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