Para Validar controles de entrada de formularios (sprint 4)

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.

1. **RequiredFieldValidator:** Asegura que el usuario no se deja un campo en blanco.
2. **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.).
3. **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.
4. **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.

5. **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.

1. **RequiredFildValidator:**
    - **ControlToValidate:** Establece el control de entrada que se va a validar.
    - **ErrorMessage:** Establece el texto del mensaje de error que se muestra en un control.
    - **CssClass:** Agregar estilos boostrap.

```
<asp:RequiredFieldValidator ID="rfvNombre" runat="server" ErrorMessage="Poporcione el nombre"
    ControlToValidate="txtNombre" CssClass="text-danger"></asp:RequiredFieldValidator><br/>
```

2. **CompareValidator:**
    - **ControlToCompare:** Establece el control de entrada para compararlo con el control de entrada que se está validando.
3. **RangeValidator:**
    - **MaximumValue:** Establece el valor máximo del intervalo de validación.
    - **MinimumValue:** Establece el valor mínimo del intervalo de validación.
    - **Type:** Tipo de datos que especifica cómo interpretar los valores que se van a comparar.

```
<asp:RangeValidator ID="rvFechaNacimiento" runat="server" ErrorMessage="La fecha debe estar ente 01-01-1980
    y 31-12-1990" ControlToValidate="txtFechaNacimiento" CssClass="text-danger" MaximumValue="01-01-1990"
    MinimumValue="31-12-1980" Type="Date"></asp:RangeValidator>
```

4. **RegularExpressionValidator:**
    - **ValidationExpression:** Establece la expresión regular que determina el patrón utilizado para validar un campo.
    - **MatchTimeout:** Establece el intervalo de tiempo máximo para ejecutar una única operación de coincidencia antes de que se agote el tiempo de espera de la operación.

```
<asp:RegularExpressionValidator ID="revCurp" runat="server" ErrorMessage="No cumple con el formato"
    ControlToValidate="txtCURP" CssClass="text-danger" ValidationExpression="^[A-Z]{1}[AEIOU]{1}[A-Z]{2}
</asp:RegularExpressionValidator></br>
```

5. **CustomValidator:**
    - **ClientValidationFunction:** Establece el nombre de la función de script de cliente personalizada que se usa para la validación.
    - **ValidateEmptyText:** establece un valor booleano que indica si se debe validar el texto vacío.

```
<asp:CustomValidator ID="cvCURPNacimiento" runat="server" ErrorMessage="Fecha del CURP no coincide con la de
    Nacimiento"
    ControlToValidate="txtCURP" CssClass="text-danger" OnServerValidate="cvCURPNacimiento_ServerValidate">
</asp:CustomValidator>
```

CustomValidator: Realiza la validación definida por el usuario en un control de entrada.

Los controles de validación siempre realizan la validación en el servidor. También tienen una implementación completa del lado cliente que permite que los exploradores habilitados para scripts (como Microsoft Internet Explorer 4.0 y versiones posteriores) realicen la validación en el cliente. La validación del lado cliente mejora el proceso de validación comprobando la entrada del usuario antes de enviarla al servidor. Esto permite detectar errores en el cliente antes de enviar el formulario, lo que evita el recorrido de ida y vuelta de la información necesaria para la validación del lado servidor.

```
Sub ValidationFunctionName(source, arguments)
```

```
function ValidationFunctionName(source, arguments)
```

El **source** parámetro es una referencia al elemento representado para el control CustomValidator . Esto le permite controlar mediante programación la etiqueta, como modificar el InnerHtml atributo . El **arguments** parámetro es un objeto con dos propiedades: Value e IsValid. Este parámetro permite obtener el valor del control que se va a validar e indicar si el valor es válido en función de la rutina de validación personalizada.

Use la propiedad para especificar el nombre de la función de script de validación del lado **ClientValidationFunction** cliente asociada al control CustomValidator . Dado que la función de script se ejecuta en el cliente, la función debe estar en un lenguaje compatible con el explorador de destino, como VBScript o JScript.

Validación del lado del Cliente

```
<script type="text/javascript">
    function CurpMATCHFechaNac(source, args) {
        var fechaNac = $("#<%=TexBoxFechaNacimCreate.ClientID%>").val();
        var CurpPARTfecha = $("#<%=TexBoxCurpCreate.ClientID%>").val().substr(4, 6);
        var FechaNacFormatCurp = fechaNac.substr(2, 2) + fechaNac.substr(5, 2) + fechaNac.substr(8,2);
        args.IsValid = CurpPARTfecha == FechaNacFormatCurp;
    }
</script>
```

```
$("#<%=TexBoxFechaNacimCreate.ClientID%>").val();
```

<%= %>: Se agregan estos símbolos para escribir código C#

```
$("#<%=TexBoxCurpCreate.ClientID%>").val().substr(4,6);
```

<%= %>: Se agregan estos símbolos para escribir código C#

ClientIDMode: static Evita que cambie el ID de la etiqueta ASP del lado del cliente.

```
ClientIDMode
```

Validación del lado del servidor

```
protected void CustomValidator1_ServerValidate(object source, ServerValidateEventArgs args)
{
    var fechaNac = TexBoxFechaNacimEdit.Text;
    var ExtracCurpfecha = TexBoxCurpEdit.Text.Substring(4,6);
    var FechaNacFormatCurp = fechaNac.Substring(2, 2) + fechaNac.Substring(5, 2) + fechaNac.Substring(8, 2);
    args.IsValid = ExtracCurpfecha == FechaNacFormatCurp;
}
```

Para activar validación del lado del servidor al guardar datos.

```
protected void BtnGuardar_Click(object sender, EventArgs e)
{
    if(Page.IsValid)
    {
```

(sprint 5)

```
<!--Agregado de constante/variables globales-->
<appSettings>
    <add key="UMA" value="96.22"/>
</appSettings>
```

```
//Para Obtener valor de variables Globales declaradas en Web config
decimal uma = Convert.ToDecimal(ConfigurationManager.AppSettings["UMA"]);
```