بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

آموزش کنترل ASP.NET – validation

مدرس : مهندس افشین رفوآ

آموزش کنترل ASP.NET – validation

کنترل های ASP.NET داده های ورودی کاربر (user input data) را اعتبارسنجی کرده تا از این طریق از ذخیره شدن داده های نامعتبر، بلااستفاده و مغایر جلوگیری شود.

ASP.NET کنترل های validation زیر را ارائه می دهد:

- RequiredFieldValidator
- RangeValidator
- CompareValidator
- RegularExpressionValidator
- CustomValidator
- ValidationSummary

کلاس BaseValidator

کلاس های کنترل validation از کلاس BaseValidator به ارث گرفته می شوند و از این رو تمامی خاصیت ها و توابع آن را نیز به ارث می برند. بدین ترتیب نگاه مختصری بر خواص و متدهای این کلاس پایه که بین کلیه ی کنترل های validation مشترک می باشد، کمک شایانی خواهد کرد.

شرح	خواص (property) و توابع

ControlToValidate	کنترل ورودی (input control) که باید اعتبار سنجی شود را مشخص می کند.
Display	نحوه ی نمایش پیغام خطا را تعیین می کند.
EnableClientScript	یک مقدار گرفته یا تنظیم می کند که آن مقدار نشانگر فعال بودن یا نبودن اعتبار سنجی سمت سرویس گیرنده (client-side validation) می باشد.
Enabled	Validator را فعال / غیر فعال می سازد.
ErrorMessage	متن پیام خطا که در کنترل ValidationSummary به هنگام شکست خوردن validation (اعتبار سنجی) نمایش داده می شود را تعیین می کند.
Text	متن خطایی (error text) که باید در صورت مواجه شدن اعتبار سنجی با شکست نشان داده شود را مشخص می کند.
IsValid	مشخص می کند آیا مقدار کنترل مورد نظر معتبر است یا خیر.
SetFocusOnError	مشخص می کند آیا در صورت برخورد با یک کنترل نامعتبر، کنترل ورودی (input control) مربوطه باید انتخاب شود یا خیر.
ValidationGroup	اسم گروه منطقی که کنترل validation مورد نظر به آن تعلق دارد را برمی گرداند.
Validate()	این تابع کنترل را مجدد ارزیابی کرده و خاصیت IsValid را بروز رسانی می کند.

کنترل RequiredFieldValidator

کنترل RequiredFieldValidator اطمینان کسب می کند که فیلد مورد نیاز تهی نباشد. کنترل ذکر شده معمولا به یک textbox متصل می باشد که باعث می شود ورودی به داخل text box هدایت شود.

دستور نگارش کنترل مزبور به ترتیب زیر می باشد:

```
<asp:RequiredFieldValidator ID="rfvcandidate"
runat="server" ControlToValidate="ddlcandidate"
ErrorMessage="Please choose a candidate"
InitialValue="Please choose a candidate">

</asp:RequiredFieldValidator>
```

کنترل RangeValidator

کنترل RangeValidator بررسی می کند آیا مقدار ورودی (input value) داخل رینج یا دامنه ی از پیش تعریف شده قرار می گیرد یا خیر.

این کنترل دارای سه خاصیت ویژه می باشد که به شرح زیر است:

شرح	خواص (properties)

Type	این خاصیت نوع داده را تعریف می کند. مقادیر موجود عبارتند از: نوع Double، Currency، Date، Integer (عدد صحیح) و string (رشته ای).
MinimumValue	minimum value of the range. کمینه ی مقدار دامنه اعتبارسنجی (validation range) را مشخص می کند.
MaximumValue	maximum value of the range. حداکثر یا بیشینه ی مقدار دامنه اعتبارسنجی (validation range) را مشخص می کند.

دستور نگارش این کنترل به صورت زیر می باشد:

```
<asp:RangeValidator ID="rvclass" runat="server" ControlToValidate="txtclass"
 ErrorMessage="Enter your class (6 - 12)" MaximumValue="12"
 MinimumValue="6" Type="Integer">
</asp:RangeValidator>
```

کنترل CompareValidator

Properties	Description
Type	این خاصیت نوع داده را مشخص می کند.
ControlToCompare	مقدار کنترل ورودی (input control) که باید مقایسه با آن صورت گیرد را تعیین می کند.
ValueToCompare	مقدار ثابتی که باید مقایسه با آن انجام گیرد را مشخص می کند.
Operator	این خاصیت عملگر مقایسه (comparison operator) را برای پیاده کردن عملیات مقایسه مشخص می کند، مقادیر موجود به ترتیب زیر می باشند: Equal، NotEqual، GreaterThan، GreaterThanEqual، LessThan ، LessThanEqual و DataTypeCheck .

کنترل CompareValidator مقدار موجود در یک کنترل را با یک مقدار ثابت در کنترل دیگر مقایسه می کند.

دارای خواص زیر می باشد:

دستور نگارش کنترل مزبور به صورت زیر می باشد:

```
<asp:CompareValidator ID="CompareValidator1" runat="server"
   ErrorMessage="CompareValidator">
</asp:CompareValidator>
```

RegularExpressionValidator

این کنترل بررسی می کند آیا مقدار یک کنترل ورودی (input control) با الگوی مشخص شده توسط یک عبارت با قاعده (regular expression) همخوانی دارد یا خیر. regular expression در خاصیت ValidationExpression مقداردهی یا تنظیم می شود (به منظور اعتبار سنجی ورودی هایی که به وسیله ی کنترل های سرور ضبط شده اند از این کنترل بهره می گیریم).

جدول زیر ساختارهای نحوی (syntax construct) معمول که برای regular expression ها بکار می روند را فهرست می کند.

Character Escapes	شرح
\b	با backspace مطابقت دارد.
\t	با کاراکتر تَب مطابقت دارد.
\r	با کاراکتر carriage return (رفتن سر سطر) مطابقت دارد.
\v	با vertical tab (جدول بندی عمودی) مطابقت دارد.
\f	با form feed (سر گرفتن پرینت از بالای صفحه ی دیگر) مطابقت دارد.
\n	با کاراکتر new line مطابقت دارد.
\	Escape character.

جدا از تطبیق کاراکتر به صورت تکی، کلاسی از کاراکترها را می توان مشخص کرده که قابلیت تطبیق دادن (matching) آن ها وجود دارد. چنین کاراکترهایی metacharacter خوانده می شوند. متا کاراکتر در واقع کاراکتری است که بجای معنای تحت لفظی، معنای خاصی برای برنامه ی کامپیوتر داشته باشد.

Metacharacters	شرح
.	با هر کاراکتری به جز \n مطابقت دارد.
[abcd]	با هر کاراکتر موجود در مجموعه کاراکترها مطابقت دارد.
[^abcd]	با هیچ یک از کاراکترهای موجود در مجموعه مطابقت نمی کند.
[2-7a-mA-M]	با هر کاراکتری که در دامنه (range) مورد نظر مشخص شده باشد مطابقت می کند.
\w	با کاراکترهای الفبای عددی و زیرین خط (underscore) مطابقت می کند.
\W	با کاراکترهایی که کلمه نیستند مطابقت می کند.
\s	با تمامی کاراکترهایی که فاصله ایجاد می کنند مانند space، tab، new line مطابقت دارد.
\S	با هر کاراکتری که فاصله ایجاد نمی کند مطابقت می کند.
\d	با کاراکترهای اعشاری (decimal) مطابقت دارد.
\D	با هر کاراکتری که اعشاری نباشد مطابقت دارد.

با استفاده از کمیت سنج (quantifier) می توان تعداد دفعاتی که یک کاراکتر می تواند ظاهر شود را تعیین کرد.

Quantifier (کمیت سنج)	توضیح

صفر یا بیشتر تکرار.	*
یک یا بیشتر تکرار.	+
صفر یا حداکثر یک تکرار.	?
N matches.	{N}
N یا بیشتر تکرار.	{N,}
بین تکرارهای N و M.	{N,M}

دستور نگارش کنترل ذکر شده به ترتیب زیر می باشد:

```
<asp:RegularExpressionValidator ID="string" runat="server" ErrorMessage="string"
   ValidationExpression="string" ValidationGroup="string">
</asp:RegularExpressionValidator>
```

CustomValidator

این کنترل همان طور که از اسم آن پیدا است به برنامه نویس امکان می دهد برای اعتبار سنجی هر دو سمت سرویس گیرنده و سرویس دهنده، روال های (routine) اعتبار سنجی سفارشی یا اختصاصی مختص اپلیکیشن بنویسد.

اعتبار سنجی سمت سرویس گیرنده از طریق خاصیت ClientValidationFunction صورت می گیرد. روال های اعتبار سنجی سمت سرویس گیرنده (client-side validation routine) باید توسط زبان های اسکریپت نویسی همچون JavaScript یا VBscript نوشته شود تا مرورگر قادر به خواندن و اجرای آن ها باشد.

روال اعتبار سنجی سمت سرویس گیرنده (server-side validation routine) باید از مدیریت کننده ی رخداد ServerValidate (event-handler) کنترل مورد نظر فراخوانده شود. روال اعتبار سنجی (validation routine) سمت سرویس دهنده باید حتما با استفاده از زبان های .NET همچون C# یا VB.NET نوشته شود.

دستور نگارش (syntax) کنترل نام برده به شرح زیر می باشد:

```
<asp:CustomValidator ID="CustomValidator1" runat="server"
   ClientValidationFunction=.cvf_func. ErrorMessage="CustomValidator">
</asp:CustomValidator>
```

کنترل ValidationSummary

این کنترل خود هیچ گونه اعتبار سنجی انجام نمی دهد و فقط خلاصه ای از تمام خطاها در صفحه را نمایش می دهد. این خلاصه کلیه ی مقادیر خاصیت ErrorMessage (property) تمامی validation control ها که در اعتبار سنجی دارای مقادیر نامعتبر هستند و از این رو رد می شوند را به نمایش می گذارد.

دو خاصیت زیر (که امکان استفاده ی هر دوی آن ها در یک تگ وجود دارد یا به عبارتی دیگر mutually inclusive هستند) پیام های خطا را فهرست می کنند:

ShowSummary : پیام های خطا را در فرمت یا قالب مشخص شده نشان می دهد.

ShowMessageBox : پیام های خطا را در یک پنجره ی مجزا به نمایش می گذارد.

دستور نگارش این کنترل به ترتیب زیر می باشد:

```
<asp:ValidationSummary ID="ValidationSummary1" runat="server"
DisplayMode = "BulletList" ShowSummary = "true" HeaderText="Errors:" />
```

Validation Groups

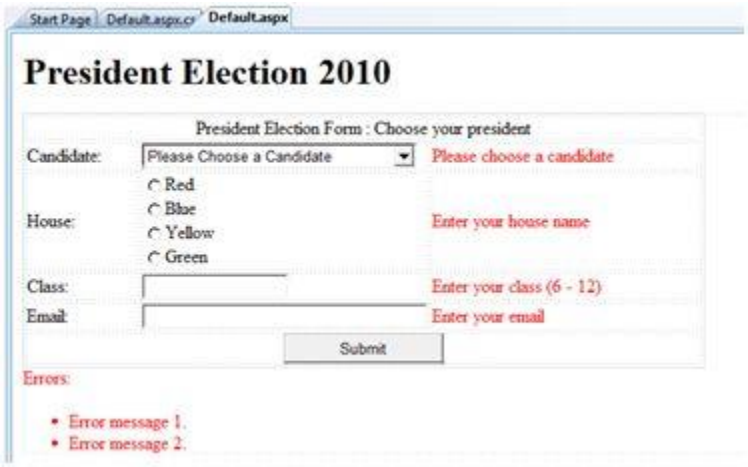
صفحات پیچیده دارای مجموعه های مختلفی از اطلاعات هستند که در صفحات (panel) متعددی نشان داده می شوند. در چنین شرایطی، ممکن است نیاز باشد هر گروه یا مجموعه اطلاعات را به صورت جداگانه اعتبار سنجی کنیم. برای این منظور از validation group (گروه های اعتبار سنجی) کمک می گیریم.

به منظور ایجاد یک validation group، باید کنترل های ورودی (input control) و کنترل اعتبار سنجی (validation control) را در گروه منطقی یکسان قرار دهیم، این کار با بکار بردن خاصیت ValidationGroup صورت می گیرد.

مثال:

مثال زیر یک فرم تعریف می کند که به چهار گروه تقسیم شده و برای انتخاب مدیر مدرسه باید توسط دانش آموزان پر شود. حال با استفاده از validation group ورودی کاربران را اعتبار سنجی می کنیم.

فرم مورد نظر در design view بدین صورت خواهد بود:



کد content file:

```
<form id="form1" runat="server">

    <table style="width: 66%;">

        <tr>
            <td class="style1" colspan="3" align="center">
                <asp:Label ID="lblmsg"
                    Text="President Election Form : Choose your president"
                    runat="server" />
            </td>
        </tr>

        <tr>
            <td class="style3">Candidate:
            </td>

            <td class="style2">
                <asp:DropDownList ID="ddlcandidate" runat="server" Style="width: 239px">
                    <asp:ListItem>Please Choose a Candidate</asp:ListItem>
                    <asp:ListItem>M H Kabir</asp:ListItem>
```

```
            <asp:ListItem>Steve Taylor</asp:ListItem>
            <asp:ListItem>John Abraham</asp:ListItem>
            <asp:ListItem>Venus Williams</asp:ListItem>
        </asp:DropDownList>
    </td>

    <td>
        <asp:RequiredFieldValidator ID="rfvcandidate"
            runat="server" ControlToValidate="ddlcandidate"
            ErrorMessage="Please choose a candidate"
            InitialValue="Please choose a candidate">
        </asp:RequiredFieldValidator>
    </td>
</tr>

<tr>
    <td class="style3">House:
    </td>

    <td class="style2">
        <asp:RadioButtonList ID="rblhouse" runat="server" RepeatLayout="Flow">
            <asp:ListItem>Red</asp:ListItem>
            <asp:ListItem>Blue</asp:ListItem>
            <asp:ListItem>Yellow</asp:ListItem>
            <asp:ListItem>Green</asp:ListItem>
        </asp:RadioButtonList>
    </td>

    <td>
        <asp:RequiredFieldValidator ID="rfvhouse" runat="server"
            ControlToValidate="rblhouse" ErrorMessage="Enter your house name">
        </asp:RequiredFieldValidator>
        <br />
    </td>
</tr>

<tr>
    <td class="style3">Class:
    </td>

    <td class="style2">
        <asp:TextBox ID="txtclass" runat="server"></asp:TextBox>
    </td>

    <td>
        <asp:RangeValidator ID="rvclass"
            runat="server" ControlToValidate="txtclass"
            ErrorMessage="Enter your class (6 - 12)" MaximumValue="12"
            MinimumValue="6" Type="Integer">
        </asp:RangeValidator>
    </td>
</tr>

<tr>
    <td class="style3">Email:
    </td>

    <td class="style2">
        <asp:TextBox ID="txtemail" runat="server" Style="width: 250px">
        </asp:TextBox>
    </td>

    <td>
        <asp:RegularExpressionValidator ID="remail" runat="server"
            ControlToValidate="txtemail" ErrorMessage="Enter your email"
            ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*">
        </asp:RegularExpressionValidator>
    </td>
</tr>
```

```
        <tr>
            <td class="style3" align="center" colspan="3">
                <asp:Button ID="btnsubmit" runat="server" OnClick="btnsubmit_Click"
                    Style="text-align: center; width: 140px;" Text="Submit" />
            </td>
        </tr>
    </table>
    <asp:ValidationSummary ID="ValidationSummary1" runat="server"
        DisplayMode="BulletList" ShowSummary="true" HeaderText="Errors:" />
</form>
```

Code behind دکمه ی ارسال (submit button):

```
    protected void btnsubmit_Click(object sender, EventArgs e)
    {
        if (Page.IsValid)
        {
            lblmsg.Text = "Thank You";
        }
        else
        {
            lblmsg.Text = "Fill up all the fields";
        }

     }
```