

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، صرفاً 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:

آزمایش چهارم							
		قیمت	گروه	تولید کننده	نام محصول	شماره	
		\$0.00	گروه 1	شرکت 1	نام 1	1	<input type="checkbox"/>
		\$1,000.00	گروه 2	شرکت 2	نام 2	2	<input type="checkbox"/>
		\$2,000.00	گروه 3	شرکت 3	نام 3	3	<input type="checkbox"/>
		\$3,000.00	گروه 4	شرکت 4	نام 4	4	<input type="checkbox"/>
		\$4,000.00	گروه 5	شرکت 5	نام 5	5	<input type="checkbox"/>
		\$5,000.00	گروه 6	شرکت 6	نام 6	6	<input type="checkbox"/>
		\$6,000.00	گروه 7	شرکت 7	نام 7	7	<input type="checkbox"/>
		\$7,000.00	گروه 8	شرکت 8	نام 8	8	<input type="checkbox"/>
		\$8,000.00	گروه 9	شرکت 9	نام 9	9	<input type="checkbox"/>
		\$9,000.00	گروه 10	شرکت 10	نام 10	10	<input type="checkbox"/>

نمایش 1 - 10 از 500 صفحه 1 از 50


روش ویرایش

☐ داخل ردیف ☐ به صورت فرم

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:

آزمایش چهارم							
	شماره	نام محصول	تولید کننده	گروه	قیمت		
 	1	نام 1	شرکت 1	گروه 1	0.00	<input checked="" type="checkbox"/>	1

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` می توان `editoptions` و سپس `dataUrl` آن را 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 `SuppliersSelect` اشاره می کند که نهایتاً لیستی از تولید کننده ها را توسط `partial` `view` ذیل بازگشت می دهد:

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

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

```
$('#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 می تواند 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 می‌شود تا در فوتر گرید، سه دکمه‌ی افزودن، ویرایش و حذف ردیف‌ها ظاهر شوند:

		\$8,000.00	گروه 9	شرکت 9	نام 9	9	<input type="checkbox"/>	9
		\$9	گروه 10	شرکت 10	نام 10	10	<input type="checkbox"/>	10

نمایش 1 - 10 از 500

صفحه 1 از 50

حذف ردیف‌های انتخابی

+

10

1

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

آزمایش چهارم

	شماره	نام محصول	تولید کننده	گروه	قیمت	
	1	نام 1			\$0.00	
	2	نام 2			\$1,000.00	
	3	نام 3			\$2,000.00	
	4	نام 4			\$3,000.00	
	5	نام 5			\$4,000.00	
	6	نام 6			\$5,000.00	
	7	نام 7			\$6,000.00	
	8	نام 8			\$7,000.00	
	9	نام 9			\$8,000.00	
	10	نام 10	شرکت 10	گروه 10	\$9,000.00	

نمایش 1 - 10 از 500

صفحه 1 از 50
 << < > >>

روش ویرایش
 داخل ردیف ☒ به صورت فرم ☐

افزافه کردن رکورد

نام محصول

تولید کننده ☐ شرکت 1 ☐

گروه ☐ گروه 1 ☐

قیمت

ثبت انصراف

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

آزمایش چهارم

	شماره	نام محصول	تولید کننده	گروه	قیمت	
	1	نام 1			\$0.00	
	2	نام 2			\$1,000.00	
	3	نام 3			\$2,000.00	
	4	نام 4			\$3,000.00	
	5	نام 5			\$4,000.00	
	6	نام 6			\$5,000.00	
	7	نام 7			\$6,000.00	
	8	نام 8			\$7,000.00	
	9	نام 9			\$8,000.00	
	10	نام 10	شرکت 10	گروه 10	\$9,000.00	

نمایش 1 - 10 از 500

صفحه 1 از 50
 << < > >>

روش ویرایش
 داخل ردیف ☒ به صورت فرم ☐

ویرایش رکورد

نام محصول

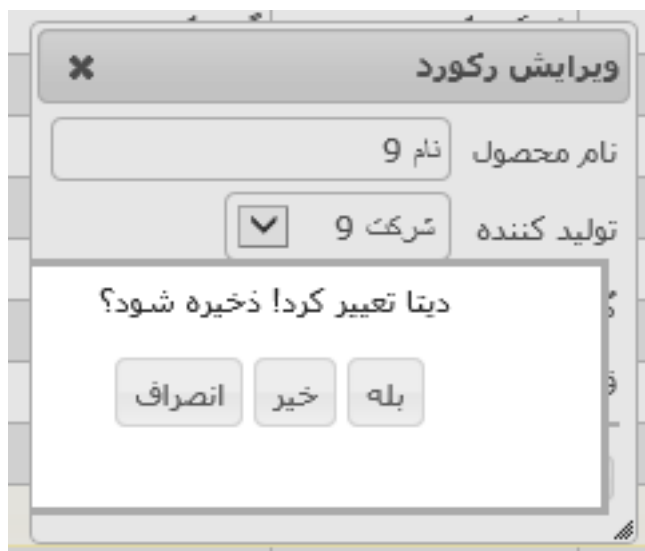
تولید کننده ☐ شرکت 9 ☐

گروه ☐ گروه 9 ☐

قیمت

ثبت انصراف

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 checkbox 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 selarrow 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 آن را بخرید.

- اخیرا یک نسخه‌ی سبک‌تر از Kendo UI با مجوز [BSD](#) ارائه شده که Grid آن را ندارد (به عمد). بنابراین از این لحاظ، مجوز jqGrid بهتر است. مجوز عمومی آن [MIT](#) است و در هر نوع پروژه‌ای قابل استفاده است. مجوز تجاری هم دارد برای حالتیکه بخواهید کامپوننت‌های ASP.NET آن را [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

« [سفرارشی 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>";
```


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: ۱۵:۱۵ ۱۳۹۳/۰۵/۲۱

تشکر از پاسختون.

با تنظیم نوع خروجی به json کار کرد

```
[HttpGet]
public IActionResult 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: ۱۵:۳۴ ۱۳۹۳/۰۵/۲۱

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());
```

نویسنده: سروش

تاریخ: ۱۳:۱۱ ۱۳۹۳/۰۹/۰۶

با سلام؛ من می‌خواهم در هنگام ارسال اطلاعات به سرور ValidateAntiForgeryToken رو هم بفرستم. چطور میشه اینکارو انجام بدم، روش‌های مختلفی رو جستجو و امتحان کردم ولی جواب نداد.

نویسنده: وحید نصیری

تاریخ: ۱۳:۳۸ ۱۳۹۳/۰۹/۰۶

این روش عمومی است و با jqGrid هم امتحان شد، کار می‌کند:

```
$(document).ready(function () {
    var securityToken = $('[name=__RequestVerificationToken]').val();
    $('body').bind('ajaxSend', function (elm, xhr, s) {
        if (s.type == 'POST' && typeof securityToken != 'undefined') {
            if (s.data.length > 0) {
                s.data += "&__RequestVerificationToken=" + encodeURIComponent(securityToken);
            }
            else {
                s.data = "__RequestVerificationToken=" + encodeURIComponent(securityToken);
            }
        }
    });
});
```

در این حالت فرقی نمی‌کند که از چه ابزاری برای ارسال اطلاعات به سرور استفاده می‌کنید. همینقدر که در پشت صحنه از JQuery Ajax استفاده می‌کند، رخداد ajaxSend آن هوک شده و پارامترهای لازم، به اطلاعات ارسالی به سرور اضافه می‌شوند.

نویسنده: سروش
تاریخ: ۱۶:۵۹ ۱۳۹۳/۰۹/۰۶

property خاصی از jqgrid باید مقدار دهی بشه؟ متاسفانه جواب نداد!

نویسنده: وحید نصیری
تاریخ: ۱۷:۴۲ ۱۳۹۳/۰۹/۰۶

@Html.AntiForgeryToken() را روی صفحه قرار دادید؟ بررسی `name=__RequestVerificationToken` در کد فوق، بر مبنای وجود این فیلد در صفحه است.

آزمایش چهارم

	نام محصول	تولید کننده	گروه	قیمت	
1	نام 1	شرکت 1	گروه 1	\$0.00	
2	نام 2	شرکت 2	گروه 2	\$1,000.00	
3	نام 3	شرکت 3	گروه 3	\$2,000.00	
4	نام 4	شرکت 4	گروه 4	\$3,000.00	
5	نام 5	شرکت 5	گروه 5	\$4,000.00	
6	نام 6	شرکت 6	گروه 6	\$5,000.00	
7	نام 7	شرکت 7	گروه 7	\$6,000.00	
8	نام 8	شرکت 8	گروه 8	\$7,000.00	
9	نام 9	شرکت 9	گروه 9	\$8,000.00	
10	نام 10	شرکت 10	گروه 10	\$9,100.00	

نمایش 1 - 10 از 500 صفحه 1 از 50

روش ویرایش
☐ داخل ردیف
☐ به صورت فرم

Network
 SUMMARY DETAILS 1 / 1 http://localhost:3504/Home/EditProduct
 Request headers Request body Response headers Response body Cookies Initiat
 1 0&Price=9100.00&oper=edit&Id=10&__RequestVerificationToken=DQUc0M-GpM3jQJHGT1HbpI

نویسنده: سروش
تاریخ: ۸:۳۹ ۱۳۹۳/۰۹/۰۸

سلام

بله @Html.AntiForgeryToken() در صفحه وجود دارد.

نویسنده: وحید نصیری
تاریخ: ۱۲:۴۰ ۱۳۹۳/۰۹/۰۸

قبل از ارسال کدی که ملاحظه کردید، [این موارد طی شدند](#) :

- به ابتدای View مثال 4 (مثال بحث جاری که کدهای کامل آن در پایان مطلب پیوست شده‌اند) ابتدا @Html.AntiForgeryToken() اضافه شد.

- بعد در قسمت اسکریپت‌های صفحه، کد مرتبط با ajaxSend که توکن امنیتی را به انتهای اطلاعات درخواست اضافه می‌کند، اضافه شد.

- در کدهای کنترلر، روی اکشن متدهای ثبت، ویرایش و حذف، ویژگی ValidateAntiForgeryToken اضافه شد. تصویر فوق هم بر همین مبنا تهیه شده‌است.

برای آزمایش بیشتر، کدهای اسکریپتی ajaxSend حذف شدند. بعد سعی در ویرایش. نتیجه دریافت استثناء از طرف سرور بود. با برگرداندن کدهای اسکریپتی، مشکل صدور استثنای نبود توکن امنیتی برطرف شد.

نویسنده: امیر نوروزیان
تاریخ: ۲۱:۲۲ ۱۳۹۳/۱۰/۰۸

سلام

زمانی که می‌خام حذف کنم بجای اینکه حذف براساس ایدی بانک باشه براساس ایدی ردیف گرید هست

نویسنده: وحید نصیری
تاریخ: ۱:۲۲ ۱۳۹۳/۱۰/۰۹

- مثال‌های سری jqGrid تغییرات زیادی داشتند. برای دریافت آن‌ها [به این مخزن کد](#) مراجعه کنید.
- برای نمونه، [این فایل](#) بهبود یافته مثال جاری است. در آن نحوه‌ی تعریف ستون Id، به صورت مخفی و کلید، معرفی شده.
همچنین در ستون actions آن نحوه‌ی معرفی آدرس حذف به نحو بهتری درج شده‌است. به علاوه نحوه‌ی استفاده از anti-forgery token در آن ذکر شده، به همراه StronglyTyped.PropertyName ها.

نویسنده: امیر نوروزیان
تاریخ: ۲۱:۱۸ ۱۳۹۳/۱۰/۰۹

ممنون. ولی تو تعریف کدهای شما، ردیف رو به عنوان ایدی به تابع حذف می‌فرسته و با این تابع میشه نوع داده ارسالی رو مشخص کرد

```
onClickButton: function (e) {
    var id = $('#grid').jqGrid('getGridParam', 'selrow');

    if (id != null) {
        var idHtml = $('#grid').jqGrid('getCell', id, 'ID_Baker');
        //var idHtml = id.ID_Baker;
        //var idHtml = $('#'+ id).children().first().html();
        $.ajax({
            url: '@(Url.Action("DeleteProfile"))' + '?Id=' + idHtml + '&RowId=' + id,
            success: function (data) {
                if (data.Success) {
                    $('#grid').jqGrid('delRowData', data.RowId);
                }
            }
        });
    }
    else {
    }
}
```

},

نویسنده: وحید نصیری
تاریخ: ۲۲:۳ ۱۳۹۳/۱۰/۰۹

[مثال به روز شده](#) را اجرا کردید؟ شماره Id اولین ردیف را به عمد اینبار [از 100](#) تنظیم کردم (تا با شماره ردیف اشتباه نشود) و این حاصل کلیک بر روی دکمه‌ی حذف اولین ردیف است؛ بدون نیاز به کد اضافه‌تری (در این مثال، حذف ردیف شماره یک، عدد id مساوی 100 را به سرور ارسال کرده):

```

55 [ValidateAntiForgeryToken]
56 [HttpPost]
57 public ActionResult DeleteProduct(string id)
58 {
59     //todo: Delete product
60     var product = ProductDataSource.LatestProducts.FirstOrDefault(x => x.Id == int.Parse(id));
61     if (product == null)
62         return Json(new { success = false }, JsonRequestBehavior.AllowGet);
63
64     ProductDataSource.LatestProducts.Remove(product);
65
66     return Json(new { success = true }, JsonRequestBehavior.AllowGet);
67 }

```