

Capítulo 9

Introducción a ASP.NET y WebForms

Objetivo

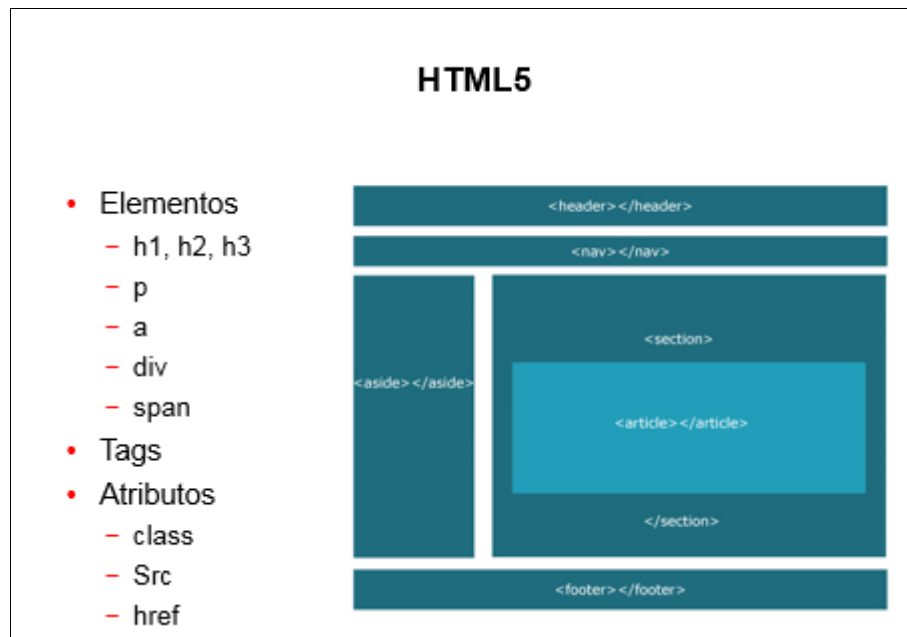
Al finalizar el capítulo, el alumno:

- Comprender las características de un WebForm.
- Identificar los tipos de controles ASP.NET.
- Crear un formulario usando controles Web ASP.NET.
- Invocar un control de usuario.

Temas

1. HTML5
2. Introducción a ASP.NET
3. Introducción a WebForms
4. Controles ASP.NET
5. Controles de validación
6. Elementos de una aplicación ASP.NET
 - Carpeta Bin
 - Archivo Global.asax
 - Archivo de configuración Web.config

1. HTML5



HTML5 es la última versión del estándar que define HTML (HyperText Markup Language). Esta versión incorpora nuevos elementos, atributos y comportamientos.

HTML nos sirve para definir la estructura de las páginas Web que son mostradas en el browser. HTML es ampliamente adoptado en la implementación de sitios web, al igual que otras tecnologías como CSS y Javascript.

Términos usados en HTML:

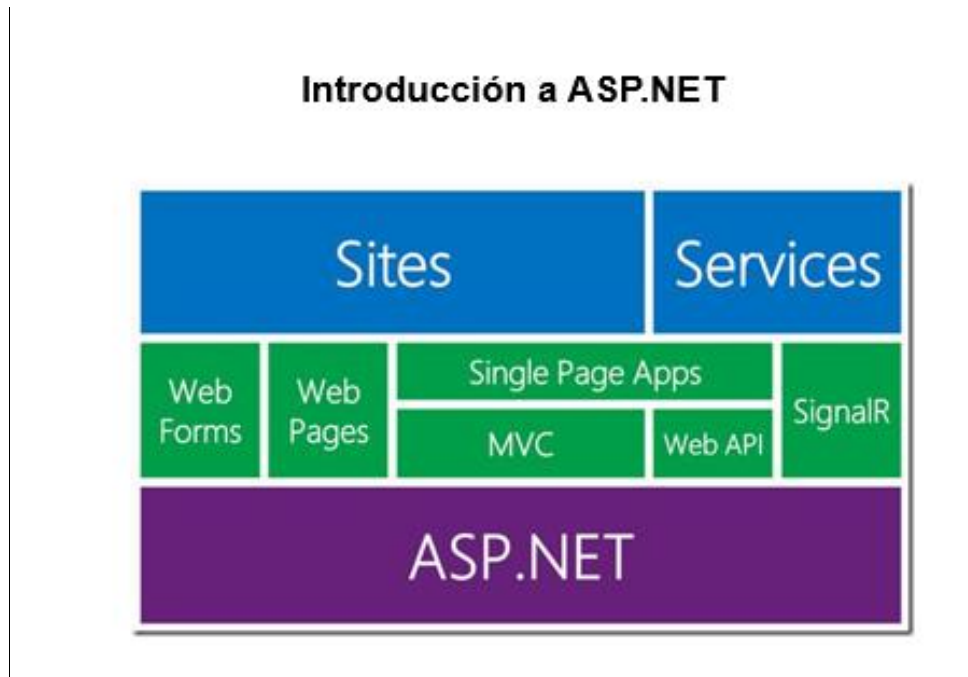
- **Elementos:** definen la estructura y contenido de objetos dentro de una página, entre ellos tenemos los headings (h1, h2, h3, etc.) los párrafos (p) las dimensiones (a) y otros (div, span).
- **Tags:** se utilizan para delimitar un elemento indicando el opening y ending del mismo: Ejemplo: <a> . . .
- **Atributos:** son las propiedades que proporcionan información adicional sobre un elemento. Los más comunes son: id (para identificar un elemento), class (el cual clasifica a un elemento), src (que especifica la ubicación de la fuente del contenido para el elemento) y href (que proporciona una referencia a un recurso enlazado).

Ejemplo de HTML:

```
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Hello World</title>
</head>
<body>
  <h1>Hello World</h1>
  <p>This is a web page.</p>
</body>
</html>
```

HTML trae consigo nuevos elementos conocidos como elementos semánticos, algunos de estos elementos son: header, footer, section, article, aside, etc.

2. Introducción a ASP.NET



ASP.NET forma parte de .NET Framework y al codificar las aplicaciones, tiene acceso a las clases en .NET Framework. El código de las aplicaciones puede escribirse en cualquier lenguaje compatible con el Common Language Runtime (CLR) y entre ellos se encuentran Microsoft Visual Basic y C#.

Estos lenguajes permiten desarrollar aplicaciones ASP.NET que se benefician del Common Language Runtime tales como seguridad de tipos, herencia, etc.

WebForms y MVC

Se tienen ambas opciones para el desarrollo de aplicaciones Web robustas, cada una de estas con sus fortalezas y debilidades.

IIS Express

IIS Express reemplaza el Servidor de desarrollo de ASP.NET como servidor web predeterminado para probar dentro de Visual Studio.

IIS Express es una versión ligera, autónoma de IIS que se ha optimizado para desarrolladores. Tiene todas las funciones básicas de IIS, así como de características adicionales diseñadas para facilitar el desarrollo de sitios Web e incluye lo siguiente:

- No se ejecuta como un servicio, no requiere derechos de usuario administrador para realizar las tareas.
- IIS Express funciona bien con aplicaciones ASP.NET.

Edición de HTML, CSS y JavaScript

Por el lado de HTML se soporta a los controles de ASP.NET WebForms y los HTML Helpers de Razor.

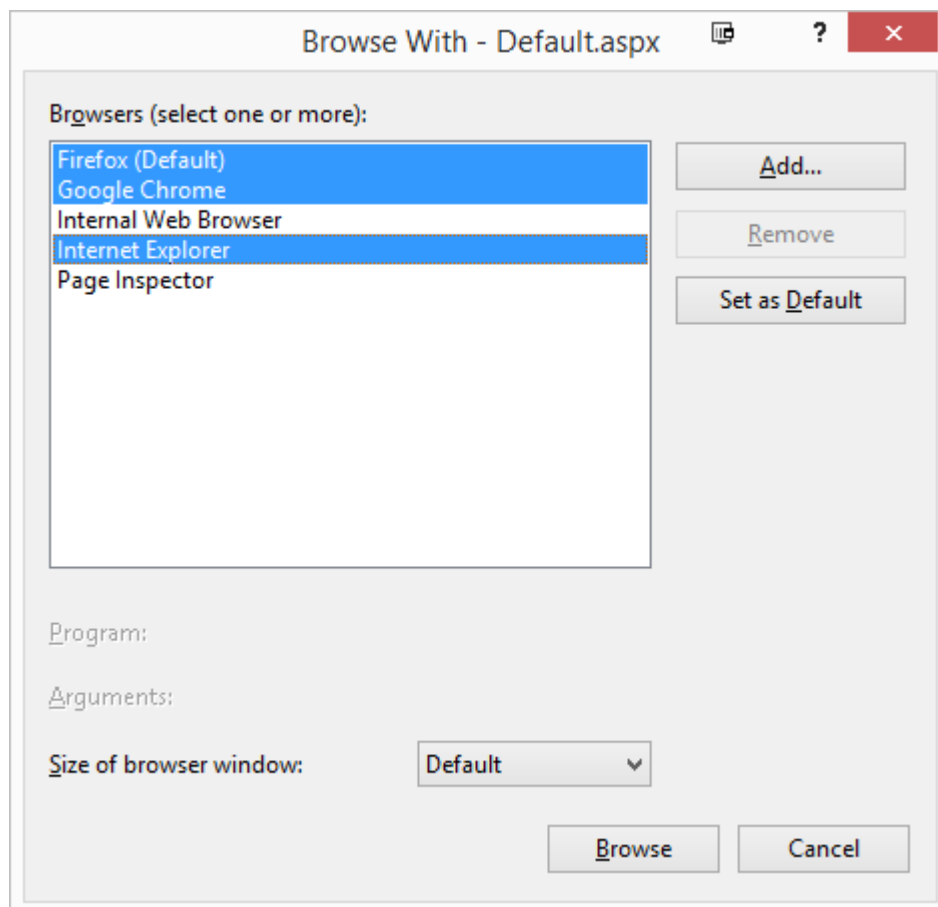
Por el lado de estilos se tiene soporte a CSS (Cascading Style Sheets) y LESS y SASS (Syntactically Awesome Style Sheets).

Por el lado de JavaScript se soporta a TypeScript y se tienen extensiones para múltiples tecnologías basadas en JS.

Compatibilidad Multi Browser

Los exploradores instalados aparecen en una nueva lista desplegable situada junto al botón de Iniciar depuración en Visual Studio. Esta característica le permite probar la misma página, aplicación o sitio en exploradores diferentes.

Para probar en varios exploradores simultáneamente, elija la opción de Browse With en la lista desplegable y seleccione los navegadores a utilizar.



3. Introducción a WebForms

Introducción a WebForms

Ventajas de WebForms

- Se puede construir toda la interfaz de usuario como se construyen los formularios Windows o WPF.
- Permite mantener el estado del formulario de forma automática entre distintas peticiones.
- Permite utilizar controles de servidor que generarán código HTML.
- Permite reutilizar controles de UI que encapsulan su funcionalidad.

Los WebForms (formularios Web) ofrecen un modelo de programación los cuales van a permitir generar contenidos dinámicos de una forma más sencilla. Estos WebForms vienen a sustituir e integrar los formularios HTML, ya que en última instancia, va a generar el código HTML correspondiente al formulario que se está diseñando.

Una de las grandes ventajas que ofrecen los WebForms es un modelo de programación muy similar al de los tradicionales formularios, incluso se puede realizar el tratamiento de eventos del cliente desde el lado del servidor.

Ventajas que ofrecen los WebForms

- Permiten reutilizar los controles de interfaz de usuario (UI) que encapsulan su funcionalidad y reducen de esta forma el código que el desarrollador debe escribir.
- Se puede construir toda la interfaz del WebForm a través del Visual Studio, tal como se construyen los formularios WinForms o WPF, arrastrando y soltando los controles (desarrollo rápido de aplicaciones o RAD).
- Permite mantener el estado del formulario de forma automática entre distintas peticiones, es decir, si se ingresa una serie de valores en los controles del WebForm, dichos valores se van a mantener entre las peticiones y respuestas del Servidor Web.
- Permite utilizar un gran conjunto de controles de servidor que generarán el código HTML necesario para mostrar el WebForm en la página cargada por el cliente en su navegador.

Para crear un WebForm, se debe utilizar la etiqueta <FORM>, al estilo de las etiquetas del lenguaje HTML, pero esta etiqueta debe poseer la propiedad runat, con el valor server.

```
<form id="frmCliente" method="post" runat="server">  
</form>
```

Dentro del WebForm se deben encontrar los distintos controles de servidor ASP.NET, necesarios para recoger la información correspondiente.

Estos controles se dividen en los siguientes tipos:

- HTML Server Controls (Controles que son equivalentes a elementos HTML, se utilizan para tener mayor control sobre estos elementos usando ASP.NET)
- Web Server Controls (Controles que generan varios elementos HTML, ahorrándonos el tiempo que tomaría construir desde cero toda la funcionalidad que presentan)
 - Controles Estándar
 - Controles de Datos
 - Controles de Navegación
 - Controles de Login
 - Controles WebPart
 - AJAX Controls
- Validation Controls
- Users Controls

Ciclo de vida de una página

Etapa	Descripción
Page request	Cuando una página es solicitada por un usuario, ASP.NET determina si la página tiene que ser parseada y compilada, o si una versión en caché de la página puede ser enviada sin tener que ejecutar la página.
Start	En la etapa inicial las propiedades de la página como Request y Response son seteadas. La página también determina si el request es un postback o no. Adicionalmente se setea el UICulture.
Initialization	Durante esta etapa los controles de la página están disponibles y cada ID único del control es seteado. Si el request es un postback los datos del postback aún no se cargaron y los valores de los controles aún no han sido restaurados con los valores del view state.
Load	Durante esta etapa, si el request es un postback, las propiedades de los controles son cargados con información del view state.

Postback event handling	Si el request es un postback, se llaman a los manejadores de eventos.
Rendering	Antes de realizar el rendering de la página el view state es guardado. Durante la fase de Rendering la página llama al método Render de cada control proveendo un text writer que escribe en el OutputStream de la propiedad Response de la página.
Unload	Unload es llamado después de que la página ha sido renderizada totalmente, enviada al cliente, y está lista para ser desechada. En este punto las propiedades como Response y Request son liberadas y se desarrolla todo el proceso de limpieza de recursos.

Eventos de las páginas WebForms

PreInit

- Comprobar la propiedad IsPostBack para determinar si esta es la primera vez que la página se está procesando.
- Crear o recrear controles dinámicos.
- Establecer la página principal de forma dinámica.
- Establecer la propiedad Theme dinámicamente.
- Leer o establecer valores de propiedad de perfil.
- Si la petición es un postback:
 - Los valores de los controles aún no han sido restaurados a partir de estado de vista.
 - Si establece la propiedad de control en esta etapa, su valor puede ser sobrescrito en el próximo evento.

Init

- Este evento se utiliza para inicializar las propiedades de los controles.

InitComplete

- El seguimiento del ViewState se activa.
- Los cambios realizados en el view state en este evento se guardan aún después del siguiente postback.

PreLoad

- Este evento procesa los datos del postback que se incluyen con el request.

Load

- En este caso el objeto Page llama al método OnLoad de dicha página, luego se llama al método OnLoad de los controles.
- El evento Load de los controles individuales se produce después del evento Load de la página.

ControlEvents

- Este evento se utiliza para controlar eventos de control específicos como el evento Click de un control Button o el evento TextChanged de un control TextBox.
- En caso de postback:
 - Si la página contiene controles de validación, la propiedad Page.IsValid y la validación de los controles se lleva a cabo antes de que se disparen los eventos de los controles.

LoadComplete

- Este evento se produce después de la etapa de manejo de eventos.
- Este evento se utiliza para tareas como el cargar todos los demás controles en la página.

PreRender

- Este método se utiliza para hacer los cambios finales a los controles de la página como asignar el DataSourceID y llamar al método DataBind.

PreRenderComplete

- Este evento se lanza después de que las propiedades de PreRender de cada control se obtienen.

SaveStateComplete

- Se lanza después de que el control state y el view state se han salvado para la página y para todos los controles.

RenderComplete

- El objeto Page llama a este método en cada control que está presente en la página.
- Este método escribe el markup del control para enviarlo al navegador.

Unload

- Este evento se genera para cada control y luego para el objeto Page.
- Utilice este evento en los controles para limpieza de recursos, como el cierre de las conexiones de base de datos, cierre los archivos abiertos, etc.

4. Controles ASP.NET

Controles ASP.NET	
Nombre	Código
Caja Texto	<code><input type="text" name="nombrecontrol"></code>
Botón	<code><input type="button" name="nombrecontrol" value="textocontrol"></code>
Botón Submit	<code><input type="submit" name="nombrecontrol" value="textocontrol"></code>
Botón Reset	<code><input type="reset" name="nombrecontrol" value="textocontrol"></code>
Combo	<code><select name="nombrecontrol"> <option value="valoropcion">textoopcion</option> </select></code>
Checkbox	<code><input type="checkbox" name="nombrecontrol" value="textocontrol"></code>
Radio	<code><input type="radio" name="nombrecontrol" value="textocontrol"></code>
Textarea	<code><textarea name="nombrecontrol" rows="40" cols="5"></textarea></code>

Los controles ASP.NET son una serie de objetos de servidor que generarán todo el código equivalente con el lenguaje HTML para que así pueda ser interpretado por cualquier navegador Web.

Los numerosos controles de servidor incluidos en ASP.NET se pueden agrupar en seis categorías o familias:

- Controles estándar, los cuales son los comunes controles de formulario.
- Controles de lista o de datos, utilizados para distribuir y mostrar datos en una página.
- Controles de validación, ofrecen distintos tipos de validación de entrada de datos.
- Controles de navegación, ofrecen la navegabilidad por la página web con controles menús, treeview, etc.
- Controles de login, ofrecen la autenticación de un usuario, empleando el modelo Membership.
- WebParts, controles para aplicaciones SharePoint

Cada control ASP.NET es capaz de ofrecer un completo modelo de los objetos que contienen propiedades, métodos y eventos, puesto que estos controles son instancias de objetos que se pueden encontrar en los espacios de nombre: System.Web.UI.HtmlControls y System.Web.UI.WebControls.

Es preferible usar controles HTML en las siguientes situaciones:

- Cuando se prefiere un modelo de objetos similar al lenguaje HTML.
- El control debe interactuar con script de cliente y de servidor.

Sintaxis

Al igual que sucedía con los controles HTML, para permitir hacer referencia a los controles dentro del código fuente de la página, se debe utilizar el atributo `id`, de esta forma, el objeto que se corresponde con el control de servidor, se puede utilizar en el código del servidor para utilizar sus métodos o manipular sus propiedades.

```
<asp:nombreControl id="identificador" runat="server"/>
```

```
<asp:nombreControl id="identificador" runat="server"></asp:nombreControl>
```

A continuación, se describen algunos de los controles ASP.NET más importantes

- **Label**

El control Label muestra un texto en una ubicación específica de la página a la que pertenece. Se utiliza para colocar textos que estén relacionados a los TextBox.

```
<asp:Label id="lblNombres" runat="server"></asp:Label>
```

- **Textbox**

El control Textbox, permite recolectar información del usuario, a través de su propiedad `Text`, la cual puede ser leída y modificada. Además, el control tiene una propiedad llamada `TextMode`, la que puede ser `SingleLine`, `MultiLine` o `Password`.

Cuando el `TextMode` está seteado en `SingleLine` y en `Password`, puede utilizarse la propiedad `MaxLength`, la cual restringe la cantidad máxima de caracteres que soporta el control. Esta propiedad no funciona cuando el `TextMode` está seteado en `MultiLine`.

Cuando el `TextMode` está seteado en `MultiLine`, se pueden utilizar las propiedades `Columns` y `Rows`; estas determinan la cantidad de columnas y de filas que visualiza el TextBox. La propiedad `Wrap`, si está seteada en `True`, permite visualizar el texto en varias líneas, tomando como ancho base la longitud del TextBox.

```
<asp:Textbox id="txtComentarios" runat="server" TextMode="MultiLine"
Wrap="True" Columns="40" Rows="5"></asp:Textbox>
```

- **Button**

El control Button muestra un botón del tipo PushButton sobre la página Web, al que el usuario puede hacerle clic, para disparar unPostBack hacia el servidor.

El control Button puede ser del tipo Submit Button o Command Button. Un Submit Button no tiene la propiedad CommandName configurada y simplemente ejecuta unPostBack hacia el servidor. Los botones de tipo Command Button, requieren tener configurada la propiedad CommandName, por lo que se pueden crear varios controles Button en una página Web y determinar por código cuál fue clickeado. A través de la propiedad CommandArgument se puede proveer información adicional al evento.

Si se utilizan controles de validación y se quiere evitar la validación de estos controles, hay que configurar la propiedad CausesValidation en False. Por defecto, esta propiedad está en True. Los botones Reset y de ayuda comúnmente tienen esta propiedad en False.

```
<asp:Button id="Button1" runat="server">
```

- **CheckBox**

El control CheckBox, permite mostrar opciones múltiples de selección. Además, la propiedad Text especifica el texto que acompaña al CheckBox. Asimismo, la propiedad TextAlign permite especificar de qué lado del CheckBox aparece el texto (Right|Left).

El evento CheckedChanged es disparado cuando el estado del CheckBox cambia, por defecto, la propiedad AutoPostBack del CheckBox está seteada en False, por lo que si se quiere notar elPostBack, se debe configurar esta propiedad en True.

```
<asp:CheckBox id="CheckBox1" runat="server" AutoPostBack="True"
Checked="True"></asp:CheckBox>
```

- **RadioButton**

El control RadioButton permite al usuario escoger una opción de entre varias. Para poder crear varios grupos de RadioButtons, pero se debe especificar la propiedad GroupName, cuyo valor deberá repetirse entre todos los RadioButtons del mismo grupo.

```
<asp:RadioButton id="RadioButton1" runat="server"
GroupName="GroupName1"></asp:RadioButton>
```

- **Table, TableRow y TableCell**

El control Table se corresponde con una tabla del lenguaje HTML, es decir, permite generar tablas del lenguaje HTML. Esta clase posee una colección llamada Rows, que contiene objetos de la clase TableRow.

```
<asp:Table id="Table1" runat="server">
  <asp:TableRow id="TableRow1" runat="server">
    <asp:TableCell id="TableCell1" runat="server"></asp:TableCell>
  </asp:TableRow>
</asp:Table>
```

- **Image**

El control Image permite mostrar una imagen dentro de una página ASP.NET. Su propiedad ImageURL indica la ruta de la imagen que se ha de mostrar.

Este control está relacionado con la etiqueta y la propiedad ImageURL está relacionada con la propiedad href.

La propiedad AlternateText permite mostrar un mensaje cuando la imagen no esté disponible. La propiedad GenerateEmptyAlternateText configurada en True, agrega a la imagen el atributo alt="".

La propiedad ImageAlign, la cual hace referencia a la alineación que pueda tener la imagen con respecto a los otros objetos de la página, puede tener los siguientes valores: NotSet, Left, Right, Baseline, Top, Middle, Bottom, AbsBottom, AbsMiddle, o TextTop.

La propiedad DescriptionUrl, es utilizada para proveer una página que contiene una explicación con más detalle del contenido y del significado de la imagen, para navegadores no visuales, generando el atributo longdesc.

```
<asp:Image id="Imagen1" runat="server" AlternateText="Tooltip de la
Imagen"></asp:Image>
```

El control Image no tiene el evento Click, por lo que para esto debería utilizarse el control ImageButton o el ImageMap.

- **Calendar**

El control Calendar muestra un calendario para un mes seleccionado. Permite al usuario seleccionar fechas y moverse hacia el siguiente o el anterior mes. El evento SelectionChanged genera un PostBack, cuando el usuario ha seleccionado una nueva fecha y el evento VisibleMonthChanged causa un PostBack cuando el usuario selecciona un mes distinto a ser visitado.

El control Calendar renderiza su salida en una tabla HTML.

```
<asp:Calendar id="Calendar1" runat="server"></asp:Calendar>
```

- **DropDownList**

El control DropDownList representa una lista desplegable, y su correspondiente en HTML es la etiqueta <select>. Al igual que el control CheckBox también posee la propiedad AutoPostBack. Si se establece a True esta propiedad, al seleccionar un elemento de la lista se generará un PostBack al servidor.

Cada uno de los elementos del control DropDownList se indica con un objeto de la clase ListItem.

```
<asp:DropDownList id="DropDownList1" runat="server">  
  <asp:ListItem value="valor1"></asp:ListItem>  
  <asp:ListItem value="valor2" Selected></asp:ListItem>  
</asp:DropDownList>
```

- **ListBox**

Este nuevo control representa una lista de selección sencilla o múltiple, es similar al control DropDownList, pero en este caso se muestran varios elementos de la lista y se permite la selección múltiple.

Este control Web, también, se corresponde con una etiqueta <select> en el código HTML generado, de la misma forma que sucedía con el control DropDownList, pero en este caso con distintas propiedades.

En la propiedad Rows del control ListBox se indica el número de filas visibles que va a mostrar el control, y en la propiedad SelectionMode indicar si se permite la selección múltiple, mediante el valor Múltiple, o bien la selección simple mediante el valor Single, este es el valor por defecto.

```
<asp:ListBox id="ListBox1" runat="server">  
  <asp:ListItem value="valor1"></asp:ListItem>  
  <asp:ListItem value="valor2" Selected></asp:ListItem>  
</asp:ListBox>
```

- **FileUpload**

El control FileUpload es utilizado para mostrar un Textbox y un botón Browse, que permite al usuario, seleccionar la ruta de un archivo. Este control se renderiza en un `<input type="file">`.

El control FileUpload no genera ningúnPostBack al servidor, luego de que el usuario ha seleccionado su archivo; por ello, es necesario que se genere unPostBack a través de otro control. ElPostBack origina que el archivo sea enviado al servidor, luego de que todo el archivo sea cargado en la memoria del servidor, el código de la página recién se ejecuta.

```
<asp:FileUpload ID="FileUpload1" runat="server" />
```

5. Controles de validación

Controles de validación

- ASP.NET WebForms proporciona 6 controles para validación:
 - **RequiredFieldValidator**: valor requerido.
 - **CompareValidator**: valida contra un valor constante o contra otro control.
 - **RangeValidator**: valor dentro de un rango de tipos.
 - **RegularExpressionValidator**: valida contra un patrón o expresión regular.
 - **CustomValidator**: lógica de validación proporcionada por nosotros.
 - **ValidationSummary**: no es un validador, solo muestra mensajes de error “agrupados”.



Los controles de validación, van a permitir de una manera fácil y rápida, realizar validaciones de los datos y de la información que el usuario ingresa en los controles que conforman la página, realizando acciones incluso en el lado del cliente sin que tengamos que escribir ningún código para esto.

Estos controles, al heredar de la clase **BaseValidator**, contienen propiedades comunes:

- La propiedad **ControlToValidate** es común para todos los otros controles de validación y en ella se debe ingresar el id del control a validar.
- La propiedad **Display** también es común para todos los controles y permite definir la forma de visualización del mensaje de error del control; si su valor es **None**, el mensaje de error no se visualizará, pero si su valor es **Dynamic**, el mensaje de error solo aparecerá si la validación determina que debe aparecer, es decir, cuando el dato es válido el mensaje de error no ocupa espacio, y si es **Static**, el mensaje de error siempre ocupa espacio.
- La propiedad **ErrorMessage** contiene el mensaje de error a mostrar.
- La propiedad **EnableClientSideScript**, por defecto configurada en **True**, permite que la validación se realice del lado del navegador/cliente.

Controles de validación

- **RequiredFieldValidator**

Se encarga de verificar que algún control de la página tenga por lo menos un valor diferente a nulo o vacío. Los otros controles de validación no contienen la funcionalidad de verificar si el control a validar se encuentra vacío.

El control posee una propiedad llamada **InitialValue**, la cual permite configurar un valor inicial para el control que se va a validar y verificar que el valor ha cambiado.

```
<asp:RequiredFieldValidator id="RequiredFieldValidator1" runat="server" InitialValue="0"
ControlToValidate="nombrecontrol" ErrorMessage="Este es el mensaje de error"
Display="Dynamic"> </asp:RequiredFieldValidator>
```

- **CompareValidator**

Ejecuta una validación utilizando un operador de comparación, además de poder verificar el tipo de dato.

La propiedad **ValueToCompare** puede tener como valor una constante, la cual es utilizada para realizar la comparación.

La propiedad **Operator**, define al operador de comparación que puede ser: **Equal**, **NotEqual**, **GreaterThan**, **GreaterThanEqual**, **LessThan**, **LessThanEqual**, o **DataTypeCheck**.

La propiedad **ControlToCompare**, permite definir otro control contra el cual se realizará la comparación. Esta propiedad tiene prioridad sobre la propiedad **ValueToCompare**.

```
<asp:CompareValidator id="CompareValidator1" runat="server" Type="String"
ControlToValidate="nombrecontrol1" ControlToCompare="nombrecontrol2"
Operator="Equal" ErrorMessage="Mensaje de Error"> </asp:CompareValidator>
```

- **RangeValidator**

Permite verificar si el valor de cierto control se encuentra dentro de un rango específico. Para realizar esta comparación, deben definirse valores a las propiedades **MinimumValue**, **MaximumValue** y **Type**.

Se debe tener especial cuidado en configurar la propiedad **Type**, pues su valor por defecto es **String**. En el caso se desee comparar números, si la propiedad **Type** no es cambiada a **Integer** o **Double**, los valores serán tratados como **Strings**, produciendo posibles errores.

```
<asp:RangeValidator id="RangeValidator1" runat="server" Type="Integer"
ControlToValidate="nombrecontrol" MinimumValue="0" MaximumValue="100">
</asp:RangeValidator>
```

- **RegularExpressionValidator**

Permite verificar que el valor de un control, tiene la forma, estructura y el tipo de una expresión regular. Una expresión regular es una excelente forma de identificar una simple o compleja secuencia de caracteres, sin la necesidad de escribir código para poder realizar dicha validación.

La propiedad **ValidationExpression** contiene la expresión regular a validar. Esta expresión regular puede ser construida por uno mismo, o se puede reutilizar una ya definida por otras personas.

Como recursos de esta última forma se tiene:

- <http://www.regexlib.com/>
- <http://www.regular-expressions.info/>

```
<html>
<body>
<form id="frmDatos" method="post" runat="server">
Valor:<asp:TextBox id="txtDigitos" runat="server"></asp:TextBox>
<br>
<asp:RegularExpressionValidator id="revDigitos" runat="server"
ControlToValidate="txtDigitos" ErrorMessage="Debe ingresar dos dígitos con
formato X-Y" Display="Dynamic" ValidationExpression="[0-9]-[0-9]">
</asp:RegularExpressionValidator>
<br>
<asp:Button ID="btnEnviar" runat="server" Text="Enviar datos"></asp:Button>
</form>
</body>
</html>
```

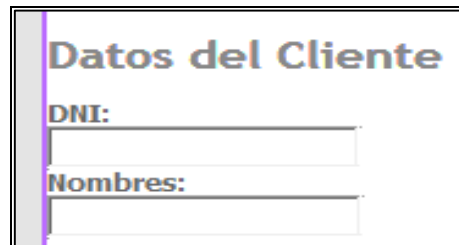
- **CustomValidator**

Permite realizar una validación personalizada, en base a un código implementado por uno mismo. Esta validación puede ser del lado del cliente, utilizando **JavaScript** o una validación del lado del servidor utilizando código **.Net**.

- **Validación del lado del cliente:** el parámetro **source**, contiene una referencia al control de validación, y el parámetro **arguments** es un objeto que contiene una propiedad llamada **Value**, que contiene la data a ser validada y una propiedad **IsValid**, la cual puede cambiar a **False** si la data es inválida o a **True** si la data es válida. Se debe utilizar la propiedad **ClientFunctionName** para definir la función JavaScript de validación.

- **Validación del lado del servidor:** para ello, se utiliza el evento **ServerValidate** en donde se define el algoritmo de validación correspondiente.

Por ejemplo, para validar el DNI de una persona ingresado desde una página:

A screenshot of a web form titled "Datos del Cliente". It contains two input fields: one labeled "DNI:" and another labeled "Nombres:". The form has a simple border and a light background.

Validación en el lado del cliente:

```
function ValidaDNI(source, arguments)
{
    //Obtiene valor del DNI ingresado
    var Valor = document.all.txtDNI.value;
    // Si no es numérico (NaN = Not a Number) en base decimal
    if (isNaN(parseInt(Valor, 10)))
    {
        arguments.IsValid = false;
        return;
    }
    arguments.IsValid = true; //La validación fue exitosa
}
```

Validación en el lado del servidor (C#):

```
private void ControlName_EventName(System.Object source,
ServerValidateEventArgs args)
{
    // Obtiene valor del DNI ingresado
    int Valor = 0;
    args.IsValid = int.TryParse(this.txtDNI.value, Valor);
}
```

6. Elementos de una aplicación ASP.NET

Elementos de una aplicación ASP.NET

- Carpeta Bin
- Archivo Global.asax
- Archivo de configuración Web.config

Una aplicación Web desarrollada en ASP .NET está compuesta por los siguientes elementos:

- Carpeta BIN
- Archivo de configuración Global.asax
- Archivo de configuración Web.config

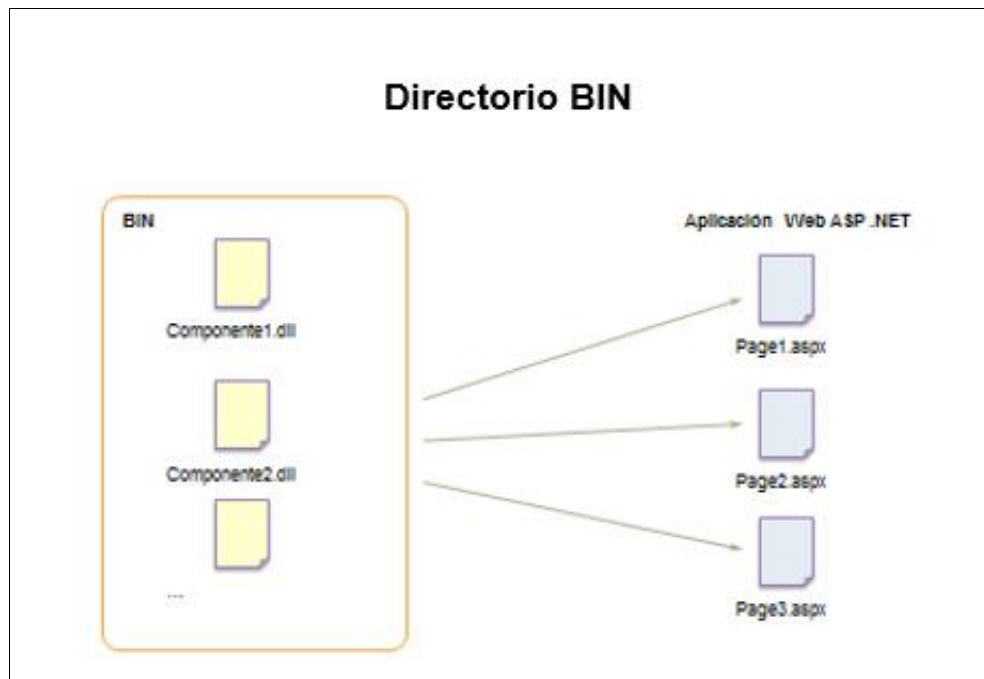
La carpeta BIN sirve para almacenar todos los ensamblados (archivos .DLL) que son necesarios para que la aplicación Web se ejecute correctamente. Esta carpeta es generada y cumple el mismo propósito en todos los tipos de proyecto que se crea con Visual Studio. Esta carpeta, a su vez, contiene 2 carpetas Debug y Release cada una de las cuales contiene los ensamblados y recursos correspondientes a la configuración de desarrollo y de publicación, respectivamente.

El archivo de configuración **Global.asax** es uno de los puntos de partida de las aplicaciones Web de ASP.NET, podemos configurar en él, eventos o acciones que queremos que se desempeñen a nivel de toda la aplicación.

El archivo de configuración **Web.config** es parte de la aplicación Web. El contenido de este archivo está en formato XML y puede configurar aspectos como:

- Conexiones de base de datos
- Manejo de errores
- Seguridad

7. Directorio BIN



El directorio BIN es el lugar donde se guardan los ensamblados compilados (archivos .DLL) de controles, componentes u otro código, al cual se desea hacer referencia dentro de la aplicación Web. Cualquier clase que se haya codificado y guardado en la carpeta BIN es referenciada automáticamente en la aplicación y se encuentra disponible para todas las páginas.

Los ensamblados en la carpeta BIN no necesitan ser registrados. La presencia de una DLL dentro de la carpeta BIN, es suficiente para que sea reconocida por ASP .NET. Si se escribe una nueva versión de una DLL y se coloca en la carpeta BIN, ASP.NET detecta la actualización y utiliza la nueva versión del archivo DLL.

8. Archivo Global.asax

Archivo Global.asax

El archivo **Global.asax**, también conocido como **archivo de aplicación ASP .NET**, es un archivo opcional que contiene código para responder a los eventos disparados a nivel de aplicación y sesión. Se ubica en el directorio raíz de la aplicación ASP.NET.

El archivo **Global.asax**, también conocido como **archivo de aplicación ASP.NET**, es un archivo opcional que contiene código para responder a los eventos disparados a nivel de aplicación y sesión. Se ubica en el directorio raíz de la aplicación ASP.NET.

En tiempo de ejecución, el archivo Global.asax es analizado sintácticamente y compilado en una clase derivada de la clase **HttpApplication**.

Los manejadores de eventos presentes, por defecto, dentro del archivo Global.asax son:

- Application_Start
- Application_End
- Application_Error
- Session_Start
- Session_End