true,
                treeGridModel: 'adjacency',
                ExpandColumn: '@(StronglyTyped.PropertyName<BlogComment>(x => x.Body))'
            }).jqGrid('gridResize', { minWidth: 400 });
        });
```

تنظیم treeGrid: true سبب فعال سازی treeGrid می‌شود. توسط treeGridModel حالات Nested Set model و Adjacency model قابل تنظیم هستند و ExpandColumn نام ستونی را مشخص می‌کند که قرار است فرزندان آن نمایش داده شوند.

یک نکته‌ی تکمیلی

اگر می‌خواهید دقیقا به شکل زیر برسید:

تنظیم rownumbers: true گرید را حذف کنید. همچنین ستون Id را نیز با تنظیم‌های hidden:true, key: true مخفی نمائید (در تعاریف colModel).

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید

jqGrid13.zip

برای مطالعه بیشتر

Tree Grid

Nested Set Model

Adjacency Model

نظرات خوانندگان

نویسنده: ابوالفضل رجب پور
تاریخ: ۱۳۹۳/۰۵/۲۲ ۱۲:۳۲

- آیا در این ساختار سلسله مراتبی adjacency می‌شود inline edit انجام داد؟
در صفحه‌ی ویکی پلاگین، این خط در قسمت محدودیت‌های adjacency نوشته شده و سپس خط خورده.

آیا راه حلی هست؟
- در ادامه سوال بالا، فرض بفرمایید یک سری دسته بندی و یک سری محصول داریم. می‌خواهیم این دسته بندی‌ها با عمق نامحدود را باز کنیم و وقتی به یک محصولی رسید، بتواند inline edit کند. این مدل رو با کدام حالت باید پیاده کرد؟ گروه بندی grouping یا سلسله مراتبی treemode ؟ نامحدود بودن عمق گروه‌های محصولات را هم در نظر بگیرید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۲۲ ۱۲:۵۱

چند مثال در این زمینه:
Functionality -> Add tree node (مثال رسمی)
Insert a row (روی صفحه کلیک راست کرده و سورس آن‌را مطالعه کنید)
Add new rows to jqGrid Treegrid model (در استک اورفلو جستجو کنید، مطالب خوبی در مورد jqGrid دارد. تا امروز 8000 سؤال مرتبط دارد. امکان ندارد در زمینه‌ی jqGrid مشکلی داشته باشید و در آنجا مطرح نشده باشد)