الفصل الثالث

سلسلة

ASP.NET

خطوة بخطوة حتى الاحتراف

C# & VB

اعداد المهندس

محمد عمر الحاج خلف

الفصل الثالث

استخدام أدوات التحقق

في هذا الفصل

- ✓ **استخدام أداة التحقق RequiredFieldValidator**
- ✓ **استخدام أداة التحقق RangeValidator**
- ✓ **استخدام أداة التحقق CompareValidator**
- ✓ **استخدام أداة التحقق RegularExpressionValidator**
- ✓ **استخدام أداة التحقق CustomValidator**
- ✓ **استخدام الأداة ValidationSummary**
- ✓ **نقاط هامة حول تطبيقات أدوات التحقق**

استخدام أدوات التحقق :

في هذا الفصل سنتعلم كيف نتحقق من مدخلات المستخدم عند محاولة إرساله بيانات نموذج ما إلى السيرفر , كما سنتعلم كيفية استخدام أدوات التحقق لمنع المستخدم من إرسال أنماط خاطئة من البيانات إلى جداول قاعدة البيانات , على سبيل المثال منع المستخدم من إرسال كلمة " تفاحة " في حقل مخصص لتاريخ الميلاد .

في القسم الأول من هذا الفصل سنتعرف على أدوات التحقق القياسية المضمنة في إطار العمل ASP.net 4.0 , حيث سنتعلم كيفية التحكم بعرض رسائل الأخطاء الناتجة عن عملية التحقق , وكيفية ضم عناصر التحقق في مجموعات , وسيتم إرفاق مثال توضيحي لكل أداة من أدوات التحقق على حدى .

بعد ذلك سنتوسع في مفاهيم أدوات التحقق , حيث سنتعلم كيفية التعامل مع أدوات التحقق الخاصة , مثلاً ً كيفية استخدام تقنية الأجاكس للتحقق "AjaxValidator control" والذي يمكننا من استدعاء تابع تحقق موجود على السيرفر من قبل العميل (client).

أدوات التحقق القياسية

تحتوي منصة العمل ASP.net 4.0 على ست أدوات قياسية للتحقق من مدخلات المستخدم وهم :

1- **RequiredFieldValidator :** تجبر المستخدم على إدخال قيمة ما في الحقل المرتبط بهذه الأداة . ومن خصائصها :
   - ✓ ControlToValidate : لتحديد أداة التحكم المرتبطة بأداة التحقق هذه .
   - ✓ ErrorMessage : لتحديد رسالة الخطأ التي ستظهر في حال عدم إدخال قيمة في أداة التحكم المرتبطة بأداة التحقق هذه .
   - ✓ InitialValue : تجبر المستخدم على إدخال قيمة مغايرة للقيمة المحددة بهذه الخاصية .

مثال :

في هذا المثال سنقوم بإنشاء نموذج بسيط , حيث يقوم المستخدم (طالب الثانوية) بإدخال رقم اكتتابه ونوع شهادته للحصول على نتائجه الامتحانية . أنشئ صفحة جديدة , أضف عليها أداة تحكم TextBox وأداة DropDownList واجعلها تحتوي على ثلاثة عناصر (غيرمحدد , علمي , أدبي) ,أضف زر سمّـه "إرسال البيانات" , أضف أداتي تحقق RequiredFieldValidator واضبط خصائصهم كالتالي :

الأداة	الخاصية	القيمة
RequiredFieldValidator1	ControlToValidate	TextBox1
RequiredFieldValidator1	ErrorMessage	املأ هذا الحقل

RequiredFieldValidator2	ControlToValidate	DropDownList1
RequiredFieldValidator2	ErrorMessage	املأ هذا الحقل
RequiredFieldValidator2	IntialValue	غيرمحدد

كود الصفحة :

كود ASP.net

```
<div>
    <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
        ErrorMessage="الحقل هذا املأ" ControlToValidate="TextBox1"
        ForeColor="Red"></asp:RequiredFieldValidator>
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    الاكتتاب رقم<br />
    <asp:RequiredFieldValidator ID="RequiredFieldValidator4" runat="server"
        ControlToValidate="DropDownList1" ErrorMessage="الحقل هذا املأ"
        ForeColor="Red" InitialValue="غيرمحدد">
    </asp:RequiredFieldValidator>
    <asp:DropDownList ID="DropDownList1" runat="server" Height="20px"
     Width="125px">
        <asp:ListItem Value="none">واحدة اختر</asp:ListItem>
        <asp:ListItem>علمي</asp:ListItem>
        <asp:ListItem>أدبي</asp:ListItem>
    </asp:DropDownList>
    الشهادة نوع<br />
    <asp:Button ID="Button1" runat="server" Text="البيانات إرسال" />
</div>
```

قم بتنفيذ الصفحة السابقة , اضغط على الزر دون ادخال بيانات ولاحظ ظهور رسائل الخطأ , لاحظ بأن حقل نوع الشهادة ليس فارغ حيث يظهر فيه النص "غيرمحدد" والذي قيمته "none" ومع ذلك فقد ظهرت رسالة الخطأ "هذا الحقل مطلوب " والسبب هو تحديد القيمة "none" للخاصية initialValue ومعنى هذا أنه يجب إدخال قيمة مغايرة للقيمة "none" في DropDownList1 .

2- **RangeValidator** : تفحص فيما إذا كانت القيمة المدخلة تقع ضمن مجال محدد من القيم أم لا . ومن خصائصها :

- ✓ ControlToValidate : لتحديد أداة التحكم المرتبطة بأداة التحقق هذه .
- ✓ ErrorMessage : لتحديد رسالة الخطأ التي ستظهر في حال إدخال قيمة تقع خارج المجال المحدد .
- ✓ MaximumValue : أعلى قيمة يمكن قبولها .
- ✓ MinimumValue : أصغر قيمة يمكن قبولها .

- ✓ Type : نمط البيانات , حيث يجب وضع نمط البيانات المناسب لتتم عملية فحص القيمة المدخلة بشكل سليم , ويأخذ إحدى القيم :

String , Integer , Double , Date, Currency .

مثال :

في نموذج إدخال علامات الامتحان حيث إن العلامة يجب أن تقع ضمن المجال من 0 إلى 100 . أنشئ صفحة جديدة , أضف عليها أداة تحكم TextBox,أضف زر سمّـه "إرسال البيانات" , أضف أداة تحقق RangeValidator واضبط خصائصها كالتالي :

الخاصية	القيمة
ControlToValidate	TextBox1
ErrorMessage	علامة خاطئة
MinimumValue	0
MaximumValue	100
Type	Integer

كود الصفحة :

كود ASP.net

```
<div>
    <asp:RangeValidator ID="RangeValidator1" runat="server"
        ErrorMessage="خاطئة علامة" ControlToValidate="TextBox1" ForeColor="Red"
        MaximumValue="100" MinimumValue="0" Type="Integer">
    </asp:RangeValidator>
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    العلامة أدخل
    <br />
    <asp:Button ID="Button1" runat="server" Text="البيانات إرسال" />
</div>
```

قم بتنفيذ الصفحة السابقة وأدخل رقم أكبر من 100 أو أصغر من 0 لتظهر لك رسالة "علامة خاطئة " , جرب بعد ذلك إدخال قيمة نصية – مثلاً : تفاحة – ستجد أن رسالة الخطأ ستظهر أيضاً قم بعد ذلك بترك الأداة TextBox1 فارغة واضغط على الزر ستلاحظ عدم ظهور رسالة الخطأ , وذلك لأن أداة التحقق هذه تقوم فقط بفحص القيمة المدخلة إن وجدت وإن لم توجد فلا تطالب بإدخالها , لإجبار المستخدم على إدخال علامة يجب استخدام الأداة RequiredFieldValidator وربطها مع TextBox1 حيث من الممكن ربط أكثر من أداة تحقق مع نفس حقل الإدخال .

3- **CompareValidator** : يوجد ثلاث استخدامات لأداة التحقق هذه :

I. التحقق من نمط البيانات المدخلة , مثلاً التحقق من أن البيانات المدخلة في حقل السعر هي من النمط Integer , وفي حقل تاريخ الميلاد من النمط Date وهكذا ..

II. مقارنة القيمة المدخلة بقيمة أخرى محددة مسبقاً , مثلاً عند بناء موقع للمزايدة العلنية يجب أن تكون القيمة المدخلة أعلى من القيمة البدائية للمزاد .

III. مقارنة القيم المدخلة في حقلي إدخال (أداتي تحكم TextBox أو غيرها) كما في حالة إدخال كلمة المرور وإعادة كتابتها مرة أخرى للتأكيد , أو للتحقق من أن تاريخ بداية الاجتماع أصغر من تاريخ انتهائه وهكذا ..

ومن خصائص أداة التحقق CompareValidator :

- ✓ ControlToValidate : لتحديد أداة التحكم المرتبطة بأداة التحقق هذه .
- ✓ ErrorMessage : لتحديد رسالة الخطأ التي ستظهر في حال عدم تحقق القيد المحدد بالخصائص الأخرى .
- ✓ Operator : معامل التحقق , أي عملية المقارنة التي يجب أن تتحقق (مساواة , عدم مساواة , أكبر , أصغر , أم التحقق من نمط البيانات المدخلة) .

Equal, NotEqual, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual, DataTypeCheck.

- ✓ ValueToCompare : قيمة ثابتة لتتم مقارنتها مع القيمة المدخلة بأداة التحكم .
- ✓ ControlToCompare : تحديد ID لأداة تحكم أخرى لتتم المقارنة بالقيمة المدخلة فيها .

Type : للتحقق من أن البيانات المدخلة هي من نمط محدد. ومكن أن تأخذ إحدى القيم String , Integer , Double , Date, Currency .

مثال :

نستعرض في هذا المثال كيفية التحقق من نمط البيانات المدخلة , سنقوم بإنشاء نموذج بسيط لإدخال تاريخ الميلاد و ستظهر رسالة خطأ إن قام المستخدم بإدخال قيمة من نمط بيانات آخر غير Date , أنشئ صفحة جديدة,أضف أدوات التحكم Button,TextBox وأداة التحقق CompareValidator واضبط فيها الخصائص التالية :

الخاصية	القيمة
ControlToValidate	TextBox1
ErrorMessage	نمط بيانات خاطئ
Type	date
Operator	DataTypeCheck

كود الصفحة :

كود ASP.net

```
<div>
    <asp:CompareValidator ID="CompareValidator1" runat="server"
        ErrorMessage="خاطئ بيانات نمط"
        ControlToValidate="TextBox1"
        ForeColor="Red"
        Operator="DataTypeCheck" Type="Date">
    </asp:CompareValidator>
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    الميلاد تاريخ<br />
    <asp:Button ID="Button1" runat="server" Text="البيانات إرسال" />
</div>
```

نفذ الصفحة السابقة وأدخل 1/1/2000 واضغط على الزر , سيتم قبول البيانات , أما لو أدخلت قيمة من نمط بيانات خاطئ فستظهر رسالة الخطأ المحددة .

مثال عن المقارنة مع قيمة ثابتة :

سنقوم في هذا المثال بإنشاء نموذج بسيط يتم فيه قبول الدخل إذا كانت القيمة المدخلة أكبر من 10 وإلا ستظهر رسالة خطأ , أنشئ صفحة جديدة , أضف أدوات التحكم Button,TextBox وأداة التحقق CompareValidator واضبط فيها الخصائص التالية :

الخاصية	القيمة
ControlToValidate	TextBox1
ErrorMessage	قيمة خاطئة
Type	Integer
Operator	GreaterThan
ValueToCompare	10

كود الصفحة :

كود ASP.net

```
<div>
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    <asp:CompareValidator ID="CompareValidator1" runat="server"
        ErrorMessage="خاطئة قيمة"
        ControlToValidate="TextBox1"
        ForeColor="Red"
        Operator="GreaterThan"
```

```
        Type="Integer"
        ValueToCompare="10">
        </asp:CompareValidator>
        <br />
        <asp:Button ID="Button1" runat="server" Text="Button" />
    </div>
```

مثال عن مقارنة قيم أدوات التحكم :

سنقوم في هذا المثال بتطبيق أحد استخدامات أدوات التحقق الشائعة , وهي إعادة كتابة كلمة المرور للتأكد من صحتها , أنشئ صفحة جديدة , أضف أدوات التحكم Button,TextBox1,TextBox2 وأداة التحقق CompareValidator واضبط فيها الخصائص التالية :

الخاصية	القيمة
ControlToValidate	TextBox2
ControlToCompare	TextBox1
ErrorMessage	كلمة المرور وتأكيدها غير متطابقين
Type	String
Operator	Equal

كود الصفحة :

كود ASP.net

```
<div>
    <asp:TextBox ID="TextBox1" runat="server" TextMode="Password">
    </asp:TextBox>
    أدخل كلمة المرور<br />
    <asp:TextBox ID="TextBox2" runat="server" TextMode="Password">
    </asp:TextBox>
    أعد كتابة كلمة المرور
    <asp:CompareValidator ID="CompareValidator1" runat="server"
        ErrorMessage="كلمة المرور وتأكيدها غير متطابقين"
        ControlToCompare="TextBox1"
        ControlToValidate="TextBox2"
        ForeColor="Red">
    </asp:CompareValidator>
    <br />
    <asp:Button ID="Button1" runat="server" Text="إرسال البيانات" />
</div>
```

نفذ الصفحة السابقة وقم بإدخال كلمتي مرور مختلفتين في حقلي الإدخال لتظهر لك رسالة الخطأ , وهذا ما نراه عادة في المنتديات عند تسجيل عضوية جديدة .

4- **RegularExpressionValidator** : فحص القيمة المدخلة إن كانت تتوافق مع تعبير منتظم محدد أم لا . ومن خصائصها :

- ✓ ControlToValidate : لتحديد أداة التحكم المرتبطة بأداة التحقق هذه .
- ✓ ErrorMessage : لتحديد رسالة الخطأ التي ستظهر في حال عدم توافق القيمة المدخلة مع التعبير المنتظم المحدد .
- ✓ ValidationExpression : لتحديد التعبير المنتظم المراد تطبيقه .

ولكن ماهو التعبير المنتظم ؟ بإيجاز : هو صيغة كتابة شيء ما . فمن المعلوم على سبيل المثال أن البريد الإلكتروني يملك الصيغة التالية : xxxxx@xxxxxxx.xxx أي أنه يتألف من (اسم الحساب @ اسم الجهة المستضيفة . النطاق) وبالتالي فإن إنقاص أي جزء من الصيغة السابقة يؤدي للحصول على بريد إلكتروني غير صحيح , مانريد عمله هو التحقق من أن المستخدم أدخل قيمة تتوافق مع صيغة محددة , يتم تحديد الصيغة بما يدعى بالتعبير المنتظم, التعبير المنتظم للبريد الالكتروني :

```
\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*
```

مثال :

سنقوم في هذا المثال بعمل نموذج بسيط لإدخال البريد الإلكتروني والتحقق من صحة صيغته , أنشئ صفحة جديدة , أضف أدوات التحكم TextBox , Button وأداة التحقق RegularExpressionValidator واضبط فيها الخصائص التالية :

الخاصية	القيمة
ControlToValidate	TextBox1
ErrorMessage	صيغة بريد إلكتروني غير صحيحة
ValidationExpression	\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*

كود الصفحة :

كود ASP.net

```
<div>
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    الالكتروني بريدك أدخل
    <br />
    <asp:RegularExpressionValidator ID="RegularExpressionValidator1"
        runat="server"
        ErrorMessage="صحيحة غير صيغة"
        ControlToValidate="TextBox1" ForeColor="Red"
        ValidationExpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*">
```

```
        </asp:RegularExpressionValidator>
        <br />
        <asp:Button ID="Button1" runat="server" Text="البيانات إرسال" />
    </div>
```

نفذ الصفحة السابقة ولاحظ ظهور رسالة الخطأ عند إدخالك لبريد إلكتروني ذو صيغة غير صحيحة

المزيد حول التعابير المنتظمة :

عناوين البريد الإلكتروني ليست الأمر الوحيد الذي يخضع لقواعد التعابير المنتظمة , فأرقام الهواتف , أرقام الضمان الاجتماعي , عناوين مواقع الانترنت .... إلخ جميعهم يخضعون لأسلوب كتابة محدد بواسطة التعابير المنتظمة . تحتوي منصة العمل Visual Studio على صيغ التعابير المنتظمة الأكثر شيوعاً , كما يوجد العديد من مواقع الانترنت التي تقدم هذه الخدمة ومن أهمها :

http://regexlib.com

5- **CustomValidator** : إن لم تؤدي أياً من أدوات التحقق السابقة وظيفة التحقق التي تريدها فبإمكانك تصميم أسلوب التحقق الخاص بك وذلك باستخدام الأداة CustomValidator ومن الخصائص الهامة لأداة التحقق هذه :

- ✓ ControlToValidate : لتحديد أداة التحكم المرتبطة بأداة التحقق هذه .
- ✓ ErrorMessage : لتحديد رسالة الخطأ التي ستظهر في حال عدم توافق القيمة المدخلة مع قيد التحقق المحدد .
- ✓ ClientValidationFunction : اسم تابع (function) التحقق الذي سيتم تطبيقه على جانب العميل (ClientSide) .

كما أن أداة التحقق CustomValidator تدعم الحدث ServerValidate والذي يتم إطلاقه عند بدء عمل أداة التحقق هذه .

يتم تطبيق تابع التحقق عند إطلاق الحدث ServerValidate .

مثال :

لو أردنا مثلاً جعل دخل المستخدم لا يتجاوز 10 أحرف , وإلا ستظهر رسالة خطأ تخبر المستخدم بأنه تجاوز الطول المسموح به . أنشئ صفحة جديدة , أضف أدوات التحكم TextBox , Button وأداة التحقق CustomValidator واضبط فيها الخصائص التالية :

الخاصية	القيمة
ControlToValidate	TextBox1
ErrorMessage	تجاوزت الطول المسموح به

أما في الحدث ServerValidate الخاص بالأداة CustomValidator فاكتب الكود التالي :

كود C#

```
protected void CustomValidator1_ServerValidate(object source,
                                               ServerValidateEventArgs args)
{
    if (args.Value.Length > 10)
        args.IsValid = false;
    else
        args.IsValid = true;
}
```

كود VB

```
Protected Sub CustomValidator1_ServerValidate(ByVal source As Object, ByVal
    args As System.Web.UI.WebControls.ServerValidateEventArgs) Handles
    CustomValidator1.ServerValidate

    If args.Value.Length > 10 Then
        args.IsValid = False
    Else
        args.IsValid = True
    End If
End Sub
```

البارامتر الثاني (args) في الدالة السابقة هو غرض من الصف ServerValidateEventArgs حيث يملك الخصائص التالية :

- ✓ Value : تمثل القيمة المدخلة في أداة التحكم المرتبطة بأداة التحقق CustomValidator .
- ✓ IsValid : تحدد هل تم تجاوز شرط التحقق بنجاح أم فشل .
- ✓ ValidateEmptyText : تحدد هل يتم تطبيق تابع التحقق اذا لم يدخل المستخدم أي قيمة في حقل الإدخال أم لا .

في الكود السابق قمنا بفحص طول القيمة المدخلة فإن تجاوزت 10 محارف يتم إعطاء الخاصية IsValid القيمة false (أي فشل اختبار التحقق) وإلا يتم إعطائه القيمة true وتجاوز اختبار التحقق بنجاح .

كود الصفحة :

كود ASP.net

```
<div>
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    <asp:CustomValidator ID="CustomValidator1" runat="server"
        ErrorMessage="به المسموح الطول تجاوزت"
        ControlToValidate="TextBox1"
        ForeColor="Red"
        onservervalidate="CustomValidator1_ServerValidate">
    </asp:CustomValidator>
    <br />

    <asp:Button ID="Button1" runat="server" Text="البيانات إرسال" />
</div>
```

إن كود C# أو VB السابق موجود على السيرفر وبالتالي فإن عملية التحقق من دخل المستخدم ستتم على الجانب السيرفر وهذا يتطلب إرسال بيانات الصفحة مما يؤدي إلى بطئ في عملية التحقق , بإمكاننا جعل عملية التحقق تتم على جانب المستخدم والسرفر معاً وذلك حيث نستفاد من سرعة التحقق على جانب المستخدم ومن سلامة عملية التحقق وضمانها على جانب السيرفر , يتم التحقق على جانب المستخدم بواسطة لغة الجافاسكريبت , حيث تتم إضافة دالة التحقق في صفحة ASP وهذا ماسيتضح من خلال المثال التالي .

مثال عن التحقق على جانب العميل

أعد تطبيق المثال السابق وأضف في صفحة ASP دالة الجافاسكربت التالية (حيث تتم إضافة أكود الجافاسكربت بين وسمي <head>.... </head> أعلى الصفحة) الكود :

كود JavaScript

```
<head runat="server">
    <title></title>
    <script type="text/javascript">
    function My_ClientValidate(source, args)
    {
    if (args.Value.length > 10)
        args.IsValid = false;
    else
        args.IsValid = true;
    }
    </script>
</head>
<body>
. . . . . باقي كود الصفحة . . . .
```

بعد الإضافة السابقة قم بتنفيذ الصفحة وأدخل نص أطول من 10 محارف ثم اضغط على الزر , سوف تظهر رسالة الخطأ , ولكن لاحظ عدم إعادة تحميل الصفحة (عدم ظهور شريط التحميل أسفل المتصفح) وهذا يعني أن الصفحة لم ترسل للسيرفر بعد , وأن عملية التحقق قد تمت على طرف المستخدم , وفي هذا سرعة أكبر وتخفيف من العبء على السيرفر .

طرف التحقق	السرعة	الأمن
التحقق على جانب المستخدم	سرعة أكبر	قابل للاختراق بشكل أسهل
التحقق على جانب السيرفر	سرعة أقل	قابلية اختراق صعبة جدا

وبالتالي فإن الحل الأفضل هو التحقق على كلا الجانبين معاً كما فعلنا في المثال السابق وبهذا نضمن مزايا كلا الأسلوبين ونتجاوز عيوبهما .

6- **ValidationSummary** : إظهار ملخص بجميع أخطاء التحقق التي تم اكتشافها في الصفحة هذا الأمر مفيد عند التعامل مع نماذج إدخال طويلة , حيث يتم عرض جميع الأخطاء الحاصلة أسفل الصفحة (أو في أي مكان ترغب به). من خلال تعاملك مع أدوات التحقق السابقة لابد وأنك لاحظت أنهم جميعاً يمتلكون الخاصية ErrorMessage والخاصية Text , الفرق بينهما هو القيمة المخصصة لـ ErrorMessage يتم إرسالها أيضاً كرسالة خطأ للأداة ValidationSummary , في حين القيم المسندة للخاصية Text تظهر كجزء من الصفحة نفسها ويفضل أن تكون قصيرة , في حين تحتوي ErrorMessage على تفصيل أكثر. على العموم , إن لم تسند قيمة للخاصية Text فإن الخاصية ErrorMessage ستعمل عملها وتقوم بعرض قيمتها كجزء من الصفحة (وهذا ما كنا نقوم به في الأمثلة السابقة) . للتوضيح أكثر حول هذه النقاط دعنا نطبق المثال التالي :

مثال :

سنقوم في هذا المثال بالتعامل مع كلا الخاصيتين ErrorMessage, Text وذلك بغية توضيح وظيفة كلاً منهما , أنشئ صفحة جديدة , أضف أدوات التحكم واضبط خصائصها كما هو موضح في الجدول التالي : (وهو نموذج بسيط يطلب من المستخدم ادخال اسمه وكلمة المرور)

اسم الأداة	الخاصية	القيمة
TextBox1		
TextBox2		
Button1	Text	إرسال البيانات
RequiredFieldValidator1	Text	*
RequiredFieldValidator1	ErrorMessage	لم تقم بإدخال اسمك
RequiredFieldValidator1	ControlToValidate	TextBox1

RequiredFieldValidator2	Text	*
RequiredFieldValidator2	ErrorMessage	لم تقم بإدخال كلمة المرور
RequiredFieldValidator2	ControlToValidate	TextBox2
ValidationSummary1		

قم بتنفيذ الصفحة السابقة واضغط على الزر دون أن تدخل أي بيانات في حقول الإدخال , ستلاحظ ظهور الرمز * بجانب كلاً من حقلي الإدخال وهي القيمة المحددة بالخاصية Text , كما ستلاحظ أسفل الصفحة ظهور رسالتي الخطأ المحددتين بالخاصية ErrorMessage لكلاً من أداة التحقق الأولى والثانية , حيث يتم عرضهما كجزء من الأداة ValidationSummary1 . كما هو واضح في الصورة .

كود الصفحة :

كود ASP.net

```
<div>

    <asp:TextBox ID="TextBox1" runat="server" ></asp:TextBox>
    <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
        ControlToValidate="TextBox1"
        ErrorMessage="اسمك بإدخال تقم لم"
        ForeColor="Red">*</asp:RequiredFieldValidator>
    <br />
    <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
    <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
        ControlToValidate="TextBox2"
        ErrorMessage="المرور كلمة بإدخال تقم لم"
        ForeColor="Red">*</asp:RequiredFieldValidator>
    <br />
    <asp:Button ID="Button1" runat="server" Text="البيانات إرسال" />
    <br />
    <asp:ValidationSummary ID="ValidationSummary1" runat="server" />

</div>
```

كما أن الأداة ValidationSummary تدعم الخصائص التالية :

- ✓ DisplayMode : لتحديد أسلوب عرض رسائل الأخطاء , مثلاً كقائمة أم فقرة ... وتأخذ إحدى القيم : List , BulletList , SingleParagraph .
- ✓ HeaderText : لتحديد عنوان يظهر أعلى رسائل الخطأ المعروضة في هذه الأداة .

- ✓ ShowMessageBox : تمكننا من إظهار الأخطاء كصندوق رسالة منبثق كما في تطبيقات ويندوز .
- ✓ ShowSummary : إظهار أو إخفاء هذه الأداة .

تمرين غير محلول :

قم ببناء نموذج شبيه إلى حد ما بنموذج إنشاء بريد إلكتروني جديد وقم بتطبيق قيود التحقق على مدخلات المستخدم , بالشكل التالي :

اسم الحقل	القيد
اسم المستخدم	غير فارغ
البريد الإلكتروني المراد إنشاؤه	غير فارغ , صيغة بريد إلكتروني نظامية
كلمة المرور	غير فارغ , لا تتجاوز 16 محرف
تأكيد كلمة المرور	يجب أن تطابق كلمة المرور

واجعل رسائل الخطأ تظهر كقائمة أسفل الصفحة وباللون الأحمر .

ملاحظة

يمكن للخاصية Text أن تقبل كود HTML كدخل لها , وبالتالي فإنه من الممكن إظهار صورة صغيرة بجانب كل حقل يخالف شرط التحقق وذلك بدلاً من الإكتفاء بإظهار رسالة توضيحية كما فعلنا في الأمثلة السابقة .
إن كنت مهتماً بهذه النقطة , قم بالعودة لأحد الأمثلة السابقة واعط الخاصية Text لأحد أدوات التحقق القيمة التالية :

```
Text="<img src='Error.gif' alt='Required' />"
```

مع الأخذ بعين الاعتبار وجود الصورة المطلوبة واسمها ومسارها وامتدادها .

أدوات التحقق والجافاسكربت

بشكل افتراضي , فإن أدوات التحقق التي تعرفنا عليها تقوم بعملية التحقق على كلا الجانبين (السيرفر والعميل) , فعلى جانب العميل تتم عملية التحقق على مستوى متصفح الانترنت وباستخدام لغة الجافاسكربت , الإصدارات قبل ASP.net 3.5 كانت تدعم التحقق على جانب العميل فقط عند استخدام المتصفح Internet Explorer , أما الإصدارات الحديثة فأصبحت مدعومة من قبل متصفحات عديدة Firefox , Opera بالإضافة طبعاً لـ Internet Explorer وغيرهم . وفي

حال لم يدعم المتصفح المستخدم لغة الجافاسكربت فإن أدوات التحقق ستكتفي بإجراء عملية التحقق على جانب السيرفر فقط حيث لن تكون قادرة على العمل على جانب العميل .

على العموم , فإن إجراء عملية التحقق على كلا الجانبين يعد الحل الأفضل كما أسلفنا في فقرة سابقة , حيث نستفاد من سرعة إجراء التحقق على جانب العميل بالإضافة لضمان سلامة العملية وعدم التلاعب بها على جانب السيرفر .

أخيراً , بإمكانك إن شئت إلغاء عملية التحقق على جانب العميل والإكتفاء بإجرائها على جانب السيرفر وذلك بإسناد القيمة False للخاصية EnableClientScript الموجودة مع جميع أدوات التحقق السابقة .

التعامل مع الخاصية Page.IsValid

جميع أدوات التحقق تمتلك الخاصية IsValid والتي تعيد القيمة True اذا كانت البيانات المدخلة موافقة لشرط التحقق المطلوب وإلا فإنها تعيد القيمة False , الصفحة (Page) تمتلك أيضاً الخاصية IsValid والتي تعيد True اذا وفقط اذا أعادت جميع أدوات التحقق الموجودة في تلك الصفحة القيمة True (أي إن لم يكن هناك أي أخطاء في دخل المستخدم) , على العموم فإن الخاصية السابقة تعمل على جانب السيرفر وبالتالي فإن كانت عملية التحقق من المدخلات تتم على كلا الجانبين (سيرفر – عميل) فلا داعي لاستخدامها , وذلك لأن بيانات الصفحة لن ترسل أساسا إلى السيرفر حتى تجتاز اختبار التحقق على جانب العميل وفي هذه الحالة فإن Page.IsValid ستعيد حتماً True , أما إن تم إيقاف عملية التحقق على جانب العميل (أو كان المتصفح المستخدم لايدعم الجافاسكربت) عندئذ يصبح استخدام الخاصية السابقة ضرورياً .

إعداد خاصية العرض Display

جميع أدوات التحقق السابقة تمتلك الخاصية Display والتي تحدد آلية عرض رسالة الخطأ على الصفحة , وهي تأخذ إحدى القيم التالية :

- ✓ Static : حيث يتم حجز مساحة رسالة الخطأ على الصفحة حتى إن لم يكن هناك خطأ في دخل المستخدم ولم تكن هناك ضرورة لعرض هذه الرسالة , ويكون كود HTML الناتج عنها :

كود HTML

```
<span id="RequiredFieldValidator1" style="visibility:hidden;">*</span>
```

- ✓ Dynamic : لا يتم حجز مساحة رسالة الخطأ على الصفحة حتى ظهورها , ويكون كود HTML الناتج عنها :

كود HTML

```
<span id="RequiredFieldValidator1" style="display:none;">*</span>
```

- ✓ None : إن تحديد هذا الخيار يؤدي لعدم ظهور رسالة الخطأ ضمن أداة التحقق , والاكتفاء بإرسالها للأداة ValidationSummary حيث تقوم بعرضها .

استخدام مجموعات التحقق

في كثير من الحالات نحتاج لبناء أكثر من نموذج تصميم ضمن نفس الصفحة , مثلاً في الصفحة الرئيسية لموقع فيسبوك facebook.com يوجد نموذجين , أحدهما لتسجيل حساب جديد والآخر لإجراء عملية تسجيل الدخول , ولكل نموذج شروط التحقق الخاصة به , حيث إن الضغط على زر "تسجيل الدخول" يؤدي إلى عمل أدوات التحقق الموجودة ضمن نموذج تسجيل الدخول , وبالمقابل فإن الضغط على زر "إنشاء حساب" يؤدي إلى عمل أدوات التحقق الموجودة في هذا النموذج فقط , لنقم الآن بتطبيق السيناريو السابق لتتوضح المسألة بشكل أكبر .

قم بتصميم صفحة مشابهة للصفحة التالية :

حيث إن جميع أدوات التحقق المضافة هي من النوع RequiredFieldValidator1 , كود الصفحة الناتج :

كود ASP.net

```
<div>
    <asp:Label ID="Label1" runat="server" Text="جديد حساب تسجيل"></asp:Label>
    <br />
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
        ControlToValidate="TextBox1" ErrorMessage="(Required)"
        ForeColor="Red"></asp:RequiredFieldValidator>
    <br />
    <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
    <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
        ControlToValidate="TextBox2" ErrorMessage="(Required)"
        ForeColor="Red"></asp:RequiredFieldValidator>
    <br />
    <asp:Button ID="Button1" runat="server" Text="حساب إنشاء" />
    <br />
    <hr />
    <asp:Label ID="Label2" runat="server" Text="دخول تسجيل"></asp:Label>


    <br />
    <asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
    <asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server"
        ControlToValidate="TextBox3" ErrorMessage="(Required)"
        ForeColor="Red"></asp:RequiredFieldValidator>
    <br />
    <asp:TextBox ID="TextBox4" runat="server"></asp:TextBox>
    <asp:RequiredFieldValidator ID="RequiredFieldValidator4" runat="server"
        ControlToValidate="TextBox4" ErrorMessage="(Required)"
        ForeColor="Red"></asp:RequiredFieldValidator>
    <br />
    <asp:Button ID="Button2" runat="server" Text="دخول تسجيل" />

    <br />

</div>
```

قم بتنفيذ الصفحة السابقة , املأ بيانات النموذج الأول واضغط على الزر "إنشاء حساب" , ستجد أنه لم يتم إرسال الصفحة إلى السيرفر إنما ظهرت رسائل تطالبك بإدخال بيانات في حقول نموذج تسجيل الدخول ! , السبب وراء ذلك هو أن الضغط على أي زر سؤدي إلى عمل جميع أدوات التحقق الموجودة في تلك الصفحة , في الإصدارات التالية لـ ASP.net 2.0 تم التغلب على هذه المشكلة عبر إمكانية ضم العديد من أدوات التحقق في مجموعة تحقق واحد وربط زر ما بمجوعة التحقق هذه وبالتالي فإن الضغط على هذا الزر سؤدي إلى عمل أدوات التحقق المنتمية إلى مجموعته فقط , ولن تعمل أي أداة تحقق منتمية إلى مجموعة أخرى حتى يتم الضغط على على زر ينتمي إلى نفس تلك المجموعة .

يتم تحديد المجموعة التي تنتمي لها أداة التحقق من خلال الخاصية ValidationGroup , حيث يتم إسناد نفس القيمة إلى جميع أدوات التحقق التي تقع ضمن نفس النموذج (نفس المجموعة) , إن هذه الخاصية موجودة أيضاً في أدوات التحكم Button, ImageButton, LinkButton حيث ينغي إعطاؤها نفس القيمة بحسب النموذج التي تقع فيه والمسؤولة عن تشغيل أدوات التحقق الخاصة به .

لنعد إلى الصفحة السابقة , أسند القيمة "signup" إلى الخاصية ValidationGroup لأدوات التحقق الموجودة في نموذج "تسجيل حساب جديد" وكذلك أيضاً للزر "إنشاء حساب" .

ثم أسند القيمة "login" إلى الخاصية ValidationGroup لأدوات التحقق الموجودة في نموذج "تسجيل دخول" وكذلك أيضاً للزر "تسجيل دخول" .

حيث سيصبح كود الصفحة بالشكل التالي :

كود ASP.net

```
<div>
    <asp:Label ID="Label1" runat="server" Text="جديد حساب تسجيل"></asp:Label>
    <br />
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
        ControlToValidate="TextBox1"
        ErrorMessage="(Required)"
        ForeColor="Red"
        ValidationGroup="signup">
    </asp:RequiredFieldValidator>
    <br />
    <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
    <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
        ControlToValidate="TextBox2"
        ErrorMessage="(Required)"
        ForeColor="Red"
        ValidationGroup="signup">
    </asp:RequiredFieldValidator>
    <br />
    <asp:Button ID="Button1" runat="server" Text="حساب إنشاء"
        ValidationGroup="signup" />
    <br />
    <hr />
    <asp:Label ID="Label2" runat="server" Text="دخول تسجيل"></asp:Label>
    <br />
    <asp:TextBox ID="TextBox3" runat="server"></asp:TextBox>
    <asp:RequiredFieldValidator ID="RequiredFieldValidator3" runat="server"
        ControlToValidate="TextBox3"
        ErrorMessage="(Required)"
        ForeColor="Red"
        ValidationGroup="login">
    </asp:RequiredFieldValidator>
    <br />
```

```
        <asp:TextBox ID="TextBox4" runat="server"></asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator4" runat="server"
            ControlToValidate="TextBox4"
            ErrorMessage="(Required)"
            ForeColor="Red"
            ValidationGroup="login">
        </asp:RequiredFieldValidator>
        <br />
        <asp:Button ID="Button2" runat="server" Text="دخول تسجيل"
            ValidationGroup="login" />

        <br />

    </div>
```

قم بتنفيذ الصفحة السابقة , لاتقم بملء أي بيانات , عند الضغط على زر "إنشاء حساب" ستجد أن أدوات التحقق الموجودة في نموذج "تسجيل حساب جديد" هي وحدها التي أظهرت رسائل تطالبك بإدخال بيانات , قم بعدها بالضغط على الزر في نموذج "تسجيل دخول" ستجد أن أدوات التحقق الموجودة في هذا النموذج هي وحدها التي ستعمل .

وبهذه الطريقة بإمكانك تخصيص أدوات التحقق والأزرار المرتبطة بها في مجموعات , وبالتالي فإن عملها لن يتداخل أو يتضارب مع بعضها البعض .

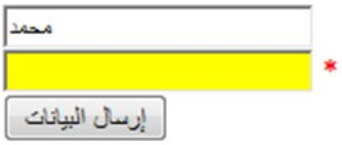
إلغاء عملية التحقق

وجدنا من خلال الفقرات السابقة أن أدوات التحكم Button, ImageButton, LinkButton هي التي تقوم بتشغيل أدوات التحقق ,ولكن من الممكن جعل الأدوات السابقة لاتسبب تشغيل أدوات التحقق وذلك بإسناد القيمة False للخاصية CausesValidation , من الحالات الشائعة لاستخدام هذه الخاصية عند تصميم نموذج تسجيل وإضافة زر "إلغاء الأمر" حيث يجب ألا يسبب الضغط على هذا الزر عمل أدوات التحقق الموجودة في ذلك النموذج , وبالتالي نقوم بإعطاء الخاصية CausesValidation لذلك الزر القيمة False .

تمرين غير محلول :

قم بإنشاء نموذج بسيط لإجراء عملية تسجيل الدخول , قم بإضافة أدوات التحقق المناسبة , أضف أسفل النموذج زري أمر أحدهما " تسجيل الدخول" والآخر "إلغاء الأمر" , يؤدي الضغط على الزر "تسجيل الدخول" إلى عمل أدوات التحقق , أما الضغط على الزر "إلغاء الأمر" يؤدي إلى نقل المستخدم إلى صفحة أخرى .

تمييز الحقول المخالفة باستخدام الألوان



سنقوم في الفقرة بجعل الحقول المخالفة لشرط التحقق تظهر بلون أصفر(أو أي لون ترغب به) , وذلك حتى تظهر بوضوح للمستخدم خاصة عند بناء نماذج كبيرة الحجم , كنماذج السيرة الذاتية وغيره .

قم بإنشاء صفحة جديدة , صمم نموذج بسيط لإدخال اسم المستخدم وكلمة المرور(كما يظهر في الصورة بالأعلى) , واربط كل حقل منهما بأداة تحقق من النوع RequiredFieldValidator واجعل الخاصية EnableClientScript تأخذ القيمة False لكلاً من أداتي التحقق السابقتين , وذلك حتى نلغي عملية التحقق على جانب العميل ونبقيها فقط على جانب السيرفر لأن السيرفر هو الذي سيقوم بتلوين الحقول المخالفة بلون مختلف .

أضف الدالة التالية لصفحة الكود الخلفي (C# أو VB) للصفحة السابقة :

كود C#

```
void Page_PreRender()
{
    foreach (BaseValidator valControl in Page.Validators)
    {
        WebControl assControl =
                  (WebControl)Page.FindControl(valControl.ControlToValidate);
        if (!valControl.IsValid)
            assControl.BackColor = System.Drawing.Color.Yellow;
        else
            assControl.BackColor = System.Drawing.Color.White;
    }
}
```

كود VB

```
Sub Page_PreRender()

    For Each valControl As BaseValidator In Page.Validators
        Dim assControl As WebControl =
                          Page.FindControl(valControl.ControlToValidate)
```

```
            If Not valControl.IsValid Then
                assControl.BackColor = Drawing.Color.Yellow
            Else
                assControl.BackColor = Drawing.Color.White
            End If
        Next

    End Sub
```

حيث يقوم الكود السابق بالمرور على جميع أدوات التحقق الموجودة ضمن الصفحة , وفحصها فإن كان الحقل المرتبط بها غير موافق للشرط يتم تلوينه باللون الأصفر , وإلا يتم إبقاؤه باللون الأبيض

كود الصفحة :

كود ASP.net

```
<div>

    <asp:TextBox ID="TextBox1" runat="server" ></asp:TextBox>
    <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
        ControlToValidate="TextBox1"
        ErrorMessage="*"
        ForeColor="Red"
        EnableClientScript="False">
    </asp:RequiredFieldValidator>
    <br />
    <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
    <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
        ControlToValidate="TextBox2"
        ErrorMessage="*"
        ForeColor="Red"
        EnableClientScript="False">
    </asp:RequiredFieldValidator>
    <br />
    <asp:Button ID="Button1" runat="server" Text="البيانات إرسال" />
    <br />

</div>
```

قم بتنفيذ الصفحة السابقة , اضغط على الزر دون ملئ بيانات ولاحظ تغيير لونها إلى الأصفر , قم بملأ أحد الحقول ثم أعد الضغط على الزر لتجد أنه سيتم إعادة لونه إلى الأبيض . تفيد هذه الطريقة كما ذكرت سابقاً عند العمل مع نماذج كبيرة الحجم وذلك ليسهل على المستخدم تمييز الحقول المخالفة لشرط التحقق .

الخاتمة

إلى هنا نأتي لنهاية هذا الفصل , والذي تناولنا فيه كيفية التعامل مع أدوات التحقق على اختلاف أنواعها واستخداماتها , كالتحقق من أن المستخدم قد ملأ الحقول المطلوبة , وبنمط بيانات صحيح وموافق للصيغة المطلوبة ولمجال القيم المحدد , ثم تعلمنا كيفية إظهار جميع رسائل الأخطاء الحاصلة في الصفحة ضمن أداة وحيدة ومكان واحد , ثم ناقشنا العديد من الأمور والخصائص والتي تستخدم بكثرة في المواقع , كبناء أكثر من نموذج ضمن نفس الصفحة واستخدام مجموعات التحقق . من الضروري جداً أن تقوم بتطبيق جميع الأمثلة الواردة ضمن هذا الفصل وجميع الفصول ولاتكتفي بمجرد القراءة وذلك لتثبيت المعومة لديك والاعتياد على الممارسة في بناء مواقع الانترنت .

نلتقي في فصل قادم ومواضيع جديدة إن شاء الله , لأي اقتراح أو نقد أرجو التواصل على البريد الاكتروني , والله ولي التوفيق .

للتواصل :

m-hajjkhalaf@hotmail.com

http://www.facebook.com/mohammed.hajkhalaf

محمد عمر الحاج خلف – سوريا

+963 932 033250