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.

Realizando Validaciones Personalizadas (CustomValidator)

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.

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

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

➢ 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 myfunction(source, args)
…….
……
```

donde ***source*** es el objeto CustomValidator del lado cliente, y ***args*** 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.

La propiedad ValidateEmptyText

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

A diferencia de RangeValidator , CompareValidator y RegularExpressionValidator, es posible validar un campo del formulario con el control CustomValidator incluso cuando el campo de formulario se deja en blanco. El control CustomValidator incluye una propiedad denominada ValidateEmptyText que podemos utilizar para hacer que el control de CustomValidator valide un campo incluso cuando el usuario no ha introducido un valor en el campo de formulario .

Por ejemplo, la página del siguiente ejemplo contiene un cuadro de texto que requiere de un código de producto que ha de contener exactamente cuatro caracteres. *(en amarillo la validación de servidor)*

```
<%@ Page Language="VB" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">

    Sub valProductCode_ServerValidate(ByVal source As Object, ByVal args As ServerValidateEventArgs)
        If args.Value.Length = 4 Then
            args.IsValid = True
        Else
            args.IsValid = False
        End If
    End Sub
</script>
<html xmlns="http://www.w3.org/1999/xhtml" >
<head id="Head1" runat="server">
    <title>Show Validate Empty Text</title>
</head>
<body>
    <form id="form1" runat="server">
```

```
    <div>

    <asp:Label
        id="lblProductCode"
        Text="Product Code:"
        AssociatedControlID="txtProductCode"
        Runat="server" />
    <br />
    <asp:TextBox
        id="txtProductCode"
        Runat="server" />


<asp:CustomValidator
        id="valProductCode"
        ControlToValidate="txtProductCode"
        Text="(Invalid product code)"
        ValidateEmptyText="true"
        OnServerValidate="valProductCode_ServerValidate"
        Runat="server" />

    <br /><br />

    <asp:Button
        id="btnSubmit"
        Text="Submit"
        Runat="server" />

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

Observa que el control CustomValidator incluye la propiedad ValidateEmptyText a True. De no ser así, al enviar el formulario sin introducir datos, no se mostrará ningún mensaje error de validación .

ojo!!, fíjate cómo el primer bloque de código pintado de amarillo **NO es código de cliente**, sino visual basic de servidor, y es que también de esta manera se puede indicar el código de servidor (<script runat="server">) *incluyendo el atributo runat="server" en el tag <script>*

A continuación se muestre el mismo ejemplo incluyendo validación en cliente y en servidor:

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="Default.aspx.vb" Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>

    <script type="text/javascript">
  function validarLongitud(oSrc, args){
    args.IsValid = (args.Value.length =4 );
}
</script>


</head>
<body>
    <form id="form1" runat="server">
    <div>
    <asp:Label
        id="lblProductCode"
        Text="Product Code:"
        AssociatedControlID="txtProductCode"
        Runat="server" />
    <br />
    <asp:TextBox
        id="txtProductCode"
        Runat="server" />
<asp:CustomValidator
        id="valProductCode"
        ControlToValidate="txtProductCode"
        Text="(Invalid product code)"
        ValidateEmptyText="true"
        OnServerValidate="valProductCode_ServerValidate"
        Runat="server" ClientValidationFunction="validarLongitud" />

    <br /><br />

    <asp:Button
        id="btnSubmit"
        Text="Submit"
        Runat="server" />

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

```
 Protected Sub valProductCode_ServerValidate(source As Object, args As
System.Web.UI.WebControls.ServerValidateEventArgs) Handles
valProductCode.ServerValidate
        If args.Value.Length = 4 Then
            args.IsValid = True
        Else
            args.IsValid = False
        End If
    End Sub
```

SetFocusOnError

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