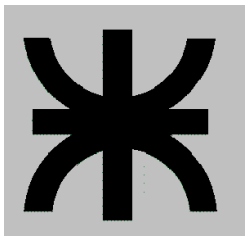


UNIVERSIDAD TECNOLOGICA NACIONAL
FACULTAD REGIONAL GENERAL PACHECO

TÉCNICO SUPERIOR EN PROGRAMACIÓN

PROGRAMACIÓN III

UNIDAD 2:

Controles de Validación

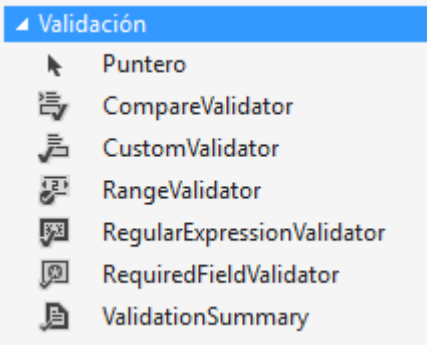
TSSI TAMARA HERRERA

Contenidos de la unidad

1. Controles de tipo validación
    - RequiredFieldValidator
    - CompareValidator
    - RangeValidator
    - RegularExpressionValidator
    - CustomValidator
    - ValidationSummary
    - Vincular controles de validación a botones específicos

Controles de tipo validación

Es importante poder asegurarnos que el usuario haya ingresado bien los datos. En el cuadro de herramientas, podemos encontrar controles que nos ayudarán a validar la información ingresada por el usuario.



Función de los controles

Control	Función
RequiredFieldValidator	Valida que el usuario ingrese algún valor en el control.
CompareValidator	Compara el control validado contra una variable o contra otro control.
RangeValidator	Verifica que lo que haya ingresado el usuario se encuentre entre un rango de valores.
RegularExpressionValidator	Compara lo ingresado por el usuario contra una expresión regular creada por el programador.
CustomValidator	Sirve para validar en el servidor, podemos controlar el evento que este produce.
ValidationSummary	Sobre este control, se puede mostrar una lista de errores obtenidos de los controles validación

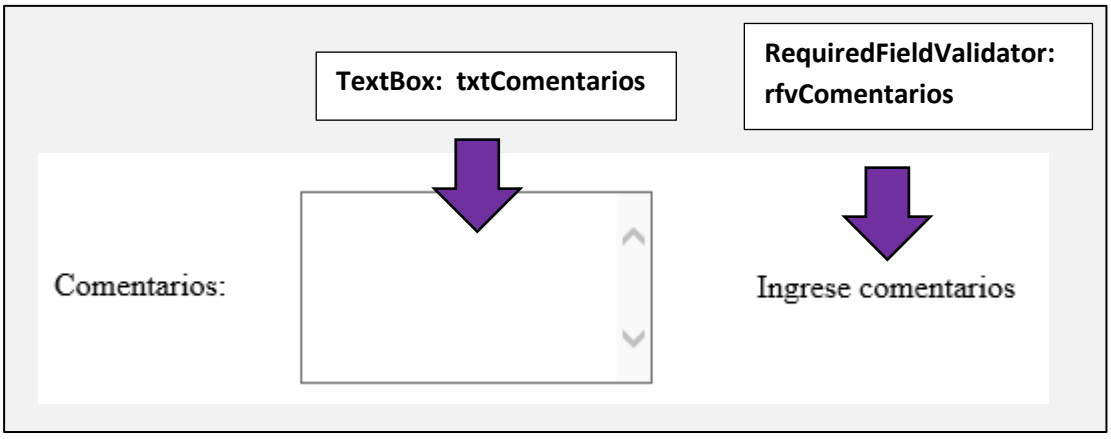
RequiredFieldValidator

El RequiredFieldValidator se utiliza para saber si el usuario escribió o seleccionó algo en un control. En el caso de un TextBox se fijará si escribió y en el caso de un control de tipo lista, se fijará si seleccionó algo de esa lista. El objetivo del RequiredFieldValidator es que el usuario escriba un valor dentro del control que va a ser validado. Entre las propiedades más utilizadas se encuentran:

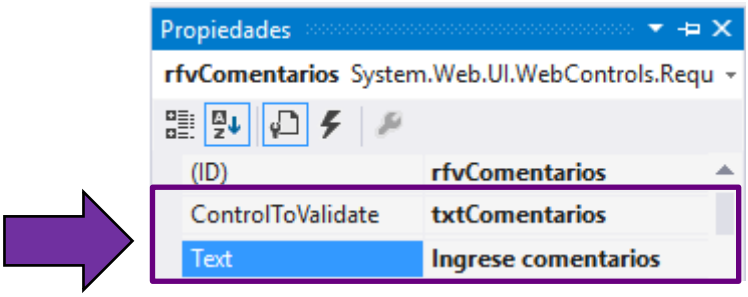
Propiedad	Función
ControlToValidate	Aquí colocaremos el ID del control a validar.
Text	Es el texto que se va a mostrar sobre el validador cuando el control validado no sea válido.
InitialValue	Es un string que contiene el valor inicial que puede tener el control. Si al momento de validar todavía sigue con este valor, quiere decir que el usuario no lo modificó, por ende se mostrará el mensaje de error

Ejemplo 1A

Vamos a validar que el usuario haya escrito algo sobre el TextBox. Si el usuario no completa el TextBox entonces se le mostrará un mensaje que diga “Ingrese comentarios”.



Para validar que el usuario escriba algo sobre el TextBox, tenemos que cambiar las siguientes propiedades del RequiredFieldValidator:

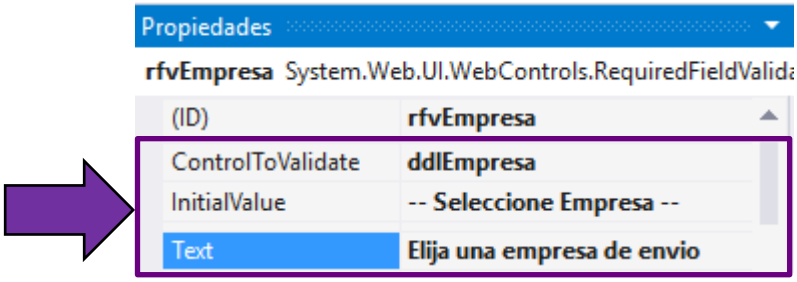


Observación: Con solo vincular el RequiredFieldValidator a un control, ya empieza a validar que el usuario complete algo. El Text es el mensaje de error que se le va a mostrar.

Ejemplo 1B:

Vamos a validar que el usuario seleccione alguna empresa de envío del DropDownList.

Si sobre el DropDownList, se encuentra las palabras “--Seleccione Empresa--”, significa que el usuario aún no ha seleccionado nada, por ende se le mostrará el mensaje de error. Para lograr eso cambiaremos las siguientes propiedades:



Observación: Lo único distinto que hicimos, fue agregar la propiedad InitialValue. Si sobre el control se encuentra seleccionado el InitialValue entonces se mostrará el mensaje de error.

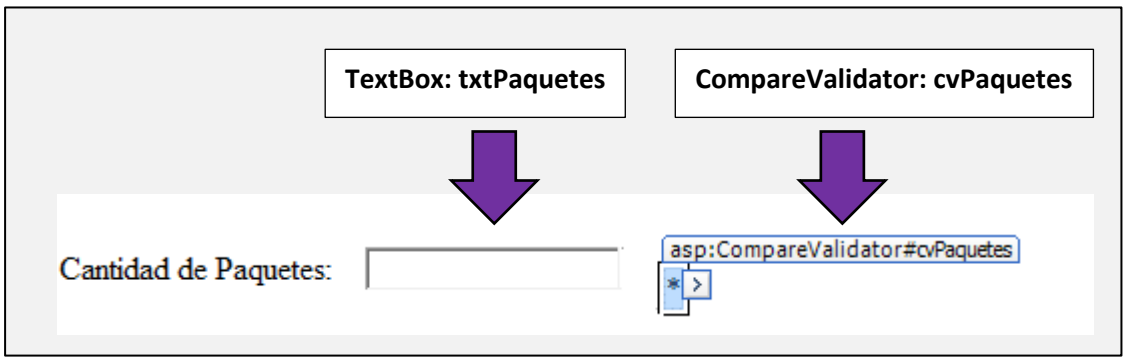
CompareValidator

Este control sirve para comparar el valor ingresado por el usuario contra un valor constante o contra otro control. Entre las propiedades más utilizadas se encuentran:

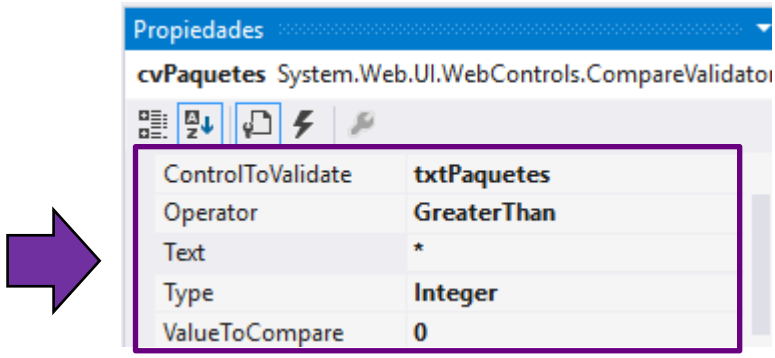
Propiedad	Función
ControlToValidate	Aquí colocaremos el ID del control a validar.
Text	Es el mensaje que se mostrará sobre el validador, si el usuario no cumple con la condición.
Type	Tipo de datos que se van a comparar.
Operator	Operador a utilizar entre los valores que se van a comparar.
ValueToCompare	En el caso de que se compare contra un valor constante, aquí se coloca el valor constante contra el que se va a comparar
ControlToCompare	En el caso de que se comparen controles, aquí se coloca el ID del control contra el que se va a comparar.

Ejemplo 2A:

Validar que el usuario ingrese solo valores positivos en un TextBox. Si ingresa un valor negativo entonces se mostrará un asterisco como mensaje de error.



Para validar que el usuario solo ingrese valores positivos tendremos que cambiar las siguientes propiedades en el CompareValidator.

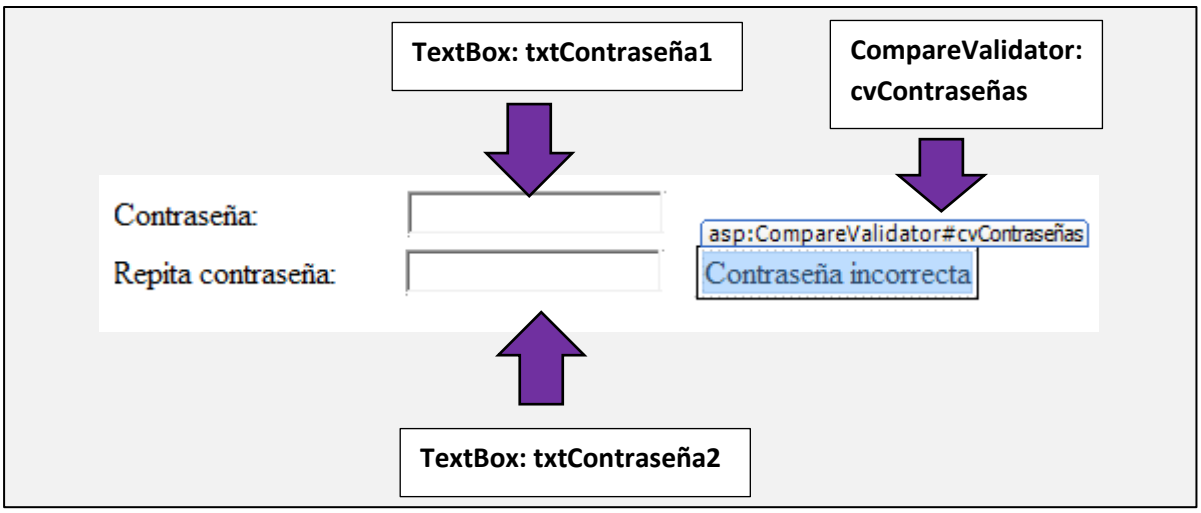


Observaciones: De esta manera, le estoy indicando al CompareValidator que solo se pueden ingresar valores de tipo enteros mayores que 0.

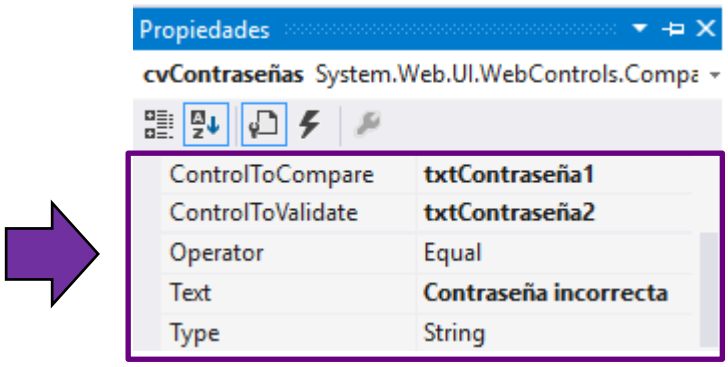
Si se quiere comparar contra otro control, entonces lo que tendríamos que hacer es utilizar la propiedad “ControlToCompare”, en lugar de “”ValueToCompare”.

Ejemplo 2B

Validar que el usuario escriba lo mismo en el TextBox1 y en el TextBox2. Si en el TextBox2 el usuario no repite lo del TextBox1, se le mostrará el mensaje “Contraseña incorrecta”.



Para eso tendremos que cambiar las siguientes propiedades sobre el CompareValidator:



Observación: En este caso estoy comparando el valor que ingrese en *txtContraseña1*, contra el valor que ingrese en *txtContraseña2*. Trabajo con el operador "*Equal*" para determinar que sean iguales y colocó el tipo de dato con el cuál voy a trabajar: "*String*"

RangeValidator

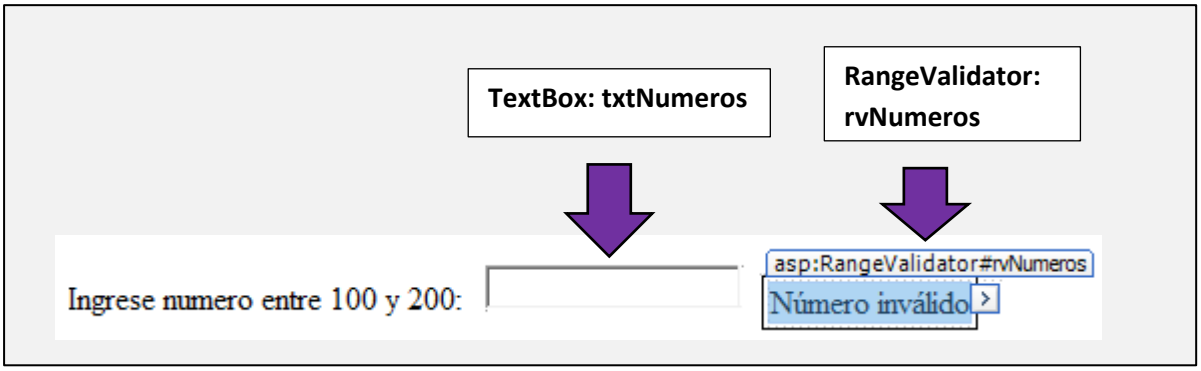
Compara si lo que ingreso el usuario se encuentra entre un rango de valores.

Propiedades más utilizadas:

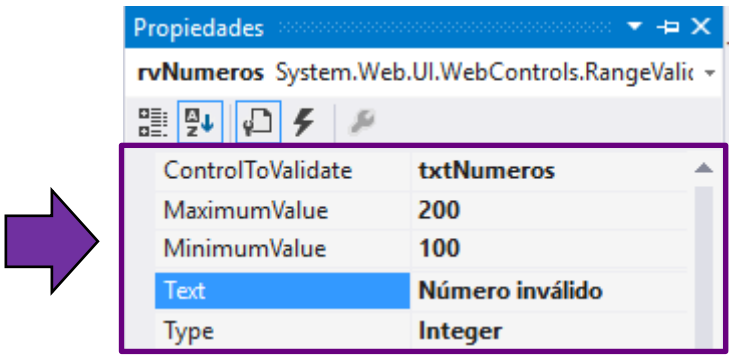
Propiedad	Función
ControlToValidate	Aquí colocaremos el ID del control a validar.
MaximumValue	Valor máximo del control que se va a validar.
MinimunValue	Valor mínimo posible del control que se va a validar
Type	Establece el tipo de datos en que se van a convertir los valores antes de realiza la comparación.
Text	Texto que va a mostrar el validador, cuando no sea válido

Ejercicio 3:

Validar que el usuario ingrese un número entre 100 y 200. En el caso de que ingrese un número que no esté entre ese rango, se le mostrará al usuario el siguiente mensaje “Número inválido.”



Para realizar eso, configuremos las siguientes propiedades en el RangeValidator:



Observación: Si el usuario ingrese un número fuera del rango se mostrará sobre el RangeValidator el texto “Número Invalido”,

RegularExpressionValidator

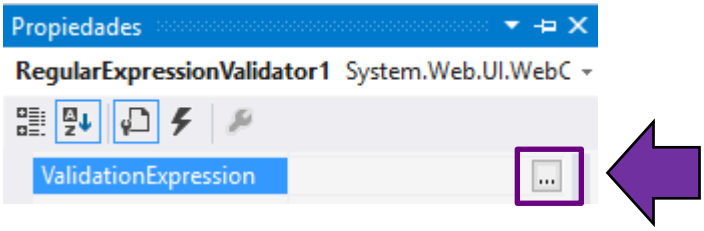
Comprueba que la entrada del usuario coincide con un modelo definido por una expresión regular. Este tipo de validación permite comprobar secuencias de caracteres predecibles, como los que aparecen en las direcciones de correo electrónico, números de teléfono, códigos postales, etc.

Propiedades más utilizadas:

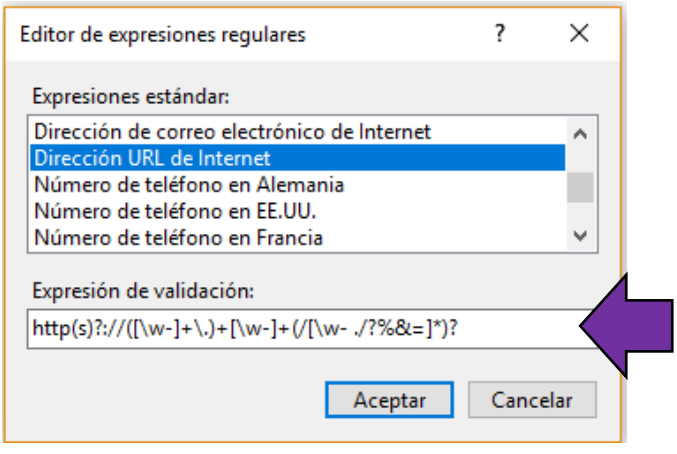
Propiedad	Función
ControlToValidate	Aquí colocaremos el ID del control a validar.
ValidationExpression	Expresión regular para determinar la validez.
Text	Texto que se va a mostrar cuanto el control validado no sea válido.

Una expresión regular es un tipo de notación ya definida mediante un lenguaje especial, es así que por ejemplo, la expresión regular que hace mención a los números es: **^\d+$**. *Hay algunas expresiones regulares que están predefinidas:*

i. Si vamos a la propiedad "*ValidationExpression*", que se encuentra dentro del control RegularExpressionValidator

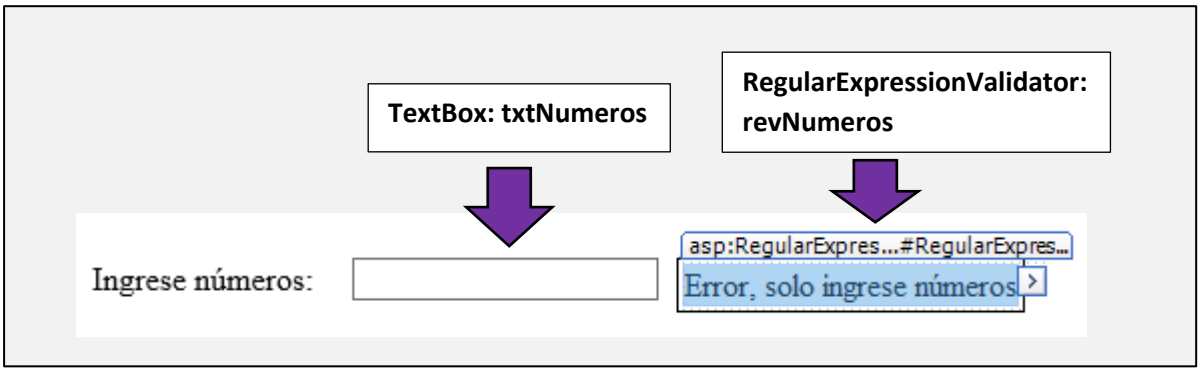


ii. Vamos a poder seleccionar entre algunas expresiones regulares predefinidas.

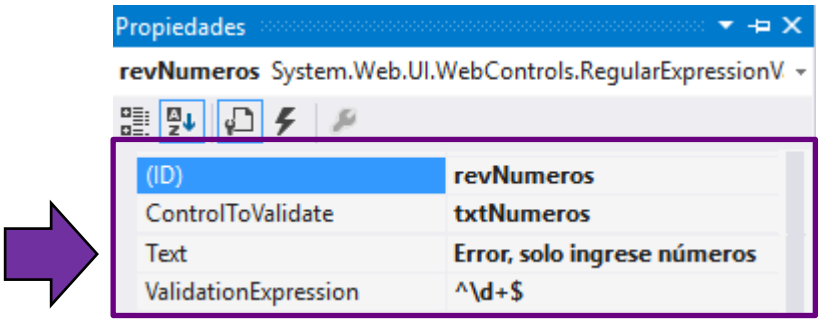


Ejercicio 4:

Validar que el usuario ingrese solo números sobre un TextBox. En el caso de que el usuario ingrese algo no válido se le mostrará el siguiente mensaje: “Error, solo ingrese números”.



Para eso vamos a cambiar las siguientes propiedades sobre el RegularExpressionValidator:



Para obtener referencias de cómo crear una expresión regular, pueden verificar los siguientes link de Microsoft:

https://msdn.microsoft.com/en-us/library/az24scfc(v=vs.110).aspx

https://msdn.microsoft.com/en-us/library/ae5bf541(v=vs.100).aspx

CustomValidator

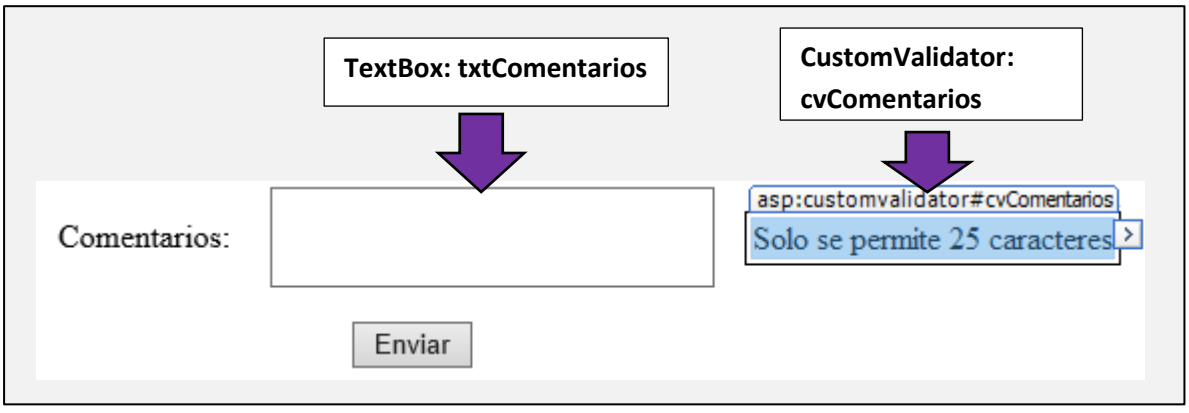
Este control, nos permite realizar validaciones más flexibles, ya sea del lado del cliente o del lado del servidor. Para realizar validaciones del lado del servidor, utilizaremos el evento *ServerValidate*.

Propiedades más utilizadas:

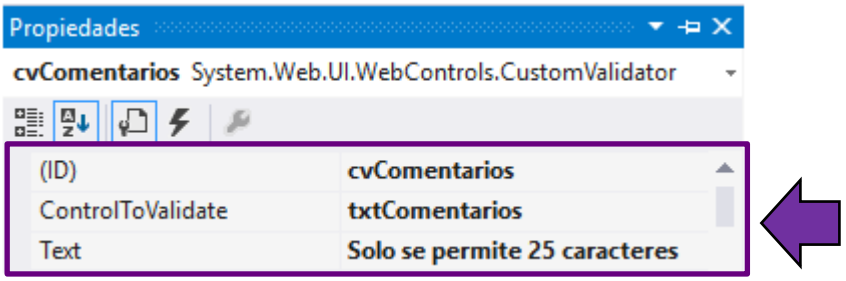
Propiedad	Función
ControlToValidate	ID del control que se va a validar.
Text	Texto que se va a mostrar cuanto el control validado no sea válido.

Ejercicio 5:

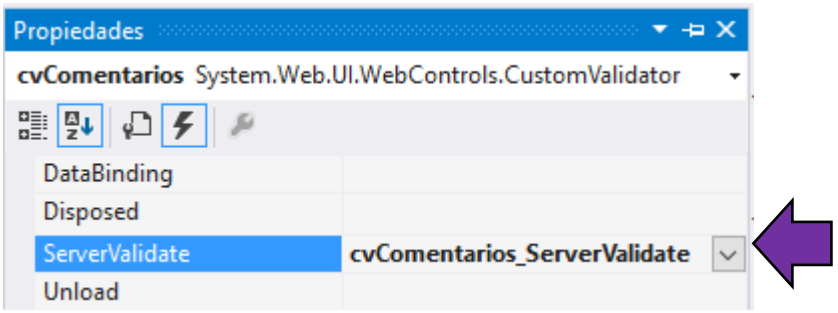
Vamos a validar que se ingresen solo 25 caracteres en el TextBox. Si se ingresan más, le mostraremos al usuario el siguiente mensaje: “Solo se permite 25 caracteres”.



A. Vamos a modificar las siguientes propiedades del CustomValidator:



B. Generamos el evento ServerValidate



C. Colocamos lo siguiente en el código:

```
protected void cvComentarios_ServerValidate(object source,
                                            ServerValidateEventArgs args)
{
    if (args.Value.Length <= 25)
        args.IsValid = true;
    else
        args.IsValid = false;
}
```

Al ser un evento que se valida en el servidor, recién se validará luego de que se haga un PostBack en la página. En este caso, se validará luego de que se haga click en botón enviar.

ValidationSummary

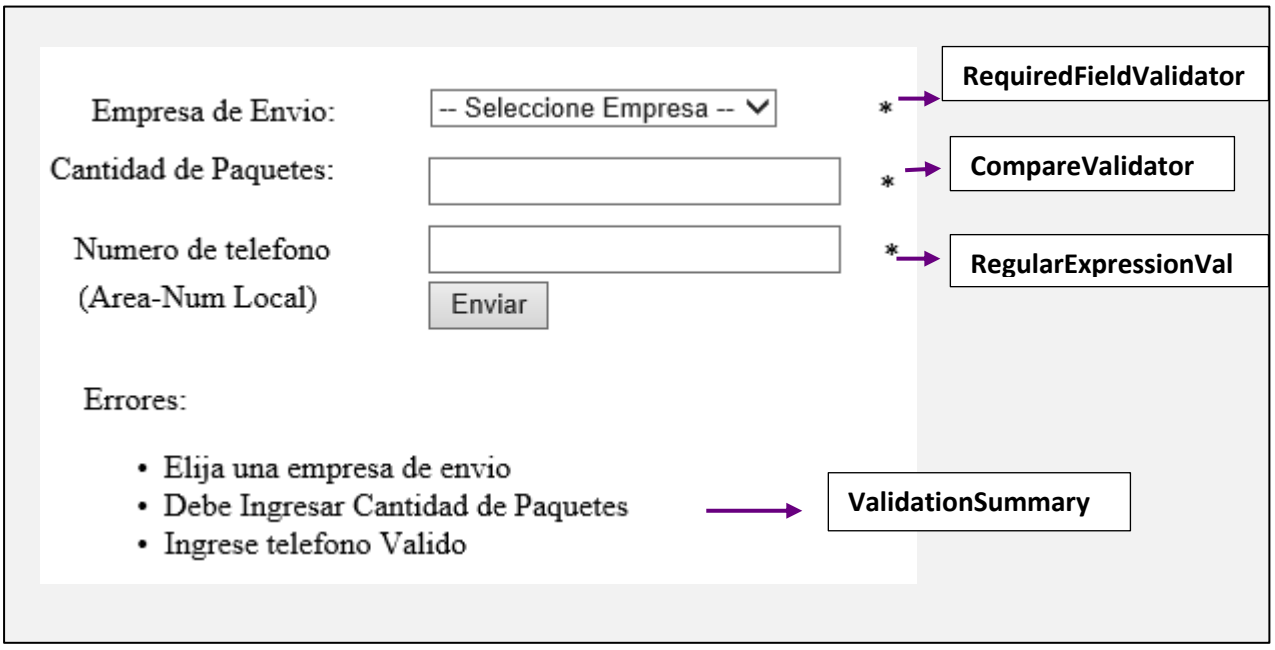
El control ValidationSummary sirve para mostrar una lista con todos los errores de validación que se hayan producido durante el envío del formulario.

Propiedades más comunes:

Propiedad	Función
ShowSummary	Si está en True, vamos a ver un cuadro dentro de nuestro formulario con los errores de los controles de validación.
ShowMessageBox	Si está en True, muestra un pop up con los errores que se produjeron en nuestro formulario.
HeaderText	Texto que se va a mostrar en el encabezado del listado de errores.

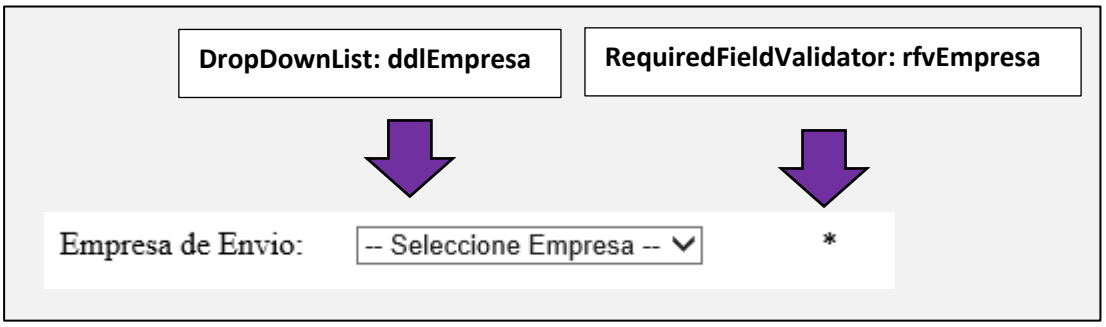
Ejemplo 6:

Hacer un programa para que al validar los controles de una página, muestre al lado del control un asterisco (*) en referencia que el usuario se equivocó. Además especificar los errores en un listado en la parte posterior de nuestro formulario.

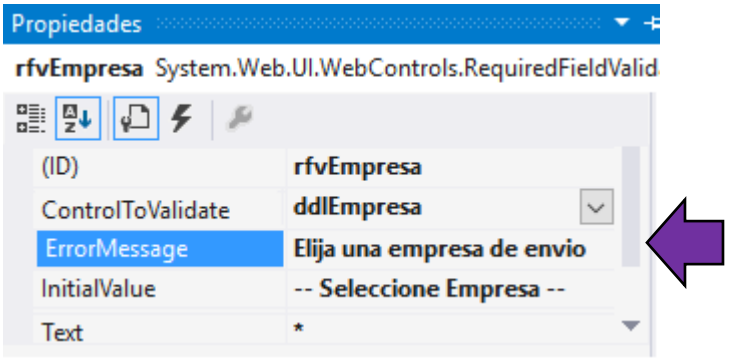


¿Cómo hacemos para que se muestren los mensajes de error de cada control en el ValidationSummary?

Sencillo, lo que se muestra en el ValidationSummary es la propiedad ErrorMessage configurada en cada control de validación. Por ejemplo, tenemos el RequiredFieldValidator vinculado a una Empresa de envío.



A ese RequiredFieldValidator, vamos a colocarle en la propiedad *ErrorMessage* "Elija una empresa de envío":



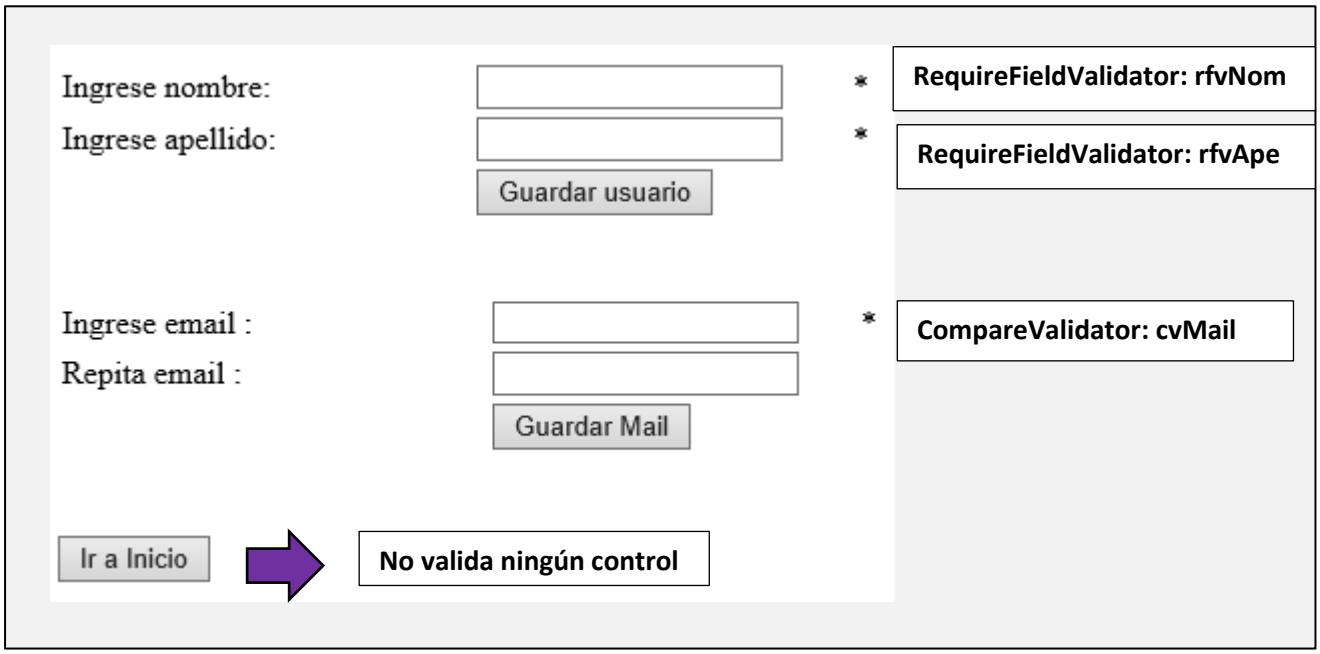
Al realizar esto, lo que va a suceder es que: cuando el usuario se equivoque en el ingreso, sobre el control se va a mostrar la propiedad Text *(*),* pero en el ValidationSummary, se va a mostrar lo que colocamos en la propiedad ErrorMessage *(Elija una empresa en envío).*

En conclusión, debe completar la propiedad ErrorMessage de todos los controles los cuáles usted quiera que se visualicen en el ValidationSummary.

Vincular un control de validación a un botón

En el siguiente formulario, tendremos tres botones.

- **Guardar usuario:** Guarda el usuario luego de que se completen los TextBox de Nombre y Apellido.
- **Guardar mail:** Guarda el mail luego de que el usuario repita el mail de manera correcta en los dos TextBox de Emails.
- **Ir al inicio:** Redirecciona a la página Inicio.aspx



A continuación, vamos a ver cómo podemos agrupar las validaciones para que trabajen en conjunto con un botón. Lo que queremos hacer es que ciertas validaciones solo se produzcan al realizar click en cierto botón.

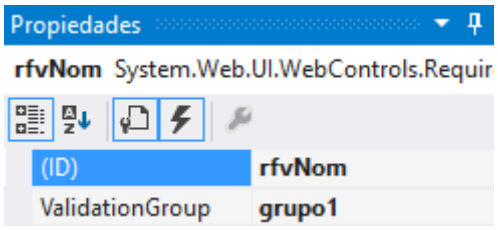
- Necesitamos que cuando se haga click en el botón “Guardar usuario” solo se realicen las validaciones del RequiredFieldValidator (rfvNom y rfvApe).
- A su vez, también necesitamos que cuando se haga click en el botón “Guardar mail”, solo se realice la validación del CompareValidator (cvMail).

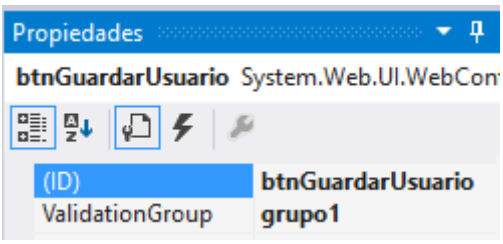
- El botón “Ir al Inicio” no tiene que estar vinculado a ningún control de validación.

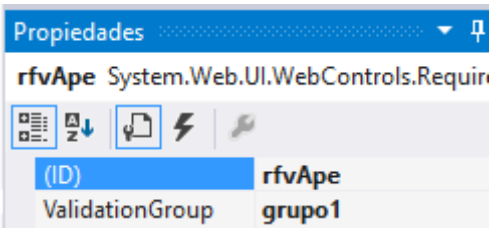
Para que los botones y validadores trabajen en conjunto, deberemos a ese mismo grupo colocarle el mismo nombre en la propiedad ValidationGroup. El nombre es a elección.

Configuración del botón guardar usuario

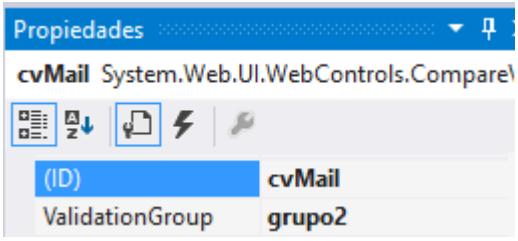
Tanto a los RequiredFieldValidator, como al botón le cambiamos la propiedad ValidationGroup = grupo1.

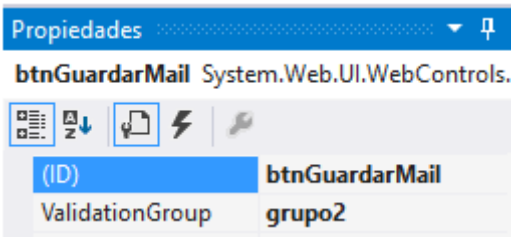






Configuración del botón guardar mail





Configuración del botón Ir a Inicio

Este botón, trabajará de manera independiente sin validaciones. Por lo que tendremos que cambiarle la propiedad CauseValidation= False