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