VALIDANDO CONTROLES DE ENTRADA DE FORMULARIOS

El Framework de Formularios Web incluye un conjunto de controles de servidor de validación que proporcionan una forma sencilla pero poderosa de comprobar los formularios de entrada en busca de errores y, si es necesario, mostrar mensajes al usuario.

Podemos "ligar" más de un control de validación a un control de entrada. Por ejemplo, podríamos especificar tanto que un campo es obligatorio y que debe contener un rango específico de valores.

Los controles de validación trabajan un limitado subconjunto de controles de servidor HTML y Web. Para cada control, una propiedad específica contiene el valor que se validará. La siguiente tabla muestra los controles de entrada que pueden ser validados.

Control	Propiedad de Validación
HtmlInputText	Value
HtmlTextArea	
HtmlSelect	
HtmlInputFile	
TextBox	Text
ListBox	SelectedItem.Value
DropDownList	
RadioButtonList	
FileUpload	FileName

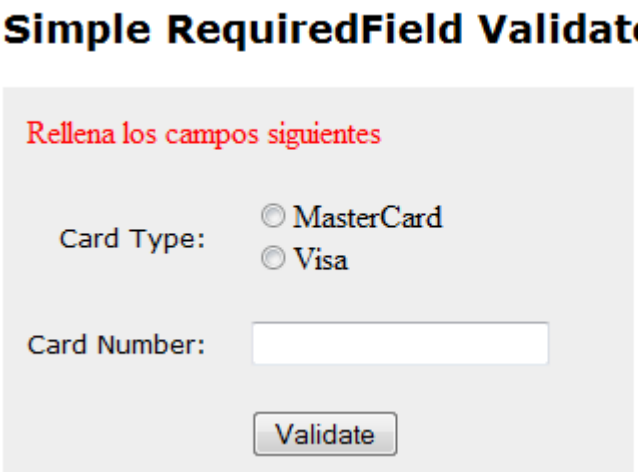
Tipos de Controles de Validación

El formulario de validación más sencillo es un campo obligatorio. Si el usuario introduce un valor en el campo, es válido. Si todos los campos de la página son válidos, la página es válida.

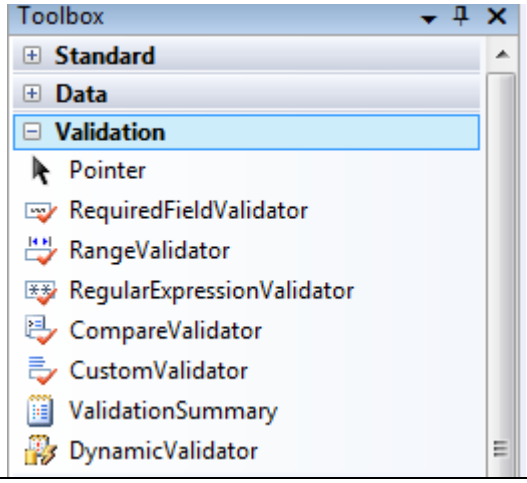
```
<html>
<head>
    <script language="C#" runat=server>
        void ValidateBtn_Click(Object Sender, EventArgs E) {
           if (Page.IsValid == true) {
              lblOutput.Text = "Page is Valid!";
           }
           else {
```

```
                lblOutput.Text = "Some of the req fields are empty";
            }
        }
    </script>
</head>
<body>

<h3><font face="Verdana">Simple RequiredField Validator
Sample</font></h3>

<form runat="server">

    <table bgcolor="#eeeeee" cellpadding=10>
  <tr valign="top">
      <td colspan=3>
            <asp:Label ID="lblOutput" Text="Rellena los campos
            siguientes " ForeColor="red" runat=server /><br>
      </td>
    </tr>

    <tr>
      <td align=right>
        <font face=Verdana size=2>Card Type:</font>
      </td>
      <td>
        <ASP:RadioButtonList id=RadioButtonList1 RepeatLayout="Flow"
        runat=server>
            <asp:ListItem>MasterCard</asp:ListItem>
            <asp:ListItem>Visa</asp:ListItem>
        </ASP:RadioButtonList>
      </td>
      <td align=middle rowspan=1>
        <asp:RequiredFieldValidator id="RequiredFieldValidator1"
            ControlToValidate="RadioButtonList1"
            Display="Static"
            InitialValue="" Width="100%" runat=server>*
        </asp:RequiredFieldValidator>
      </td>
    </tr>
    <tr>
      <td align=right>
        <font face=Verdana size=2>Card Number:</font>
      </td>
      <td>
        <ASP:TextBox id=TextBox1 runat=server />
      </td>
      <td>
        <asp:RequiredFieldValidator id="RequiredFieldValidator2"
            ControlToValidate="TextBox1"
            Display="Static"
            Width="100%" runat=server>*
        </asp:RequiredFieldValidator>
      </td>
    </tr>
    <tr>
      <td></td>
      <td>
        <ASP:Button id=Button1 text="Validate"
        OnClick="ValidateBtn_Click" runat=server />
```

```
      </td>
    </tr></table>
</form>
</body>
</html>
```



También hay controles de validación para tipos específicos de validación, cómo control de rango o coincidencia con un patrón.



Nombre del Control	Descripción
RequiredFieldValidator	Asegura que el usuario no se deja un campo
CompareValidator	Compara los datos que introduce el usuario con una constante o el valor de una propiedad de otro control mediante un operador de comparación (menor que, igual que, mayor que, etc.).
RangeValidator	Comprueba que la entrada del usuario se encuentra entre un límite superior y otro inferior. Podemos comprobar los rangos con parejas de números, caracteres alfabéticos o fechas. Los límites se pueden expresar como constantes.
RegularExpressionValidator	Comprueba que la entrada sigue un patrón definido como una expresión regular. Este tipo de validación

	nos permite comprobar secuencias predecibles de caracteres, tales como números de seguridad social, dirección de e-amil, números de teléfono, códigos postales, etc.
CustomValidator	Comprueba la entrada de usuario mediante lógica de validación que hemos programado nosotros. Este tipo de validación nos permite comprobar valores obtenidos en tiempo de validación.
ValidationSummary	Muestra los errores de validación en un formulario resumen para todos los validadores de la página.

Validación del Lado Cliente

Los controles de validación siempre realizan la comprobación de validación en el servidor. También tienen una implementación completa en el cliente que permite a los exploradores compatibles con DHTML realizar la validación en el cliente. La validación en el cliente mejora el proceso de validación ya que se comprueban los datos proporcionados por el usuario antes de enviarlos al servidor. De este modo se pueden detectar los errores en el cliente antes de enviar el formulario y se evita la acción de ida y vuelta de la información necesaria para la validación en el servidor.

Si cualquiera de los validadores encuentra un error, el envío del formulario al servidor se cancela y se muestra la propiedad Text del validador.

Esto permite al usuario corregir la entrada antes de enviar el formulario al servidor. Los valores de los campos se revalidan cuando el campo que contenía el error pierde el foco, proporcionando así una experiencia rica e interactiva de validación al usuario.

El Framework de Páginas de Formularios Web siempre realiza la validación en el servidor, incluso cuando ya se ha hecho en el cliente. Esto nos ayuda a impedir que los usuarios puedan saltarse la validación haciéndose pasar por otro usuario o una transacción previamente aprobada.

La validación del lado del cliente está permitida por defecto. Para deshabilitar la validación del lado del cliente, estableceremos la propiedad **ClientTarget** de la página a "Downlevel" ("Uplevel" fuerza la validación del lado cliente). De forma alternativa, podemos establecer la propiedad **EnableClientScript** de un control de validación a "false" para deshabilitar la validación del lado cliente para dicho control.

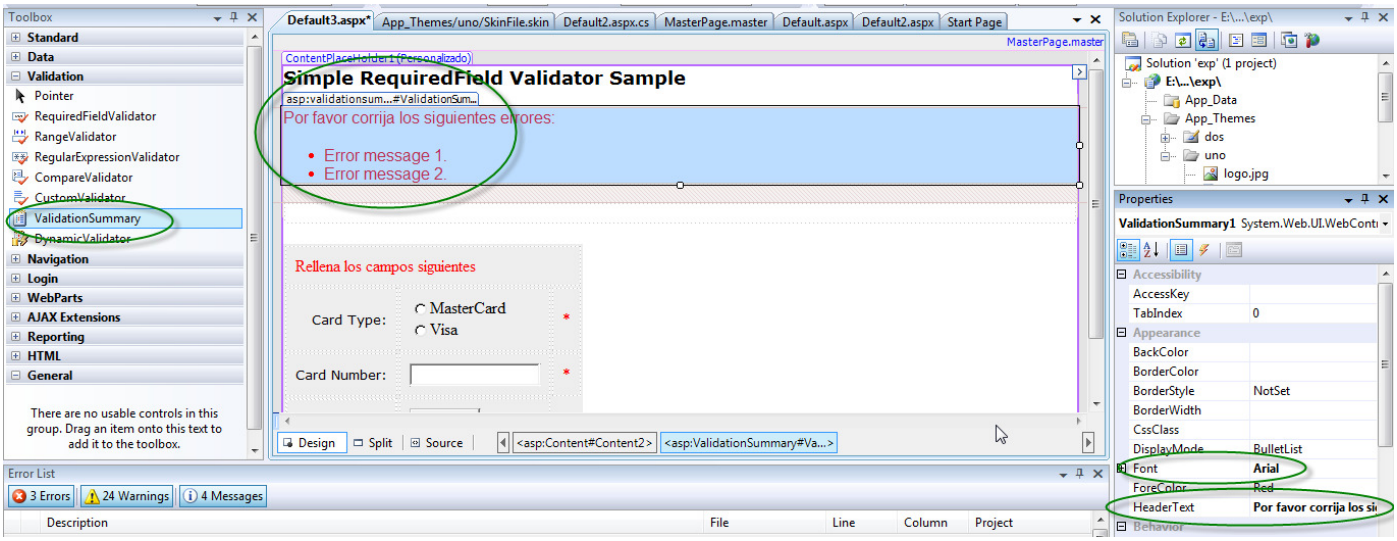
Mostrando Errores de Validación

Después de procesarse todos los controles de validación, la propiedad IsValid de la página se establece; si alguno de los controles muestra un fallo de validación, la página entera se marca como inválida (`Page.IsValid`).

Si un control de validación da un error, dicho control mostrará un mensaje de error en la página o en un control ValidationSummary. El control ValidationSummary se

muestra cuando la propiedad IsValid de la página está establecida a "false". Sondea el resto de controles de validación de la página y agrega el texto que cada uno muestra. En el siguiente ejemplo vemos cómo mostrar errores con un control ValidationSummary.

```
<asp:ValidationSummary ID="ValidationSummary1" runat="server"
Font-Names="Arial" HeaderText="Por favor corrija los siguientes
errores:" />
```



Trabajando con CompareValidator

Compara los valores de dos controles. Utiliza tres propiedades clave para realizar su validación. **ControlToValidate** y **ControlToCompare** contienen los valores a comparar. **Operator** define el tipo de comparación a realizar (por ejemplo Igual o Diferente).

Podemos especificar opcionalmente la propiedad **ValueToCompare** para realizar la comparación con un valor estático, en lugar de ControlToCompare.

El control de servidor CompareValidator también puede utilizarse para realizar la validación de Datatype (type=Date).

Por ejemplo, si la información de la fecha de nacimiento se tiene que recoger de la página de registro del usuario, el control CompareValidator se puede utilizar para asegurarnos que la fecha tiene un formato reconocido, antes de que se envíe a la base de datos.

```
<asp:CompareValidator ID="CompareValidator1" runat="server"
ControlToValidate="TextBox1" ErrorMessage="Error de validación"
Type="Date"></asp:CompareValidator>
```

Trabajando con RangeValidator

El control de servidor RangeValidator comprueba si un valor de entrada se encuentra dentro de un determinado rango. RangeValidator utiliza tres propiedades clave para

realizar su validación. **ControlToValidate** contiene el valor a validar. **MinimumValue** y **MaximumValue** definen los valores mínimo y máximo del rango válido.

Trabajando con Expresiones Regulares

Comprueba que una entrada coincida con un determinado patrón definido por una expresión regular. Este tipo de validación nos permite comprobar secuencias predecibles de carácteres, como números de la seguridad social, direcciones de e-mail, números de teléfono, códigos postales, etc.

Dos propiedades clave para realizar la validación: **ControlToValidate** contiene el valor a validar. **ValidationExpression** contiene la expresión regular con la que tiene que coincidir.

(http://es.wikipedia.org/wiki/Expresi%C3%B3n_regular)

Sintaxis Expresiones Regulares

- * cero o más ocurrencias del carácter o expresión anterior.
- + una o más ocurrencias.
- () agrupa una subexpresión que se trata como un único elemento.
- | Cualquiera de las dos partes (OR)
- [ ] se corresponde con un carácter en un intervalo de caracteres válidos [a-c]
- [^] se corresponde con un carácter que NO está en un intervalo dado [^a-b]
- . cualquier carácter excepto el salto de línea
- \s carácter de espacio en blanco (ej. tab o espacio)
- \S cualquier carácter no espacio
- \d cualquier carácter numérico
- \D cualquier carácter no dígito
- \w cualquier carácter alfanumérico (letra, número o carácter de subrayado)

Ejemplos:

1. /\(? (\d{3})? \)? (?(1) [\-\s]) \d{3}-\d{4}/x Obtener todos los números telefónicos de un segmento de texto....
2. /(<([\w]+)[^>]*>)(.*)(<\/\\2>)/ Encontrar etiquetas HTML coincidentes...
3. /[\w-\.]+@([\w-]+\.)+[\w-]{2,4}/ Valida cuentas de correo electrónico
4. /[\w-\.]{3,}@([\w-]{2,}\.)*([\w-]{2,}\.)[\w-]{2,4}/ Valida una cuenta de correo. Comprueba que delante del signo @ haya al menos 3 caracteres.
5. /<a href=\s*([^\s]*)\s*>/m Enlaces de una en un fichero HTML.

Realizando Validación Personalizada (CustomValidator)

Llama a una función definida por el usuario para realizar validaciones que los validadores estándar no pueden llevar a cabo.

La función personalizada se puede ejecutar en el servidor o en un script del lado del cliente, por ejemplo JScript o VBScript.

Para la validación personalizada en el lado del cliente, el nombre de la función debe definirse en la propiedad **ClientValidationFunction**. Dicha función debe tener la forma function myvalidator(source, arguments) donde source es el objeto **CustomValidator** del lado cliente, y arguments es un objeto con dos propiedades, Value y IsValid.

La propiedad Value es el valor que tendremos que validar y la propiedad IsValid es un Boolean en el que se devolverá el resultado de la validación.

Para validación personalizada del lado del servidor, tendremos que poner nuestra validación en el delegado de **OnServerValidate** del validador.

```
protected void CustomValidator1_ServerValidate(object source,
ServerValidateEventArgs valor)
 {
        int num;
        // ¿Número par?
        if (int.TryParse(valor.Value, out num))
         {
            valor.IsValid = (num % 2 == 0);
         }
        else
         {
            valor.IsValid = false;
         }
 }
```

```
<asp:CustomValidator ID="CustomValidator1" runat="server"
 ClientValidationFunction="js_par" ControlToValidate="TextBox1"
 ErrorMessage="Numero impar!!"
onservervalidate="CustomValidator1_ServerValidate"></asp:CustomValidat
or>
```

```
function js_par() {
    alert("alerta");
}
```

ValidateEmptyText

Podemos fijar esta propiedad a "true" para hacer que la validación personalizada se realice para valores de entrada vacíos.

Validation Groups

La propiedad ValidationGroup se utiliza cuando el usuario quiere tener escenarios de validación diferentes en la misma página.

Estableceremos el nombre del grupo en el validador y en el botón o en otro control "postback" que cause la validación. Esto es útil con controles Wizard, MultiView o controles de datos (edición). Por defecto, todos los validadores se encuentran en el grupo "" (grupo por defecto), para compatibilidad hacía atrás. Page nos proporciona

además los métodos GetValidators("group") y Validate("group"). **Page.IsValid** refleja la validez de todos los controles (acumulativa) que han sido llamados a Validate.

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

<asp:TextBox ID="TextBox1" Runat="server" ValidationGroup="First"></asp:TextBox>

<asp:TextBox ID="TextBox2" Runat="server" ValidationGroup="First"></asp:TextBox><br />

<asp:RequiredFieldValidator ID="RequiredFieldValidator1" Runat="server"
ValidationGroup="First"
ErrorMessage="TextBox1 should not be blank" ControlToValidate="TextBox1">
</asp:RequiredFieldValidator>

<asp:Button ID="Submit1" Runat="server" ValidationGroup="First" Text="Submit 1" />

<asp:TextBox ID="TextBox3" Runat="server" ValidationGroup="Second"></asp:TextBox>

<asp:TextBox ID="TextBox4" Runat="server" ValidationGroup="Second"></asp:TextBox>

<asp:RequiredFieldValidator ID="RequiredFieldValidator2" Runat="server" ErrorMessage="
TextBox3 should not be blank"
ControlToValidate="TextBox3" ValidationGroup="Second">
</asp:RequiredFieldValidator>

<asp:Button ID="Submit2" Runat="server" ValidationGroup="Second" Text="Submit 2" />

</div>
</form>
</body>
```

SetFocusOnError

Propiedad que se establece en controles de validación y hace que el primer control inválido reciba el foco.

NAVEGACIÓN

(http://msdn.microsoft.com/es-es/library/x3x8t37x.aspx)

Hipervínculos (control HyperLink)

Características

- Realiza una nueva solicitud en la página de destino.
- No pasa la información de la página actual a la página de destino.
- Requiere iniciación por parte del usuario.
- El redireccionamiento se produce a cualquier página, no sólo a aquéllas incluidas en la misma aplicación Web.
- Permite compartir información entre las páginas utilizando una cadena de consulta o el estado de sesión. (El permite crear cadenas de direcciones URL y de consulta mediante programación.)

Uso

- Para la exploración sin ningún procesamiento adicional, como en menús o listas de vínculos.
- Cuando el desplazamiento a otra página debe realizarse bajo el control del usuario.

Envío entre páginas (control Button + PostBackUrl también llamado Cross Post Back)

Características

- Envía la información de la página actual a la página de destino.
- Hace que la información del envío esté disponible en la página de destino.
- Requiere iniciación por parte del usuario.
- El redireccionamiento se produce a cualquier página, no sólo a aquéllas incluidas en la misma aplicación Web.
- Permite que la página de destino lea las propiedades públicas de la página de origen cuando estas páginas están en la misma aplicación Web.
- Hay que usar en página destino **PreviousPage.IsValid**

Uso

- Para pasar la información de la página actual a la página de destino (como en los formularios de varias páginas).
- Cuando el desplazamiento debe realizarse bajo el control del usuario.

Redireccionamiento del explorador (HttpResponse.Redirect)

Características

- Realiza una nueva solicitud en la página de destino.
- Pasa la cadena de consulta (querystring) a la página de destino.
- Permite controlar dinámicamente y mediante programación la dirección URL de destino y la cadena de consulta.
- Permite el redireccionamiento a cualquier página, no sólo a aquéllas incluidas en la misma aplicación Web.
- Permite almacenar información de la página de origen en el estado de sesión antes del redireccionamiento para compartirla con la página de destino.

Uso

- Para la exploración condicional, cuando desee controlar la dirección URL de destino y cuando tenga lugar la exploración. Por ejemplo, utilice esta opción si la aplicación debe determinar a qué página debe desplazarse basándose en los datos proporcionados por el usuario.

Transferencia del servidor (Server.Transfer)

Características:

- Transfiere el control a una nueva página que se representa en lugar de la página de origen.
- El redireccionamiento sólo se produce a las páginas de destino que están en la misma aplicación Web que la página de origen.
- Permite leer los valores y las propiedades públicas de la página de origen.
- **No actualiza la información del explorador con información sobre la página de destino. Al presionar los botones para actualizar o retroceder del explorador, se pueden producir resultados inesperados**.

Uso

- Para la exploración condicional, cuando se desee controlar cuándo tiene lugar la exploración y se desee obtener acceso al contexto de la página de origen.
- Esta opción se recomienda en situaciones en las que la dirección URL está oculta para el usuario.

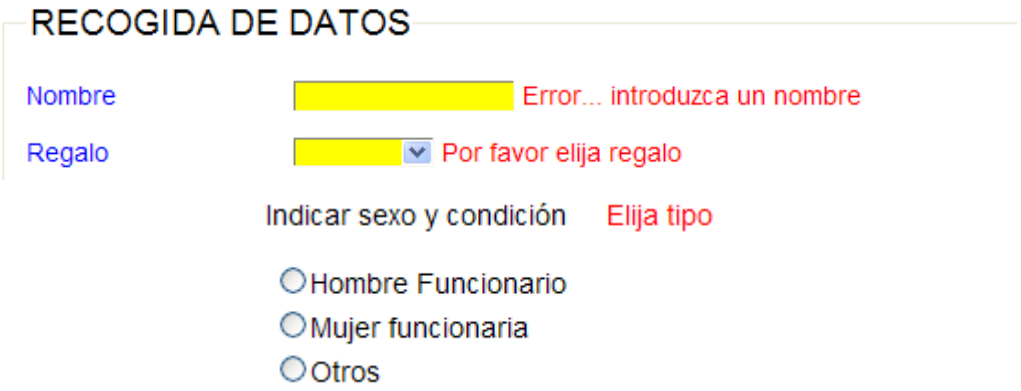
Ejercicios:

Antes de comenzar el ejercicio asegúrate que tienes un botón de servidor con `CausesValidation="true"`

ValidationSummary

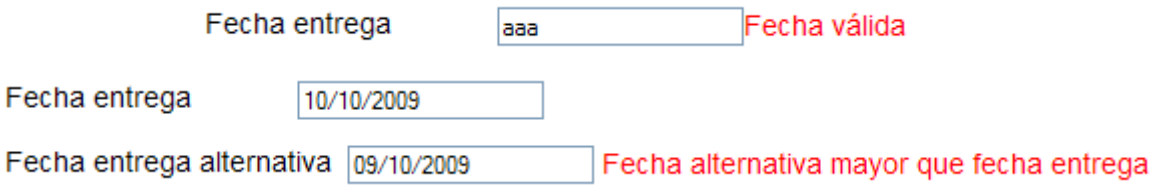
Añade un control de este tipo al final del documento.

RequiredFieldValidator



CompareValidator

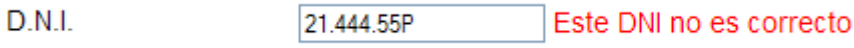
Dos controles: comprobar que fecha de entrega es una fecha y que la fecha de entrega alternativa es mayor que la fecha de entrega.



RangeValidator

RegularExpressionValidator

El DNI responde al patrón NN.NNN.NNNL donde N es número y L es letra.

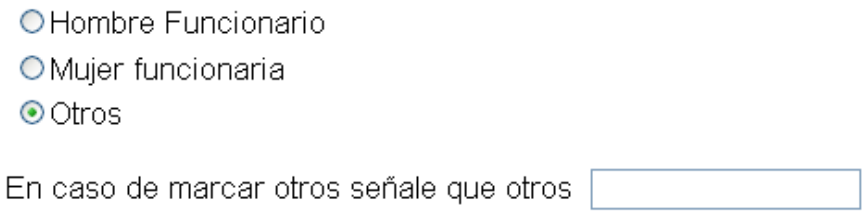


CustomValidator

Crea un control de servidor llamado “consola” y pinta al hacer click en el botón el valor de Page.IsValid

```
protected void Button1_Click(object sender, EventArgs e)     {
    consola.Text = "--->" + Page.IsValid.ToString();
}
```

Aunque en futuros temas ahondaremos en cómo hacer funciones vamos a establecer una que valide cuando seleccionamos “otros” que sea obligatorio insertar datos en la caja “txtOtros”. Importante: que considere el texto vacio.




```
protected void  ValidaOtros(object source, ServerValidateEventArgs
args) {
   args.IsValid = true;
   if (radTipo.SelectedValue == "Otros" & txtOtros.Text.Length == 0)
        {            args.IsValid = false;         }
```

A) Primero prueba con una función “ValidaOtros” usada para la parte servidor.

B) Elimina la propiedad del botón PostBackUrl y prueba a hacerlo solo desde una función javascript.

C) Junta ambas comprobaciones.