Curso de Programación Web .NET

Unidad 3

“Web: ASP.NET”

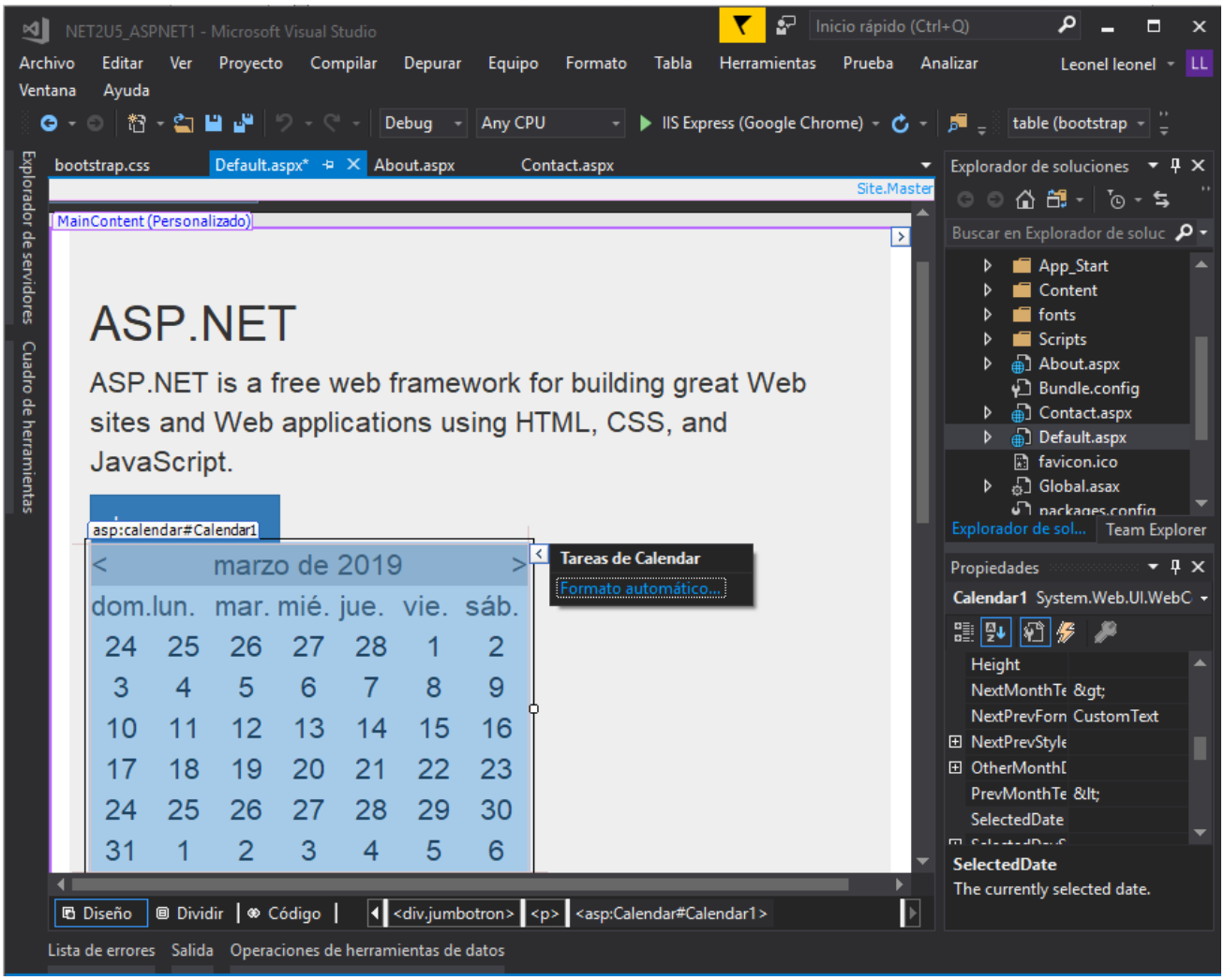
Contacto: consultas@elearning-total.com
Web: www.elearning-total.com

Índice

ASP.NET

Introducción

ASP.NET es una plataforma de desarrollo web de Microsoft que nos permite por medio de sus componentes y su arquitectura integral desarrollar sistemas web que funcionan tanto para PC como para dispositivos móviles. ASP.NET es integral porque nos permite desarrollar la vista o la página web así como también la lógica de backend de la misma y distribuirla, todo desde la misma plataforma.



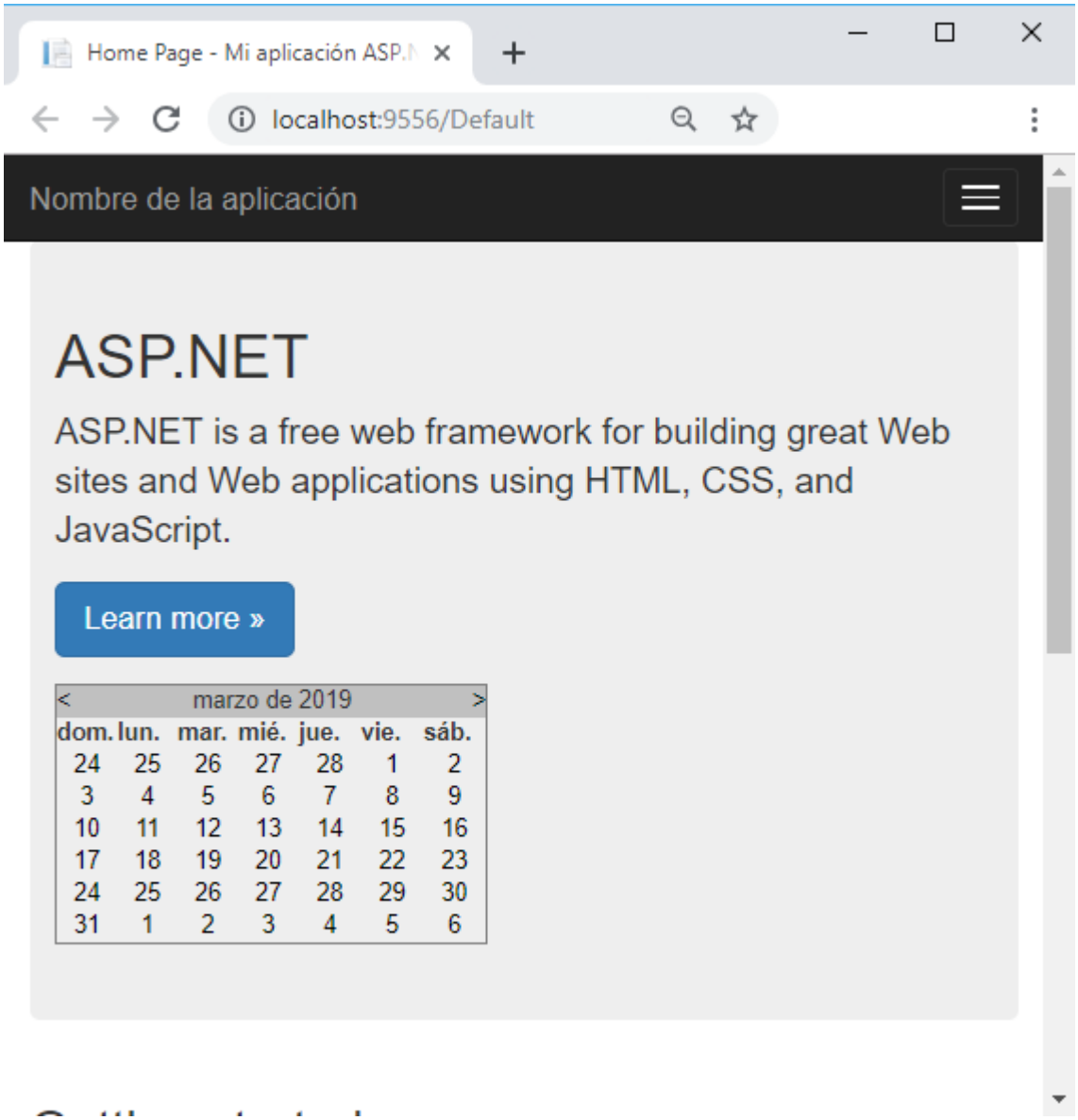
ASP.NET es una plataforma de desarrollo web, que proporciona un modelo de programación, una infraestructura de software integral y diversos servicios necesarios para desarrollar aplicaciones web robustas para PC, así como dispositivos móviles.

La misma funciona sobre el protocolo HTTP y utiliza los comandos y políticas HTTP para establecer una comunicación y cooperación bilateral entre el navegador y el servidor.

ASP.NET es parte de la plataforma Microsoft .Net. Las aplicaciones ASP.NET son códigos compilados, escritos usando los componentes u objetos extensibles y reutilizables presentes en el Framework .Net. Estos códigos pueden usar toda la jerarquía de clases en el marco .Net.

Los códigos de la aplicación ASP.NET se pueden escribir en cualquiera de los siguientes idiomas:

- C#
- Jscript
- Visual Basic .Net
- J#


Home Page - Mi aplicación ASP.
localhost:9556/Default
Nombre de la aplicación
ASP.NET
ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS, and JavaScript.
Learn more »
< marzo de 2019 >
dom. lun. mar. mié. jue. vie. sáb.
24 25 26 27 28 1 2
3 4 5 6 7 8 9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31 1 2 3 4 5 6

ASP.NET se usa para realizar aplicaciones web dinámicas basadas en datos las cuales se puede distribuir en internet o intranet. Cuenta con una gran cantidad de controles: cuadros de texto, etiquetas, botones entre otros con el fin de ensamblar, configurar y manipular código para crear páginas HTML.

Modelo de formularios web ASP.NET

El navegador envía un formulario web al servidor web y el mismo devuelve un XML o una página HTML como respuestas. Los formularios web de ASP.NET justamente facilitan todo este proceso de interacción por medio de los eventos.

Todas las actividades del usuario del lado del cliente se reenvían al servidor para su procesamiento con estado. El servidor procesa la salida de las acciones del cliente y desencadena las reacciones.

ASP.NET Framework ayuda a almacenar el estado de la aplicación, es decir si yo compre un producto y el mismo está en el carrito de compras quiero persistir esa información si cierro y vuelvo a abrir el navegador. ASP.NET mejora el protocolo HTTP ya que le mismo por definición no tiene estado.

Estos estados que maneja ASP.NET son:

Estado de la página y Estado de la sesión

El estado de la página es el estado del cliente, es decir, el contenido de varios campos de entrada en el formulario web. El estado de la sesión es la información colectiva obtenida de varias páginas que el usuario visitó y trabajó, es decir, el estado general de la sesión. Para aclarar el concepto volvamos al concepto del carrito de compras:

El usuario agrega artículos a un carrito de compras. Los artículos se seleccionan desde una página, por ejemplo la página de artículos, luego el total de artículos recolectados y el precio se muestran en una página diferente. Por si solo HTTP no puede realizar un seguimiento de toda la información proveniente de varias páginas. El estado de la sesión de ASP.NET y la infraestructura del lado del servidor realizan un seguimiento de la información recopilada globalmente durante una sesión.

El runtime de ASP.NET lleva el estado de la página hacia y desde el servidor a través de las solicitudes de la página mientras genera códigos de tiempo de ejecución de ASP.NET, e incorpora el estado de los componentes del lado del servidor en campos ocultos.

De esta manera, el servidor se da cuenta del estado general de la aplicación y opera de manera conectada en dos vías.

El modelo de componente ASP.NET

El modelo de componente ASP.NET proporciona varios bloques de construcción de páginas ASP.NET. Básicamente es un modelo de objeto, que describe:

- Los controles del servidor, que ayudan en el desarrollo de una interfaz de usuario compleja. Por ejemplo, el control `Gridview`.
- Las contrapartes del lado del servidor de casi todos los elementos o etiquetas HTML, como `<input>` y `<form>`.

Una aplicación ASP.NET está hecha de páginas, y cuando un usuario solicita una página sucede lo siguiente:

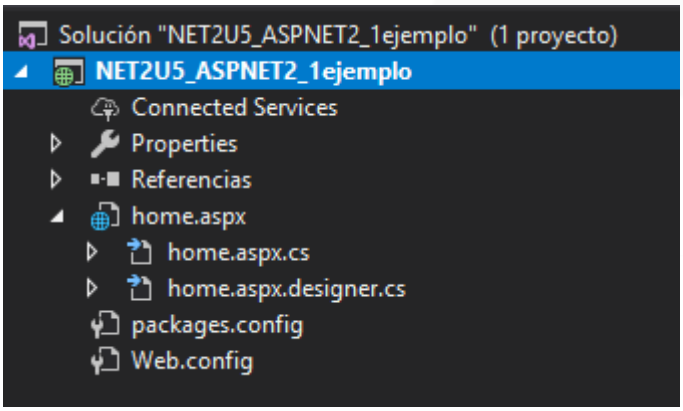
- IIS (El servidor de páginas web) recibe y valida la solicitud luego la delega al runtime de ASP.NET
- El runtime ASP.NET transforma la pagina ASP.NET (.aspx) en una instancia de clase que hereda del framework .NET

Cada página ASP.NET es un objeto y todos sus componentes, es decir, los controles del lado del servidor también son objetos.

El Entorno Visual Studio y ASP.NET

Proyectos y Soluciones

Una aplicación típica de ASP.NET consta de muchos elementos: los archivos de contenido web (.aspx), los archivos de origen (archivos .cs), los conjuntos (archivos .dll y .exe), los archivos de origen de datos (archivos .mdb), las referencias, los iconos , controles de usuario y varios otros archivos y carpetas. Todos estos archivos que conforman el sitio web están contenidos en una Solución.



Mínimamente un proyecto contiene los siguientes archivos de contenido:

- Archivo de página (.aspx)
- Página maestra (.master)

- Archivo de configuración del sitio web (.config)

También puede contener archivos de tipo **html** o de código **javascript** u hojas de estilos **css,**

WebForms

Si venimos del mundo del desarrollo de aplicaciones escritorio o **WinForms** nos vamos a familiarizar muy pronto con su análogo de la Web: **WebForms.**

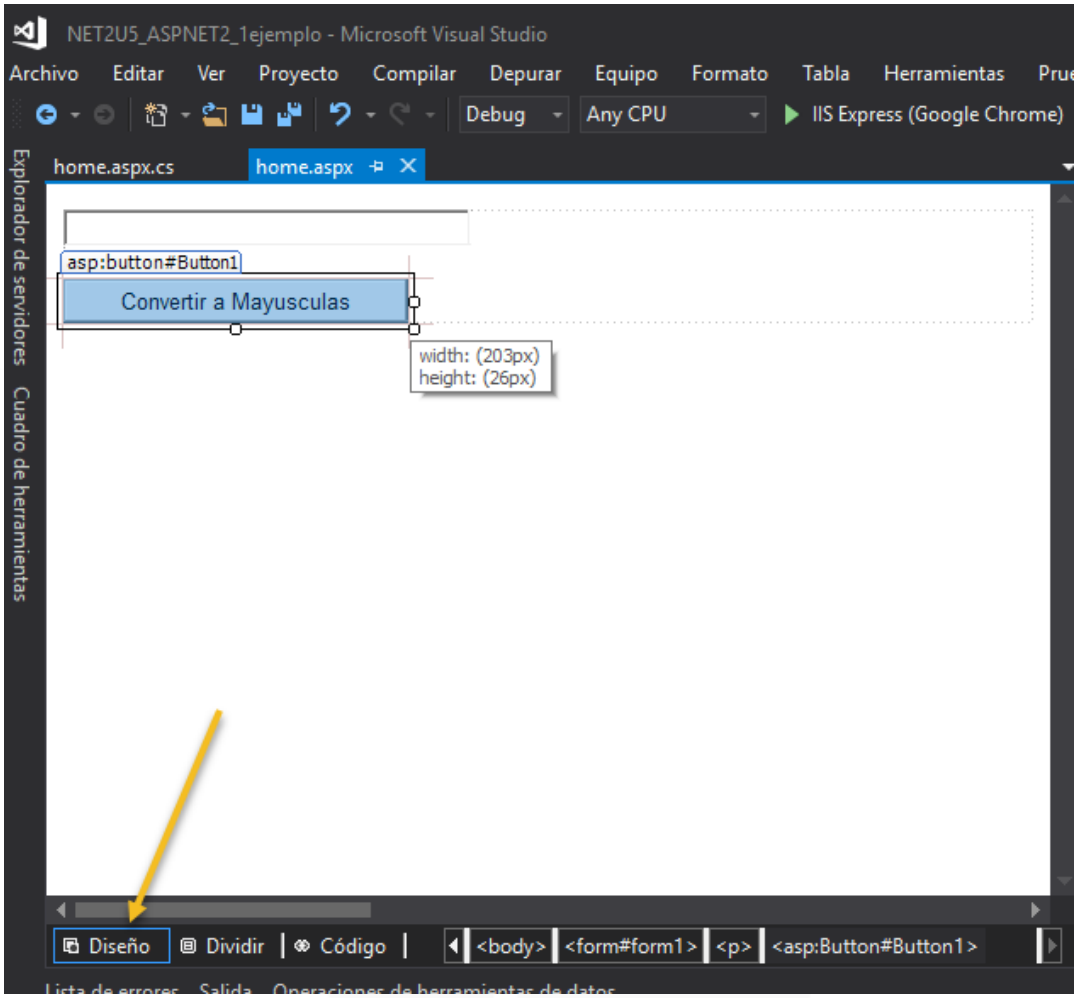
Básicamente el concepto es el mismo, podemos arrastrar los controles en el formulario e ir posicionandolos visualmente para ir armando la estructura de nuestra página web. Pero como es una página web también podemos editar su html y esa es la principal diferencia con el WinForms.

Visual Studio nos brinda 3 herramientas para editar las páginas web forms:

Diseño: Editor visual

El editor visual me permite arrastrar y colocar de manera visual los distintos controles web: texto, botones, menús etc.

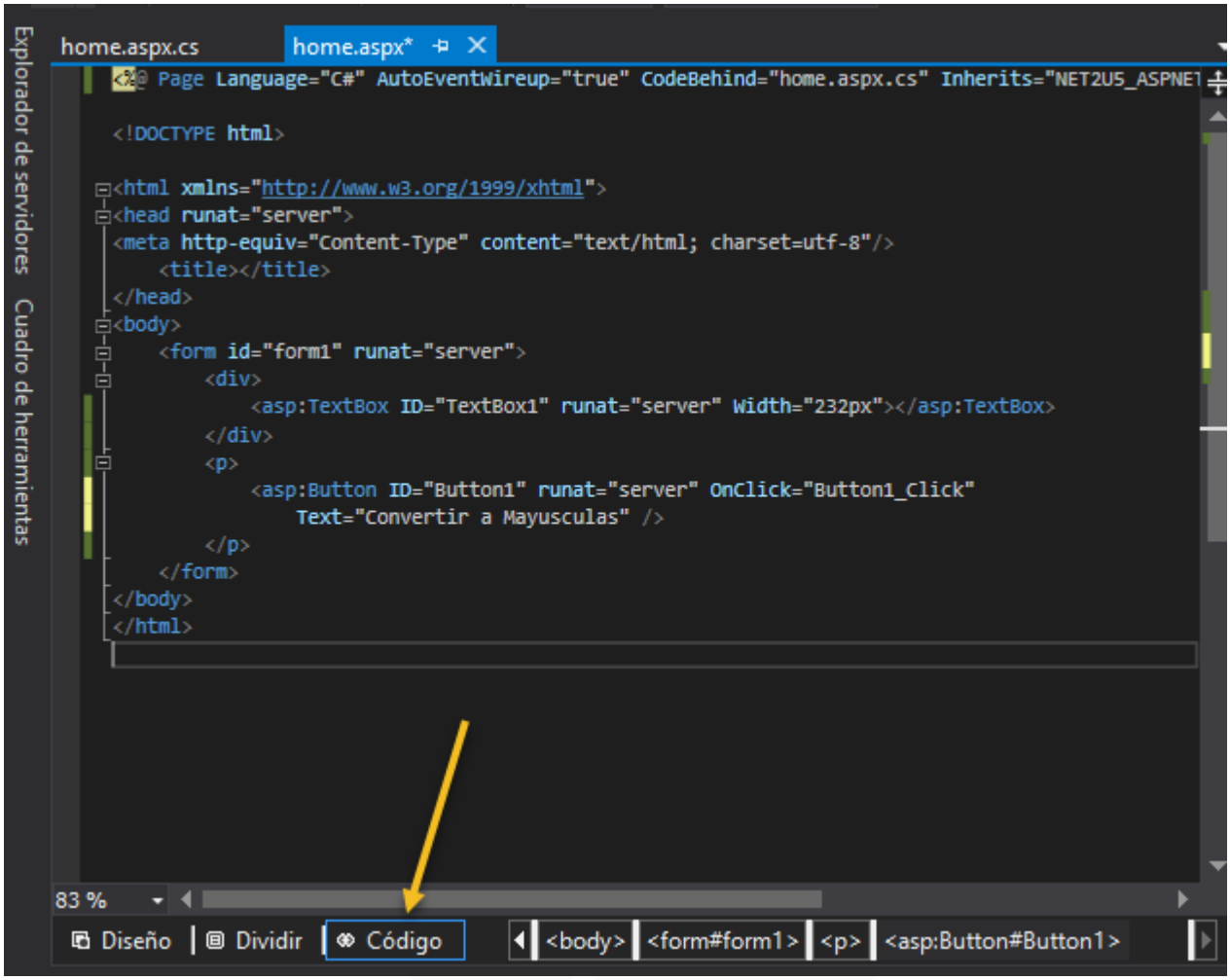
Al hacer doble click en cualquier elemento me permite ir al código C# o VB asociado para trabajar con los eventos, igual que sucede con Windows Forms en aplicaciones escritorio.



Unicamente con el diseño de form no vamos a poder realizar páginas web modernas y completamente dinámicas, en cuyo caso necesitamos poder editar el código HTML y esto también está disponible en Visual Studio, tenemos la posibilidad de editar el código HTML de nuestra web.

Diseño: Editor HTML

El editor HTML como su nombre lo indica me permite modificar la estructura de la página web por medio de su código html.



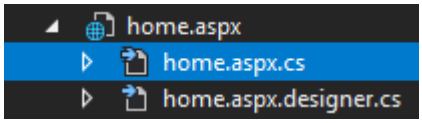
Además de las etiquetas **html** clásicas tengo las etiquetas **asp** que me permiten hacer referencia a los elementos propios de ASP.NET desde el html, como en el ejemplo que puedo modificar las características del botón.

Como todo código html puedo aplicarle estilos o agregar funcionalidad javascript por medio de este editor.

Algo a notar es que en la parte superior de la página se ve un código que no es HTML pero que es parte de una directiva especial de ASP.NET que agrega comportamiento y me permita que este simple HTML hable también el lenguaje de ASP.NET ya que en el html clásico no tenemos estas etiquetas especiales o tampoco podemos referenciar funciones de C# como en el caso de `Button1_Click` o mejor dijo code behind.

Editor de Code behind

Si abrimos nuestro archivo WebForm vamos a ver que tenemos varios archivos:



home.aspx contiene el diseño visual y el diseño html que ya hemos visto, **home.aspx.cs** por su parte contiene el código .NET y es donde va a ocurrir la mayor parte del procesamiento y la lógica de nuestra aplicación.

```
1   using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Web;
5   using System.Web.UI;
6   using System.Web.UI.WebControls;
7
8   namespace NET2U5_ASPNET2_1ejemplo
9   {
10      public partial class WebForm1 : System.Web.UI.Page
11      {
12          protected void Page_Load(object sender, EventArgs e)
13          {
14
15          }
16
17          protected void Button1_Click(object sender, EventArgs e)
18          {
19              TextBox1.Text = TextBox1.Text.ToUpper();
20          }
21      }
22  }
```

Observen que el el editor de code behind es muy similar a trabajar con un formulario WinForm. Puedo utilizar eventos como en el caso del botón, por lo tanto también puedo modificar el estado de mis formularios para generar una salida al usuario en mi página web.

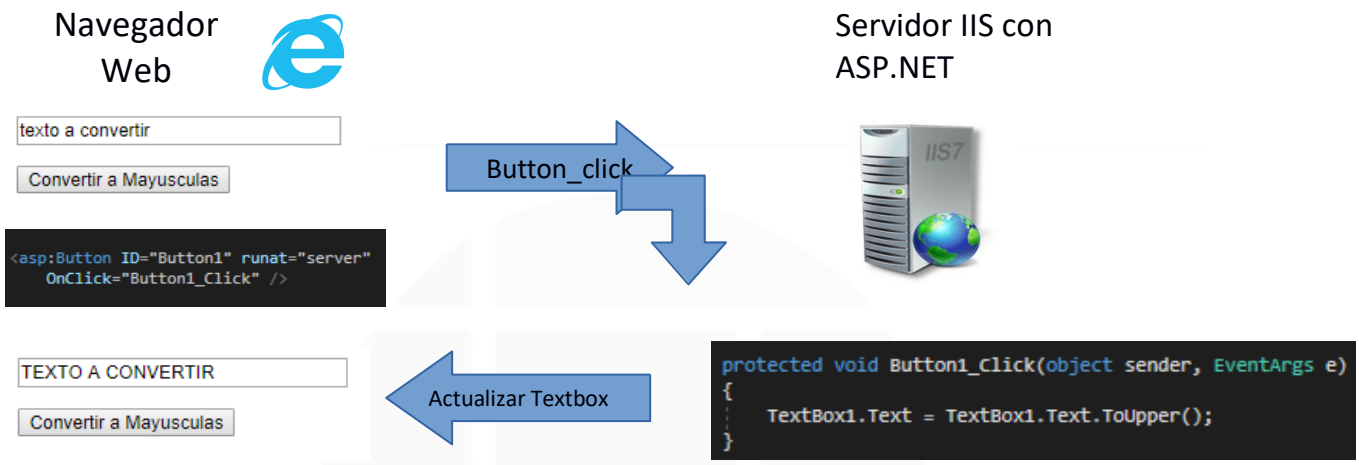
Básicamente se pueden aplicar los mismos principios que utilizo para crear aplicaciones de tipo escritorio.

Eventos en ASP.NET

Un evento es una acción como el click del mouse, o presionar una tecla o cualquier otra notificación generada por el mismo sistema.

Los eventos en ASP.NET se levantan desde la máquina del usuario (del explorador específicamente) y son enviados al servidor (IIS). Por ejemplo, Un usuario hace click en un botón desde una página web. En ese

momento un evento Click es enviado al servidor. El servidor recibe este evento generado por el lado del cliente y busca la rutina correspondiente para atenderlo:



En el ejemplo de arriba vemos como el modelo Cliente Servidor se aplica perfectamente en el caso de los eventos por medio del protocolo HTTP. En la izquierda una página ASP.NET por medio de este protocolo le avisa al servidor que quiere convertir “texto a convertir” en mayúsculas, la misma sabe esto por medio del código html generado en Visual Studio al configurar un evento Click (OnClick=**Button1_Click**).

El servidor en la derecha entiende que al recibir Button1_Click debe buscar un función con ese nombre, si la encuentra la ejecuta y en este caso esto significa actualizar un componente (**Texbox1.Text = TextBox1.Text.ToUpper();**) por lo tanto además de convertir el texto a mayúsculas debe actualizar la página web del usuario para que aparezca el nuevo valor en la caja de texto. Todo este proceso de enviar datos de controles a la pagina web por medio de un pedido del cliente se denomina **PostBack** el cual vamos a ver más a fondo en la próxima unidad de ASP.NET.

Todo este proceso se logra por medio del a comunicación cliente servidor que existe entre el explorador Web y el servidor IIS, cuyo lenguaje es el HTTP.

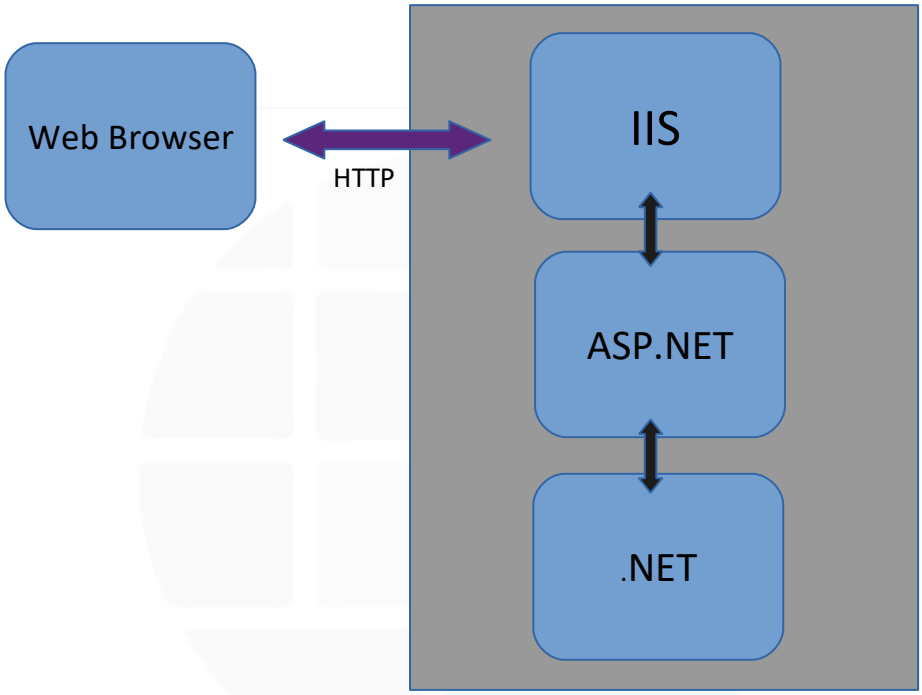
Compilando y ejecutando ASP.NET

En los proyectos de aplicaciones de escritorio al compilar se genera un ensamblado de tipo ejecutable (exe) en el cual al hacer doble click sobre él se ejecuta gracias a la ayuda del runtime del Framework .NET. Por otra parte los proyectos de tipo Web ASP.NET presentan algunas diferencias con entre proceso.

Este tipo de proyectos no generan archivos ejecutables (exe) en cambio generan archivos de tipo dll que se distribuyen junto a los archivos aspx.

Obviamente para ejecutar con este tipo de archivos no puedo hacer doble click ya que son compilados para ejecutarse con un Servidor Web con el componente ASP.NET más el runtime .NET y finalmente son distribuidos al usuario por medio del protocolo HTTP hacia el usuario final en su navegador web.

En el siguiente gráfico vemos un diagrama simplificado de la comunicación del usuario por medio del navegador web hacia un servidor Web IIS con ASP.NET, el cuadrado gris representa un servidor.



A efectos prácticos la principal diferencia es que vamos a necesitar montar un servidor web para distribuir la aplicación. Esto simplifica mucho la distribución ya que simplemente necesitamos copiar los archivos aspx y dll en este servidor para que luego cada nuevo usuario que quiera utilizar la aplicación puede acceder a ella por medio de la dirección de nuestro servidor web que hemos montado.

Para debug y pruebas Visual Studio viene con un **IIS express** que funciona de manera local y se ejecuta directamente cuando hacemos click en el botón Debug o Ejecutar en nuestro IDE, algo muy conveniente.

Controles Básicos

Introducción

Una de las ventajas de ASP.NET es su orientación a controles. Estos controles como botones, campos de texto, imágenes entre otros nos van a ayudar a construir nuestra página Web de manera más rápida comparada con el clásico modelo html y javascript.

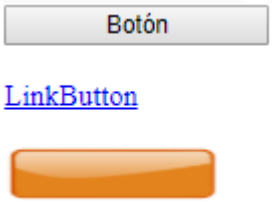
¿Qué sucede cuando agregamos un control?

Cuando arrastramos un control en nuestro form además de estar creando visualmente el elemento, estamos estructurando el código html y javascript que luego ASP.NET va a generar para que el cliente en su explorador web lo utilice, además del lado del servidor estamos definiendo su lógica por medio del "code behind" para su procesamiento en el IIS.

Todo esto es una ventaja que ASP.NET nos brinda como buen Framework: tener ensamblados para acelerar nuestra productividad además de requerir muy poco conocimiento de html para crear una Web estándar.

Control Button

El botón es uno de los controles fundamentales en lo que es Web, se utiliza generalmente cuando queremos enviar formularios de información (hacer un post) por lo tanto no es de extrañar que generalmente se emplea junto a otros controles.



Sintaxis:

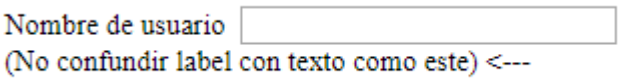
```
<asp:Button ID="MiBoton" runat="server" onclick="Button1_Click" Text="Boton" / >
```

Algunas Propiedades destacables:

- **Text:** Contiene el texto a mostrar en el botón
- **ImageURL:** La dirección de la imagen del ImageButton
- **AlternateText:** Si por algún motivo el botón no puede mostrarse en la página web se mostrará este texto alternativo. Esto es muy importante ya que las personas que necesitan de accesibilidad especial usaran este campo alternativo para navegar por la web.

Controles TextBox y Label

Estos controles se usan para pedir input al usuario y devolver información al mismo.



Sintaxis:

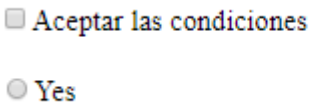
```
<asp:TextBox ID="mitextbox" runat="server"></asp:TextBox>
```

Algunas Propiedades destacables:

- **Text:** Contiene el texto del textbox. Muy utilizado para editar y trabajar su contenido.
- **TextMode:** si configuramos este campo como “SingleLine” obtenemos un textbox de una línea en cambio si lo hacemos como “Multiline” podemos tener un campo multilinea, por ultimo “password” nos permite esconder los caracteres con asteriscos (***) para ingresar passwords
- **MaxLength:** La cantidad máxima de caracteres a ingresar.

Controles CheckBox y RadioButton

Ambos controles sirven para especificar mejor que tipo de acción queremos que nuestro usuario tome en cuanto a decisiones de tipo verdadero o falso, si o no.



Sintaxis de un CheckBox

```
<asp:CheckBox ID="chkcondiciones" runat="Server" Text="Aceptar las condiciones"> </asp:CheckBox>
```

Sintaxis de un RadioButton

```
<asp:RadioButton ID="radioYes" runat= "Server" Text="Yes"></asp:RadioButton>
```

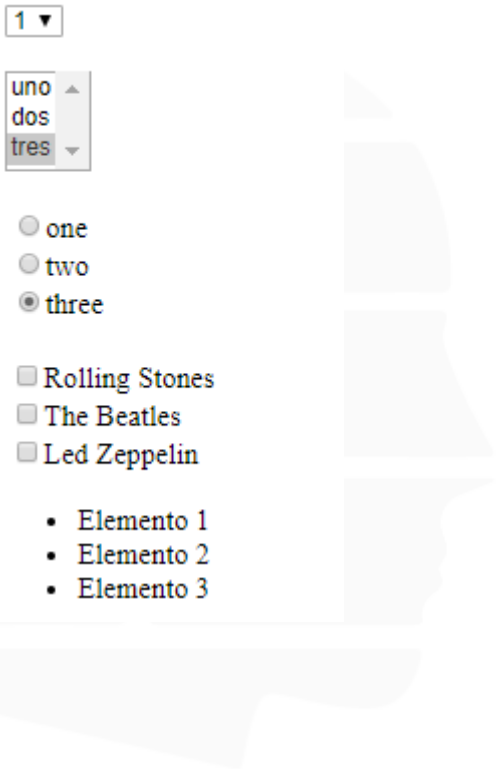
Algunas Propiedades destacables:

- **Text:** el texto a mostrar en el control.

- **Checked:** indica si el campo fue seleccionado (true) o no (false).
- **GroupName:** Sirve para agruparlos bajo un mismo nombre.

Controles Listas: DropDownList, ListBox, CheckBoxList, BulletedList

Todos estos controles nos ayudan a mostrar colecciones o lista de elementos a nuestros usuarios, algunos de estos controles admiten la interacción con el usuario directamente y otros son meramente informaciones como la Bulleted list. Estas listas son estáticas (se cargan los ítems de manera visual) o dinámicas (se pueden llenar por medio de código),



sintaxis de un listbox

```
<asp:ListBox ID="ListBox1" runat="server" AutoPostBack="True"
OnSelectedIndexChanged="ListBox1_SelectedIndexChanged"></asp:ListBox>
```

Fíjense que se puede interactuar con un listbox por medio de un evento de cambio de selección, “ListBox1_SelectedIndexChange” va a ser una función de code behind que va a operar sobre esta acción.

Algunas Propiedades de estos controles:

- **Items:** esta es una lista de tipo ListItemCollection la cual puedo operar en .NET como cualquier otra colección. (cada ítem contiene otras propiedades: Text, Selected y Value)

- **SelectedIndex y Selected Value:** la primera contiene el índice numérico del objeto seleccionado y el segundo su valor.

Control HyperLink (hipervínculo)

Este control no es más que una máscara del elemento html <a> utilizado para generar hipervínculos en las páginas web.

Link a Google

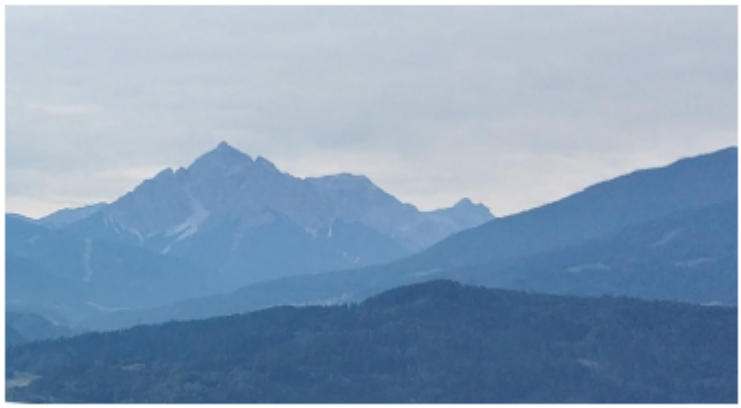
```
<asp:HyperLink ID="HyperLink1" runat="server">Link a Google</asp:HyperLink>
```

Algunas Propiedades destacables:

- **ImageURL:** Si queremos utilizar una imagen como hipervínculo pondremos su dirección aquí.
- **NavigationURL: URL** Destino al hacer click en el mismo.
- **Text:** El texto que va a mostrar el hipervínculo
- **Target: Acá** podemos especificar si el hipervínculo abrirá la dirección en la misma ventana o en una emergente (tipo popup).

Control Image

Permite mostrar imágenes en la página web.



```
<asp:Image ID="Imagen" runat="server">
```

Algunas Propiedades destacables:

- **ImageURL: Dirección de la imagen que queremos mostrar.**
- **AlternateText**: Si por algún motivo la imagen no puede mostrarse en la página web se mostrará este texto alternativo. Esto es muy importante ya que las personas que necesitan de accesibilidad especial usaran este campo alternativo para navegar por la web.

Paneles

El control panel se usa como un contenedor de otros controles en la página. El mismo permite manejar la visibilidad de los controles que el mismo contiene. También permite generar controles por medio de código.

```
<asp:Panel ID="Panel1" runat = "server">ACA VAN LOS OTROS CONTROLES</asp:Panel>
```

Algunas Propiedades destacables:

- **HorizontalAlign:** Permite definir como se alinean los elementos en su interior (izquierda, derecha o centro)
- **ScrollBars**: Permite mostrar, ocultar o configurar las barras de desplazamiento del panel.

Los panales son clave para maquetar páginas web en ASP.NET ya que luego de que se renderiza toda la página los paneles se transforman en elementos `<div>`.