



ASP.NET

En Visual Studio 2019



	UNIDAD Nº 1: Sitios web con Visual Studio 2019.	TEMAS: Controles visuales.	Clase 0
--	--	---	------------------------------

Objetivos:

- Pasos para crear un sitio web básico.
- WebForm.
- Controles visuales y sus eventos.

Introducción

ASP.Net es la última tecnología de Microsoft para la construcción de aplicaciones y servicios basados en la web, sucesor de las Active Server Pages (ASP).

Para la programación de páginas dinámicas se pueden emplear: Visual Basic .Net, C#, Jscript .Net y cualquier otro lenguaje desarrollado para la .Net

¿Por qué y cuándo deberíamos utilizar ASP.NET en nuevos proyectos?

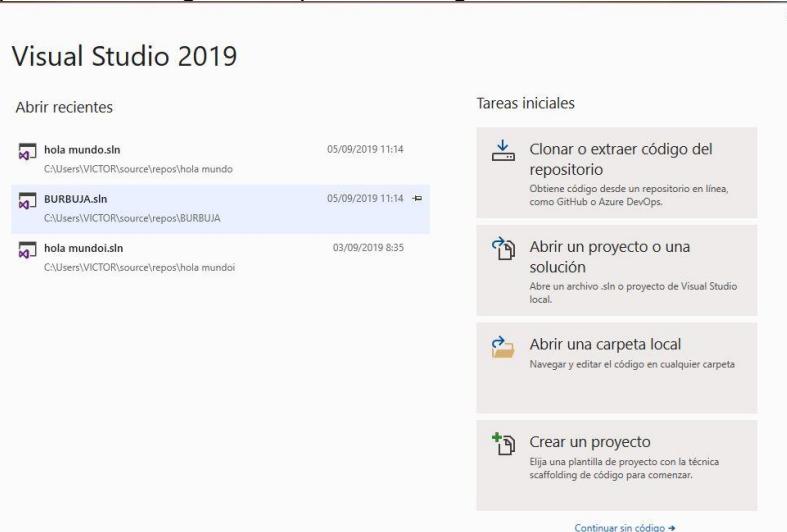
- Para reducir el tiempo de desarrollo.
- Incrementar la performance.
- Incrementar la estabilidad de la aplicación.
- Incrementar la escalabilidad.

Los costos asociados con el desarrollo de aplicaciones con ASP.NET:

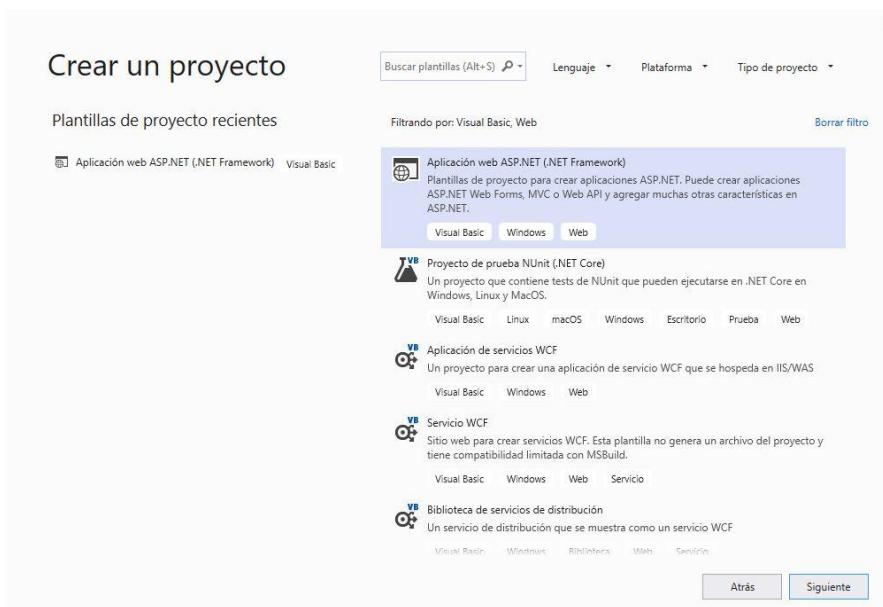
- Sistema operativo (Windows 7,10, Windows XP professional Edition, Windows Server)
- Entorno de desarrollo de Visual Studio .Net

Pasos para crear un sitio web con ASP.Net

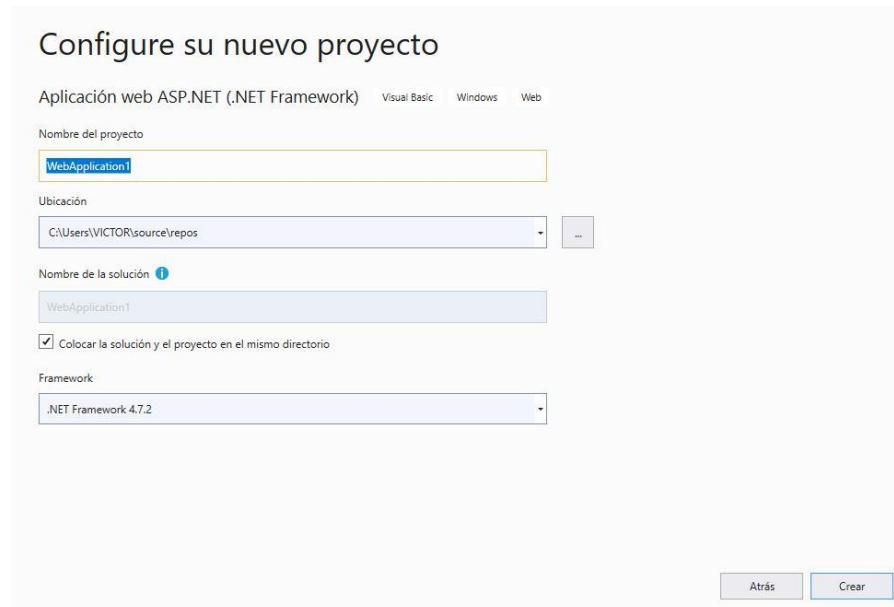
Para crear un proyecto web debemos seleccionar desde inicio el entorno del Visual Studio 2019. Inmediatamente aparece un diálogo donde podemos elegir la tarea inicial



Elegimos Crear un proyecto Aplicación Web ASP.NET Framework



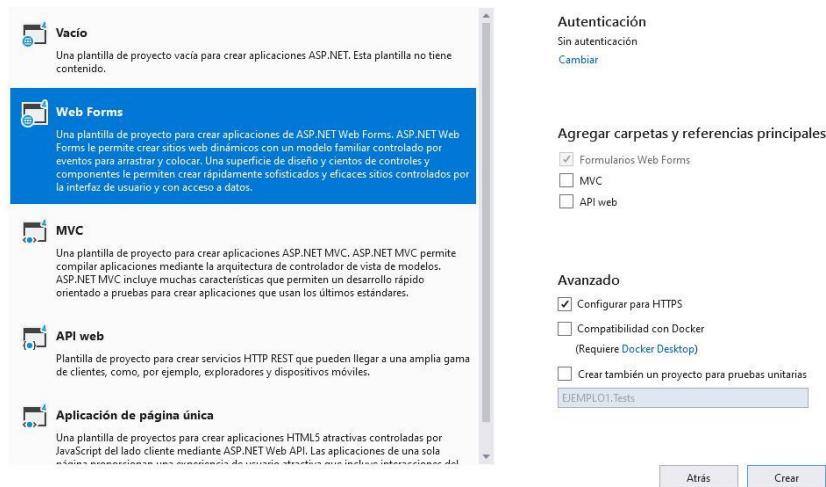
En la siguiente pantalla especificamos el nombre del proyecto y nos permitirá configurar entre otras cosas, que versión de Framework será compatible nuestra aplicación, la ubicación dentro del disco se almacenará, el lenguaje de programación a utilizar:
Cambio el nombre de proyecto a EJEMPLO11



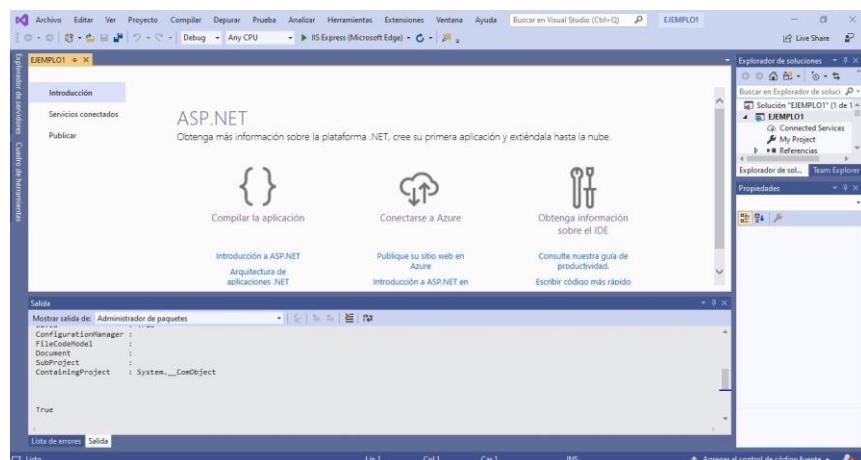
En este ejemplo C:\users\victor\source\repos\ejemplo11
(en esta última carpeta es donde se almacenará nuestra aplicación web)
Colocar la solución y el proyecto en el mismo directorio

Hemos dejado seleccionado por defecto el framework a utilizar (.NET framework 4.7.2.)

Crear una aplicación web ASP.NET

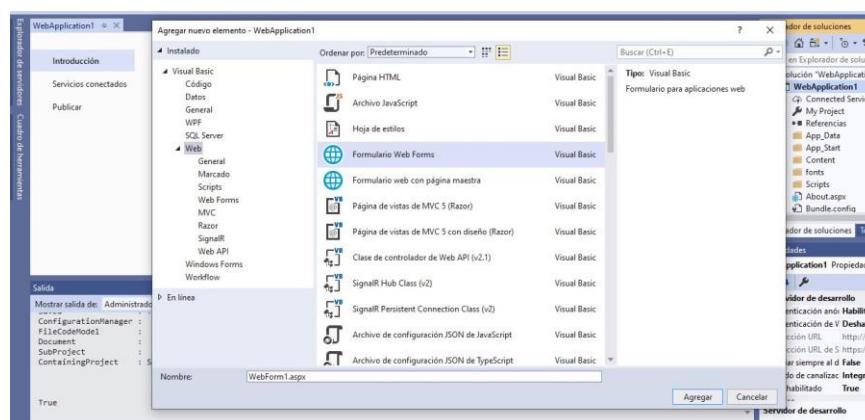


Elegimos Web Forms y a continuación Crear.
Nos crea el proyecto y tras unos segundos
Temos el esqueleto básico para iniciar nuestro sitio web empleando la tecnología de Microsoft.



Como hemos usado una plantilla deberemos de borrar en el explorador de soluciones todos los ficheros con extensión ASPX que encontramos.

En la Parte derecha que es donde tenemos el Explorador de Soluciones
desde debajo de EJEMPLO11 Agregar, Nuevo elemento Formulario Web Forms



Le Cambiaremos el Nombre y le llamaremos DEFAULT.ASPX.

En la parte izquierda tenemos el “Cuadro de herramientas”, en esta aparecen las componentes visuales (Label, TextBox, Button etc.) que tenemos disponibles para crear nuestro formulario Web.

En el centro aparece la página en vista de código (se puede ver en “vista de diseño”, “Divisor” y “Código”)

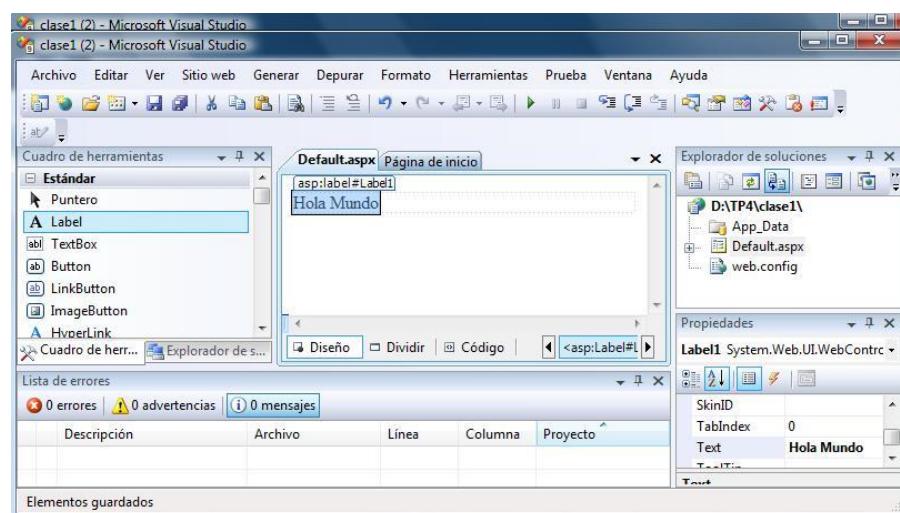
En la parte derecha disponemos del “Explorador de soluciones” donde podemos identificar el directorio donde se almacena nuestra aplicación web y los archivos contenidos en dicho directorio. Siempre que creamos una aplicación web nos crea un formulario web inicial y lo almacena en el archivo Default.aspx (la extensión aspx indica que se trata de una página dinámica ASP.Net, así como la extensión php indica que su contenido está programado en PHP)

Además del archivo Default.aspx se crea otro archivo llamada Default.aspx.vb (este archivo contiene la codificación en Visual Basic de los eventos que definamos a los controles del formulario)

Otro archivo que veremos más adelante y que se crea en forma automática es el web.config.

Por último se crea una carpeta llamada App_Data.

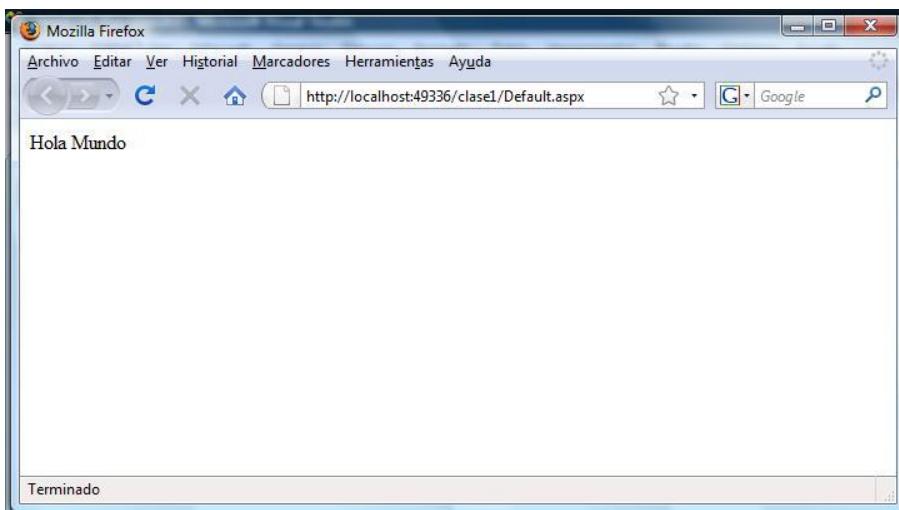
Para nuestro primer ejemplo implementaremos el “Hola Mundo” para ello en el recuadro central seleccionamos la pestaña “Diseño” y desde el cuadro de herramientas arrastramos un control de tipo Label. Seguidamente desde el cuadro de propiedades ubicado en la parte inferior derecha de la pantalla inicializamos la propiedad text con el mensaje que queremos que muestre nuestra “Label”, en nuestro caso “Hola Mundo”. Una vez modificada la propiedad Text con el mensaje que queremos mostrar y presionada la tecla Enter podemos ver como se actualiza la ventana de Diseño en la parte central de nuestra pantalla:



Para probar nuestra pequeña aplicación desarrollada debemos presionar el triángulo verde que se encuentra en la barra de botones, o desde el menú de opciones: Depurar->Iniciar depuración, o presionar la tecla “F5”.

Inmediatamente nos aparece un diálogo que nos invita a modificar el archivo Web.config para que la página se pueda ejecutar en modo depuración (esto nos permite disponer puntos de interrupción o ejecutar paso a paso una aplicación) Seleccionamos el botón “Aceptar” para activar el modo depuración.

Podemos ver que inmediatamente aparece el navegador configurado por defecto con el resultado de la ejecución de la página:



El Visual Studio 2019 instala un servidor web propio que está escuchando en el puerto 44372.

Luego de cerrar la ventana del navegador debemos detener la depuración de nuestra aplicación para poder modificarla, para esto podemos seleccionar desde el menú Depurar -> Detener Depuración o presionar desde la barra de botones el cuadradito azul (luego de esto aparece el “Cuadro de herramientas”)

Eventos

Modificaremos ahora nuestra aplicación para que muestre la fecha del servidor en una Label. Cuando desde el navegador solicitamos una página aspx lo primero que se ejecuta es el evento Page_Load. Para poder definir un método para dicho evento hacemos doble clic sobre el WebForm con lo que inmediatamente se abre el archivo Default.aspx.vb y genera dicho método:

```
Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As _
        System.EventArgs) Handles Me.Load

        End Sub
    End Class
```

Luego codificamos dentro del método Page_Load el algoritmo que muestra la fecha actual del servidor:

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As _
    System.EventArgs) Handles Me.Load
    Me.Label1.Text = Date.Now.Day & "/" & Date.Now.Month & _
        "/" & Date.Now.Year
    End Sub
```

Mediante el objeto Date y accediendo a la propiedad Now obtenemos el día, mes y año disponible en el servidor.

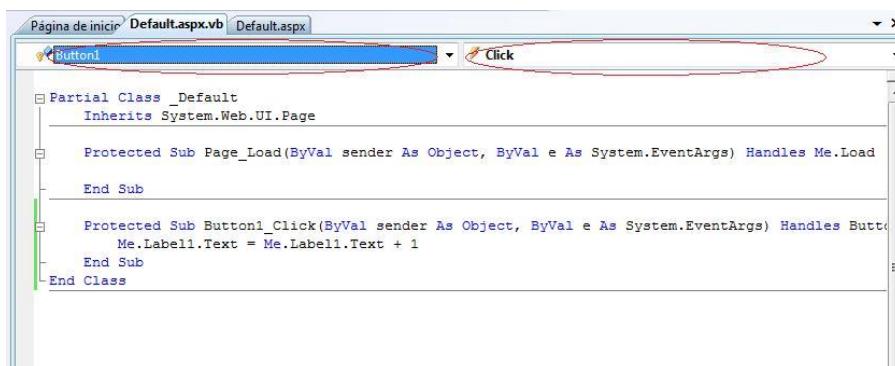
Captura del evento click de un objeto de la clase Button.

Ahora nuevamente modificaremos nuestra pequeña aplicación para que muestre un objeto de una clase Button y una Label.

La propiedad Text de la Label la inicializamos con el valor “0” y la propiedad Text del objeto Button lo inicializamos con la cadena “Sumar”.

El objetivo es que cada vez que se presione el botón se actualice el contenido de la Label con el valor actual más uno.

La forma más sencilla de generar dicho evento es hacer doble clic sobre el objeto Button. Otra forma es seleccionar dicho objeto de la lista que aparece en la parte superior del editor y al lado derecho según el objeto seleccionado nos aparecen todos los métodos disponibles para dicho objeto:



La palabra reservada Me en Visual Basic .NET se usa para hacerse referencia a los propios elementos de una clase como pueden ser los atributos o nuestros método o funciones.

El símil en Java sería la palabra reservada this.

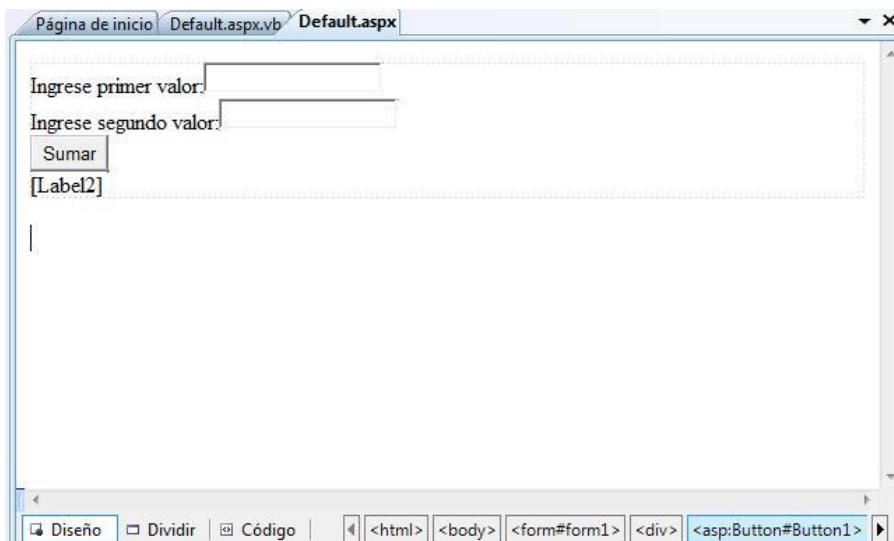
Luego para el evento Button1_Click actualizamos el contenido de la propiedad Text de la Label con el valor actual más uno. El operador que utilizamos es el + y no el & (que nos concatenaría el valor y no lo sumaría numéricamente como necesitamos en este problema) Podemos utilizar directamente el operador + ya que el segundo operando es un número y por lo tanto el Visual Basic convierte automáticamente el primer operando a entero.

Controles Label, Button y TextBox.

Hasta ahora hemos utilizado los controles de tipo Label y Button, ahora utilizaremos el control TextBox. Crearemos una aplicación que nos permita ingresar dos números y luego en una label muestre la suma de los mismos.

(Si tenemos un proyecto abierto podemos cerrarlo seleccionando la opción: Archivo->Cerrar proyecto y luego seguir los mismos pasos que vimos anteriormente para crear un nuevo proyecto ASP.NET)

Crearemos un proyecto llamado pruebatextbox y desarrollaremos la siguiente interface:



Disponemos tres Label, dos TextBox y un Button. Las dos primeras Label las inicializamos con los textos: "Ingrese primer valor", "Ingrese segundo valor" y la tercera Label borramos todo el contenido de la propiedad Text (como podemos ver el visor del Visual Studio muestra el nombre del objeto encerrado entre corchetes cuando la propiedad Text está vacía)

Inicializamos la propiedad Text del objeto de tipo Button con la etiqueta "Sumar".

Luego codificamos el evento click del objeto de la clase Button (en este evento debemos extraer el contenido de los dos controles de tipo TextBox y proceder a convertirlos a tipo de dato entero y sumarlos):

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal _  
    e As System.EventArgs) Handles Button1.Click  
    Dim s As Integer  
    s = Integer.Parse(Me.TextBox1.Text) + _  
        Integer.Parse(Me.TextBox2.Text)  
    Me.Label2.Text = "La suma de los dos valores es :" & s  
End Sub
```

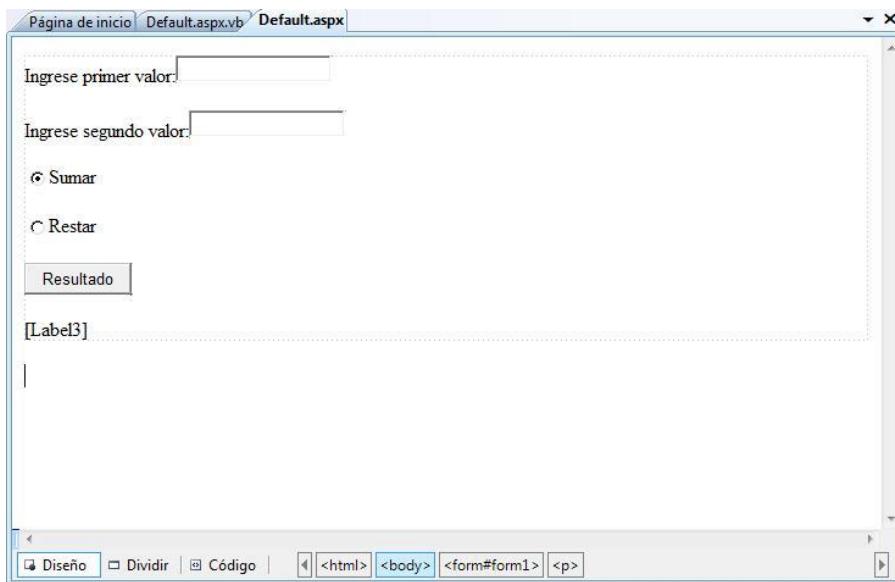
La clase Integer tiene un método estático llamado Parse y que tiene por objetivo recibir un String y retornar el valor del mismo convertido a entero.

Luego de sumar mostramos en la tercer label el resultado de la suma de los dos valores ingresados.

Control RadioButton.

Para probar el funcionamiento del control RadioButton crearemos un nuevo sitio web llamado pruebaradiobutton.

Crearemos una interface similar al problema anterior, con la salvedad que le agregaremos dos controles de tipo RadioButton para poder indicar si queremos sumar o restar los valores ingresados:



Como vemos agregamos dos controles de tipo RadioButton, inicializamos las propiedades text con los textos “Sumar” y “Restar”. Luego para indicar que los controles RadioButton están en el mismo grupo debemos inicializar la propiedad GroupName con el mismo valor (con esto logramos que al seleccionar un RadioButton el otro se desmarca), si nos olvidamos inicializar la propiedad GroupName luego los dos controles de tipo RadioButton se podrán seleccionar en forma simultánea.

Si queremos que alguno de los dos RadioButton aparezca seleccionado por defecto debemos inicializar la propiedad Checked con el valor True.

La codificación del evento click del objeto Button1 es el siguiente:

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim x1 As Integer = Me.TextBox1.Text
    Dim x2 As Integer = Me.TextBox2.Text
    Dim resultado As Integer
    If Me.RadioButton1.Checked Then
        resultado = x1 + x2
        Me.Label3.Text = "La suma de los dos valores es:" & resultado
    Else
        If Me.RadioButton2.Checked Then
            resultado = x1 - x2
            Me.Label3.Text = "La diferencia de los dos valores es:" & resultado
        End If
    End If
End Sub
```

Cuando se presiona el botón se ejecuta el método Button1_Click donde primero extraemos el contenido de los dos controles TextBox.

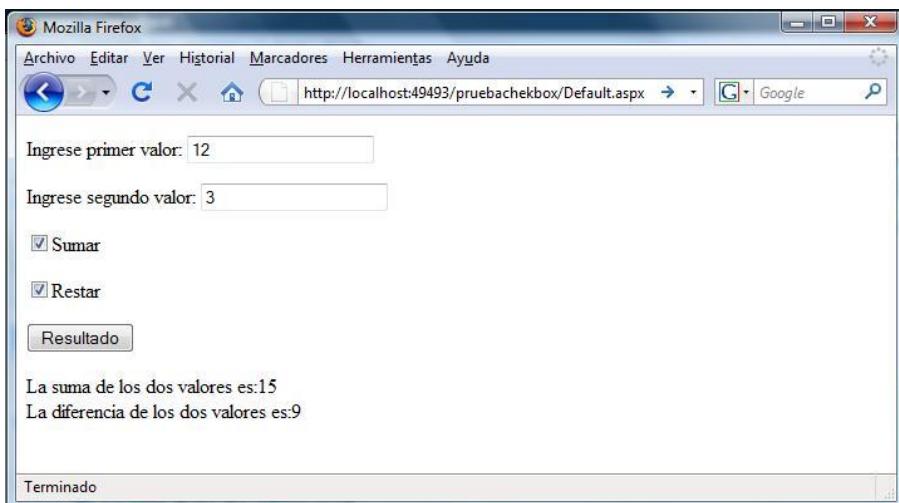
Verificamos con if cual de los dos controles RadioButton se encuentra seleccionado. La propiedad Checked del RadioButton indica si está seleccionado el control o no.

Control CheckBox.

Los controles CheckBox permiten que más de uno esté seleccionado. Similar a los controles RadioButton tiene dos estados (seleccionado o no seleccionado) y esto lo sabemos según el estado de la propiedad Checked.

Codificaremos un nuevo sitio web que permita cargar dos valores y luego calcule la suma y/o resta de los valores ingresados. Como podemos seleccionar ambas operaciones utilizaremos los controles de tipo CheckBox.

La interface visual es la siguiente:



La codificación del evento click del botón es:

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Button1.Click
    Dim x1 As Integer = Me.TextBox1.Text
    Dim x2 As Integer = Me.TextBox2.Text
    Dim resultado As Integer
    Me.Label3.Text = ""
    If Me.CheckBox1.Checked Then
        resultado = x1 + x2
        Me.Label3.Text = "La suma de los dos valores es:" _
            & resultado
    End If
    If Me.CheckBox2.Checked Then
        resultado = x1 - x2
        Me.Label3.Text = Me.Label3.Text & "<br />La" & _
            "diferencia de los dos valores es:" & resultado
    End If
End Sub
```

Disponemos dos if a la misma altura ya que ambos CheckBox pueden estar seleccionados. Previo a los if borramos el contenido de la Label en el caso que tenga el resultado de operaciones anteriores. Luego en el primer if verificamos si el primer CheckBox está seleccionado y procedemos a inicializar la propiedad Text de la Label con el resultado de la suma de los dos valores ingresados, seguidamente verificamos con un segundo if si el siguiente CheckBox está seleccionado, en caso afirmativo agregamos al contenido actual de la Label el resultado de la diferencia de los valores ingresados (Como vemos podemos añadir marcas HTML a la propiedad Text de una Label, luego estas serán interpretadas por el navegador)

Control ListBox.

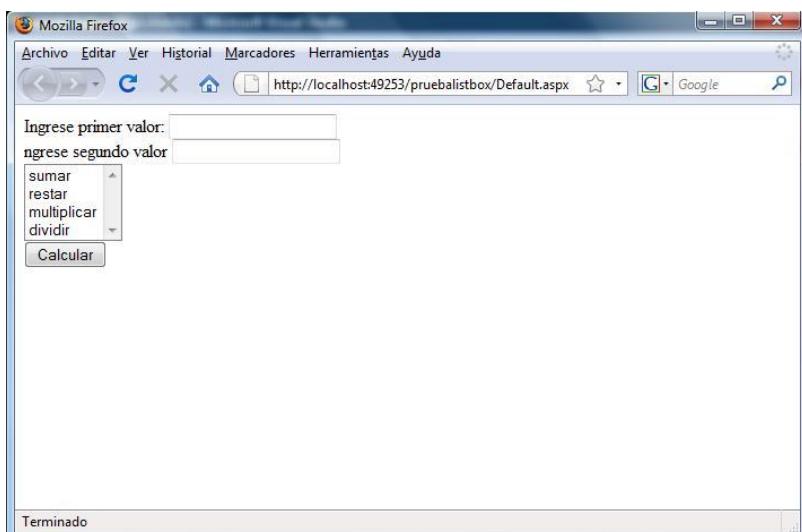
El control ListBox permite crear una lista de valores.

La propiedad Item permite definir los miembros de la lista (cada item define las propiedades Text (valor a mostrar), Value (valor a retornar en caso de estar seleccionado), Selected (con un valor lógico))

Otra propiedad muy importante del control ListBox es SelectionMode, esta admite dos valores: Single o Multiple.

Crearemos una aplicación que permita cargar dos valores y mediante un control ListBox poder seleccionar si queremos sumar, restar, multiplicar o dividir dichos valores (como podemos seleccionar varias operaciones en forma simultánea configuraremos la propiedad SelectionMode del ListBox con el valor Multiple)

Luego la interface visual a crear es la siguiente (insertamos también una Label luego del botón "Calcular", con el objetivo de mostrar los resultados):



Cuando se presiona el botón calcular verificamos cual de las opciones está seleccionada y procedemos a calcular y mostrar los resultados.

```
Protected Sub Button1_Click(ByVal sender As Object, _
    ByVal e As System.EventArgs) Handles Button1.Click
    Dim x1 As Integer = Me.TextBox1.Text
    Dim x2 As Integer = Me.TextBox2.Text
    Me.Label3.Text = ""
    If Me.ListBox1.Items(0).Selected Then
        Dim suma As Integer = x1 + x2
        Me.Label3.Text &= "La suma es:" & _
            suma & "<br />"
    End If
    If Me.ListBox1.Items(1).Selected Then
        Dim resta As Integer = x1 - x2
        Me.Label3.Text &= "La diferencia:" & _
            resta & "<br />"
    End If
    If Me.ListBox1.Items(2).Selected Then
        Dim multi As Integer = x1 * x2
        Me.Label3.Text &= "El producto:" & _
            multi & "<br />"
    End If
    If Me.ListBox1.Items(3).Selected Then
```

```

        Dim divi As Integer = x1 / x2
        Me.Label3.Text &= "La division:" & _
        divi & "<br />"
    End If
End Sub

```

Como podemos ver primero vaciamos el contenido de la Label3 y procedemos mediante cuatro if a verificar cuales de los elementos del ListBox se encuentran seleccionados:

```
If Me.ListBox1.Items(0).Selected Then
```

Si por ejemplo el primer elemento del ListBox se encuentra seleccionado procedemos a sumar los dos valores almacenados en los TextBox y los agregamos a la Label:

```

        Dim suma As Integer = x1 + x2
        Me.Label3.Text &= "La suma es:" & _
        suma & "<br />"
```

Control DropDownList.

El control DropDownList permite crear una lista de valores y luego seleccionar solo uno de ellos, esta es la diferencia fundamental con el control ListBox.

Para probar este control implementaremos el problema propuesto con el control ListBox, ahora la interface es la siguiente:



Cargamos las 4 operaciones básicas en DropDownList y para el evento clic del botón tenemos que codificar:

```

Dim x1 As Integer = Me.TextBox1.Text
Dim x2 As Integer = Me.TextBox2.Text
If Me.DropDownList1.Items(0).Selected Then
    Dim suma As Integer = x1 + x2
    Me.Label3.Text = "La suma es:" & _
    suma & "<br />"
ElseIf Me.DropDownList1.Items(1).Selected Then
    Dim resta As Integer = x1 - x2
    Me.Label3.Text = "La diferencia:" & _
    resta & "<br />"
ElseIf Me.DropDownList1.Items(2).Selected Then
    Dim multi As Integer = x1 * x2
    Me.Label3.Text = "El producto:" & _
    multi & "<br />"
ElseIf Me.DropDownList1.Items(3).Selected Then
    Dim divi As Integer = x1 / x2
    Me.Label3.Text = "La division:" & _
    divi & "<br />"
End If

```

Como solo un elemento del control DropDownList puede estar seleccionado disponemos una serie de if/elseif para verificar cual de ellos es el seleccionado. Cuando identificamos el item seleccionado procedemos a efectuar el cálculo correspondiente y mostrarlo en la Label3.

Ejercicios Propuestos

1 – Confeccionar una página que solicite el ingreso del nombre y apellido de una persona (cada elemento en un TextBox), luego al presionar un botón mostrar en una label si alguno de los datos no se cargó.

2 – Confeccionar una página que muestre un examen múltiple choice (disponer 4 preguntas y tres respuestas posibles por pregunta) utilizar controles de tipo RadioButton para la selección de la respuesta correcta.

Mostrar la cantidad de respuestas correctas luego que se presiona un botón.

3 – Solicitar el ingreso de un número en un textbox. Verificar con la función IsNumeric si se trata de un número. En caso que se trate de un número mostrar la cantidad de dígitos que tiene.

4 – Disponer un conjunto de RadioButton agrupados. Mostrar en las leyendas de cada RadioButton distintos buscadores (Google, Bing, Yahoo etc.)

Cuando se presione un botón redireccionar a dicho servidor (para redireccionar debemos utilizar la siguiente sintaxis response.redirect(<http://www.google.com>)

5 – Confeccionar el siguiente formulario para registrarse en un sitio web (utilizar controles de la pestaña estandar)

En una Label mostrar los datos cargados en cada control (disponer la Label al final del formulario)

Hacer por lo menos 5 validaciones y mostrar mensajes de errores en una Label.

Nombre de usuario:

Clave:

Repita la clave:

Correo electrónico:

Apellido:

Nombre:

País de origen: Argentina

Provincia:

Código Postal:

Sexo: Hombre Mujer

Fecha de nacimiento
(dd/mm/aaaa):

Comentarios:

Acepto los términos y condiciones: Acepto

	UNIDAD Nº 1: Variables,operadores If y Bucles con Visual Studio 2019	TEMAS: Variables,operadores Instrucción If y bucles con Visual Studio	Clase 1
--	---	--	------------------------------

Objetivos:

- Funciones condicionales y bucles con Visual Studio
- Variables y Operadores

Introducción

VARIABLES

Una variable es una posición de memoria con un nombre que mantiene un valor que puede cambiar en el transcurso del programa

Las variables se tienen que declarar

```
Dim num1, num2 As Integer
```

Se les puede asignar un valor en el momento de su declaración

```
Dim startVal As Integer = 1
```

En .NET, existen los siguientes tipos de variables (en la tabla se muestran también los alias de cada tipo de dato en VB)

Tipo .NET	Alias VB	¿Con Signo?	Bytes	Rango
System.SByte	SByte	Sí	1	-128 a 127
System.Int16	Short	Sí	2	-32768 a 32767
System.Int32	Integer	Sí	4	-2147483648 a 2147483647
System.Int64	Long	Sí	8	-9223372036854775808 a 9223372036854775807
System.Byte	Byte	No	1	0 a 255
System.UInt16	UShort	No	2	0 a 65535
System.UInt32	UInteger	No	4	0 a 4294967295
System.UInt64	ULong	No	8	0 a 18446744073709551615
System.Single	Single	Sí	4	Aprox. $\pm 1.5 \times 10^{-45}$ a $\pm 3.4 \times 10^{38}$ con 7 decimales
System.Double	Double	Sí	8	Aprox. $\pm 5.0 \times 10^{-324}$ a $\pm 1.7 \times 10^{308}$ con 15 o 16 decimales
System.Decimal	Decimal	Sí	12	Aprox. $\pm 1.0 \times 10^{-28}$ a $\pm 7.9 \times 10^{28}$ con 28 o 29 decimales
System.Char	Char	N/A	2	Cualquier carácter Unicode
System.Boolean	Boolean	N/A	Depende de la plataforma	True o False

Ámbito de vida de las variables

Habitualmente las variables mantienen su contenido en el procedimiento en el que se declaran, se las denomina Locales. Tambien hay Variables que mantienen su valor en todas las paginas de una solución se las denomina Globales

Variables globales en un sitio ASP.NET

En el Web.config

Para hacer esto he visto varias maneras, la primera de todas se trata de guardar los datos en el Web.config en la zona de “appSettings”, esto lo haría de la siguiente forma:

Primero añadiendo una clave a la zona de appSettings en el Web.config

```
<appSettings>
    <add key="username" value="valor" />
</appSettings>
```

Y luego en el código de la página en la que lo necesitemos podemos obtener o modificar el valor de la clave mediante:

```
System.Configuration.ConfigurationSettings.AppSettings["username"]
```

En el Global.asax

Otra manera que he visto es utilizando el archivo Global.asax. Este archivo opcional en los desarrollos de ASP.NET nos permite manejar eventos que ocurren a nivel de la aplicación y de sesión, también nos permite declarar valores que necesitemos entre las diferentes solicitudes.

El archivo Global.asax para que funcione tiene que permanecer en la raíz de la aplicación y solo se admite uno por aplicación ejecutada. Si tenemos algún otro archivo de este tipo en algún subdirectorio no se tendrán en cuenta por la aplicación.

Cuando declaremos una variable dentro de Global.asax estará disponible en todas las páginas de la aplicación. Lo podemos hacer de la siguiente manera:

En el fichero Global.asax declaramos una variable string y luego en Session_Start otorgamos el valor de la cadena a un objeto de tipo Session que en mi caso se llama valorCadena.

```
public class Global : System.Web.HttpApplication
{
    string cadena="valor inicial";
    protected void Application_Start(object sender, EventArgs e) { }
    protected void Session_Start(object sender, EventArgs e)
    {
        Session["valorCadena"] = cadena;
    }
}
```

Luego en la página que lo necesitemos podemos escribir valores:

```
Session["valorCadena"] = "la_cadena_que_quiera_poner";
```

y leer de la variable:

```
Label1.Text = (string)Session["valorCadena"];
```

Los operadores

Los operadores son palabras claves del lenguaje que permiten la ejecución de operaciones sobre el contenido de ciertos elementos, en general variables, constantes, valores literales o resultados de funciones. La combinación de uno o varios operadores y elementos en los cuales los operadores

van a apoyarse se llama una expresión. Estas expresiones se valoran en el momento de su ejecución, en función de los operadores y valores que tienen asociados.

Los operadores se pueden repartir en 4 categorías.

1. Los operadores de asignación

El único operador disponible en esta categoría es el operador `=`. Permite asignar un valor a una variable. El mismo operador se utiliza sea cual sea el tipo de la variable (numérica, cadena de caracteres...).

2. Los operadores aritméticos

Los operadores aritméticos permiten efectuar cálculos en el contenido de las variables:

Operador	Operación realizada	Ejemplo	Resultado
<code>++</code>	Suma	<code>6+4</code>	10
<code>-</code>	Resta	<code>12-6</code>	6
<code>*</code>	Multiplicación	<code>3*4</code>	12
<code>/</code>	División	<code>25/3</code>	8.3333333333
<code>\</code>	División entera	<code>25/3</code>	8
Mod	Modulo (resto de la división entera)	<code>25 mod 3</code>	1
<code>^^</code>	Potencia	<code>5 ^ 3</code>	125

3. Los operadores binarios (Encadenadores de condiciones)

Estos operadores encadenan condiciones en las que se han de dar unas premisas para que se efectue un resultado. (Ver mas adelante en instrucción if).

Operador	Operación realizada	Ejemplo	Resultado
And	Y Binario	<code>A=1 b=1 if (A=1 and b=1)</code>	true
Or	O Binario	<code>A=1 b=1 if (A=1 or b=1)</code>	true
Not	Negación	<code>A=1 if a not isnumeric</code>	false

4. Los operadores lógicos

Devuelven verdadero o falso en base al contenido de las variables

Operador	Operación realizada	Ejemplo	Resultado
<code>></code>	Mayor que	<code>45 > 255</code>	false
<code><</code>	Menor que	<code>9 < 46</code>	true
<code><></code>	Distinto	<code>99 <> 46</code>	true
<code>=</code>	Igual	<code>25 =24</code>	false

La Instrucción if

La declaración condicional IF-THEN-ELSE está utilizando para verificar las condiciones que proporcionamos en el encabezado de la declaración if y para tomar una decisión basada en esa condición. La declaración condicional que examina los datos utilizando operadores de comparación y operadores lógicos.

```
If [Condicion] Then  
    Instrucciones  
Else  
    Instrucciones  
End If
```

Si la condición dada es verdadera, entonces el control va al cuerpo del bloque if, es decir, el programa ejecutará el código dentro del bloque if. Si la condición es falsa, el control pasa al siguiente

nivel, es decir, si proporciona el bloque else, el programa ejecutará el bloque de código de la instrucción else; de lo contrario, el control pasará a la siguiente línea de código.

La instrucción else es opcional, por lo que puede usar la instrucción if sin la instrucción else.

El siguiente programa ASP.NET declara e inicializa valores a dos números, num1 y num2 y luego comprueba cuál es mayor usando la instrucción if.

Default.aspx

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Button" />
            <br />
            <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
        </div>
    </form>
</body>
</html>
```

Default.aspx.vb

```
Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Button1.Click
        Dim num1, num2 As Integer
        num1 = 100
        num2 = 100

        If num1 > num2 Then
            Label1.Text = "Num1 es mayor que num2 "
        Else
            Label1.Text = "Num2 es mayor que num1 "
        End If
    End Sub
End Class
```

BUCLAS FOR - NEXT

Para las instrucciones bucles, que permiten que el código se ejecute repetidamente. En la programación ASP.NET, a veces tiene que enfrentarse a una situación para repetir una tarea varias veces o debe repetir una tarea hasta llegar a una condición, en estas situaciones puede usar declaraciones de bucle para lograr los resultados deseados.

En VB.NET, el bucle FOR NEXT, ejecuta el cuerpo del bucle (el código fuente dentro del bloque de código For ..Next) en un número fijo de veces.

```
For var=[startValue] To [endValue] [Step]
[loopBody]
Next [var]
```

- var: El contador para que el ciclo repita los pasos.
- startValue: el valor inicial asignado a la variable de contador.
- endValue: cuando la variable de contador alcanza el valor final, el bucle se detendrá.
- loopBody: el código fuente entre el cuerpo del bucle

El siguiente ejemplo de ASP.NET declara dos variables startVal y maxVal y luego asigna los valores 1 y 10 respectivamente y muestra el número 1 a 10 usando un bucle for .

Default.aspx

```

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <br />
            <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Button" />
            <br />
            <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
        </div>
    </form>
</body>
</html>

```

Default.aspx.vb

```

Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Button1.Click
        Dim startVal As Integer = 1
        Dim maxVal As Integer = 10
        Dim i As Integer

        For i = startVal To maxVal
            Label1.Text = i
        Next i
    End Sub
End Class

```

BUCLE FOR - EACH

El bucle for each ejecuta un bloque de código en cada elemento de una matriz o una colección de elementos. Al ejecutar for each loop, atraviesa elementos en una colección o matriz.

```

For Each [Item] In [Group]
    [loopBody]
Next [Item]

```

- Item : el Item en el grupo
- Group : el grupo que contiene items

El siguiente programa ASP.NET agrega una matriz de elementos a un cuadro de lista utilizando ... cada ciclo.

Default.aspx

```

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="Button" />
            <br />
            <asp:ListBox ID="ListBox1" runat="server"></asp:ListBox>
        </div>
    </form>
</body>
</html>

```

Default.aspx.vb

```

Partial Class _Default
    Inherits System.Web.UI.Page

```

```

Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Button1.Click
Dim days As String() = {"Lunes", "Martes", "Miercoles", "Jueves", "Vierbes", "Sabado", "Domingo"}
    ListBox1.Items.Clear()
    For Each day As String In days
        ListBox1.Items.Add(day)
    Next
End Sub
End Class

```

BUCLE WHILE (Mientras)

El bucle While ejecuta el cuerpo del código hasta que cumpla con la condición especificada.

Al igual que la instrucción if, la instrucción while evalúa la expresión, que debe devolver un valor booleano. Si la expresión se evalúa como verdadera, la instrucción while ejecuta las declaraciones en el bloque while. La instrucción while continúa probando la expresión y ejecutando su bloque hasta que la expresión se evalúe como falsa.

```

While [condition]
[loop body]
End While

```

El siguiente programa ASP.NET cuenta un número del 1 al 10 usando el bucle while.

Default.aspx

```

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
Text="Button" />
            <br />
            <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
        </div>
    </form>
</body>
</html>

```

Default.aspx.vb

```

Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Button1.Click
        Dim count As Integer = 10
        While (count <= 10)
            Label1.Text = "El valor de i es : " & count
            count = count + 1
        End While
    End Sub
End Class

```

Select Case

Select case le permite elegir entre muchas declaraciones basadas en múltiples selecciones pasando el control a una de las declaraciones de caso dentro de su cuerpo. La declaración de cambio puede incluir cualquier número de instancias de casos.

```

Select Case expression
    [Case expression list]
    [statements]
End Select

```

El siguiente programa ASP.NET evalúa qué valor pasa en la declaración de caso.

Default.aspx

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
Text="Button" />
            <br />
            <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
        </div>
    </form>
</body>
</html>
```

Default.aspx.vb

```
Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Button1.Click
        Dim val As Integer = 2

        Select Case val
            Case 1
                Label1.Text = " El valor es 1"
            Case 2
                Label1.Text = " El valor es 2"
            Case 3
                Label1.Text = " El valor es 3"
        End Select
    End Sub
End Class
```

ASP.NET Exceptions

Las excepciones son la aparición de algunas condiciones que cambian el flujo normal de ejecución. Se producen excepciones en situaciones como que su programa se quede sin memoria, el archivo no existe en la ruta dada, las conexiones de red se cortan, etc.

Todas las excepciones en Common Language Runtime se derivan de una sola clase base, también puede crear sus propias clases de excepción personalizadas.

Puede manejar Excepciones usando la instrucción Try..Catch..Finally. El código en el bloque finalmente se ejecutará incluso si no se produjeron Excepciones.

```
Try
    statements
Catch [Exception [As Type]]
    statements
Finally
    statements
```

El siguiente programa ASP.NET intenta dividir un número entre cero.

Default.aspx

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
    <title>Untitled Page</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Button ID="Button1" runat="server" onclick="Button1_Click"
Text="Button" />
            <br />
            <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label><br />
            <asp:Label ID="Label2" runat="server" Text="Label"></asp:Label><br />
            <asp:Label ID="Label3" runat="server" Text="Label"></asp:Label>
        </div>
    </form>
</body>
</html>
```

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

Default.aspx.vb

```
Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Button1.Click
        Try
            Dim i As Integer = 10
            Dim res As Integer
            res = i / 0
            Label1.Text = "resultado : " & res.ToString()
        Catch ex As Exception
            Label2.Text = "la excepcion vendria aqui !! "
        Finally
            Label3.Text = "ha entrado eb el bloque Finally !!!"
        End Try
    End Sub
End Class
```

	UNIDAD Nº 1: Archivos de texto.	TEMAS: Creación, lectura y escritura.	Clase 2
--	--	--	------------------------------

Objetivos:

- Creación de un archivo de texto..
- Agregar datos a un archivo de texto.
- Lectura de un archivo de texto..

Introducción

En muchas situaciones es necesario almacenar información en el servidor, tenemos dos alternativas, si la cantidad de datos a almacenar es pequeña podemos utilizar un archivo de texto para ello (en la próxima clase veremos como almacenar en una base de datos)

Un archivo de texto no requiere grandes recursos del servidor.

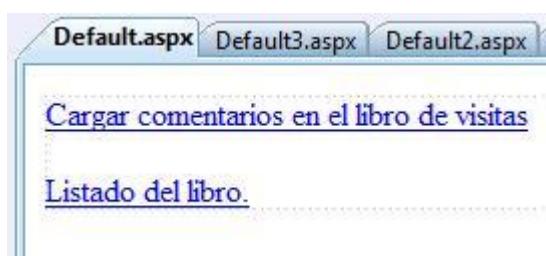
Creación, carga y lectura.

Creación y carga del archivo de texto.

Confeccionaremos un libro de visitas de un sitio web.

La página principal dispondremos de dos hipervínculos (HyperLink), el primero irá al formulario de carga y el segundo el listado del contenido del archivo.

La página Default.aspx:



El primer control HyperLink nos lleva al formulario web que permite cargar el nombre del visitante, su país y los comentarios.

El formulario web requiere tres objetos de la clase TextBox, al tercero donde se ingresan los comentarios debemos inicializar la propiedad TextMode con el valor MultiLine.

Disponemos un control de tipo HyperLink para poder retornar a la página principal.

Cuando se presiona el botón confirmar procedemos a almacenar los datos del formulario en el archivo de texto, si existe los agregamos al final, en caso que no exista se crea el archivo. Mostramos finalmente en una label que se almacenaron los datos.

El código completo para registrar los datos es:

```
Imports System.IO
Partial Class Default2
    Inherits System.Web.UI.Page
    'Hace que la clase o interfaz actual herede los atributos, variables, propiedades,
    'procedimientos y eventos de otra clase o conjunto de interfaces.
    Protected Sub Button1_Click(ByVal sender As Object,
        ByVal e As System.EventArgs) Handles Button1.Click
        Dim arch As New StreamWriter(Me.Server.MapPath(".") _
            & "/" & "visitas.txt", True)
        arch.WriteLine("Nombre:" & Me.TextBox1.Text)
        arch.WriteLine("<br />")
        arch.WriteLine("Pais:" & Me.TextBox2.Text)
        arch.WriteLine("<br />")
        arch.WriteLine("Comentarios<br />")
        arch.WriteLine(Me.TextBox3.Text)
        arch.WriteLine("<br />")
        arch.WriteLine("<hr>")
        arch.Close()
        Me.Label1.Text = "Datos Guardados"
    End Sub
End Class
```

Primero importamos el espacio de nombres donde está declarada la clase StreamWriter:

```
Imports System.IO
```

Cuando se presiona el botón confirmar creamos un objeto de la clase StreamWriter y le pasamos en el constructor el path y nombre del archivo a abrir o crear según sea el caso. Para obtener el directorio actual utilizamos el método MapPath del objeto Server:

```
Dim arch As New StreamWriter(Me.Server.MapPath(".") _
    & "/" & "visitas.txt", True)
```

El valor true que le pasamos en el constructor significa que si el archivo no existe en el servidor en la carpeta especificada se procede a su creación y en el caso que si existe se procede a su apertura y posicionado del puntero de archivo al final del mismo.

El método WriteLine de la clase StreamWriter permite almacenar una línea en el archivo de texto y el posterior ingreso en forma automática del salto de línea:

```
arch.WriteLine("Nombre:" & Me.TextBox1.Text)
arch.WriteLine("<br />")
arch.WriteLine("Pais:" & Me.TextBox2.Text)
arch.WriteLine("<br />")
arch.WriteLine("Comentarios<br />")
arch.WriteLine(Me.TextBox3.Text)
arch.WriteLine("<br />")
arch.WriteLine("<hr>")
```

Cuando finalizamos de trabajar con el archivo procedemos a cerrarlo:

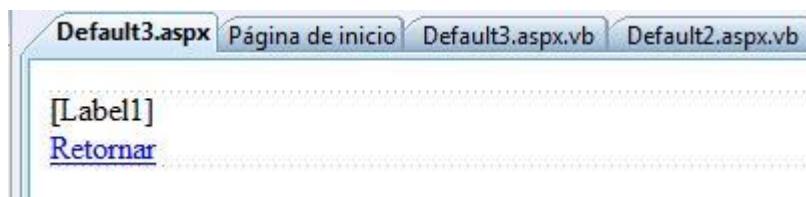
```
arch.Close()
```

Lectura del archivo de texto.

Creamos una página donde mostraremos todos los datos almacenados en el archivo de texto visitas.txt.

El archivo de texto almacena los datos de los visitantes y las marcas HTML básicas para hacer los saltos de línea y líneas separadoras entre comentarios, solo nos queda leer el archivo e ir almacenándolo en la Label para que lo muestre.

Podemos disponer un control de tipo HyperLink para retornar a la página principal.



El código necesario para leer y cargar los datos en la Label1 es:

```
Imports System.IO
Partial Class Default3
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object,
                           ByVal e As System.EventArgs) Handles Me.Load
        Dim arch As New StreamReader(Me.Server.MapPath(".") _
                                   & "/" & "visitas.txt")
        Dim linea As String
        linea = arch.ReadLine
        Do While Not linea Is Nothing
            Me.Label1.Text = Me.Label1.Text & linea
            linea = arch.ReadLine
        Loop
        arch.Close()
    End Sub
End Class
```

Importmos el espacio de nombres donde está declarada la clase StreamReader:

```
Imports System.IO
```

Procedemos a realizar la apertura del archivo, indicando el camino donde se encuentra:

```
Dim arch As New StreamReader(Me.Server.MapPath(".")) _
```

```
& "/" & "visitas.txt")
```

Antes de la estructura repetitiva procedemos a leer la primer línea del archivo:

```
Dim linea As String  
linea = arch.ReadLine
```

El método ReadLine de la clase StreamReader retorna el contenido de toda una línea del archivo de texto. En caso que no haya más líneas en el archivo de texto el método ReadLine retorna el valor Nothing.

El while nos permite recorrer todo el archivo y cargar cada línea leída en la Label1:

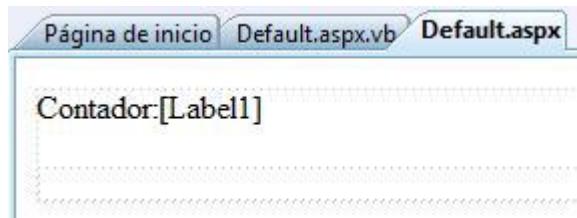
```
Do While Not linea Is Nothing  
    Me.Label1.Text = Me.Label1.Text & linea  
    linea = arch.ReadLine  
Loop
```

Finalmente procedemos a cerrar el archivo:

```
arch.Close()
```

Contador de páginas vistas.

Confeccionaremos ahora un simple contador de páginas utilizando un archivo de texto de una sola línea. Cada vez que un navegador solicite la página mostraremos el contador.



Disponemos una Label para mostrar el valor del contador que se almacena en el archivo de texto.

En el evento Page_Load hacemos todo el algoritmo:

```
Imports System.IO  
Partial Class _Default  
    Inherits System.Web.UI.Page  
  
    Protected Sub Page_Load(ByVal sender As Object,  
        ByVal e As System.EventArgs) Handles Me.Load  
        If File.Exists(Me.Server.MapPath(".") & "/" &  
            & "contador.txt") Then  
            Dim arch1 As New StreamReader(Me.Server.MapPath(".") &  
                & "/" & "contador.txt")  
            Dim conta As Integer  
            conta = arch1.ReadLine  
            conta = conta + 1  
            arch1.Close()  
            Dim arch2 As New StreamWriter(Me.Server.MapPath(".") &  
                & "/" & "contador.txt")  
            arch2.WriteLine(conta)  
            arch2.Close()  
            Me.Label1.Text = conta  
        Else  
            Dim arch As New StreamWriter(Me.Server.MapPath(".") &  
                & "/" & "contador.txt")  
            arch.WriteLine("1")  
            arch.Close()  
        End If  
    End Sub
```

```

        Me.Label1.Text = "1"
    End If
        End Sub
    End Class

```

Mediante un if verificamos si existe el archivo que almacena el contador (la clase File tiene un método estático llamado Exists que retorna true si existe el archivo en la carpeta indicada):

```

If File.Exists(Me.Server.MapPath(".") & "/" _
& "contador.txt") Then

```

En caso que el archivo existe procedemos a abrirlo con el objetivo de leer su única línea:

```

Dim arch1 As New StreamReader(Me.Server.MapPath(".") _
& "/" & "contador.txt")
Dim conta As Integer
conta = arch1.ReadLine
conta = conta + 1
arch1.Close()

```

Luego procedemos a crearlo nuevamente y almacenar el valor que acabamos de leer, previo a incrementarlo en uno (mostramos en la Label el valor del contador actual):

```

Dim arch2 As New StreamWriter(Me.Server.MapPath(".") _
& "/" & "contador.txt")
arch2.WriteLine(conta)
arch2.Close()
Me.Label1.Text = conta

```

Si el archivo no existe procedemos a crearlo almacenando el valor 1:

```

Dim arch As New StreamWriter(Me.Server.MapPath(".") _
& "/" & "contador.txt")
arch.WriteLine("1")
arch.Close()
Me.Label1.Text = "1"

```

Ejercicios Propuestos

1 – Confeccionar un sitio que permita ingresar los datos del curriculum de una persona y almacenarlos en un archivo de texto.

	UNIDAD Nº 2: Acceso a una Base de Datos con ASP.Net.	TEMAS: Clases SQLConnection, SqlCommand.	Clase 3
--	---	---	------------------------------

Objetivos:

- Crear una cadena de conexión con SQL Server.
- Configuración de la clase SQLConnection.
- Altas, Bajas, Modificaciones y consultas a nivel de código.

Introducción

Con ASP.Net podemos comunicarnos a distintos gestores de base de datos como pueden ser SQL Server, Oracle, Access, MySQL etc.

Nosotros trabajaremos con el gestor de base de datos SQL Server, uno por ser el más empleado cuando se utiliza la tecnología de ASP.Net en el desarrollo de sitios web dinámicos.

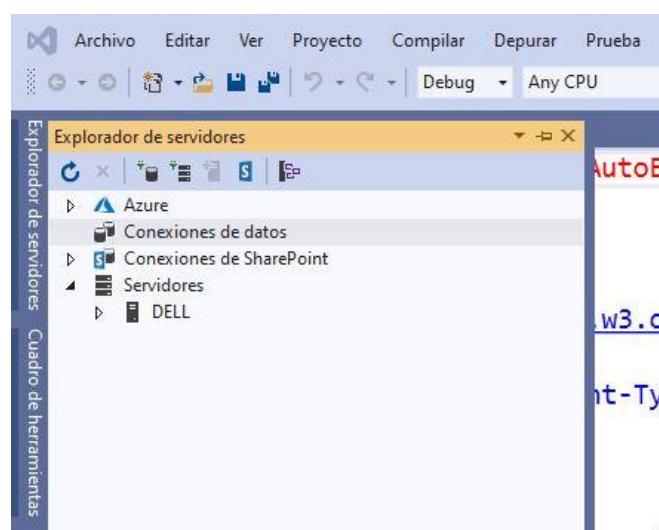
En esta clase especificaremos todo el código necesario para acceder al gestor de base de datos, desde la cadena de conexión hasta la implementación de los comandos SQL a enviar.

clase3

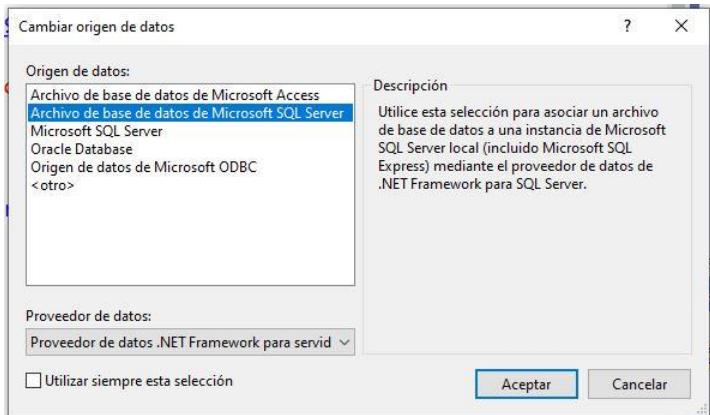
Crearemos un sitio web en el Visual Studio 2019 llamado CLASE3 y una vez creado en su interior creamos una carpeta DATOS

Dejamos un único Default ASPX y comenzamos a crear la base de datos

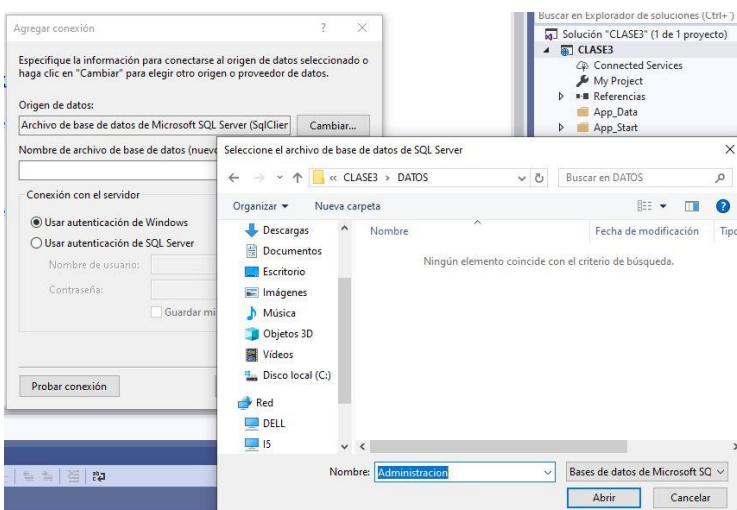
Conexiones de datos click izq agregar conexión



Archivo de base de datos de Microsoft sql server – aceptar

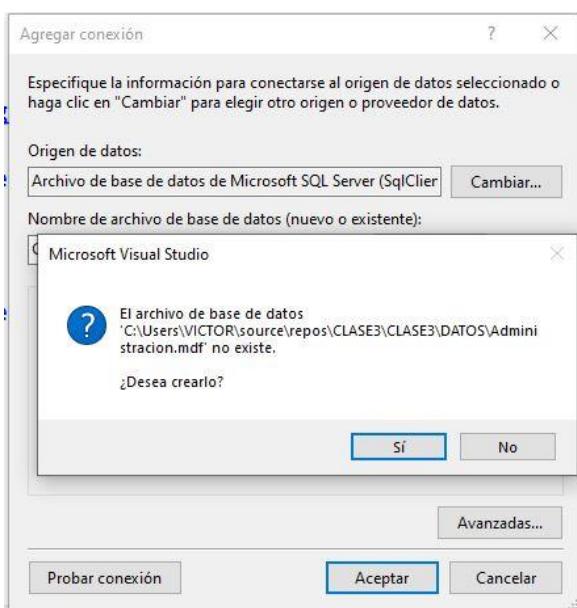


Examinamos y elegimos la carpeta DATOS de nuestro proyecto y como nombre ponemos Administración y pulsamos abrir

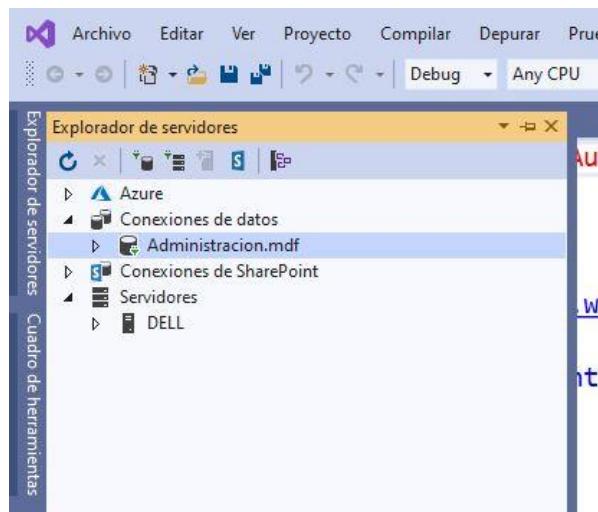


Pulsamos aceptar

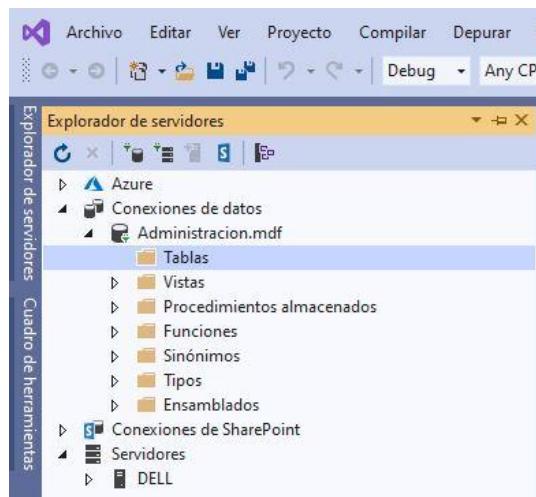
Nos pregunta si queremos crearla



En el Explorador de servidores ya nos aparecerá Administracion.mdf



Hacemos doble click sobre Administracion.mdf y aparecerán los objetos de la base de datos recién creada



Click izq sobre Tablas y crearemos una tabla llamada usuarios con tres campos:

nombre varchar(30) Clave primaria

clave varchar(30)

mail varchar(30)

The screenshot shows the 'dbo.Table [Diseño]' (Design) tab in SSMS. The table structure is defined with three columns: 'Nombre' (varchar(30)), 'clave' (varchar(30)), and 'mail' (varchar(30)). The 'Nombre' column is designated as the primary key (PK_Table). The 'T-SQL' tab at the bottom displays the generated T-SQL code for creating the table:

```
CREATE TABLE [dbo].[Table]
(
    [nombre] VARCHAR(30) NOT NULL,
    [clave] VARCHAR(30) NULL,
    [mail] VARCHAR(30) NULL,
    CONSTRAINT PK_Table PRIMARY KEY CLUSTERED ([nombre])
)
```

Se nos crea una consulta de creación de tabla en T-sql (Esta manera de crear tablas es similar a la que tiene mysql).

Cambiamos [Tabla] por [usuarios]

```

CREATE TABLE [dbo].[usuarios] (
    [nombre] VARCHAR (30) NOT NULL,
    [clave] VARCHAR (30) NULL,
    [mail] VARCHAR (30) NULL,
    CONSTRAINT [PK_Table] PRIMARY KEY CLUSTERED ([nombre] ASC)
);

```

Marcamos todas las instrucciones Clicj izq y Ejecutar código seleccionado.

```

CREATE TABLE [dbo].[usuarios] (
    [nombre] VARCHAR (30) NOT NULL,
    [clave] VARCHAR (30) NULL,
    [mail] VARCHAR (30) NULL,
    CONSTRAINT [PK_Table] PRIMARY KEY CLUSTERED ([nombre] ASC)
);

```

Ya tendremos creada la tabla usuarios

La primer página (Default.aspx) solo tendrá los hipervínculos a otras páginas que tendrán por objetivo efectuar una el alta de usuarios, otra la baja, otra las modificaciones y por último otra la consulta:

Default.aspx* Página de inicio

[Alta de usuarios.](#)

Consulta de usuarios.

Baja de usuarios.

Modificación de usuarios.

Para crear esta interface insertaremos cuatro objetos de la clase HyperLink, como mínimo debemos inicializar las propiedades text (es el texto que mostrará el hipervínculo en el navegador y la propiedad NavigateUrl que indica el nombre de la página que debe cargar el navegador cuando se presione el hipervínculo)

Como todavía no tenemos creada las otras cuatro páginas no podemos inicializar la propiedad NavigateUrl de cada HyperLink.

Como segundo paso creemos las cuatro páginas, para ello desde el menú: Archivo -> Nuevo archivo... seleccionamos la plantilla "Web Forms" y en la parte inferior definimos el nombre del archivo aspx.

Los nombres de las cuatro páginas a crear serán:

altausuario.aspx
consultausuario.aspx
bajausuario.aspx
modificacionusuario.aspx

Una vez que hemos creado las cuatro páginas aspx podemos proceder a enlazar la propiedad NavigateUrl de cada control HyperLink.

Cuando seleccionamos la propiedad NavigateUrl aparece un dialogo que nos permite seleccionar la página aspx a enlazar.

Una vez inicializada la propiedad NavigateUrl podremos ver que el texto aparece subrayado (indicando que se trata de un hipervínculo)

Definición de la cadena de conexión con la base de datos en el archivo web.config

Web.config es el archivo principal de opciones de configuración para una aplicación web en ASP.NET.

El archivo es un documento XML que define información de configuración concerniente a la aplicación web. El archivo web.config contiene información que controla la carga de modulos, configuraciones de seguridad, configuraciones del estado de la sesión, opciones de compilación y el lenguaje de la aplicación.

El archivo web.config contiene también la cadenas de conexión a la base de datos.

Debemos modificar la sección o crearla si no existe

<connectionStrings/>
y remplazarlo por:

```
<connectionStrings>
  <add name="Administracion"
    connectionString=" Data Source=(LocalDB)\MSSQLLocalDB;
    Initial Catalog=Administracion;

    AttachDbFilename=C:\Users\VICTOR\source\repos\CLASE3\CLASE3\DATOS\Administracion.mdf;
    Integrated Security=True;
    Connect Timeout=30 ;"/>
</connectionStrings>
```

Como vemos en la propiedad connectionString indicamos en Initial Catalog el nombre de la base de datos que hemos creado en SQL Server.

Altas.

Activemos desde el Visual Studio 2019 la pestaña altausuario.aspx para elaborar la interface visual que nos permita efectuar la carga de datos de usuarios:



Como podemos ver disponemos tres controles de tipo TextBox, el que solicita el ingreso de la clave modificamos la propiedad TextMode con el valor Password, los otros dos los dejamos con el valor SingleLine.

Disponemos un objeto de la clase Button y una Label donde mostraremos un mensaje si el alta se efectuó correctamente.

Por último disponemos un objeto de la clase HyperLink configurando la propiedad NavigateUrl con la dirección de la página principal (Default.aspx)

Ahora codificamos el evento clic del botón de alta:

```
Imports System.Data.SqlClient
Partial Class altausuario
    Inherits System.Web.UI.Page
    Protected Sub Button1_Click(ByVal sender As Object,
        ByVal e As System.EventArgs) Handles Button1.Click
        Try
            Dim s As String
            s = ConfigurationManager.ConnectionStrings("administracion").ConnectionString
            Dim conexion As New SqlConnection(s)
            conexion.Open()
            Dim comando As New SqlCommand("insert into
                usuarios(nombre,clave,mail) values('"
                & Me.TextBox1.Text & "','" & Me.TextBox2.Text & "','" &
                & Me.TextBox3.Text & "')", conexion)
            comando.ExecuteNonQuery()
            Me.Label1.Text = "Se registró el usuario"
            conexion.Close()
        Catch ex As SqlException
            Me.Label1.Text = ex.Message
        End Try
    End Sub
End Class
```

Lo primero que debemos hacer es importar el espacio de nombres donde se encuentra definida la clase SQLException:

```
Imports System.Data.SqlClient
```

Al presionar el botón primero extraemos la cadena de conexión que tenemos almacenada en el archivo web.config:

```
s = ConfigurationManager.ConnectionStrings("administracion").ConnectionString
```

Creamos un objeto de la clase SQLConnection indicando como parámetro la cadena de conexión que rescatamos anteriormente:

```
Dim conexion As New SqlConnection(s)
```

Abrimos la conexión:

```
conexion.Open()
```

Creamos un objeto de la clase SqlCommand creándolo con los datos cargados en los controles TextBox:

```
Dim comando As New SqlCommand("insert into
    usuarios(nombre,clave,mail) values('"
    & Me.TextBox1.Text & "','" & Me.TextBox2.Text & "','" -
    & Me.TextBox3.Text & "')", conexion)
```

Pedimos a SQLServer que ejecute el comando especificado anteriormente:

```
comando.ExecuteNonQuery()
```

Cerramos la conexión:

```
conexion.Close()
```

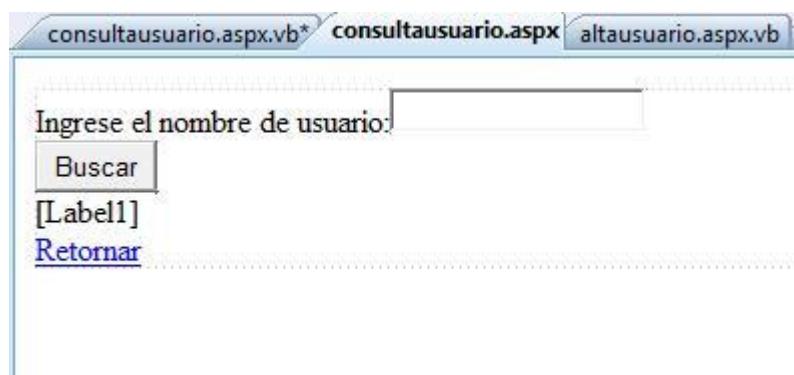
Todo esto lo hacemos capturando la excepción SQLException.

En caso de dispararse un error será capturado por el Catch y procederemos a mostrar en una Label el mensaje de error respectivo:

```
Catch ex As SqlException
    Me.Label1.Text = ex.Message
End Try
```

Consultas.

Seleccionamos del Explorador de soluciones la página consultausuario.aspx y procedemos a elaborar la siguiente interface visual (disponemos un TextBox, un Button, una Label y un HyperLink):



El código del evento click es:

```
Imports System.Data.SqlClient
Partial Class consultausuario
```

```

Inherits System.Web.UI.Page

Protected Sub Button1_Click(ByVal sender As Object,
    ByVal e As System.EventArgs) Handles Button1.Click
    Try
        Dim s As String
        s = ConfigurationManager.ConnectionStrings("administracion").ConnectionString
        Dim conexion As New SqlConnection(s)
        conexion.Open()
        Dim comando As New SqlCommand("select nombre,clave,mail from usuarios " +
            & " where nombre='" & Me.TextBox1.Text & "'", conexion)
        Dim registro As SqlDataReader = comando.ExecuteReader
        If registro.Read Then
            Me.Label1.Text = "Clave:" & registro("clave") & "<br>" +
                & "Mail:" & registro("mail")
        Else
            Me.Label1.Text = "No existe un usuario con dicho nombre"
        End If
    Catch ex As SqlException
        Me.Label1.Text = ex.Message
    End Try
End Sub
End Class

```

Para poder recuperar los datos lo hacemos creando un objeto de la clase SqlDataReader e inicializándolo mediante la llamada del método ExecuteReader de la clase SqlCommand:

```
Dim registro As SqlDataReader = comando.ExecuteReader
```

Luego recorremos el SqlDataReader (como este caso puede retornar cero o una fila lo hacemos mediante un if:

```
If registro.Read Then
```

Si el método Read retorna true luego podemos acceder a la fila recuperada con el select.

Bajas.

Seleccionamos del Explorador de soluciones la página bajausuario.aspx y procedemos a elaborar la siguiente interface visual:



Luego el código a ejecutar cuando se presiona el botón “Borrar” de la página es:

```

Imports System.Data.SqlClient

Partial Class bajausuario
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(ByVal sender As Object,
        ByVal e As System.EventArgs) Handles Button1.Click
        Try

```

```

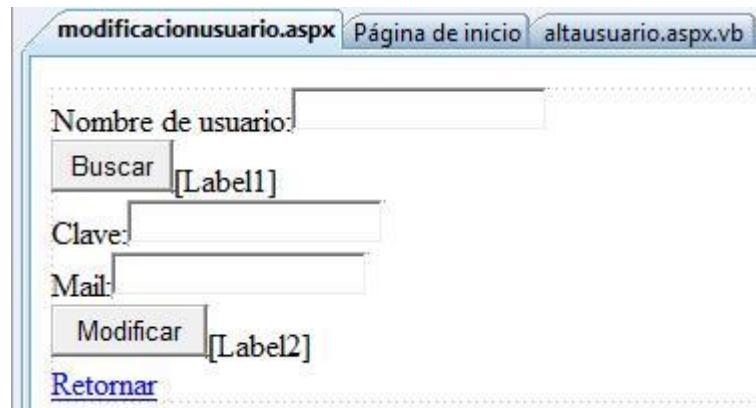
Dim s As String
s = ConfigurationManager.ConnectionStrings("administracion").ConnectionString
Dim conexion As New SqlConnection(s)
conexion.Open()
Dim comando As New SqlCommand("delete from usuarios where nombre='"
& Me.TextBox1.Text & "'", conexion)
Dim cant As Integer = comando.ExecuteNonQuery
If cant = 1 Then
    Me.Label1.Text = "Se borró el usuario"
Else
    Me.Label1.Text = "No existe un usuario con dicho nombre"
End If
conexion.Close()
Catch ex As SqlException
    Me.Label1.Text = ex.Message
End Try
End Sub
End Class

```

El método ExecuteNonQuery retorna un entero y representa la cantidad de filas borradas de la tabla.

Modificaciones.

Por último implementaremos la modificación de datos. Seleccionamos del Explorador de soluciones la página modificacionusuario.aspx y procedemos a elaborar la siguiente interface visual:



Para efectuar la modificación de datos de un usuario procederemos primero a la búsqueda de los datos actuales.

Luego el código para los eventos click de los dos botones es:

```

Imports System.Data.SqlClient
Partial Class modificacionusuario
Inherits System.Web.UI.Page
Protected Sub Button1_Click(ByVal sender As Object,
                           ByVal e As System.EventArgs) Handles Button1.Click
    Try
        Dim s As String
        s = ConfigurationManager.ConnectionStrings("Administracion").ConnectionString
        Dim conexion As New SqlConnection(s)
        conexion.Open()
        Dim comando As New SqlCommand("select nombre,clave,mail from usuarios " -
                                      & " where nombre=''" & Me.TextBox1.Text & "'", conexion)

```

```

Dim registro As SqlDataReader = comando.ExecuteReader
If registro.Read Then
    Me.TextBox2.Text = registro("clave")
    Me.TextBox3.Text = registro("mail")
Else
    Me.Label1.Text = "No existe un usuario con dicho nombre"
End If
Catch ex As SqlException
    Me.Label1.Text = ex.Message
End Try
End Sub

Protected Sub Button2_Click(ByVal sender As Object,
                           ByVal e As System.EventArgs) Handles Button2.Click
Try
    Dim s As String
    s = ConfigurationManager.ConnectionStrings("Administracion").ConnectionString
    Dim conexion As New SqlConnection(s)
    conexion.Open()
    Dim comando As New SqlCommand("update usuarios set " & _
        "clave=''" & Me.TextBox2.Text & _
        "'','mail=''" & Me.TextBox3.Text & _
        "' where nombre=''" & Me.TextBox1.Text & "'", conexion)
    Dim cant As Integer = comando.ExecuteNonQuery()
    If cant = 1 Then
        Me.Label1.Text = "Datos Modificados"
    Else
        Me.Label1.Text = "No existe el usuario"
    End If
    conexion.Close()
    Catch ex As SqlException
        Me.Label1.Text = ex.Message
    End Try
End Sub
End Class

```

El botón “Buscar” hace lo mismo que vimos en la consulta. Luego cuando se presiona el botón “Modificar” procedemos a hacer un update de la tabla usuarios con los datos cargados en los TextBox.

Ejercicios Propuestos

1 – Crear una tabla:
alumnos (dni varchar(8), apellidonom varchar(50), provincia varchar(30))

Confeccionar una serie de páginas que permitan efectuar altas, bajas, modificaciones y consultas.

	UNIDAD Nº 2: Acceso a una Base de Datos con ASP.Net.	TEMAS: Altas, Bajas, Modificaciones y Consultas de una tabla con el SQLDataSource.	Clase 4
--	---	---	-------------------

Objetivos:

- Conexión al servidor de base de datos SQL Server.
- Configuración de la componente SqlDataSource.
- Altas, Bajas, Modificaciones y consultas de una tabla.

Introducción

Con ASP.Net podemos comunicarnos a distintos gestores de base de datos como pueden ser SQL Server, Oracle, Access, MySQL etc.

Nosotros trabajaremos con el gestor de base de datos SQL Server, uno por ser el más empleado cuando se utiliza la tecnología de ASP.Net en el desarrollo de sitios web dinámicos.

Clase4

Crearemos una base de datos en SQL Server llamada: Administracion y dentro de la misma definiremos una tabla llamada usuarios con tres campos:

```
nombre varchar(30) Clave primaria
clave  varchar(30)
mail   varchar(30)
```

Crearemos un sitio web en el Visual Studio 2019 llamado CLASE4.

La primer página solo tendrá los hipervínculos a otras páginas que tendrán por objetivo efectuar una el alta de usuarios, otra la baja, otra las modificaciones y por último otra la consulta:

Para crear esta interface insertaremos cuatro objetos de la clase HyperLink, como mínimo debemos inicializar las propiedades text (es el texto que mostrará el hipervínculo en el navegador y la propiedad NavigateUrl que indica el nombre de la página que debe cargar el navegador cuando se presione el hipervínculo)

Como todavía no tenemos creada las otras cuatro páginas no podemos inicializar la propiedad NavigateUrl de cada HyperLink.

Como segundo paso creemos las cuatro páginas, para ello desde el menú: Archivo -> Nuevo archivo... seleccionamos la plantilla "Web Forms" y en la parte inferior definimos el nombre del archivo aspx.

Los nombres de las cuatro páginas a crear serán:

altausuario.aspx
consultausuario.aspx
bajausuario.aspx
modificacionusuario.aspx

Una vez que hemos creado las cuatro páginas aspx podemos proceder a enlazar la propiedad NavigateUrl de cada control HyperLink.

Cuando seleccionamos la propiedad NavigateUrl aparece un dialogo que nos permite seleccionar la página aspx a enlazar.

Una vez inicializada la propiedad NavigateUrl podremos ver que el texto aparece subrayado (indicando que se trata de un hipervínculo)

Altas.

Activemos desde el Visual Studio 2008 la pestaña altausuario.aspx para elaborar la interface visual que nos permita efectuar la carga de datos de usuarios:



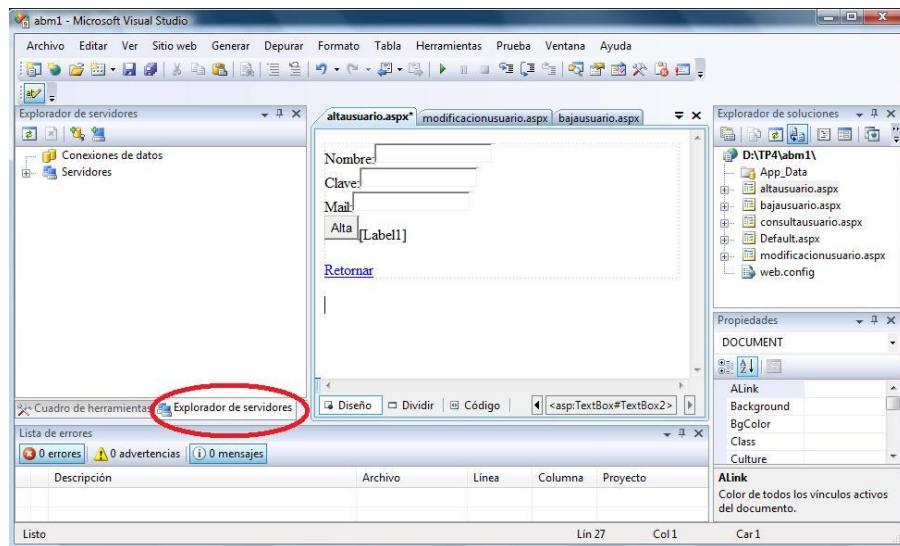
Como podemos ver disponemos tres controles de tipo TextBox, el que solicita el ingreso de la clave modificamos la propiedad TextMode con el valor Password, los otros dos los dejamos con el valor SingleLine.

Disponemos un objeto de la clase Button y una Label donde mostraremos un mensaje si el alta se efectuó correctamente.

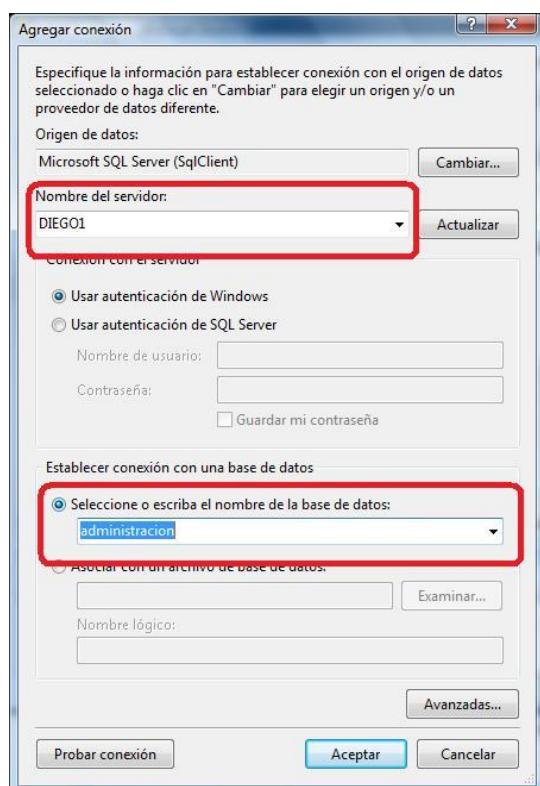
Por último disponemos un objeto de la clase HyperLink configurando la propiedad NavigateUrl con la dirección de la página principal (Default.aspx)

Conexión con la base de datos.

Ahora veremos como crear una conexión con nuestra base de datos que hemos creado desde SQL Server.

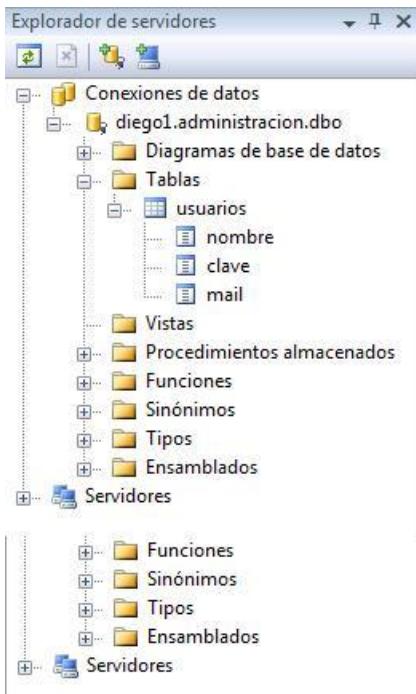


Seleccionamos la solapa “Explorador de servidores” y presionamos botón derecho sobre “Conexiones a datos”, elegimos la opción “Agregar conexión”, aparece una ventana de configuración:



Debemos seleccionar el nombre del servidor y posteriormente el nombre de la base de datos. Podemos presionar luego el botón “Probar conexión” para comprobar la correcta configuración de la conexión.

Presionamos por último el botón “Aceptar”, veremos luego que aparece la pestaña del “Explorador de servidores” la conexión de datos que acabamos de crear. Podemos presionar sobre el signo más que aparece en nuestra conexión y ver que tenemos disponible el acceso a las Tablas, Vistas, Procedimientos almacenados etc. definidos para la base de datos:

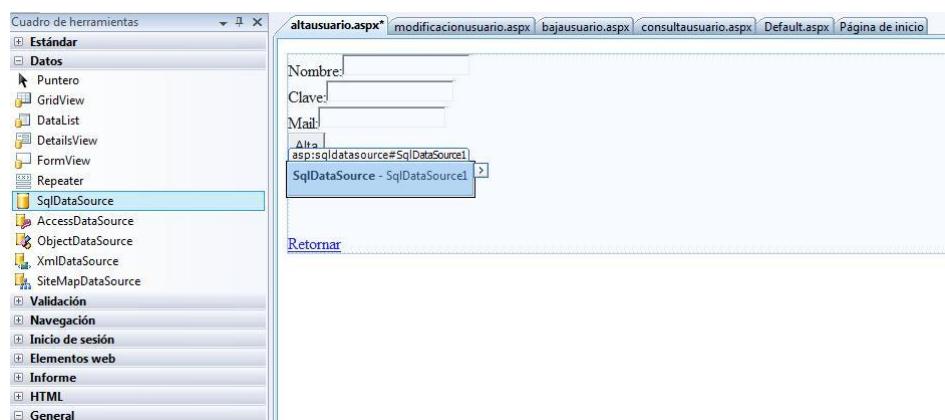


Podemos ahora desde el “explorador de servidores” ver las tablas, campos de una tabla e inclusive si presionamos el botón derecho del mouse sobre la tabla usuarios podemos ver los datos de la misma.

Ahora podemos empezar con el alta de usuarios.

Del cuadro de herramientas seleccionamos un control de la clase `SqlDataSource` (tener en cuenta que está en el grupo “Datos” y no en el grupo “Estándar”)

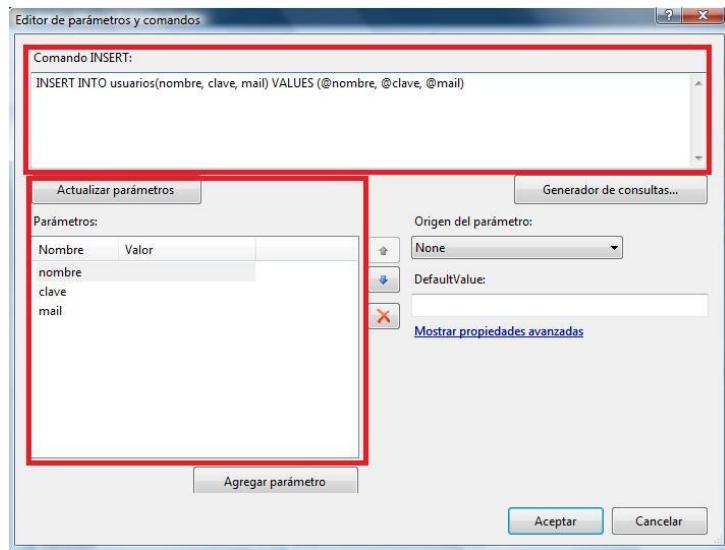
Ahora podemos ver que nuestro formulario tiene un componente llamado `SqlDataSource1` que es de la clase `SqlDataSource`. Este componente visual no se verá en tiempo de ejecución como veremos más adelante.



Seleccionamos el control `SqlDataSource1` del formulario y en la ventana de “Propiedades” inicializamos la propiedad **ConnectionString** con el valor que aparece al presionar la pestaña de la derecha (básicamente es la cadena de conexión que creamos anteriormente)

Ahora nos queda configurar la propiedad **InsertQuery** con el comando SQL que nos permite insertar un registro en una tabla.

La propiedad InsertQuery nos abre una ventana de diálogo donde debemos configurar el comando INSERT:

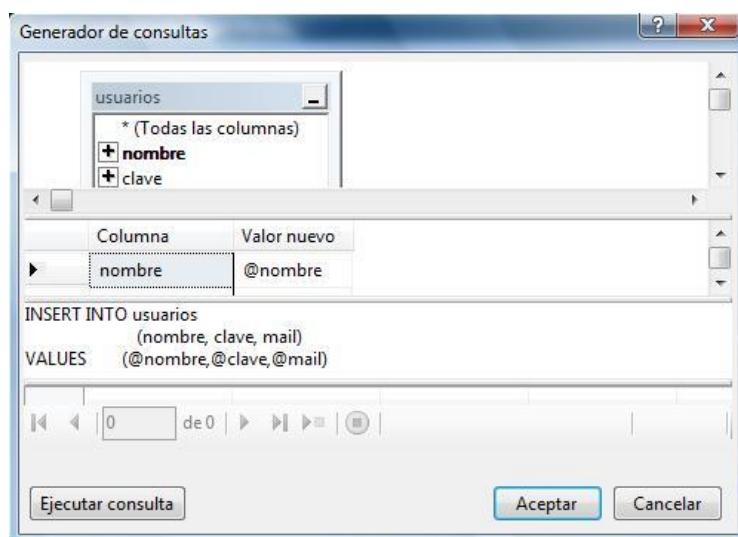


Este diálogo es muy importante ingresar correctamente el comando SQL parametrizando los valores que serán remplazados en tiempo de ejecución con los datos que cargue el operador.

Los parámetros se indican con un nombre antecediéndole el carácter @.

Luego de crear completamente el comando Insert procedemos a presionar el botón "Actualizar parámetros".

Si queremos comprobar si nuestro comando SQL está correcto presionamos el botón "Generador de consultas..." y desde este nuevo diálogo presionamos el botón "Ejecutar consulta":



(También desde este diálogo podemos codificar la consulta, probarla y finalmente confirmarla)

Lo que no hay que olvidarse nunca es que cada vez que agregamos o borramos un parámetro de nuestro comando SQL es presionar el botón "Actualizar parámetros".

Ahora codificamos el evento clic del botón de alta:

```
Imports System.Data.SqlClient  
Partial Class altausuario
```

```

Inherits System.Web.UI.Page

Protected Sub Button1_Click(ByVal sender As Object,
    ByVal e As System.EventArgs) Handles Button1.Click
    Try
        Me.SqlDataSource1.InsertParameters("nombre").DefaultValue = _
            Me.TextBox1.Text
        Me.SqlDataSource1.InsertParameters("clave").DefaultValue = _
            Me.TextBox2.Text
        Me.SqlDataSource1.InsertParameters("mail").DefaultValue = _
            Me.TextBox3.Text
        Me.SqlDataSource1.Insert()
        Me.Label1.Text = "Se efectuó la carga."
    Catch ex As SqlException
        Me.Label1.Text = ex.Message
    End Try
End Sub
End Class

```

Lo primero que debemos hacer es importar el espacio de nombres donde se encuentra definida la clase `SqlException`:

```
Imports System.Data.SqlClient
```

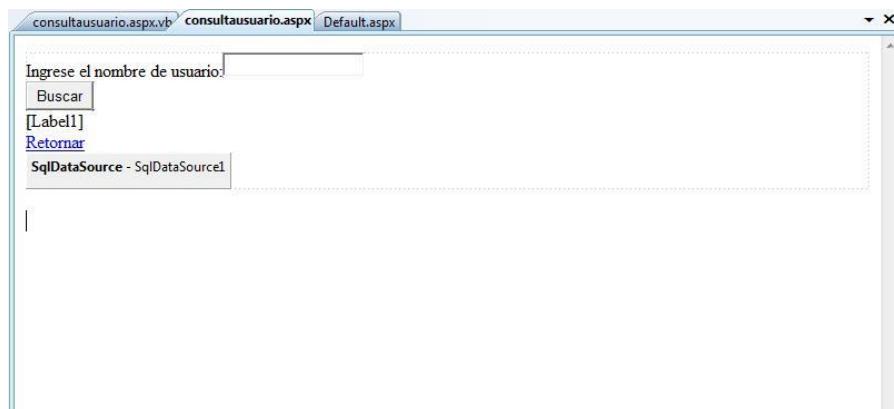
Al presionar el botón inicializamos los tres parámetros del comando SQL que hemos configurado en el `SqlDataSource`. Para acceder a dichos parámetros disponemos de la propiedad `InsertParameters` que le indicamos como subíndice el nombre del parámetro.

Luego de inicializar los tres parámetros procedemos a llamar al método `Insert` de la clase `SqlDataSource`.

Todo esto lo hacemos capturando la excepción `SQLException`.

Consultas.

Seleccionamos del Explorador de soluciones la página `consultausuario.aspx` y procedemos a elaborar la siguiente interface visual (disponemos un `TextBox`, un `Button`, una `Label`, un `HyperLink` y un `SqlDataSource`):



Configuramos el `SqlDataSource1`, luego de seleccionarlo con el mouse modificamos la propiedad **ConnectionString** (con la conexión creada anteriormente) y en el diálogo que abre la propiedad **SelectQuery** procedemos a codificar el comando select:

```
select clave,mail from usuarios where nombre=@nombre
```

Es decir recuperamos la clave y mail de un determinado usuario (recordar de presionar el botón **Actualizar parámetros**)

El código del evento click es:

```

Imports System.Data.SqlClient
Partial Class consultausuario
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(ByVal sender As Object,
        ByVal e As System.EventArgs) Handles Button1.Click
        Try
            Me.SqlDataSource1.SelectParameters("nombre").DefaultValue = _
                Me.TextBox1.Text
            Me.SqlDataSource1.DataSourceMode = _
                SqlDataSourceMode.DataReader
            Dim datos As SqlDataReader
            datos =
                Me.SqlDataSource1.Select(DataSourceSelectArguments.Empty)
            If datos.Read Then
                Me.Label1.Text = "Clave:" & datos("clave") & "<br />" -
                    & "Mail:" & datos("mail")
            Else
                Me.Label1.Text = "No existe un usuario con dicho nombre"
            End If
        Catch ex As SQLException
            Me.Label1.Text = ex.Message
        End Try
    End Sub
End Class

```

Para poder recuperar los datos mediante un objeto de la clase SqlDataReader debemos configurar el SqlDataSource indicando a la propiedad DataSourceMode el siguiente valor:

```
Me.SqlDataSource1.DataSourceMode = SqlDataSourceMode.DataReader
```

Ahora cuando llamemos al método select del SqlDataSource lo hacemos con la siguiente sintaxis:

```
Dim datos As SqlDataReader
datos = Me.SqlDataSource1.Select(DataSourceSelectArguments.Empty)
```

Luego recorremos el SqlDataReader (como este caso puede retornar cero o una fila lo hacemos mediante un if:

```
If datos.Read Then
```

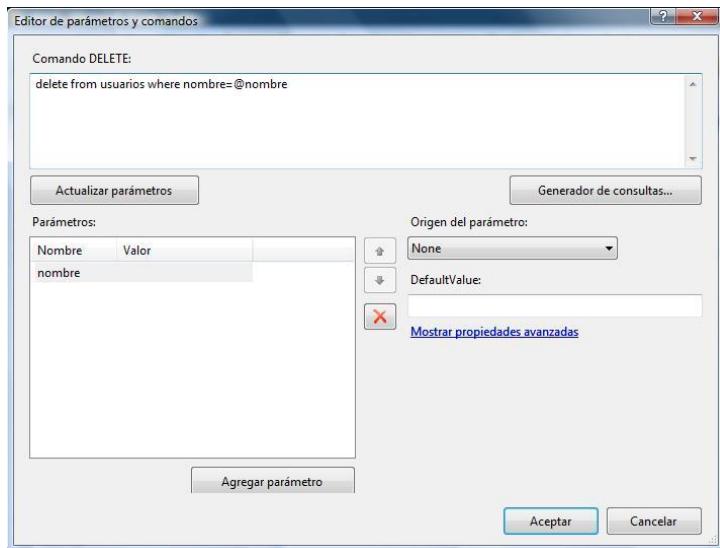
Si el método Read retorna true luego podemos acceder a la fila recuperada con el select.

Bajas.

Seleccionamos del Explorador de soluciones la página bajausuario.aspx y procedemos a elaborar la siguiente interface visual:



Inicializamos la propiedad **ConnectionString** del **SqlDataSource1** con la conexión que habíamos creado e inicializamos la propiedad **DeleteQuery**:



Luego el código a ejecutar cuando se presiona el botón “Borrar” de la página es:

```
Imports System.Data.SqlClient

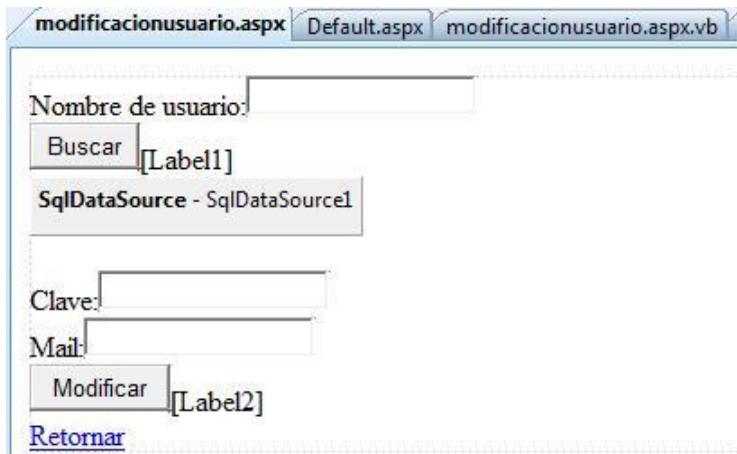
Partial Class balausuario
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(ByVal sender As Object,
        ByVal e As System.EventArgs) Handles Button1.Click
        Try
            Me.SqlDataSource1.DeleteParameters("nombre").DefaultValue =
                = Me.TextBox1.Text
            Dim cant As Integer
            cant = Me.SqlDataSource1.Delete
            If cant = 1 Then
                Me.Label1.Text = "Se borró el usuario"
            Else
                Me.Label1.Text = "No existe dicho nombre"
            End If
        Catch ex As SqlException
            Me.Label1.Text = ex.Message
        End Try
    End Sub
End Class
```

Procedemos a inicializar el parámetro y luego llamamos al método Delete del **SqlDataSource**. El método Delete retorna un entero y representa la cantidad de filas borradas de la tabla.

Modificaciones.

Por último implementaremos la modificación de datos. Seleccionamos del Explorador de soluciones la página **modificacionusuario.aspx** y procedemos a elaborar la siguiente interface visual:



Para efectuar la modificación de datos de un usuario procederemos primero a la búsqueda de los datos actuales. Esto hace que necesitemos un control de la clase **SqlDataSource**.

El **SqlDataSource1** nos permite consultar los datos de un usuario por su nombre y mostrar el resultado en dos controles de tipo **TextBox**. La propiedad **SelectQuery**:

`select clave,mail from usuarios where nombre=@nombre`

Luego también configuramos la propiedad **UpdateQuery**:

`UPDATE usuarios SET clave = @clave, mail = @mail WHERE (nombre = @nombre)`

Luego el código para los eventos click de los dos botones es:

```

Imports System.Data.SqlClient

Partial Class modificacionusuario
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles Button1.Click
        Try
            Me.SqlDataSource1.SelectParameters("nombre").DefaultValue =
                Me.TextBox1.Text
            Me.SqlDataSource1.DataSourceMode =
                SqlDataSourceMode.DataReader
            Dim datos As SqlDataReader
            datos =
                Me.SqlDataSource1.Select(DataSourceSelectArguments.Empty)
            If datos.Read Then
                Me.TextBox2.Text = datos("clave")
                Me.TextBox3.Text = datos("mail")
            Else
                Me.Label1.Text = "No existe dicho usuario"
            End If
        Catch ex As SqlException
            Me.Label1.Text = ex.Message
        End Try
    End Sub

    Protected Sub Button2_Click(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles Button2.Click
        Try
            Me.SqlDataSource1.UpdateParameters("clave").DefaultValue =
                Me.TextBox2.Text
            Me.SqlDataSource1.UpdateParameters("mail").DefaultValue =
                Me.TextBox3.Text
            Me.SqlDataSource1.UpdateParameters("nombre").DefaultValue =
                Me.TextBox1.Text
            Dim cant As Integer
            cant = Me.SqlDataSource1.Update()
            If cant = 1 Then
                Me.Label2.Text = "Se modificaron los datos"
            End If
        End Try
    End Sub

```

```
    Else
        Me.Label2.Text = "No existe dicho usuario"
    End If
Catch ex As SqlException
    Me.Label2.Text = ex.Message
End Try
End Sub
End Class
```

	UNIDAD Nº 2: Acceso a una Base de Datos con ASP.Net.	TEMAS: Altas, Bajas, Modificaciones y Consultas de varias tablas.	Clase 5
--	---	--	------------------------------

Objetivos:

- Conexión al servidor de base de datos SQL Server.
- Configuración de la componente SqlDataSource.
- Altas, Bajas, Modificaciones y consultas de varias tablas.

Introducción

En la mayoría de los problemas se requiere el acceso a varias tablas del gestor de base de datos. Algunas las accedemos para consultar, mientras a otras las modificamos o cargamos datos. Seguiremos trabajando con el gestor de base de datos SQL Server.

CLASE5

Continuamos con la base de datos que creamos en la clase anterior en SQL Server llamada: administración y dentro de la misma crearemos ahora dos nuevas tablas:

Tabla: articulos

codigo	int	Clave primaria e identidad.
descripcion	varchar(50)	
precio	float	
codigotitulo	int	

Tabla: titulos

Codigo	int	Clave primaria e identidad
descripción	varchar(50)	

Crearemos un sitio web en el Visual Studio 2019 llamado CLASE5.

La primer página solo tendrá los hipervínculos a otras páginas que tendrán por objetivo efectuar una el alta de artículos, otra la baja, otra las modificaciones y por último otra la consulta.

Desde la herramienta que utilizamos para crear la base de datos y las tablas procedemos a cargar algunos títulos en forma manual (esto nos permitirá entrar de lleno en los algoritmos que requieren trabajar con las dos tablas)

Para crear esta interface insertaremos cuatro objetos de la clase HyperLink, como mínimo debemos inicializar las propiedades text (es el texto que mostrará el hipervínculo en el navegador y la propiedad NavigateUrl que indica el nombre de la página que debe cargar el navegador cuando se presione el hipervínculo)

Como todavía no tenemos creada las otras cuatro páginas no podemos inicializar la propiedad NavigateUrl de cada HyperLink.

Como segundo paso creamos las cuatro páginas, para ello desde el menú: Archivo -> Nuevo archivo... seleccionamos la plantilla "Web Forms" y en la parte inferior definimos el nombre del archivo aspx.

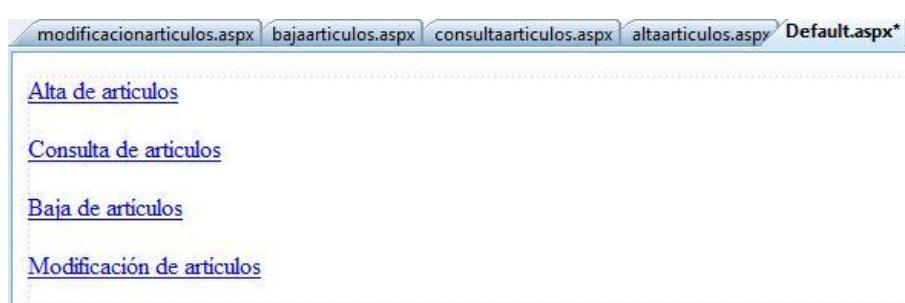
Los nombres de las cuatro páginas a crear serán:

altaarticulos.aspx
consultaarticulos.aspx
bajaarticulos.aspx
modificacionarticulos.aspx

Una vez que hemos creado las cuatro páginas aspx podemos proceder a enlazar la propiedad NavigateUrl de cada control HyperLink.

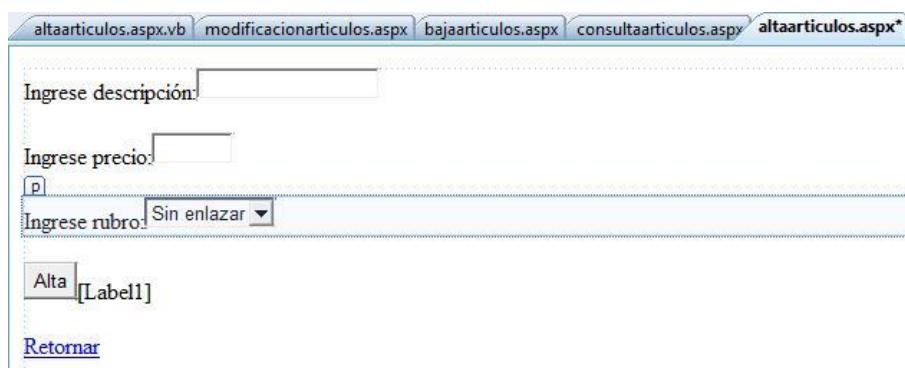
Cuando seleccionamos la propiedad NavigateUrl aparece un dialogo que nos permite seleccionar la página aspx a enlazar.

Una vez inicializada la propiedad NavigateUrl podremos ver que el texto aparece subrayado (indicando que se trata de un hipervínculo):



Altas.

Activemos desde el Visual Studio 2019 la pestaña altaarticulos.aspx para elaborar la interface visual que nos permita efectuar la carga de datos de artículos:



Como podemos ver disponemos dos controles de tipo TextBox, un control de tipo DropDownList el cual nos permitirá seleccionar el rubro que pertenece el artículo que estamos cargando. Disponemos un objeto de la clase Button y una Label donde mostraremos un mensaje si el alta se efectuó correctamente.

Por último disponemos un objeto de la clase HyperLink configurando la propiedad NavigateUrl con la dirección de la página principal (Default.aspx)

Conexión con la base de datos.

Utilizaremos la misma conexión que creamos en la clase anterior. Si desde el Visual Estudio activamos la pestaña Explorador de servidores veremos que está presente la conexión con la base de datos administración.

Ahora podemos empezar con el alta de artículos.

Del cuadro de herramientas seleccionamos un control de la clase SqlDataSource (tener en cuenta que está en el grupo “Datos” y no en el grupo “Estándar”)

Ahora podemos ver que nuestro formulario tiene un componente llamado SqlDataSource1 que es de la clase SqlDataSource.

Ahora cambiamos el nombre de este objeto (SqlDataSource1) por SqlDataSourceTitulos.

Este primer SqlDataSource nos permitirá rescatar los datos de los titulos (codigo y descripción) y poblar el control DropDownList.

Seleccionamos el control SqlDataSourceTitulos del formulario y en la ventana de “Propiedades” inicializamos la propiedad **ConnectionString** con el valor que aparece al presionar la pestaña de la derecha (básicamente es la cadena de conexión que creamos anteriormente)

Ahora nos queda configura la propiedad SelectQuery con el comando SQL que nos permite recuperar todas las filas de la tabla Titulos.

La propiedad **SelectQuery** nos abre una ventana de diálogo donde debemos configurar el comando SELECT:

```
select codigo,descripcion from titulos
```

Una vez hecho esta configuración procedemos a codificar el enlace entre el DropDonwList1 y el SqlDataSourceTitulos en el evento Load de la página:

```
Partial Class altaarticulos
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles Me.Load
        If Me.IsPostBack = False Then
            Me.DropDownList1.DataSource = Me.SqlDataSourceTitulos
            Me.DropDownList1.DataTextField = "descripcion"
            Me.DropDownList1.DataValueField = "codigo"
            Me.DropDownList1.DataBind()
        End If
    End Sub
End Class
```

La propiedad IsPostBack de la página almacena false la primera vez que solicitamos la página desde el navegador, de ahí en más cada vez que presionamos un botón de la página (Alta por ejemplo) la propiedad IsPostBack almacena true.

Con esta condición logramos que la carga del control DropDownList1 se ejecute solo la primera vez que solicitamos la página.

Debemos inicializar tres propiedades del control DropDownList para cargar los datos de los titulos. La propiedad DataSource la inicializamos con la referencia a nuestro objeto SqlDataSourceTitulos. Las

propiedades DataTextField y DataValueField deben inicializarse con los nombres de las columnas de la tabla titulos que deben visualizarse y seleccionarse en el control. Por último llamamos al método DataBind para que se genere el HTML del control.

Ahora procedemos a agregar un segundo objeto de la clase SqlDataSource que nos permitirá actualizar la tabla artículos.

Seleccionamos el control SqlDataSource1 del formulario y le asignamos un nombre más significativo (SqlDataSourceArticulos) y en la ventana de “Propiedades” inicializamos la propiedad **ConnectionString** con el valor que aparece al presionar la pestaña de la derecha.

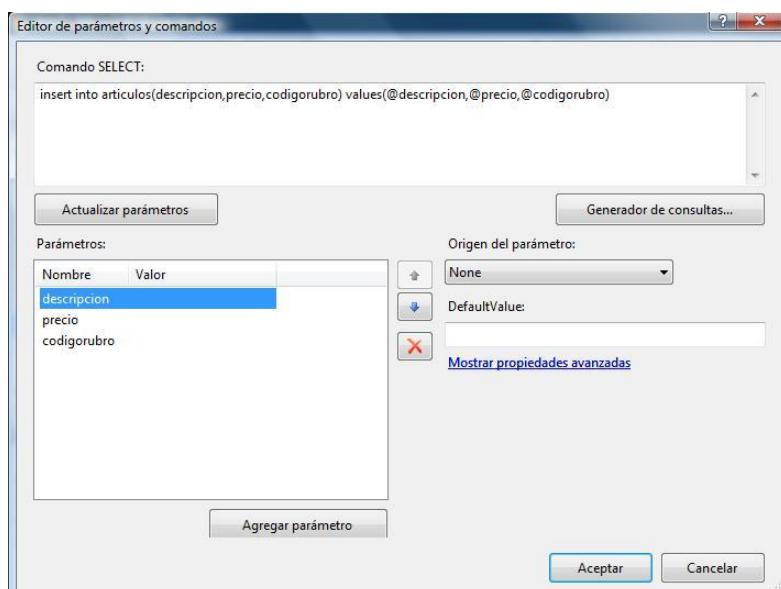
Ahora nos queda configura la propiedad **InsertQuery** con el comando SQL que nos permite insertar un registro en una tabla.

La propiedad **InsertQuery** nos abre una ventana de diálogo donde debemos configurar el comando INSERT:

Decíamos en la clase anterior que este diálogo es muy importante ingresar correctamente el comando SQL parametrizando los valores que serán remplazados en tiempo de ejecución con los datos que cargue el operador.

Los parámetros se indican con un nombre antecediéndole el carácter @.

Luego de crear completamente el comando Insert procedemos a presionar el botón “Actualizar parámetros”.



Si queremos comprobar si nuestro comando SQL está correcto presionamos el botón “Generador de consultas...” y desde este nuevo diálogo presionamos el botón “Ejecutar consulta”:

Lo que no hay que olvidarse nunca es que cada vez que agregamos o borramos un parámetro de nuestro comando SQL es presionar el botón “Actualizar parámetros”.

Ahora codificamos el evento clic del botón de alta:

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As _  
System.EventArgs) Handles Button1.Click  
    Try
```

```

Me.SqlDataSourceArticulos.InsertParameters("descripcion") _ 
    .DefaultValue = Me.TextBox1.Text
Me.SqlDataSourceArticulos.InsertParameters("precio") _ 
    .DefaultValue = Me.TextBox2.Text
Me.SqlDataSourceArticulos.InsertParameters("codigotitulo") _ 
    .DefaultValue = Me.DropDownList1.SelectedValue
Me.SqlDataSourceArticulos.Insert()
Me.Label1.Text = "Se efectuó la carga"
Catch ex As Exception
    Me.Label1.Text = ex.Message
End Try
End Sub

```

Lo primero que debemos hacer es importar el espacio de nombres donde se encuentra definida la clase `SqlException`:

```
Imports System.Data.SqlClient
```

Al presionar el botón inicializamos los tres parámetros del comando SQL que hemos configurado en el `SqlDataSource`. Para acceder a dichos parámetros disponemos de la propiedad `InsertParameters` que le indicamos como subíndice el nombre del parámetro.

Lo que difiere de la clase anterior es la inicialización del parámetro `codigotitulo` con el valor que rescatamos de la propiedad `SelectedValue` del control `DropDownList1`

Luego de inicializar los tres parámetros procedemos a llamar al método `Insert` de la clase `SqlDataSource`.

Todo esto lo hacemos capturando la excepción `SqlException`.

Consultas.

Seleccionamos del Explorador de soluciones la página `consultaarticulos.aspx` y procedemos a elaborar la siguiente interface visual (disponemos un `TextBox`, un `Button`, una `Label`, un `HyperLink` y un `SqlDataSource`):



Configuramos el `SqlDataSource1`, luego de seleccionarlo con el mouse modificamos la propiedad `ConnectionString` (con la conexión creada anteriormente) y en el diálogo que abre la propiedad `SelectQuery` procedemos a codificar el comando select:

```

select ar.descripcion as descriarticulo,
       precio,
       ti.descripcion as descriptitulo
  from articulos as ar
 join titulos as ti on ti.codigo=ar.codigotitulo
 where ar.codigo=@codigo

```

Es decir recuperamos la descripción del artículo, su precio y la descripción del rubro de la otra tabla (recordar de presionar el botón “Actualizar parámetros”)

El código del evento click es:

```
Imports System.Data.SqlClient

Partial Class consultaarticulos
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(ByVal sender As Object,
        ByVal e As System.EventArgs) Handles Button1.Click
        Try
            Me.SqlDataSource1.SelectParameters("codigo").DefaultValue = _
                Me.TextBox1.Text
            Me.SqlDataSource1.DataSourceMode = _
                SqlDataSourceMode.DataReader
            Dim datos As SqlDataReader
            datos = _SqlDataSource1.Select(DataSourceSelectArguments.Empty)
            If datos.Read Then
                Me.Label1.Text = "Descripción:" & _
                    datos("descriarticulo") & "<br>" & _
                    "Precio:" & datos("precio") & "<br>" & _
                    "Rubro:" & datos("descrirubro")
            Else
                Me.Label1.Text = "No existe un artículo con dicho código"
            End If
        Catch ex As Exception
            Me.Label1.Text = ex.Message
        End Try
    End Sub
End Class
```

Para poder recuperar los datos mediante un objeto de la clase SqlDataReader debemos configurar el SqlDataSource indicando a la propiedad DataSourceMode el siguiente valor:

```
Me.SqlDataSource1.DataSourceMode = SqlDataSourceMode.DataReader
```

Ahora cuando llamemos al método select del SqlDataSource lo hacemos con la siguiente sintaxis:

```
Dim datos As SqlDataReader
datos = Me.SqlDataSource1.Select(DataSourceSelectArguments.Empty)
```

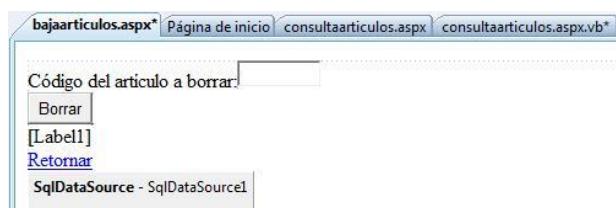
Luego recorremos el SqlDataReader (como este caso puede retornar cero o una fila lo hacemos mediante un if):

```
If datos.Read Then
```

Si el método Read retorna true luego podemos acceder a la fila recuperada con el select.

Bajas.

Seleccionamos del Explorador de soluciones la página bajaarticulos.aspx y procedemos a elaborar la siguiente interface visual:



Inicializamos la propiedad ConnectionString del SqlDataSource1 con la conexión que habíamos creado e inicializamos la propiedad DeleteQuery.

Luego el código a ejecutar cuando se presiona el botón “Borrar” de la página es:

```
Imports System.Data.SqlClient
Partial Class bajaarticulos
    Inherits System.Web.UI.Page
```

```

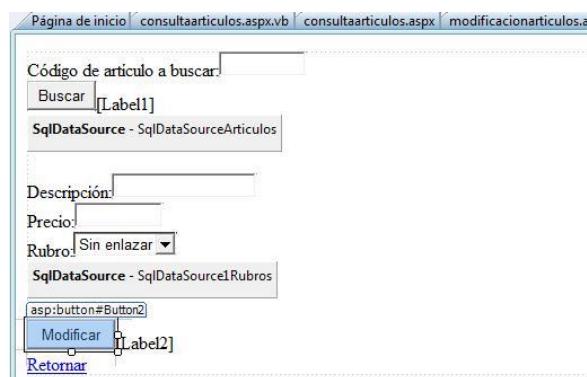
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
    Try
        Me.SqlDataSource1.DeleteParameters("codigo").DefaultValue = _
            Me.TextBox1.Text
        Dim cant As Integer
        cant = Me.SqlDataSource1.Delete
        If cant = 1 Then
            Me.Label1.Text = "Se borró el artículo"
        Else
            Me.Label1.Text = "No existe el código"
        End If
    Catch ex As Exception
        Me.Label1.Text = ex.Message
    End Try
End Sub
End Class

```

Procedemos a inicializar el parámetro y luego llamamos al método Delete del SqlDataSource. El método Delete retorna un entero y representa la cantidad de filas borradas de la tabla.

Modificaciones.

Por último implementaremos la modificación de datos. Seleccionamos del Explorador de soluciones la página modificacionarticulos.aspx y procedemos a elaborar la siguiente interface visual:



Para efectuar la modificación de datos de un artículo procederemos primero a la búsqueda de los datos actuales. Esto hace que necesitemos un primer control de tipo SqlDataSource (que llamamos SqlDataSourceArticulos).

El primer SqlDataSource nos permite consultar los datos de un artículo por su código y mostrar el resultado en dos controles de tipo TextBox y en un control de tipo DropDownList el rubro que pertenece.

Para inicializar el DropDownList debemos crear un segundo SqlDataSource (SqlDataSourceTitulos)

El SqlDataSourceArticulos debemos configurar las propiedades:

- ConnectionString.
- SelectQuery

```
select * from articulos where codigo=@codigo
```

- UpdateQuery

```
update articulos set descripcion=@descripcion, precio=@precio,
codigotitulo=@codigotitulo where codigo=@codigo
```

El SqlDataSourceTitulos debemos configurar las propiedades:

- ConnectionString.
- SelectQuery

```
select * from titulos
```

Luego el código para los eventos click de los dos botones es:

```

Imports System.Data.SqlClient
Partial Class modificacionarticulos
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles Button1.Click
        Try
            Me.SqlDataSourceArticulos.SelectParameters("codigo"). _
                DefaultValue = Me.TextBox1.Text
            Me.SqlDataSourceArticulos.DataSourceMode = _
                SqlDataSourceMode.DataReader
            Dim datos As SqlDataReader
            datos = SqlDataSourceArticulos. _
                Select(DataSourceSelectArguments.Empty)
            If datos.Read Then
                Me.TextBox2.Text = datos("descripcion")
                Me.TextBox3.Text = datos("precio")
                Me.DropDownList1.DataSource = Me.SqlDataSourceTitulos
                Me.DropDownList1.DataTextField = "descripcion"
                Me.DropDownList1.DataValueField = "codigo"
                Me.DropDownList1.SelectedValue = datos("codigotitulo")
                Me.DropDownList1.DataBind()
            Else
                Me.Label1.Text = "No existe un articulo con dicho código"
            End If
        Catch ex As Exception
            Me.Label1.Text = ex.Message
        End Try
    End Sub

    Protected Sub Button2_Click(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles Button2.Click
        Try
            Me.SqlDataSourceArticulos.UpdateParameters("descripcion") _
                .DefaultValue = Me.TextBox2.Text
            Me.SqlDataSourceArticulos.UpdateParameters("precio") _
                .DefaultValue = Me.TextBox3.Text
            Me.SqlDataSourceArticulos.UpdateParameters("codigorubro") _
                .DefaultValue = Me.DropDownList1.SelectedValue
            Me.SqlDataSourceArticulos.UpdateParameters("codigo") _
                .DefaultValue = Me.TextBox1.Text
            Dim cant As Integer
            cant = Me.SqlDataSourceArticulos.Update
            If cant = 1 Then
                Me.Label2.Text = "Datos modificados"
            Else
                Me.Label2.Text = "No existe un artículo con dicho codigo"
            End If
        Catch ex As Exception
            Me.Label2.Text = ex.Message
        End Try
    End Sub
End Class

```

	<u>UNIDAD Nº 2:</u> Sitios web con Visual Studio 2019	<u>TEMAS:</u> Control GridView.	Clase 6
--	--	--	------------------------------

Objetivos:

- Utilizar el control GridView para visualizar datos.
- Edición, modificación y consulta de datos a partir de un GridView.
- Configuración del control GridView.

Introducción

El control GridView del ASP.Net permite visualizar datos en una tabla en pantalla, editar, modificar y borrar registros del mismo.

El GridView es un control extremadamente flexible para mostrar tablas multicolumna.

Cada registro de una consulta del un select configurado en un SqlDataSource genera una fila en la grilla. Cada campo en el registro representa una columna en la grilla.

El GridView es el control más poderoso que provee el ASP.Net. Veremos que este control trae funcionalidades ya implementadas para paginación, ordenamiento y edición de sus datos.

GridView

Datos de una tabla

Crearemos un proyecto para probar el control GridView y las diferentes opciones que nos brinda Un formulario Menu un formulario Datos ,un formulario Varias ,un formulario Selección.

Luego de crear el proyecto iremos al Explorador de servidores y seleccionaremos la tabla “titulos” y la arrastraremos al formulario web. Veremos que se generan dos objetos sobre la página:

Un objeto de la clase GridView llamado GridView1.

Un objeto de la clase SqlDataSource llamado SqlDataSource1.

Si seleccionamos el objeto SqlDataSource1 y observamos el contenido de la propiedad SelectQuery, veremos que ya está configurado el comando SELECT:

```
SELECT [codigo], [descripcion] FROM [titulos]
```

El comando SELECT indica rescatar todas las filas de la tabla titulos.

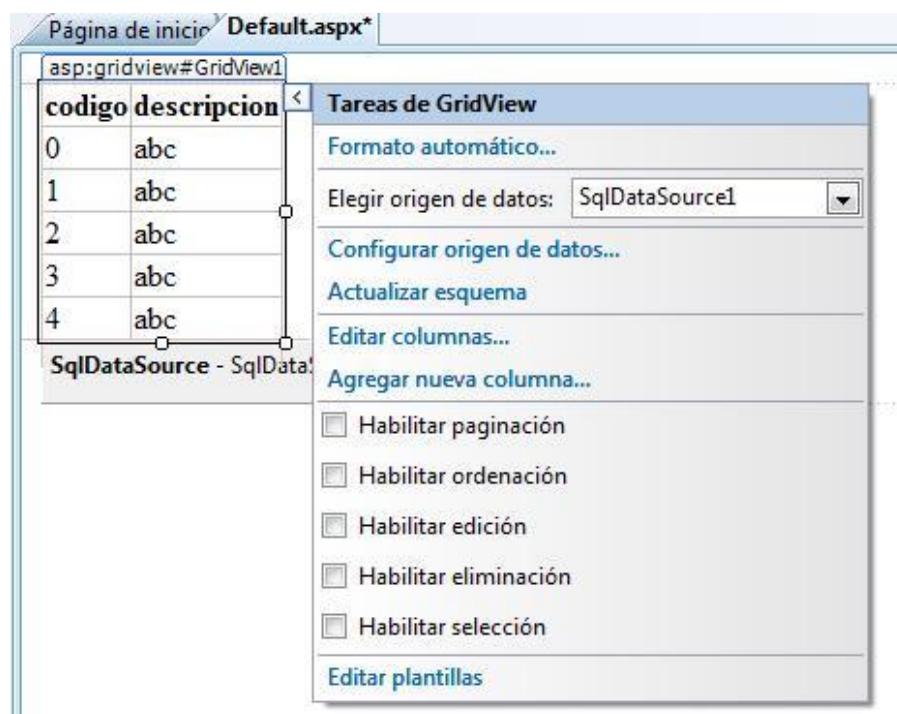
Podemos ver también que se han configurado automáticamente las propiedades **InsertQuery**, **DeleteQuery** y **UpdateQuery** con los valores:

```
INSERT INTO [titulos] ([descripcion]) VALUES (@descripcion)
```

```
DELETE FROM [titulos] WHERE [codigo] = @codigo
```

```
UPDATE [titulos] SET [descripcion] = @descripcion WHERE [codigo] = @codigo
```

Como podemos ver hasta este momento la herramienta Visual Studio .Net nos ha configurado en forma automática el control SqlDataSource1, solo nos queda configurar el control GridView1. Seleccionamos el control GridView y presionamos el botón presente en la parte superior derecha, el mismo nos muestra una serie de funcionalidades básicas del control:



Como podemos ver ya está configurado el origen de datos con el objeto SqlDataSource1. Habilitemos la paginación, ordenamiento, edición y eliminación.

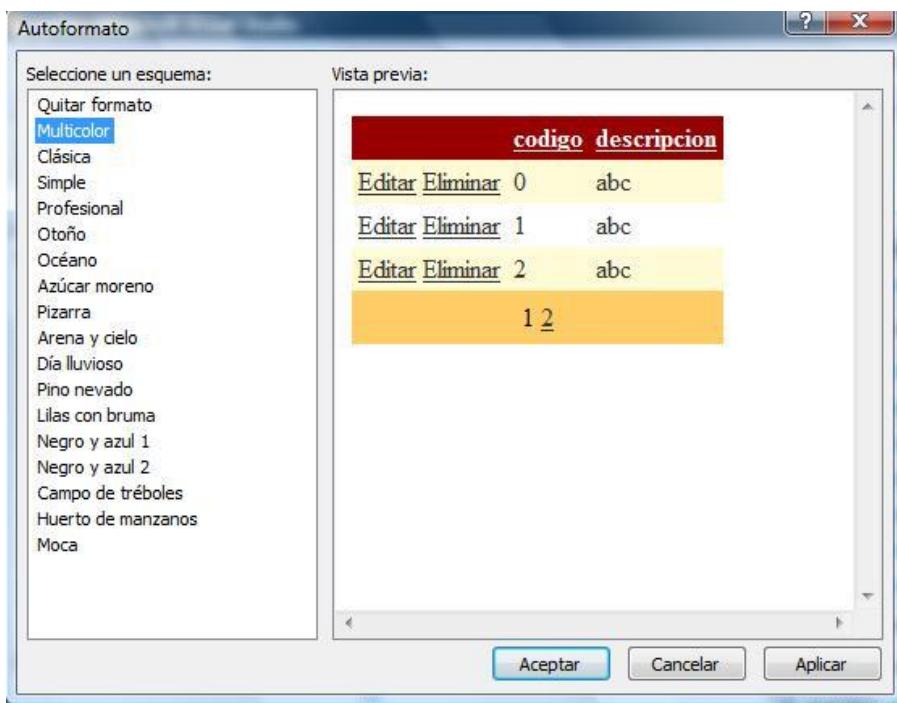
Ejecutemos el proyecto y comprobaremos que tenemos en la página los datos de la tabla “titulos” con la capacidad de modificar y borrar registros. Además está activa la paginación y ordenamiento por cualquiera de las dos columnas de la tabla.

	codigo	descripcion
Editar	Eliminar	1 frutas
Editar	Eliminar	2 verduras
Editar	Eliminar	3 carnes

1 [2](#) [3](#)

Sin escribir una sola línea de código tenemos el mantenimiento de la tabla titulos (con la excepción del alta)

Veamos otras características que podemos configurar en forma visual del control GridView. Desde el botón “>” que se encuentra en la parte superior derecha del control GridView1 podemos seleccionar la opción “Formato Automático...” para definir la presentación de la tabla con plantillas predefinidas de color y fuente:



Luego de seleccionar el esquema para el GridView, presionamos aceptar y tenemos ya definido el nuevo formato de la grilla.

Desde la ventana de propiedades configuraremos las propiedades:

Caption: Es un título que aparece en la parte superior del GridView.

PageSize: Cantidad de registros a mostrar por página.

Luego de esto ejecutamos nuevamente y tenemos como resultado:

Mantenimiento de Rubros	
codigo	descripcion
Editar Eliminar 2	verdurass
Editar Eliminar 3	carnes
Editar Eliminar 4	galletas
Editar Eliminar 5	dulces

Datos de varias tablas.

Continuamos con las tablas:

Tabla: articulos

codigo	int	Clave primaria e identidad.
descripcion	varchar(50)	
precio	float	
codigotitulo	int	

Tabla: titulos

Codigo	int	Clave primaria e identidad
descripción	varchar(50)	

Generamos un nuevo webform y seleccionamos desde el Explorador de servidores la tabla articulos y la disponemos dentro del webform. El entorno del Visual Studio .Net nos genera un objeto de la clase GridView y otro de la clase SqlDataSource.

El objetivo final es mostrar el código del artículo, su descripción, su precio y finalmente la descripción del título (no el código de título)

1 - Primero seleccionamos el control SqlDataSource1 y configuramos la propiedad SelectQuery con el comando Select que rescata los datos haciendo el emparejamiento por la columna codigotitulo de la tabla articulos y codigo de la tabla titulos:

```
SELECT ar.codigo, ar.descripcion as descriarticulo, precio, ti.descripcion as
descrititulo
from articulos as ar
join titulos as ti on ti.codigo=ar.codigotitulo
```

2 - Luego de configurar la propiedad SelectQuery debemos actualizar el esquema del SqlDataSource1, esto lo hacemos seleccionando el objeto sobre el formulario y seleccionamos la opción “Actualizar esquema”. Con esto logramos que se refresque las columnas a mostrar en el GridView1.

Si ejecutamos podemos ver que ya tenemos la tabla que rescata todos los artículos y asociado a cada artículo la descripción del rubro al que pertenece:

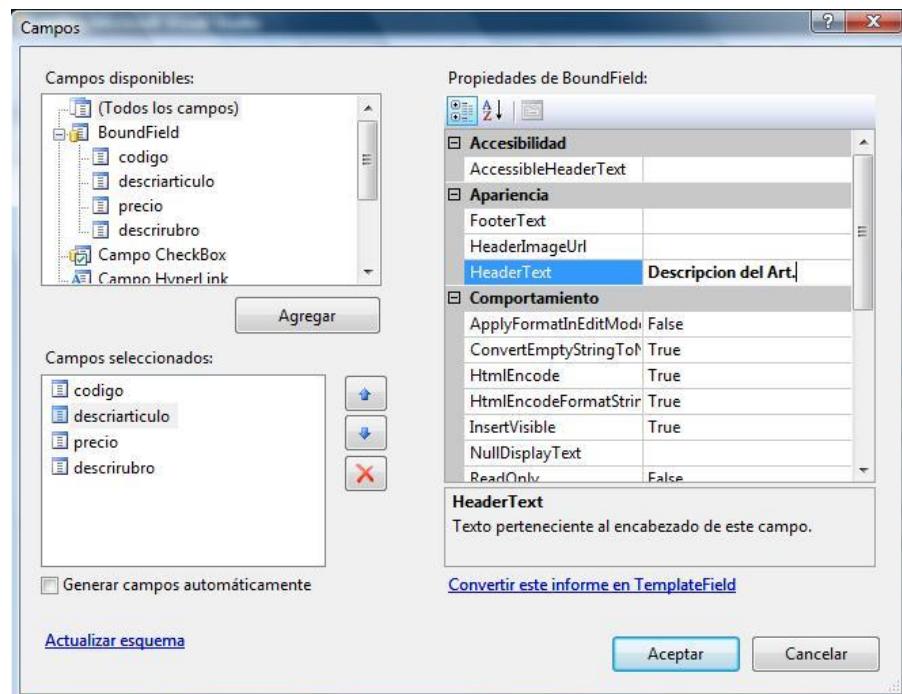
The screenshot shows a web browser window with the URL <http://localhost:49276/gr...>. Inside the browser, a GridView control is displayed, showing the following data:

codigo	descriarticulo	precio	descrirubro
1	papas	4	verduras
3	tomates	12	verduras
4	naranjas	2	frutas
5	frutillas	3	frutas
6	mandarinas	3.4	frutas

Ahora configuraremos algunas propiedades para dar un formato y presentación más adecuado de los datos:

A – Ya vimos que seleccionando el GridView1 y mediante la opción “Formato automático...” podemos definir el estilo de presentación de la grilla.

B – Ahora entramos al la opción “Editar columnas...” y seleccionamos el campo a configurar:



Cambiemos los títulos de las columnas de cada campo (por ejemplo en el campo descriarticulo mostraremos el título “Descripción del Artículo.” Modificando la propiedad HeaderText. De forma similar cambiar los otros títulos de las columnas de la grilla:

codigo	Descripción del Art.	precio	rubro
1	papas	4	verduras
3	tomates	12	verduras
4	naranjas	2	frutas
5	frutillas	3	frutas
6	mandarinas	3.4	frutas

C – La propiedad Visible de cada columna permite configurar si la columna se muestra o no.

D – La propiedad DataFormatString nos permite configurar la apariencia de números y fechas. Por ejemplo si queremos que el precio aparezca con el símbolo de moneda debemos configurar la propiedad DataFormatString con el valor:
{0:C}

codigo	Descripción del Art.	precio	rubro
1	papas	\$7.00	verduras
3	tomates	\$12.00	verduras
4	naranjas	\$2.00	frutas
5	frutillas	\$3.00	frutas
6	mandarinas	\$3.40	frutas

Algunos ejemplos de formato de campos:

Tipo	Formato	Ejemplo
Moneda	{0:C}	\$120.50
Porcentaje	{0:P}	40%
Decimales fijos	{0:F4}	12.4567
Notación científica	{0:E}	1.233E003
Fecha corta	{0:d}	12/25/2005

Formato de filas individuales de acuerdo a ciertas condiciones.

Cuando se grafica la tabla podemos capturar el evento RowDataBound y configurar como graficar dicha fila de la tabla.

A modo de ejemplo mostraremos de color amarillo las filas de los artículos con precio superior a 5. Para esto codificamos el evento RowDataBound del GridView1:

```
Protected Sub GridView1_RowDataBound(ByVal sender As Object, _  
    ByVal e As System.Web.UI.WebControls.GridViewRowEventArgs) _  
Handles GridView1.RowDataBound  
  
    If e.Row.RowType = DataControlRowType.DataRow Then  
        Dim precio As Single  
        precio = DataBinder.Eval(e.Row.DataItem, "precio")  
        If precio > 5 Then  
            e.Row.ForeColor = System.Drawing.Color.Red  
            e.Row.BackColor = System.Drawing.Color.Yellow  
            e.Row.Font.Bold = True  
        End If  
    End If  
  
End Sub
```

Con el if verificamos si el evento se disparó para una fila de datos de la grilla (ya que este método se dispara cuando dibuja la cabecera (DataControlRowType.Header), el pie de grilla (DataControlRowType.Footer) etc.

Luego rescatamos el valor del campo precio y verificamos con un nuevo if si el precio supera 5, en caso afirmativo modificamos el color de fondo (BackColor) y de frente de la fila.



The screenshot shows a web browser window with the URL 'http://localhost...'. The page displays a GridView containing five rows of data. The columns are labeled 'codigo', 'articulo', 'precio', and 'rubro'. The data is as follows:

codigo	articulo	precio	rubro
1	papas	3	verduras
3	tomates	12	verduras
4	naranjas	2	frutas
5	frutillas	3	frutas
6	mandarinas	6	frutas

Selección de una fila del GridView y posterior extracción de sus datos.

En muchas situaciones es necesario que el usuario seleccione una fila de la grilla para reflejar dicho dato en otra parte de la página o hacer otra consulta.

Para poder implementar esta característica del GridView llevaremos a cabo los siguientes pasos:

- 1 – Cambiaremos el valor de la propiedad SelectedRowStyle.BackColor por amarillo (es decir que cuando seleccionemos la fila el color de fondo de la misma se activará con este valor)
- 2 – En el menú de opciones que se despliega en la parte derecha del GridView1 activaremos el CheckBox “Habilitar selección”

3 – Dispondremos una Label en el webform para mostrar el valor seleccionado de la grilla (solo a modo de ejemplo)

4 – Para el evento SelectedIndexChanged del GridView1 codificaremos el siguiente código:

```
Protected Sub GridView1_SelectedIndexChanged(ByVal sender As Object,  
    ByVal e As System.EventArgs) Handles GridView1.SelectedIndexChanged  
    Me.Label1.Text = Me.GridView1.Rows(Me.GridView1.SelectedIndex).Cells(1).Text  
End Sub
```

El objeto Rows del GridView almacena una colección de filas, mediante el valor devuelto por la propiedad SelectedIndex de la grilla podemos acceder a la celda que almacena el código del artículo.



Esta información nos es muy útil para mostrar información adicional sobre el registro en otro control por ejemplo.

	UNIDAD Nº :3 Recursos de ASP.Net.	TEMAS: Upload de archivos.	Clase 7
--	--	---	------------------------------

Objetivos:

- Subir archivos al servidor.
- Validar la existencia de archivos con el mismo nombre.
- Controlar propiedades del archivo.

Introducción

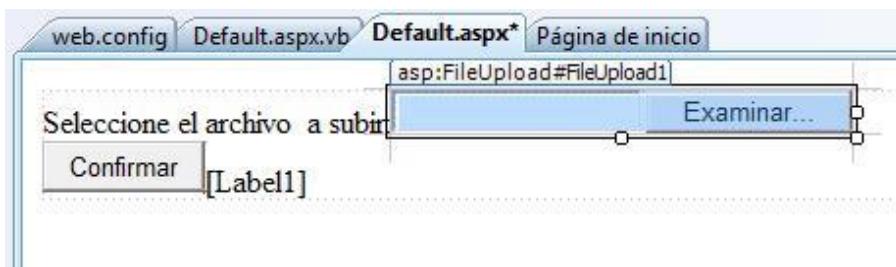
Una actividad muy común en un sitio web es el envío de archivos desde el cliente y su almacenamiento en el servidor.

Upload

1 - Componente FileUpload

La componente FileUpload encapsula el envío y recepción de un archivo en el servidor web. Confeccionaremos una serie de páginas web para aprender a utilizar los métodos y propiedades de la clase FileUpload.

Crear un webform e implementar la siguiente interface:



Disponemos en el webform un objeto de la clase FileUpload que se encuentra en la pestaña de componentes "Estándar"

Para el evento clic de botón confirmar implementamos el siguiente código:

```
Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(ByVal sender As Object _
        , ByVal e As System.EventArgs) Handles Button1.Click
        Me.FileUpload1.SaveAs(Server.MapPath(".") & "\\" & _
            Me.FileUpload1.FileName)
        Me.Label1.Text = "Archivo subido"
    End Sub
End Class
```

El método SaveAs permite grabar el archivo que se subió al servidor. Debemos indicarle el camino del directorio donde se almacena y el nombre del archivo. Para obtener el path donde se almacena la página ASPX actual el objeto Server tiene el método MapPath y para obtener el nombre del archivo del objeto FileUpload1 tiene una propiedad llamada FileName.

Con esta única línea tenemos subido en el servidor el archivo que se envió desde el navegador.

2 – Almacenar el archivo en un subdirectorio.

Es una buena costumbre evitar almacenar los archivos que suben los usuarios en la misma carpeta donde se encuentran la páginas dinámicas ASPX.

Para almacenar los archivos en otro directorio primero debemos crear el directorio (por ejemplo creamos una carpeta imágenes en el directorio donde se aloja el sitio)

Creamos un nuevo webform (Default2.aspx) y creamos el siguiente código para el evento clic del objeto Button:

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
    Me.FileUpload1.SaveAs(Server.MapPath(".") & "\imagenes\" & Me.FileUpload1.FileName)
    Me.Label1.Text = "Archivo subido"
End Sub
```

La carpeta imágenes debemos crearla en forma manual desde el administrador de archivos del sistema operativo. Luego cada archivo que se suba al servidor se almacenará en dicha carpeta.

3 – Mostrar propiedades del archivo subido.

Crearemos un tercer webform para ver como accedemos a distintas propiedades de la clase FileUpload.

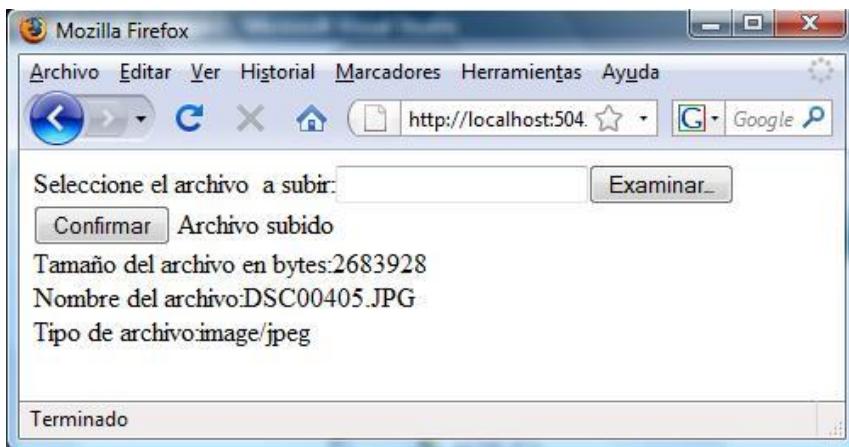
Disponemos sobre el webform un objeto de la clase FileUpload, un Button y cuatro Label.

Mostraremos el tamaño del archivo en bytes. El nombre del archivo y finalmente el tipo de archivo.

El código fuente para el evento clic es:

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
    Me.FileUpload1.SaveAs(Server.MapPath(".") & "\\" & Me.FileUpload1.FileName)
    Me.Label1.Text = "Archivo subido"
    Me.Label2.Text = Me.FileUpload1.PostedFile.ContentLength
    Me.Label3.Text = Me.FileUpload1.FileName
    Me.Label4.Text = Me.FileUpload1.PostedFile.ContentType
End Sub
```

La propiedad PostedFile del control FileUpload almacena en:
ContentLength (el tamaño del archivo en bytes)
FileName (El nombre del archivo dado en el cliente)
ContentType (El tipo de archivo)



El tamaño del archivo nos puede ser útil si queremos limitar el peso del mismo.

4 – Validar la existencia de otro archivo con el mismo nombre.

Puede suceder que en la carpeta donde se almacena el archivo exista otro con el mismo nombre. Esto conlleva a que se pise el archivo antiguo por el nuevo.

Veamos otro webform donde validamos que no exista otro archivo con el mismo nombre, en caso afirmativo no permitimos su almacenamiento e informamos de tal situación:

```
Imports System.IO
Partial Class Default4
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(ByVal sender As Object,
        ByVal e As System.EventArgs) Handles Button1.Click
        If File.Exists(Me.Server.MapPath(".") & "/" &
            Me.FileUpload1.FileName) Then
            Me.Label1.Text = "Hay archivo con ese nombre"
        Else
            Me.FileUpload1.SaveAs(Server.MapPath(".") & "\\" &
                Me.FileUpload1.FileName)
            Me.Label1.Text = "Archivo almacenado."
        End If
    End Sub
End Class
```

El método estático Exists retorna true si ya existe un archivo con el nombre que le pasamos como parámetro, en caso negativo retorna false

En caso que no exista el archivo procedemos a efectuar la registración del mismo en la carpeta de nuestro proyecto.

Ejercicios Propuestos

1 – Subir una imagen al servidor y luego mostrarla mediante el control Image. No permitir cargarla en caso que haya otra con el mismo nombre.

2 – Se tienen las tablas:

autos (#patente char(6), propietario varchar(50), precio float, codigomarca int, foto varchar(10),

modelo int)

marcas (#codigo int identity, descripcion varchar(30))

a – Confeccionar el mantenimiento de la tabla marcas (altas, bajas y modificaciones – utilizar un GridView)

b – Confeccionar una página para efectuar el alta de autos (validar que se ingresen datos en todos los controles).

c – Consulta de un auto ingresando su patente (mostrar todos los datos, incluido la foto)

d – Seleccionar de un DropDownList una marca y luego mostrar todos los autos de dicha marca.

e – Implementar el borrado de un auto ingresando su patente.

f – Ingresar un rango de años y luego mostrar todos los autos en dicho rango.

	<u>UNIDAD Nº :3</u> Recursos de ASP.Net.	<u>TEMAS:</u> Controles de validación.	Clase 8
--	--	--	-----------------------

Objetivos:

- Identificar los controles de validación propuestos en ASP.Net.
- Elegir el más adecuado para cada problema.
- Crear algoritmos de validación en el servidor.

Introducción

Hay seis controles Web para la validación de datos de entrada que se pueden incorporar en un Formulario Web.

RequiredFieldValidator: Facilita la validación de un dato del formulario chequeando que el mismo tenga algún valor.

RangeValidator: Facilita la validación de un dato del formulario contra un valor mínimo y máximo.

CompareValidator: Facilita la validación de un dato del formulario contra un valor fijo u otro campo del formulario.

CustomValidator: Facilita la validación de un dato del formulario usando una subrutina propia.

RegularExpressionValidator: Facilita la validación de un dato del formulario contra una expresión.

ValidationSummary: Agrupa los mensajes de error de otros controles en una parte de la página.

Todos los controles de validación tienen tres propiedades fundamentales: **ControlToValidate**, **Text** y **IsValid**. Todos los controles derivan de la clase **BaseValidator**.

La propiedad **ControlToValidate** contiene la referencia del control del formulario que queremos validar.

La propiedad **Text** almacena el mensaje de error que queremos que se muestre en la página.

Por último la propiedad **IsValid** almacena True en caso que el control pase el test de validación.

Cuando empleamos controles de validación con el Explorer 4.0 o superior, los controles automáticamente usan funciones en JavaScript en el cliente. Esto significa que los controles pueden inmediatamente mostrar los mensajes de error en el browser mientras el usuario está completando el formulario. Si hay algún error en la página el código JavaScript previene que el usuario pueda enviar los datos al servidor.

En caso de emplear navegadores más antiguos los controles que veremos seguirán funcionando, pero la validación se realizará en el servidor.

Controles de validación

Control: *RequiredFieldValidator*

Para probar este control haremos una página que solicite el nombre de usuario (mostraremos un error si el operador no ingresa texto en el TextBox).

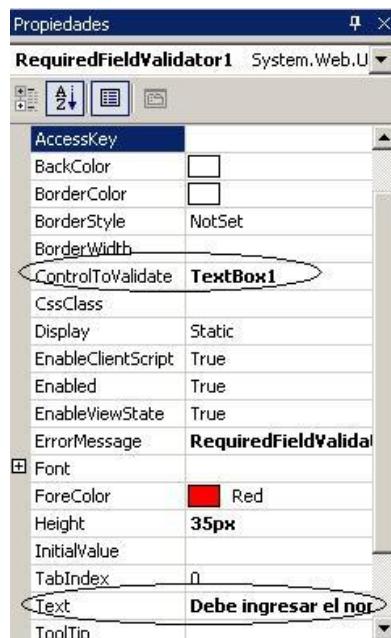
La interface visual es la siguiente:



El mensaje en rojo debe aparecer si presionamos el botón “Confirmar” y no se ingresó texto en el TextBox.

En el Visual Studio .Net confeccionamos el formulario web disponiendo uno control de tipo TextBox, un Button y un RequiredFieldValidator que se encuentra en la pestaña “Validación” del “Cuadro de herramientas”.

El control RequiredFieldValidator es importante inicializar las siguientes propiedades:



Cuando ejecutemos la página podemos ver el código que llega al navegador (en ella veremos las funciones en JavaScript que automáticamente el ASP.NET nos crea para facilitar la validación).

El código HTML completo de la página es el siguiente:

```
<%@ 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>
</head>
<body>
    <form id="form1" runat="server">
        <div>

            Ingrese nombre de usuario:<asp:TextBox ID="TextBox1"
runat="server"></asp:TextBox>
                <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat="server"
                    ControlToValidate="TextBox1"
                    ErrorMessage="RequiredFieldValidator">Debe ingresar el nombre de
usuario.</asp:RequiredFieldValidator>
                <br />
                <asp:Button ID="Button1" runat="server" Text="Confirmar" />

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

Como sabemos este código HTML se genera en forma automática cuando creamos cada control y configuramos sus propiedades.

Luego si queremos que al presionar el botón se redireccione a otra página en caso de haber ingresado un nombre de usuario debemos codificar el método Clic para dicho botón:

```
Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(ByVal sender As Object,
        ByVal e As System.EventArgs) Handles Button1.Click

        If Me.IsValid Then
            Me.Response.Redirect("Default2.aspx")
        End If

    End Sub
End Class
```

La propiedad IsValid del WebForm almacena true si todos los controles de validación dispuestos en el formulario se validan correctamente. Es decir en este problemas si se ingresa algún carácter en el control TextBox luego se puede pasar a la página "Default2.aspx".

Control: RangeValidator

El control RangeValidator especifica un valor mínimo y máximo para un control TextBox. Podemos utilizar el control para chequear el rango de enteros, fechas, cadenas o valores reales.

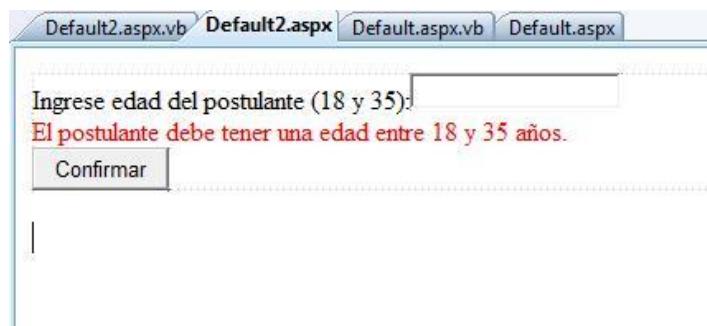
Las propiedades más importantes del control son:

- ControlToValidate El campo del formulario a validar.
- MinimumValue El valor mínimo a validar en el rango de valores.
- MaximumValue El valor máximo a validar en el rango de valores.
- Text El mensaje de error a mostrar.
- Type El tipo de comparación a ejecutar (valores posibles: String, Integer, Double, Date, Currency).

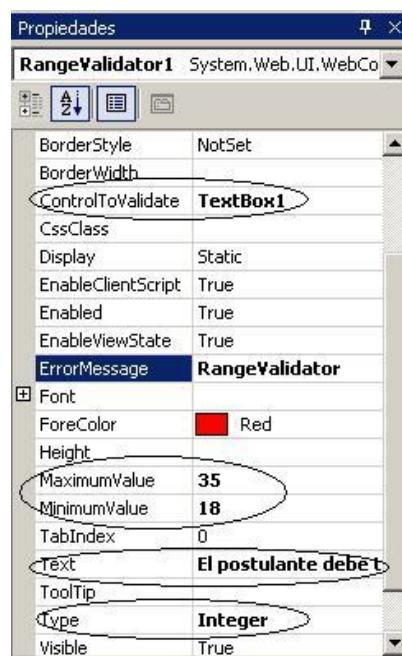
Para probar este control haremos una página que solicite ingresar la edad de una persona que se postula para un trabajo (la misma debe estar en el rango de 18 a 35 años).

Disponemos sobre el formulario los siguientes objetos: Label, TextBox, Button y un RangeValidator.

La interface que debemos implementar es la siguiente:



El objeto RangeValidator lo debemos configurar con los siguientes valores:



Si ejecutamos la página veremos que el mensaje aparece si ingresamos una edad que esté fuera del rango de 18 a 35 años.

El código a implementar al presionar el botón confirmar es el siguiente:

```
Partial Class Default2
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(ByVal sender As Object, _
        ByVal e As System.EventArgs) Handles Button1.Click

        If Me.IsValid Then
            Me.Response.Redirect("Default3.aspx")
        End If

    End Sub
End Class
```

Es decir redireccionamos a la próxima página en caso que todos los controles de validación del formulario se verifiquen correctos (en este problema solo tenemos un control de tipo RangeValidator, pero en muchos casos veremos que en un formulario puede haber más de un control de validación)

Si quisiéramos solo validar un control determinado del WebForm la condición sería:

```
If Me.RangeValidator1.IsValid Then
    Me.Response.Redirect("Default3.aspx")
End If
```

Es decir verificamos la propiedad IsValid del control RangeValidator (si tenemos un solo control en el formulario preguntar por la propiedad IsValid del webform o del RangeValidator el resultado será idéntico).

Control: CompareValidator

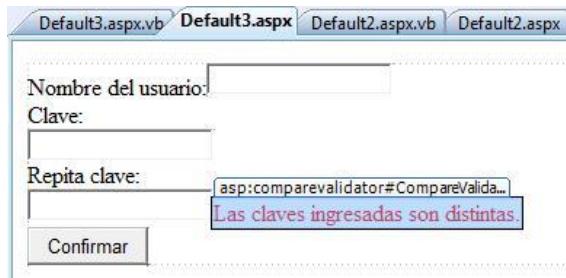
El control CompareValidator permite comparar un valor de un control con otro control o comparar el valor de un control con un valor fijo.

Las propiedades más importantes son:

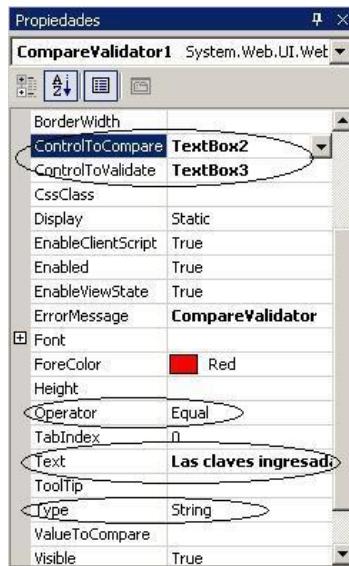
- ControlToValidate El campo del formulario a validar.
- ControlToCompare El campo del formulario contra el cual se efectúa la comparación.
- Operator El operador a utilizarse en la comparación (los valores posibles son Equal, NotEqual, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual y DataTypeCheck).
- Text El mensaje de error a mostrar.
- Type El tipo de comparación a ejecutar (valores posibles String, Integer, Double, Date, Currency).
- ValueToCompare El valor fijo a comparar.

Para probar este control implementaremos una página que realizaría el alta de la tabla usuarios (debe permitir el ingreso del nombre de usuario y su clave, esta última dos veces, con el objetivo de asegurarse que la ingresó correctamente), emplearemos un objeto de la clase CompareValidator para validar el ingreso repetido de la clave.

La interface visual de la página es:



Al objeto CompareValidator le configuramos las propiedades de la siguiente manera:



Es importante inicializar la propiedad ControlToValidate con el objeto TextBox que carga la segunda clave, luego que el operador carga la clave se procede a validar si el texto ingresado coincide en el TextBox que hemos inicializado la propiedad ControlToCompare.

El código a implementar al presionar el botón "Confirmar":

```
Partial Class Default3
    Inherits System.Web.UI.Page

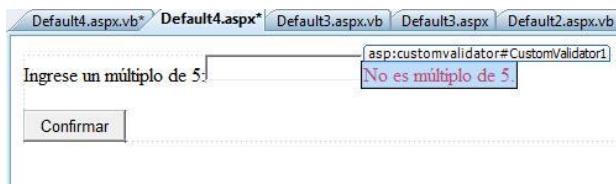
    Protected Sub Button1_Click(ByVal sender As Object,
        ByVal e As System.EventArgs) Handles Button1.Click
        If Me.IsValid Then
            Me.Response.Redirect("Default4.aspx")
        End If
    End Sub
End Class
```

Control: CustomValidator

El control CustomValidator permite validar el campo de un formulario con una función de validación propia. Debemos asociar nuestro control CustomValidator con un evento propio.

Para probar este control implementaremos una página que solicite el ingreso de un número múltiplo de 5, en caso de ingresar un valor incorrecto mostraremos un mensaje de error.

La interfaz a implementar es la siguiente:



Primero configuramos las propiedades del objeto CustomValidator con:



Ahora debemos codificar el evento ServerValidate que tiene el objeto CustomValidator1, a esto lo hacemos desde la ventana de código y lo podemos generar automáticamente haciendo doble clic sobre el control en la ventana de diseño:

```
Protected Sub CustomValidator1_ServerValidate(ByVal source As Object, ByVal args As System.Web.UI.WebControls.ServerValidateEventArgs) Handles CustomValidator1.ServerValidate
    Dim valor As Integer = args.Value
    If valor Mod 5 = 0 Then
        args.IsValid = True
    Else
        args.IsValid = False
    End If
End Sub
```

El parámetro args tiene dos propiedades fundamentales, una almacena el valor del control que estamos validando y otra llamada IsValid que debemos asignarle el resultado de nuestra validación.

En nuestro ejemplo almacenamos el número ingresado en la variable valor, luego mediante el operador Mod (resto de una división) verificamos si es cero, en caso afirmativo inicializamos la propiedad IsValid del objeto args con el valor True, en caso contrario, es decir que el número ingresado no sea un múltiplo de 5 almacenamos el valor False.

Cuando se presiona el botón confirmar tenemos:

```
Protected Sub Button1_Click(ByVal sender As Object, _  
    ByVal e As System.EventArgs) Handles Button1.Click  
    If Me.IsValid Then  
        Me.Response.Redirect("Default5.aspx")  
    End If  
End Sub
```

Control: RegularExpressionValidator

El control RegularExpressionValidator permite validar el valor de un campo de un formulario con un patrón específico, por ejemplo un código postal, un número telefónico, una dirección de mail, una URL etc.

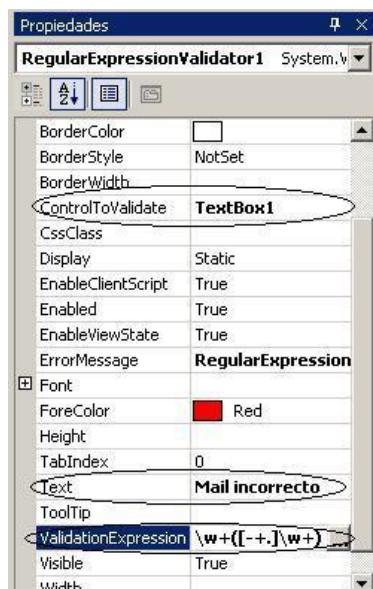
El planteo de una RegularExpression es bastante compleja, pero afortunadamente el Visual Studio .Net provee una serie de expresiones regulares preconfiguradas.

Para probar este control, haremos una página que solicite el ingreso de un mail y mostraremos un error en caso que el usuario ingrese un mail mal formado.

La interface visual de la página es la siguiente:



Al objeto de la clase RegularExpressionValidator le configuraremos las propiedades con los siguientes valores:



Si ejecutamos el programa podremos ver que al abandonar el foco del TextBox aparecerá el mensaje de error en caso de ingresar un mail incorrecto:

```
Partial Class Default5
    Inherits System.Web.UI.Page
    Protected Sub Button1_Click(ByVal sender As Object,
        ByVal e As System.EventArgs) Handles Button1.Click
        If Me.IsValid Then
            Me.Response.Redirect("Default6.aspx")
        End If
    End Sub
End Class
```

Control: ValidationSummary

Cuando tenemos formularios con gran cantidad de controles puede llegar a ser dificultoso ubicar los errores en la página. El Framework de la .Net trae otra clase llamada ValidationSummary que muestra todos los errores de la página agrupados en una parte de pantalla. Para utilizar el control ValidationSummary es necesario fijar el valor de la propiedad ErrorMessage para cada control de validación que tiene la página. Es importante no confundir la propiedad Text que aparece en la misma posición donde la disponemos con la propiedad ErrorMessage que contiene el mensaje de error que mostrará el control ValidationSummary.

Para probar este control haremos una página que solicite la carga del nombre de usuario y su clave en forma obligatoria (mostrando un mensaje de error en caso de dejar vacío los TextBox).

La interface de la página es la siguiente:

The screenshot shows a Windows application window for Microsoft Visual Studio. The title bar says "Default6.aspx". Below the title bar, there's a tab bar with four tabs: "Default6.aspx", "Default5.aspx.vb", "Default5.aspx", and "Default4.aspx.vb". The main area of the window displays a web form. The form has two text input fields: one for "Usuario" and one for "Clave", both marked with a red asterisk (*) indicating they are required fields. Below these fields is a "Confirmar" button. At the bottom of the page, there is a "ValidationSummary" control which lists two error messages: "• Mensaje de error 1." and "• Mensaje de error 2.", both displayed in red text.

Debemos disponer los siguientes objetos sobre el formulario:

1Button, 2 TextBox, 2 RequiredValidator y un objeto de la clase ValidationSummary.

La propiedad text de los objetos RequiredValidator las inicializamos con un (*) asterisco y las propiedades ErrorMessage con las cadenas: “Debe ingresar el nombre de usuario” y “Debe ingresar la clave” respectivamente. En cuanto al objeto de la clase ValidationSummary no debemos hacer ninguna configuración específica, solo ubicarlo dentro de la página.

	<u>UNIDAD Nº :3</u> Recursos de ASP.Net.	<u>TEMAS:</u> Cookies y Session.	Clase 9
--	---	---	------------------------------

Objetivos:

- Creación de Cookies, variables de sesión y aplicación.
- Recuperación de Cookies, variables de sesión.
- Usos de estas herramientas.

Introducción

Cuando un sitio web necesita identificar un usuario que visita un conjunto de páginas web puede emplear cookies y variables de sesión.

Como veremos las cookies se almacenan en el cliente (navegador) y son enviadas al servidor cada vez que le solicitamos una página a dicho servidor. En cambio las variables de sesión se almacenan en el servidor.

Cookies

El protocolo http es un protocolo desconectado. El protocolo http por si mismo no provee al servidor Web si dos peticiones provienen del mismo navegador. Desde este punto de vista, para el Servidor Web cada petición de página proviene de un nuevo usuario. Esta característica puede ser un inconveniente para ciertos sitios Web.

Netscape introdujo las cookies en su primera versión de navegador. Desde entonces, el World Wide Web Consortium (W3C) ha sumado las cookies al estandar. Muchos navegadores ahora pueden utilizar las cookies.

Las Cookies son pequeñas piezas de información que el servidor solicita que el navegador las registre en el equipo del cliente. De acuerdo a la especificación original de Netscape una cookie no puede contener más de 4 Kb (el tamaño se refiere a la combinación del nombre y valor de la cookie).

Otras son persistentes, a diferencia de las cookies de sesión, estas pueden sobrevivir meses o aún años. Los navegadores que soportan cookies mantienen uno o más archivos con la información de las cookies.

Como trabaja una Cookie.

Las Cookies son pasadas entre el navegador y el servidor a través de la cabecera HTTP.

El servidor primero crea la cookie usando la cabecera Set-Cookie en una respuesta; subsecuentemente, cada requerimiento del navegador retorna la cookie en la cabecera de la petición. Supongamos que queremos crear una cookie llamada color que contenga como valor la cadena rojo, el Servidor debería enviar una cabecera parecida a esto:

```
Set-Cookie: color=rojo; path=/; domain=issd.edu.ar;
expires=Tuesday, 01-Jan-11 00:00:01 GMT
```

El navegador agrega una cookie llamada color con el valor rojo. Además la información de la cookie deberá ser enviada al servidor cada vez que el navegador haga una petición de página.

El atributo domain define donde la cookie podrá enviarse. En este ejemplo la cookie se enviará al servidor cuando solicitemos una página en la dirección issd.edu.ar. La cookie no se enviará si hacemos una petición por ejemplo a la dirección google.com.ar o cualquier otro sitio de la Web de Internet.

Finalmente, el atributo expires especifica cuando la cookie deberá eliminarse.

Después que el servidor crea una cookie el navegador retorna la cookie en cada requerimiento que hacemos al sitio Web.

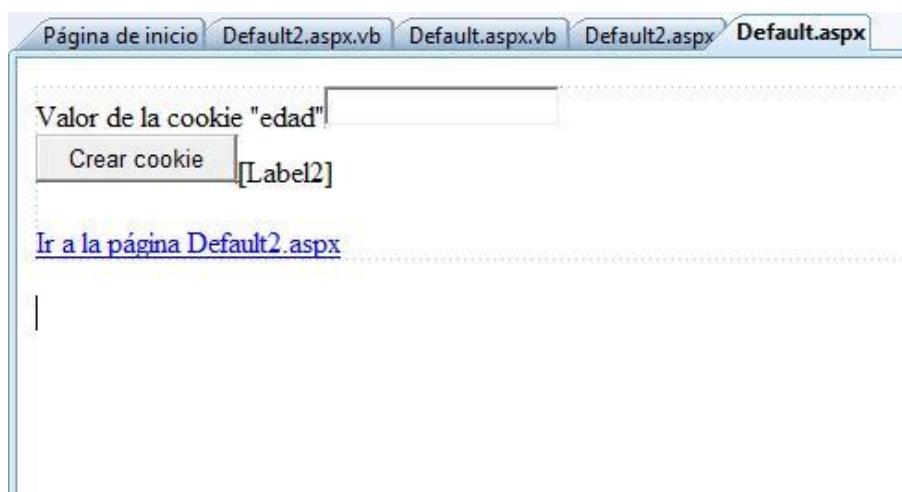
cookie persistente : Creación y acceso.

Una cookie persistente tiene una fecha de caducidad de la misma.

Cuidado: No podemos garantizar que la cookie existirá por el tiempo que hemos fijado al crearla, ya que el navegador puede en cualquier momento borrarla o inclusive el usuario del equipo cliente puede manualmente borrar las cookies.

Haremos dos páginas, en una cargaremos el valor de la cookie y en la segunda la imprimiremos.

La interface y código de la primer página es:

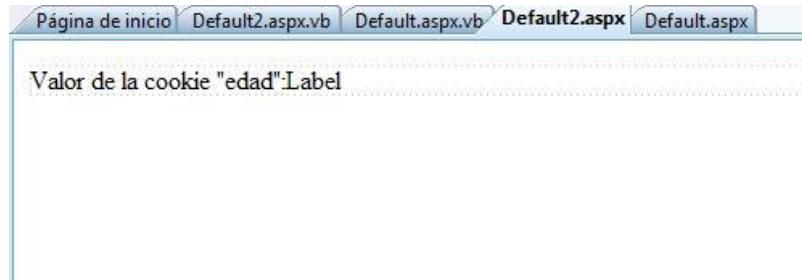


Para el evento Clic del botón “Crear cookie” escribimos el siguiente código:

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Button1.Click
    Dim cookie1 As New HttpCookie("edad", Me.TextBox1.Text)
    cookie1.Expires = #12/25/2015#
    Me.Response.Cookies.Add(cookie1)
    Me.Label2.Text = "Se creó la cookie"
End Sub
```

Creamos una cookie con el nombre “edad” y el valor almacenado en el TextBox1. Fijamos como fecha de caducidad de la cookie el 25 de diciembre de 2015, por último llamamos al método add del objeto Cookies.

La segunda página tiene por objetivo recuperar la cookie en caso que se haya creado previamente, la interface y código de la misma es:



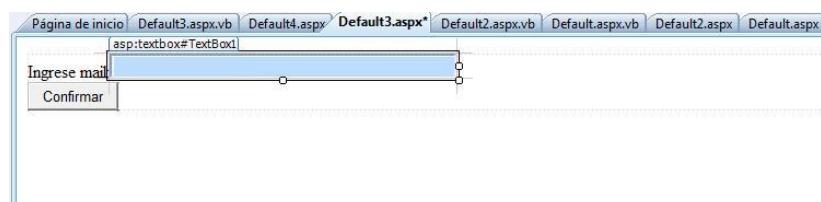
```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    If Me.Request.Cookies("edad") Is Nothing Then
        Me.Label2.Text = "No existe la cookie edad"
    Else
        Me.Label2.Text = Me.Request.Cookies("edad").Value
    End If
End Sub
```

En el evento Load de la página verificamos si existe la cookie, en caso de existir la recuperamos de la propiedad Cookies del objeto Request.

1 – Mostrar el último mail ingresado en un control TextBox.

El objeto del siguiente ejemplo es la creación de una página que solicite el ingreso del mail de una persona, si en otro momento ya lo había ingresado mostrarlo precargado en el control TextBox. Emplearemos una cookie persistente para almacenar el mail ingresado.

La página y el código es el siguiente:



```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim cookie1 As New HttpCookie("mail", Me.TextBox1.Text)
    cookie1.Expires = #12/25/2015#
    Me.Response.Cookies.Add(cookie1)
    Me.Response.Redirect("Default4.aspx")
End Sub

Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    If Me.IsPostBack = False Then
        If Me.Request.Cookies("mail") IsNot Nothing Then
            Me.TextBox1.Text = Me.Request.Cookies("mail").Value
        End If
    End If
End Sub
```

```
End If  
End If  
End Sub
```

En el método Load de la página verificamos si es la primera vez que pedimos la página (es decir que no se recarga por presionar el botón) y si existe la cookie, en dicho caso cargamos el contenido de la propiedad Text del control TextBox.

Luego de ingresar un mail en el control TextBox y presionar el botón “Confirmar” se crea una cookie llamada “mail” y almacena el valor ingresado en el control TextBox. Seguidamente redirecciona a la página “Default4.aspx”. Si Luego volvemos a ejecutar la página Default3.aspx veremos que el control TextBox aparece inicializado con el último mail ingresado (Aunque apaguemos y prendamos la máquina el último mail aparecerá dentro del control)

Como podemos ver una cookie es muy útil si queremos almacenar datos de configuración de un sitio para cada visitante.

cookie de sesión : Creación y acceso.

La diferencia de una cookie de sesión con una persistente es que las cookies de sesión permanecen mientras no cerramos la instancia del navegador, luego el código para crear una cookie de sesión es similar a las cookies persistentes con la salvedad que no debemos especificar fecha de expiración:

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles  
Button1.Click  
  
    Dim cookie1 As New HttpCookie("edad", Me.TextBox1.Text)  
    Me.Response.Cookies.Add(cookie1)  
    Me.Label2.Text = "Se creó la cookie"  
  
End Sub
```

Variables de Sesión

Una variable de sesión permite preservar el valor de una variable a través de una serie de páginas. Una variable de sesión se utiliza normalmente para almacenar una preferencia del usuario, un carrito de compras, información de seguridad del usuario, nombres de usuarios, password, etc.

Las variables de sesión son almacenadas durante el tiempo que el usuario visita el sitio Web. Cuando el servidor detecta que el usuario no hace más peticiones de páginas, la información almacenada de las variables de sesión es automáticamente destruida (por defecto está configurado para que la destrucción de dichas variables suceda luego de 20 minutos de inactividad).

Podemos modificar el tiempo de vida de las variables de sesión inicializando la propiedad Timeout del objeto Session que tiene toda página (el valor que se asigna representa minutos).

```
Me.Session.Timeout = 10
```

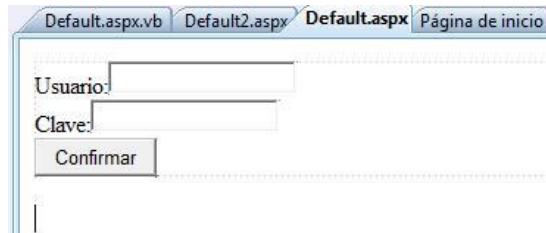
Los datos que podemos almacenar en variables de sesión pueden ser de cualquier tipo: string, integer, ArrayLists, DataSet, DataTables, etc.

Para ilustrar el uso de variables de sesión haremos una serie de páginas donde en la primera ingresaremos el nombre de usuario y clave, en la segunda los listaremos, y en esta dispondremos un

hipervínculo para dirigirnos a una tercera página, donde mostraremos nuevamente el contenido de las variables de sesión.

Página 1:

Esta página solicita la carga de los dos datos y redirecciona a la segunda página, la interface visual es la siguiente:



El código para el evento Click del botón confirmar es el siguiente:

```
Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(ByVal sender As Object,
        ByVal e As System.EventArgs) Handles Button1.Click
        'Almacenamos las dos variables de sesión
        Me.Session("usuario") = Me.TextBox1.Text
        Me.Session("clave") = Me.TextBox2.Text
        'Redireccionamos a la segunda pagina
        Response.Redirect("Default2.aspx")
    End Sub
End Class
```

Página 2:

Tiene por objetivo mostrar los contenidos de las dos variables de sesión. Además hay un hipervínculo (objeto de la clase HyperLink) que llama a la tercera página.

La interface visual es:



El código fuente de esta página es la siguiente:

```
Partial Class Default2
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object,
        ByVal e As System.EventArgs) Handles Me.Load
        'Mostramos el contenido de las dos variables
        'de sesión en las Label
        Me.Label1.Text = Me.Session("usuario")
        Me.Label2.Text = Me.Session("clave")
    End Sub
End Class
```

Es decir inicializamos las Label con los contenidos de las variables de sesión que están almacenadas en memoria y administradas por el servidor Web.

Página 3:

Por último esta tercera página tiene por objetivo mostrar nuevamente el contenido de las variables de sesión.

La interface es:



Y el código fuente del evento Load de la página es:

```
Partial Class Default3
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object,
                           ByVal e As System.EventArgs) Handles Me.Load
        'Introducir aquí el código de usuario
        'y la clave
        Label1.Text = Me.Session("usuario")
        Label2.Text = Me.Session("clave")
    End Sub
End Class
```

Las variables de sesión se verifican su existencia igual que las cookies.

Ejercicios Propuestos

1 – Confeccionar un sitio que muestre noticias Deportivas, Políticas y Culturales. Mediante una página de configuración permitir que un visitante pueda especificar que tipo de noticias quiere ver en la portada del sitio (emplear tres cookies)

	<u>UNIDAD Nº :3</u> Recursos de ASP.Net.	<u>TEMAS:</u> Imágenes dinámicas.	Clase 10
--	---	--	-------------------------------

Objetivos:

- Creación de imágenes dinámicas.
- Generación de distintos formatos gráficos..

Introducción

Cuando debemos generar imágenes a partir de datos que se extraen de una base de datos o datos ingresados desde el navegador debemos construir la imagen en dicho momento.

La ASP.Net provee de clases orientadas a generar imágenes en memoria y su posterior materialización en un archivo.

Imágenes dinámicas

1 - Programa “Hola Mundo”

Para crear una imagen en forma dinámica debemos seguir una serie de pasos:

a - Debemos modificar el archivo webform agregando lo indicado en negrita:

```
<%@ Page ContentType="image/gif" Language="vb" AutoEventWireup="false"
Codebehind="WebForm1.aspx.vb" Inherits="WebApplication27.WebForm1"%>
```

Al resto de las marcas Html debemos borrarlas ya que para generar una imagen no serán de utilidad. Es decir que en realidad no se trata de un webform lo que envía el servidor sino se trata de una imagen.

b – El archivo *.aspx.vb es:

```
' Importamos el espacio de nombre que declara la clase Bitmap, Graphics,
Font y SolidBrush
Imports System.Drawing
' Importamos el espacio de nombre que declara la clase ImageFormat
Imports System.Drawing.Imaging

Partial Class pagina
    Inherits System.Web.UI.Page

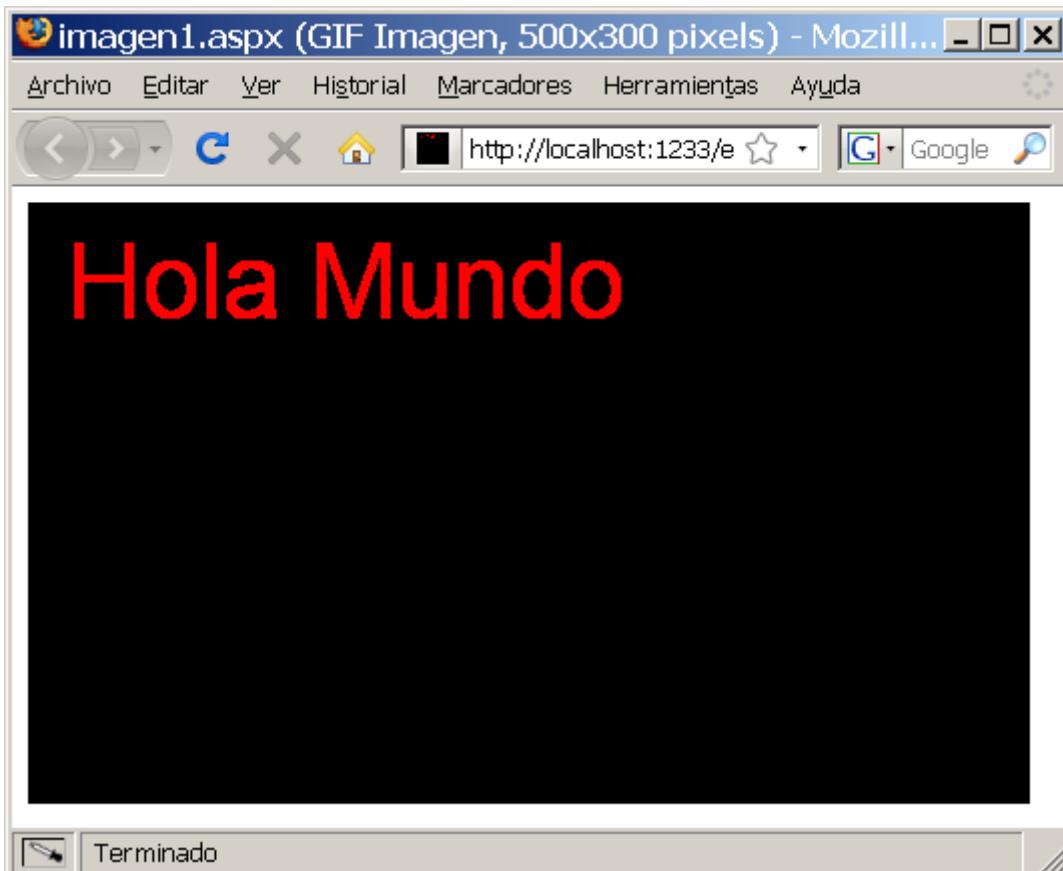
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        'Creamos un objeto de la clase Bitmap
        Dim mapabit As New Bitmap(500, 300)
        Dim lienzo As Graphics
```

```

'Inicializamos la variable de tipo Graphics con el bitmap creado
lienzo = Graphics.FromImage(mapabit)
'Creamos una fuente
Dim fuente1 As New Font("Arial", 40)
'Creamos un pincel
Dim pincell As New SolidBrush(Color.Red)
'Graficamos dentro de la imagen
lienzo.DrawString("Hola Mundo", fuente1, pincell, 10, 10)
'Enviamos la imagen al navegador que la solicitó
mapabit.Save(Response.OutputStream, ImageFormat.Gif)
End Sub
End Class

```

El resultado de la ejecución es:



Es importante tener en cuenta que cuando ejecutamos este webform en realidad lo que retorna es una imagen con formato gif y no un archivo HTML. Fácilmente podemos comprobarlo tratando de ver el código fuente de la página presionando el botón derecho del mouse dentro del navegador.

2 – Inserción de una imagen dinámica en un webform.

Para la incorporación de la imagen en una página propiamente dicha debemos hacer lo siguiente:

Creamos una página con un código similar a este:

```

<%@ Page Language="VB" AutoEventWireup="false" CodeFile="pagina1.aspx.vb"
Inherits="pagina1" %>

<!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>Página sin título</title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            </div>
        </form>
    </body>
</html>

```

Es decir que con la marca HTML Img indicamos en la propiedad src el nombre del archivo que genera la imagen.

3 –Generación de un Captcha.

Captcha es el acrónimo de *Completely Automated Public Turing text to tell Computers and Humans Apart* (Prueba de Turing pública y automática para diferenciar a máquinas y humanos). Se trata de una prueba desafío-respuesta utilizada en computación para determinar cuándo el usuario es o no humano.

Confeccionaremos una imagen con un número aleatorio entre 1000 y 9999 y para dificultar su lectura por un programa automático de lectura lo rayaremos con líneas de distintos colores. Por último incorporaremos almacenaremos el valor en una variable de sesión y validaremos el valor que ingresa el usuario en un control de tipo textbox.

a - Primero generamos un gráfico dinámico (captcha.aspx.vb y captcha.aspx):

Tener en cuenta que el archivo captcha.aspx solo debemos disponer (borramos todo el código HTML y en el bloque de definición de atributos aspx agregamos la propiedad ContentType indicando que generará una imagen en formato gif):

```

<%@ Page ContentType="image/gif" Language="VB" AutoEventWireup="false"
CodeFile="captcha.aspx.vb" Inherits="captcha" %>

```

Luego el código de la clase será:

```

Imports System.Drawing
Imports System.Drawing.Imaging
Partial Class captcha
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Dim mapabit As New Bitmap(100, 40)
        Dim lienzo As Graphics
        lienzo = Graphics.FromImage(mapabit)
        Dim nro As Integer
        nro = Rnd() * 10000

```

```

        Dim fuente1 As New Font("Arial", 22)
        Dim pincell As New SolidBrush(Color.Red)
        lienzo.DrawString(nro, fuente1, pincell, 4, 4)
        'Dibujar lineas
        For f As Integer = 1 To 10
            Dim x1 As Integer = Rnd() * 100
            Dim y1 As Integer = Rnd() * 40
            Dim x2 As Integer = Rnd() * 100
            Dim y2 As Integer = Rnd() * 40
            Dim lapiz1 As New Pen(Color.FromArgb(Rnd() * 255, Rnd() * 255,
Rnd() * 255))
                lienzo.DrawLine(lapiz1, x1, y1, x2, y2)
        Next
        Me.Session("codigo") = nro
        mapabit.Save(Response.OutputStream, ImageFormat.Gif)
    End Sub
End Class

```

Hacemos el import de los dos espacios de nombres:

```

Imports System.Drawing
Imports System.Drawing.Imaging

```

En el método Page_Load creamos un objeto de la clase BitMap indicando el ancho y el alto en píxeles.

Obtenemos una referencia de la imagen almacenándola en una variable de tipo Graphics.

```

Dim lienzo As Graphics
lienzo = Graphics.FromImage(mapabit)

```

Generamos un valor aleatorio:

```
nro = Rnd() * 10000
```

Imprimimos el número dentro de la imagen:

```

Dim fuente1 As New Font("Arial", 22)
Dim pincell As New SolidBrush(Color.Red)
lienzo.DrawString(nro, fuente1, pincell, 4, 4)

```

Dibujamos 10 líneas de distintos colores en forma aleatoria tapando parcialmente el número aleatorio:

```

For f As Integer = 1 To 10
    Dim x1 As Integer = Rnd() * 100
    Dim y1 As Integer = Rnd() * 40
    Dim x2 As Integer = Rnd() * 100
    Dim y2 As Integer = Rnd() * 40
    Dim lapiz1 As New Pen(Color.FromArgb(Rnd() * 255, Rnd() * 255,
Rnd() * 255))
        lienzo.DrawLine(lapiz1, x1, y1, x2, y2)
Next

```

Almacenamos el valor en una variable de sesión para poder recuperarlo en otra página y poder compararlo con el que ingresa el usuario por teclado:

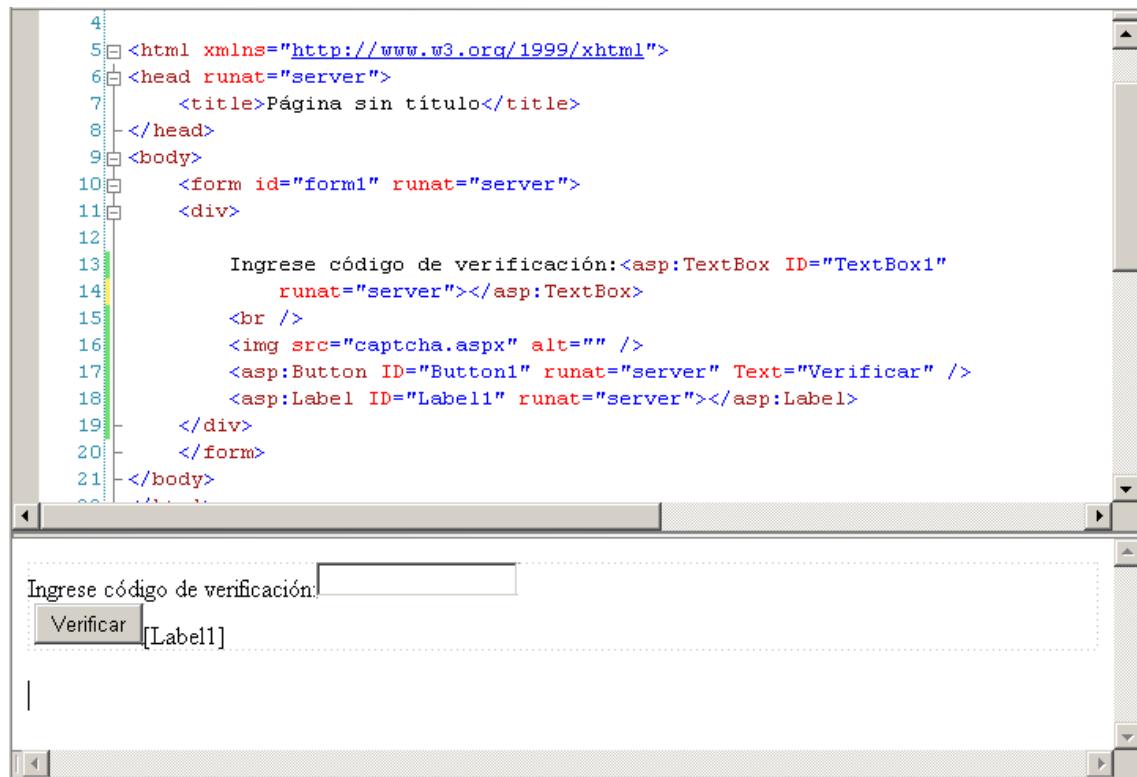
```
Me.Session("codigo") = nro
```

Enviamos la imagen al navegador que lo solicitó:

```
mapabit.Save(Response.OutputStream, ImageFormat.Gif)
```

b – Creamos un formulario web donde insertamos el Captcha (imagen) y disponemos además un control de tipo textbox y un button:

Debemos generar un webform semejante a este:



Para el evento Click del objeto Button1 disponemos el siguiente código:

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Button1.Click
    If Me.Session("codigo") = Me.TextBox1.Text Then
        Me.Label1.Text = "Ingresó el código correcto"
    Else
        Me.Label1.Text = "Incorrecto"
    End If
End Sub
```

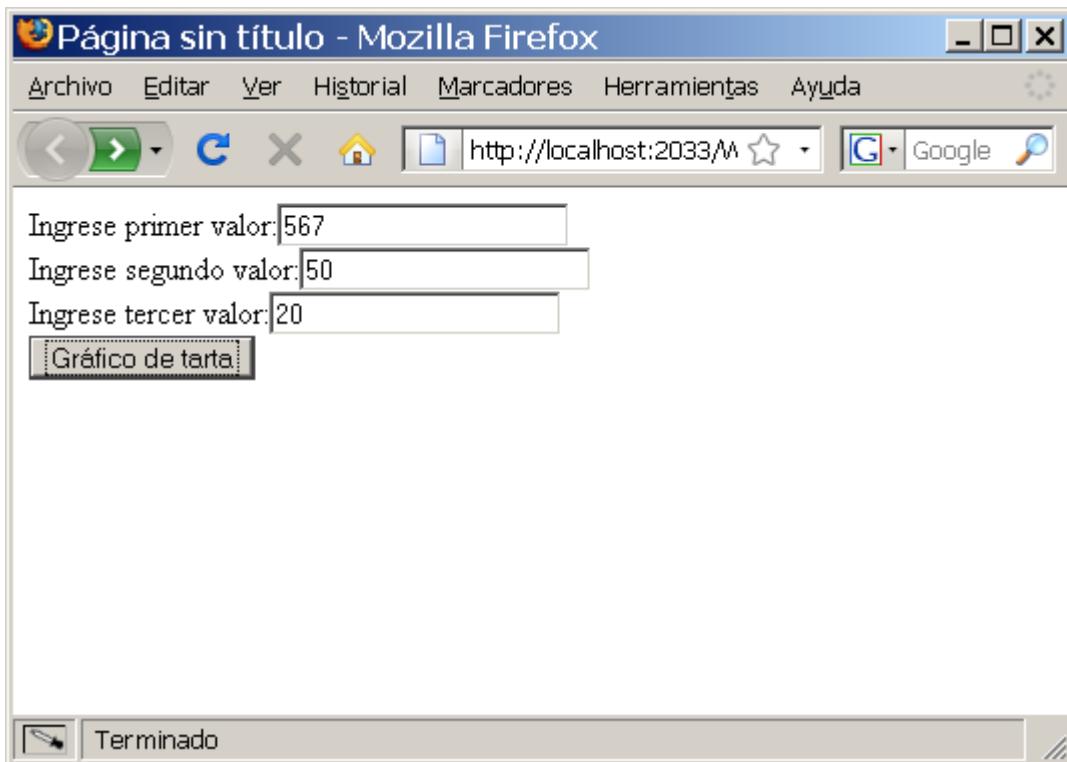
Luego mediante un if controlamos el valor del captcha y el valor ingresado por el usuario en el textbox1. Mostramos en un label el resultado de la comparación.

4 –Gráfico de tarta.

Para implementar un gráfico de tarta debemos emplear el método fillPie de la clase Graphics. Desarrollaremos una aplicación que solicite en un webform tres valores enteros por teclado y llamaremos a un gráfico dinámico que mostrará un gráfico de tarta rescatando los valores del webform.

El webform envía mediante parámetros los datos ingresados.

El webform tiene la siguiente interfaz:



Y el código que debemos implementar para el evento Click del objeto Button es:

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
    Me.Response.Redirect("graficotarta.aspx?v1=" & Me.TextBox1.Text &
    "&v2=" & Me.TextBox2.Text & "&v3=" & Me.TextBox3.Text)
End Sub
```

Es decir redireccionamos al archivo graficotarta.aspx pasando 3 parámetros en la cabecera de la llamada.

El archivo que genera la imagen dinámica (graficotarta.aspx.vb) es la siguiente:

```
Imports System.Drawing
Imports System.Drawing.Imaging
Partial Class graficotarta
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
        Dim mapabit As New Bitmap(800, 600)
        Dim lienzo As Graphics
        lienzo = Graphics.FromImage(mapabit)
        Dim valor1 As Integer = Me.Request.QueryString("v1")
        Dim valor2 As Integer = Me.Request.QueryString("v2")
        Dim valor3 As Integer = Me.Request.QueryString("v3")
        Dim total As Integer = valor1 + valor2 + valor3
        Dim grados1 As Integer = valor1 / total * 360
        Dim pincel1 As New SolidBrush(Color.Red)
        lienzo.FillPie(pincel1, 100, 100, 400, 400, 0, grados1)
        Dim grados2 As Integer = valor2 / total * 360
        Dim pincel2 As New SolidBrush(Color.Blue)
        lienzo.FillPie(pincel2, 100, 100, 400, 400, grados1, grados2)
        Dim grados3 As Integer = valor3 / total * 360
```

```

        Dim pincel3 As New SolidBrush(Color.Green)
        lienzo.FillPie(pincel3, 100, 100, 400, 400, grados1 + grados2,
grados3)
        'Referencias
        lienzo.FillRectangle(pincel1, 600, 500, 20, 20)
        Dim fuente As New Font("Arial", 10)
        lienzo.DrawString(valor1, fuente, pincel1, 630, 500)
        lienzo.FillRectangle(pincel2, 600, 530, 20, 20)
        lienzo.DrawString(valor2, fuente, pincel2, 630, 530)
        lienzo.FillRectangle(pincel3, 600, 560, 20, 20)
        lienzo.DrawString(valor3, fuente, pincel3, 630, 560)
        mapabit.Save(Response.OutputStream, ImageFormat.Gif)
    End Sub
End Class

```

También recordemos que el archivo (graficotarta.aspx) debemos borrar todo el contenido HTML y modificar la cabecera:

```

<%@ Page Language="VB" AutoEventWireup="false"
CodeFile="graficotarta.aspx.vb" Inherits="graficotarta"
contenttype="image/gif"%>

```

Veamos el algoritmo para mostrar el gráfico de tarta dentro de la imagen. En el método Load generamos la imagen. Creamos un objeto de la clase Bitmap y obtenemos la referencia al objeto de la clase Graphics contenido en el mismo:

```

Dim mapabit As New Bitmap(800, 600)
Dim lienzo As Graphics
lienzo = Graphics.FromImage(mapabit)

```

Recuperamos los tres valores ingresados en el webform:

```

Dim valor1 As Integer = Me.Request.QueryString("v1")
Dim valor2 As Integer = Me.Request.QueryString("v2")
Dim valor3 As Integer = Me.Request.QueryString("v3")

```

Calculamos la cantidad de grados que le corresponde a cada valor y dibujamos el trozo de tarta respectivo:

```

Dim total As Integer = valor1 + valor2 + valor3
Dim grados1 As Integer = valor1 / total * 360
Dim pincel1 As New SolidBrush(Color.Red)
lienzo.FillPie(pincel1, 100, 100, 400, 400, 0, grados1)
Dim grados2 As Integer = valor2 / total * 360
Dim pincel2 As New SolidBrush(Color.Blue)
lienzo.FillPie(pincel2, 100, 100, 400, 400, grados1, grados2)
Dim grados3 As Integer = valor3 / total * 360
Dim pincel3 As New SolidBrush(Color.Green)
lienzo.FillPie(pincel3, 100, 100, 400, 400, grados1 + grados2,
grados3)

```

Dibujamos los recuadros de referencias con el valor que le corresponde a cada trozo de tarta:

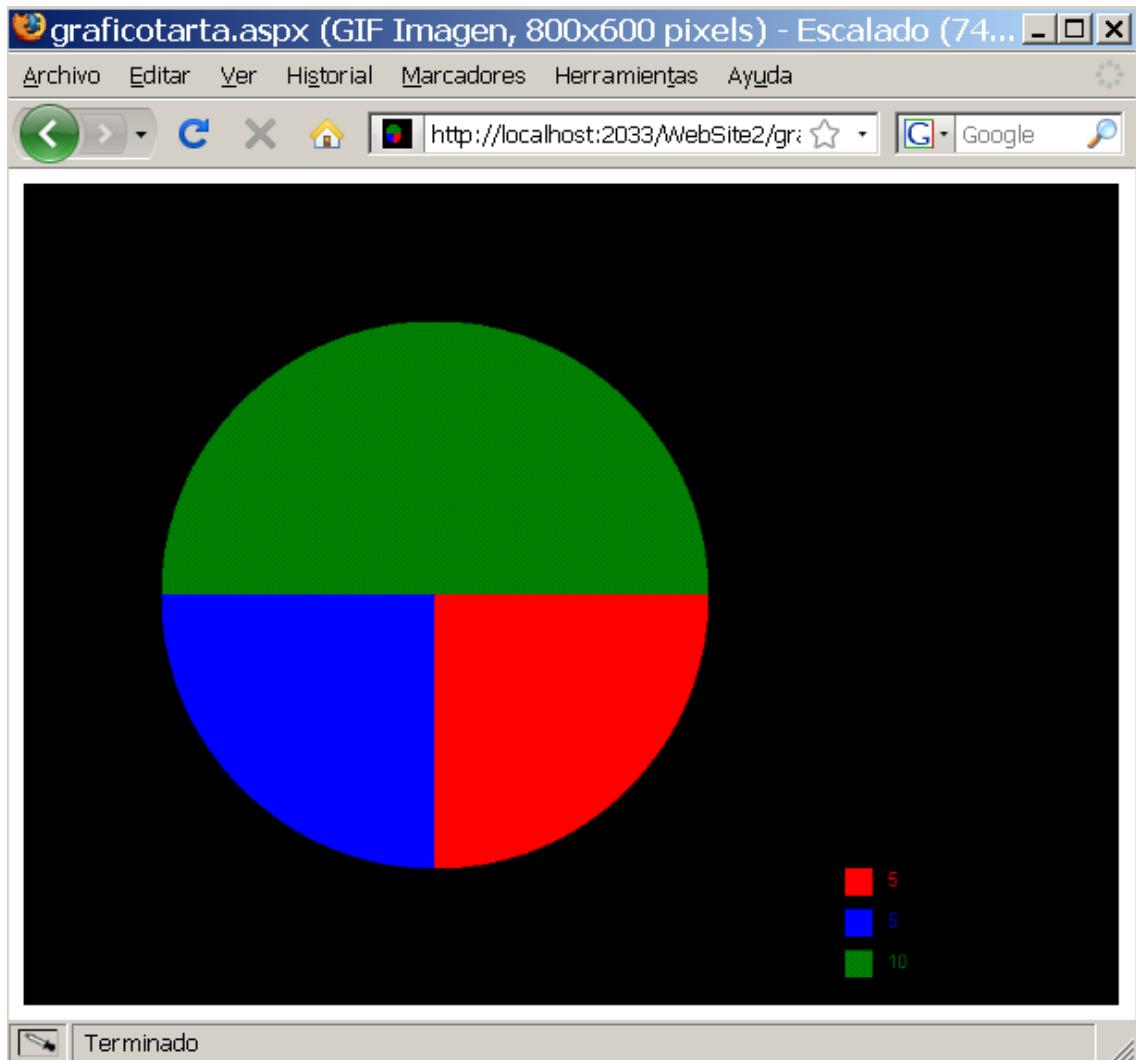
```

lienzo.FillRectangle(pincel1, 600, 500, 20, 20)
Dim fuente As New Font("Arial", 10)
lienzo.DrawString(valor1, fuente, pincel1, 630, 500)
lienzo.FillRectangle(pincel2, 600, 530, 20, 20)
lienzo.DrawString(valor2, fuente, pincel2, 630, 530)
lienzo.FillRectangle(pincel3, 600, 560, 20, 20)
lienzo.DrawString(valor3, fuente, pincel3, 630, 560)

```

```
mapabit.Save(Response.OutputStream, ImageFormat.Gif)
```

El resultado del gráfico es:



4 –Gráfico de barra horizontal.

Para implementar un gráfico de barra horizontal debemos emplear el método `fillrect` de la clase `Graphics`. Desarrollaremos una aplicación que solicite en un webform tres valores enteros por teclado y llamaremos a un gráfico dinámico que mostrará un gráfico de barra horizontal rescatando los valores del webform.

El webform envía mediante parámetros los datos ingresados.

El interfaz para la carga de los tres valores enteros en un webform es similar al problema del gráfico de tarta.

El código que debemos implementar para el evento Click del objeto Button es:

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
    Me.Response.Redirect("graficobarrahorizontal.aspx?v1=" &
    Me.TextBox1.Text & "&v2=" & Me.TextBox2.Text & "&v3=" & Me.TextBox3.Text)
End Sub
```

Es decir redireccionamos al archivo graficobarrahorizontal.aspx pasando 3 parámetros en la cabecera de la llamada.

El archivo que genera la imagen dinámica (graficobarrahorizontal.aspx.vb) es la siguiente:

```
Imports System.Drawing
Imports System.Drawing.Imaging

Partial Class Default2
    Inherits System.Web.UI.Page

    Private Function RetornarMayor(ByVal x1 As Integer, ByVal x2 As Integer, ByVal x3 As Integer) As Integer
        If x1 > x2 And x1 > x3 Then
            Return x1
        Else
            If x2 > x3 Then
                Return x2
            Else
                Return x3
            End If
        End If
    End Function

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
        Dim mapabit As New Bitmap(800, 400)
        Dim lienzo As Graphics
        lienzo = Graphics.FromImage(mapabit)
        Dim valor1 As Integer = Me.Request.QueryString("v1")
        Dim valor2 As Integer = Me.Request.QueryString("v2")
        Dim valor3 As Integer = Me.Request.QueryString("v3")
        Dim mayor As Integer = Me.RetornarMayor(valor1, valor2, valor3)
        Dim largo1 As Integer = valor1 / mayor * 400
        Dim pincel1 As New SolidBrush(Color.Red)
        lienzo.FillRectangle(pincel1, 50, 50, 50 + largo1, 80)
        Dim largo2 As Integer = valor2 / mayor * 400
        Dim pincel2 As New SolidBrush(Color.Green)
        lienzo.FillRectangle(pincel2, 50, 150, 50 + largo2, 80)
        Dim largo3 As Integer = valor3 / mayor * 400
        Dim pincel3 As New SolidBrush(Color.Blue)
        lienzo.FillRectangle(pincel3, 50, 250, 50 + largo3, 80)
        'Cantidades
        Dim pincel4 As New SolidBrush(Color.Black)
        Dim fuente1 As New Font("Arial", 30)
        lienzo.DrawString(valor1, fuente1, pincel4, 60, 80)
        lienzo.DrawString(valor2, fuente1, pincel4, 60, 160)
        lienzo.DrawString(valor3, fuente1, pincel4, 60, 260)
        mapabit.Save(Response.OutputStream, ImageFormat.Gif)
    End Sub
End Class
```

El método RetornarMayor recibe como parámetros tres enteros y retorna el mayor de los mismos:

```
Private Function RetornarMayor(ByVal x1 As Integer, ByVal x2 As Integer, ByVal x3 As Integer) As Integer
    If x1 > x2 And x1 > x3 Then
        Return x1
    End If
End Function
```

```

    Else
        If x2 > x3 Then
            Return x2
        Else
            Return x3
        End If
    End If
End Function

```

El método Load es el que crea el gráfico de barra horizontal propiamente dicho.

Para graficar cada barra:

```

Dim largol As Integer = valor1 / mayor * 400
Dim pincel1 As New SolidBrush(Color.Red)
lienzo.FillRectangle(pincel1, 50, 50, 50 + largol, 80)

```

Debemos dividir el valor correspondiente con respecto al mayor de los tres. Este valor como máximo puede ser 1. Luego multiplicamos este valor por 400 (este valor representa el largo de barra mayor)

Graficamos mediante el método FillRectangle.

Mostramos los valores enteros que les corresponde a cada barra:

```

'Cantidad
Dim pincel4 As New SolidBrush(Color.Black)
Dim fuente1 As New Font("Arial", 30)
lienzo.DrawString(valor1, fuente1, pincel4, 60, 80)
lienzo.DrawString(valor2, fuente1, pincel4, 60, 160)
lienzo.DrawString(valor3, fuente1, pincel4, 60, 260)

```



Ejercicios Propuestos

1 – Confeccionar un gráfico dinámico que genere un gráfico de barras verticales.

2 – Confeccionar un gráfico dinámico que genere una barra porcentual (mostrar los porcentajes que les corresponde a cada trozo fuera de la barra).

3 – Generar una imagen a partir de otra y agregarle un texto en forma dinámica. En algunos casos cuando se quiere proteger la propiedad de imágenes es aplicarles un texto que indica que la imagen no puede ser usada en otro sitio para fines comerciales.

4 – Confeccionar un algoritmo que permita ingresar los datos personales de una persona y luego genere una tarjeta personal con dichos datos:
ej.:



	UNIDAD Nº :3 Recursos de ASP.Net.	TEMAS: Master Page.	Clase 11
--	--	--	-------------------------------

Objetivos:

- Identificar las ventajas de la reutilización de código.
- Dominar los pasos propuestos por el Visual Studio.Net para la creación de "Master Page".

Introducción

Una Master Page o Página Principal es una estructura base para un conjunto de páginas pertenecientes a un mismo sitio Web. Este esqueleto base se almacena en un archivo independiente y luego es heredado por otras páginas que requieren esa estructura base.

Un ejemplo donde utilizar una página Master Page es la definición de la cabecera y el pie del sitio, luego todas las otras páginas del sitio heredan de dicha página. La ventaja fundamental en este ejemplo es que si hay cambios en la cabecera del sitio solo debemos hacerlos en la página principal y con esto se propaga en todas las páginas que heredan de la misma.

1 – Problema 1

Problema: Confeccionar un sitio que permita mostrar un conjunto de direcciones web agrupados por tema. Disponer una lista de hipervínculos con los siguientes temas: Buscadores – Periódicos – Blog
Mostrar en la parte inferior de la página un mensaje de Copyright.

Para crear un sitio que utilice una Master Page debemos seguir una serie de pasos:

a – Creamos un nuevo sitio web: Archivo → Nuevo Sitio Web...
y seleccionamos la plantilla “ Sitio web vacío ”

b – En la ventana del “Explorador de soluciones” presionamos el botón derecho del mouse y seleccionamos “Aregar nuevo elemento...” y seleccionamos “Página principal” (en inglés Master Page)

Las páginas Master Page tienen extensión master en lugar de aspx.

El código generado en forma automática es el siguiente:

```
<%@ Master Language="VB" CodeFile="MasterPage.master.vb"
Inherits="MasterPage" %>

<!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>
    <asp:ContentPlaceHolder id="head" runat="server">
        </asp:ContentPlaceHolder>
</head>
<body>
    <form id="form1" runat="server">
        <div>
```

```

<asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">

    </asp:ContentPlaceHolder>
</div>
</form>
</body>
</html>

```

Lo primero que vemos que es distinto con un WebForm es la directiva Master en lugar de Page:

```
<%@ Master Language="VB" CodeFile="MasterPage.master.vb"
Inherits="MasterPage" %>
```

Ahora agregaremos los hipervínculos en la parte superior y el pié de página:

```

<%@ Master Language="VB" CodeFile="MasterPage.master.vb"
Inherits="MasterPage" %>

<!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>
    <asp:ContentPlaceHolder id="head" runat="server">
        </asp:ContentPlaceHolder>
</head>
<body>

    <form id="form1" runat="server">
        <div>
            <h2> <a href="Default.aspx">Buscadores.</a>-
                <a href="Default2.aspx">Periódicos.</a>-
                <a href="Default3.aspx">Blogs.</a></h2>

            <asp:ContentPlaceHolder id="ContentPlaceHolder1" runat="server">

                </asp:ContentPlaceHolder>
            <h3>Copyright 2009</h3>

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

```

Luego veremos que todas las páginas que hereden de esta mostrarán los tres hipervínculos en la parte superior y el mensaje de Copyright en la parte inferior.

c – Agregar un Web Forms que herede del Master Page que hemos creado. En la ventana del “Explorador de soluciones” presionamos el botón derecho del mouse y seleccionamos “Aregar nuevo elemento...” y seleccionamos “Web Forms”. Es importantísimo ahora seleccionar el checkbox “Seleccionar la página principal”, también podemos cambiar el nombre por defecto de la página (podemos llamarla Default.aspx), luego de presionar el botón Agregar y si seleccionamos correctamente “Seleccionar la página principal” aparecerá un nuevo diálogo para seleccionar el Master Page del cual hereda nuestro Web Form.

El código generado en forma automática es el siguiente:

```
<%@ Page Title="" Language="VB" MasterPageFile="~/MasterPage.master"
AutoEventWireup="false" CodeFile="Default.aspx.vb" Inherits="_Default" %>
```

```

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
Runat="Server">
</asp:Content>

```

Como podemos observar no contiene las marcas HTML de la cabecera y el cuerpo, ya que se heredan de la Master Page.

El código propio de esta página debemos disponerlo en la marca asp:Content, en nuestro ejemplo agregaremos una lista de enlaces a los distintos buscadores:

```

<%@ Page Title="" Language="VB" MasterPageFile="~/MasterPage.master"
AutoEventWireup="false" CodeFile="Default.aspx.vb" Inherits="_Default" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
Runat="Server">
<a href="http://www.google.com">Google</a><br />
<a href="http://www.yahoo.com">Yahoo</a><br />
<a href="http://www.live.com">Live</a><br />
</asp:Content>

```

d - De igual forma creamos las otras páginas (Default2.aspx y Default3.aspx):

Default2.aspx

```

<%@ Page Title="" Language="VB" MasterPageFile="~/MasterPage.master"
AutoEventWireup="false" CodeFile="Default2.aspx.vb" Inherits="Default2" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
Runat="Server">
<a href="http://www.lanacion.com.ar">La Nación</a><br />
<a href="http://www.clarin.com.ar">El Clarín</a><br />
<a href="http://www.lavoz.com.ar">La Voz</a><br />
</asp:Content>

```

Default3.aspx

```

<%@ Page Title="" Language="VB" MasterPageFile="~/MasterPage.master"
AutoEventWireup="false" CodeFile="Default3.aspx.vb" Inherits="Default3" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" Runat="Server">
</asp:Content>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1"
Runat="Server">
<a href="http://www.microsiervos.com">Microsiervos</a><br />
<a href="http://www.alt1040.com">Alt 1040</a><br />
<a href="http://www.puntogeek.com">PuntoGeek</a><br />
</asp:Content>

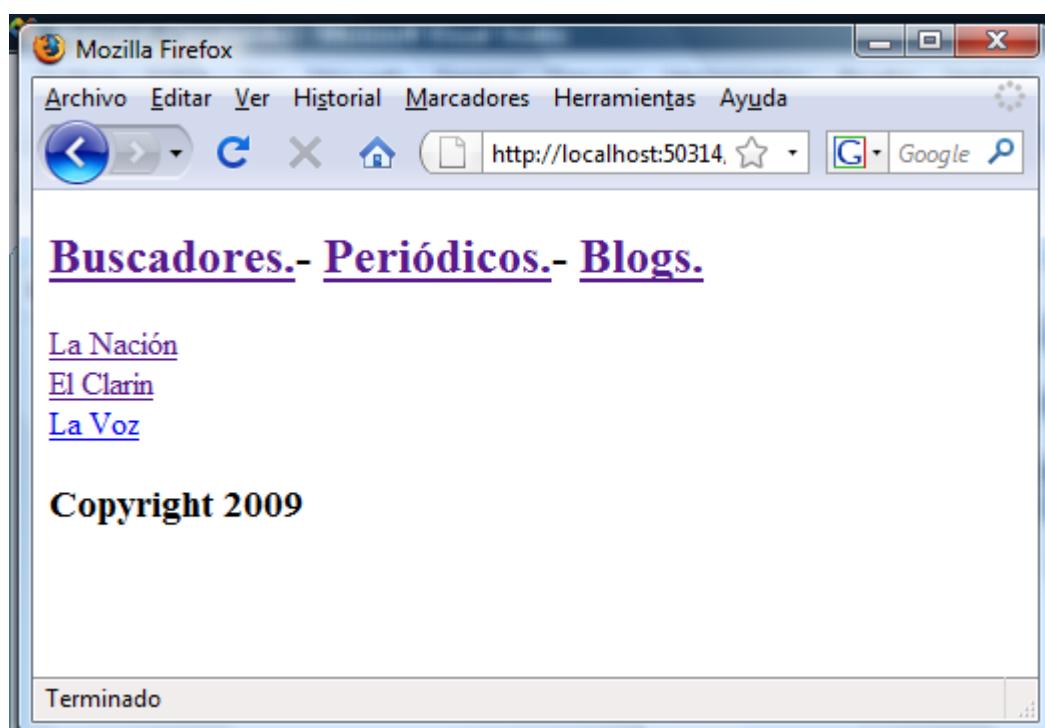
```

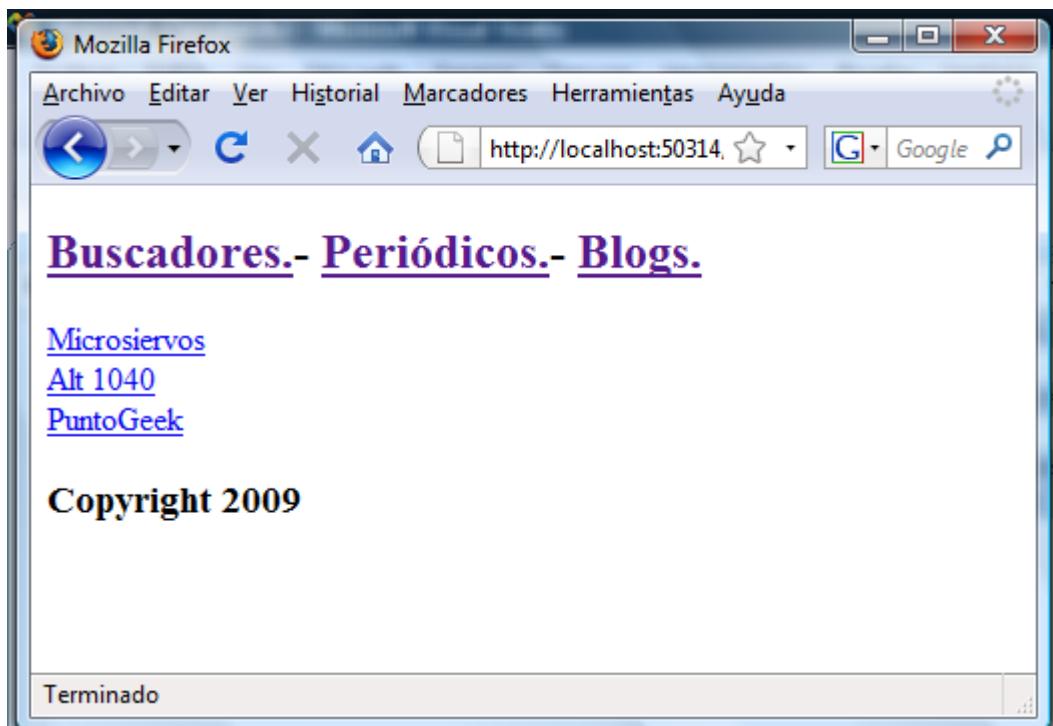
Ahora cuando ejecutamos y solicitamos la página Default.aspx el resultado es el siguiente:



Como podemos observar al contenido definido en el archivo Default.aspx (los tres hipervínculos a los buscadores) se a añadido todo lo definido en la Master Page (es decir los hipervínculos a las categorías “Buscadores”, “Periódicos” y “Blogs”, además del mensaje en la parte inferior de la página)

Luego cuando solicitamos las páginas Default2.aspx y Default3.aspx vemos que todas tienen el contenido definido en el archivo MasterPage:





	<u>UNIDAD Nº :3</u> Recursos de ASP.Net.	<u>TEMAS:</u> Ajax.	Clase 12
--	---	--	-------------------------------

Objetivos:

- Identificar y utilizar la clases que provee .Net para implementar Ajax.
- Identificar las ventajas de Ajax.

Introducción

Hasta ahora, hemos aprendido a crear páginas web que utilizan el modelo postback. Con el postback, las páginas están perpetuamente reenviándose al servidor web y regenerándose. Como desventaja del modelo postback tenemos que hay un parpadeo del contenido de la página cuando tiene que refrescarse. Además se tiene el problema que se reenvían todos los datos al servidor.

Recientemente, una nueva generación de aplicaciones web ha comenzado a parecer que se comportan más como las aplicaciones de Windows que las tradicionales páginas web. Estas aplicaciones se refrescan en forma rápida y sin parpadeos. Entre los ejemplos notables incluyen el correo electrónico basado en web como Gmail y aplicaciones de la cartografía como Google Maps. Esta nueva generación de aplicaciones web utiliza un conjunto de prácticas de diseño y tecnologías conocidas como Ajax.

Una de las características fundamentales de Ajax es la capacidad para actualizar parte de la página, mientras que deja el resto intacto.

AJAX son las siglas de **A**synchronous **J**ava**S**cript **A**nd **X**ML. No es un lenguaje de programación sino un conjunto de tecnologías (HTML-JavaScript-CSS-DHTML-PHP/ASP.NET/JSP-XML) que nos permiten hacer páginas de internet más interactivas.

La característica fundamental de AJAX es permitir actualizar parte de una página con información que se encuentra en el servidor sin tener que refrescar completamente la página. De modo similar podemos enviar información al servidor.

Veremos como ASP.Net nos esconde la complejidad de Ajax y nos permite una fácil transición entre aplicaciones web tradicionales y el nuevo modelo.

ASP.NET AJAX

Hay una variedad de formas de aplicar el Ajax en cualquier aplicación Web, incluyendo ASP.NET. Para ponerlo en práctica Ajax sin utilizar librerías, es necesario que se tenga una comprensión profunda de JavaScript, porque

es el lenguaje JavaScript el que se ejecuta en el navegador, que solicita la nueva información del servidor web cuando sea necesario y la actualización de la página en consecuencia.

Si bien JavaScript no está terriblemente complejo, es muy difícil de programar correctamente, por dos razones:

- La aplicación de los principales detalles de JavaScript varía de navegador a navegador, que significa que necesita una enorme cantidad de experiencia para escribir una buena página web que se ejecuta igual de bien en todos los navegadores.
- JavaScript es un lenguaje muy laxo que tolera muchos pequeños errores. La captura de estos errores y la eliminación de ellos es un proceso tedioso. Aún peor, el error puede ser fatal en algunos navegadores y dañina en otros, lo que complica la depuración.

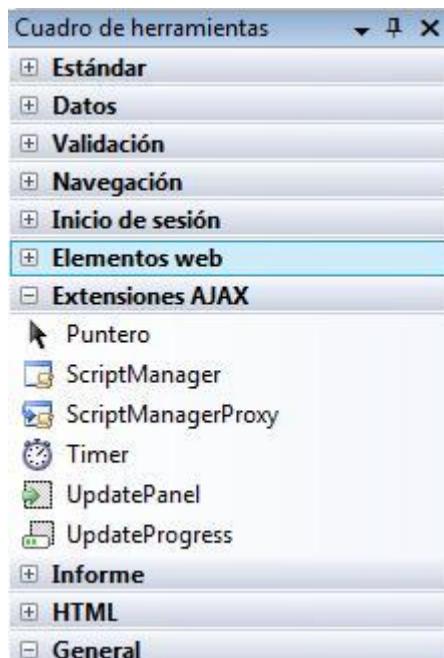
En este capítulo, usted NO va a usar JavaScript directamente. Por el contrario, utilizará un modelo de nivel superior llamado ASP.NET AJAX.

ASP.NET AJAX le da un conjunto de componentes del lado del servidor y los controles ASP.NET tradicionales que puede utilizar en el diseño de su página web. Estos componentes hacen

automáticamente todos el código JavaScript que necesita para obtener el efecto deseado. El resultado es que puede crear una página con Ajax. Por supuesto, usted no obtendrá el máximo control para personalizar hasta el último detalle de su página, pero usted recibirá una gran funcionalidad con un mínimo de esfuerzo.

Extensiones Ajax

El Visual Studio .Net 2008 agrega una pestaña que agrupa los controles referidos a Ajax:



Para poder utilizar ASP.NET AJAX es necesario un control de tipo ScriptManager. Este control es el cerebro de ASP.NET AJAX. Cuando disponemos un control de tipo ScriptManager en el formulario aparece un cuadro gris, pero cuando ejecutemos dicha página no aparecerá, es decir el control ScriptManager no genera etiquetas HTML.

El control ScriptManager genera todo el código JavaScript necesario para las llamadas asíncronas desde el navegador al servidor web.

Cada página que utiliza ASP.NET AJAX requiere solo una instancia de la clase ScriptManager, indistintamente la cantidad de regiones que vallamos a actualizar posteriormente.

Refresco parcial de página.

Confeccionaremos un problema sin utilizar AJAX y luego lo resolveremos utilizando AJAX y veremos las ventajas que presenta.

El problema a resolver es mostrar un video del sitio de youtube.com, dispondremos un botón que cuando se presione recuperará los comentarios existentes para dicho video (para hacer el problema más corto evitaremos extraer los comentarios de una base de datos)

Creamos un sitio web y disponemos en vista de código las marcas HTML que nos provee el sitio youtube.com para insertar en nuestra página el video. Luego la página Default en vista de código queda:

```
<%@ 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>
</head>
<body>

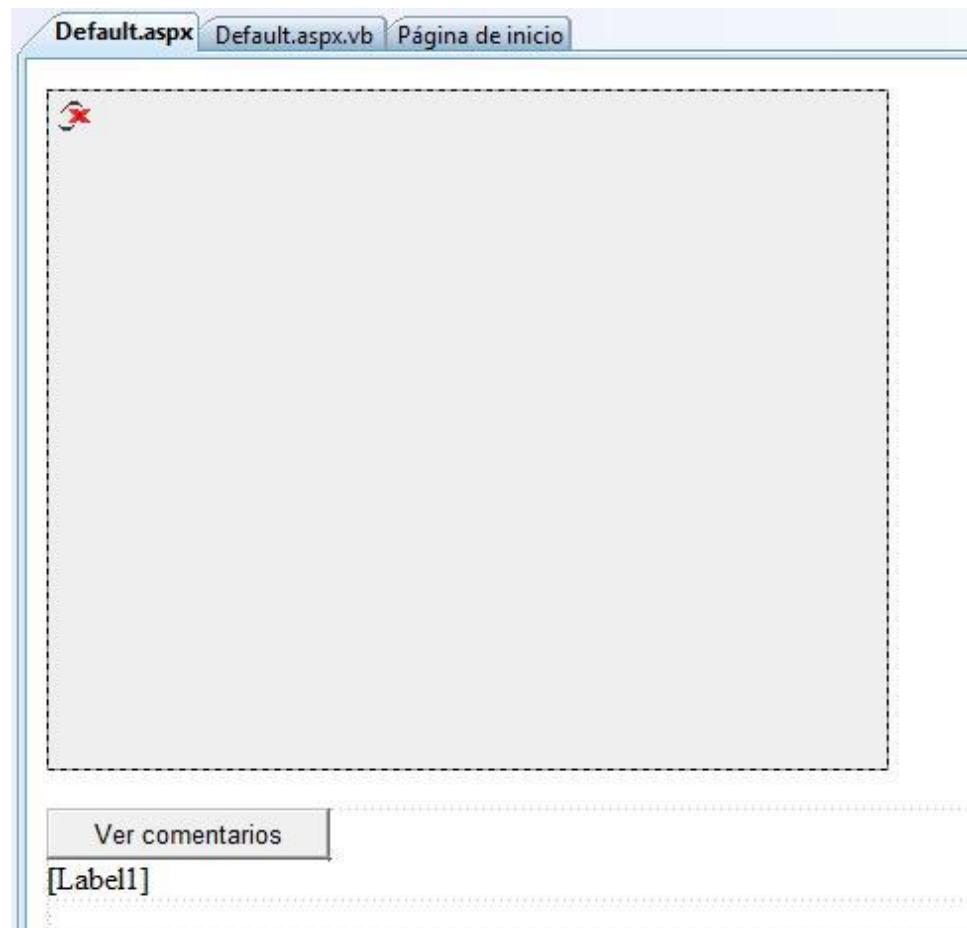
<object width="425" height="344">
<param name="movie"
value="http://www.youtube.com/v/Gi61DAYGIkE&hl=es&fs=1&"></param>
<param name="allowFullScreen" value="true"></param>
<param name="allowScriptAccess" value="always"></param>
<embed src="http://www.youtube.com/v/Gi61DAYGIkE&hl=es&fs=1&"
type="application/x-shockwave-flash" allowScriptAccess="always"
allowfullscreen="true" width="425" height="344"></embed>

</object>
<form id="form1" runat="server">
<div>

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

```

Después del bloque del body hemos dispuesto el código suministrado por youtube.com
Ahora en vista de Diseño agregamos un Button y una Label.:



El código a implementar al presionar el botón es:

```
Partial Class _Default
Inherits System.Web.UI.Page

Protected Sub Button1_Click(ByVal sender As Object, ByVal e As _
System.EventArgs) Handles Button1.Click

    Me.Label1.Text =
        "1 - Muy buen video. <br> 2 - Correcto. <br>3 - De lo mejor."

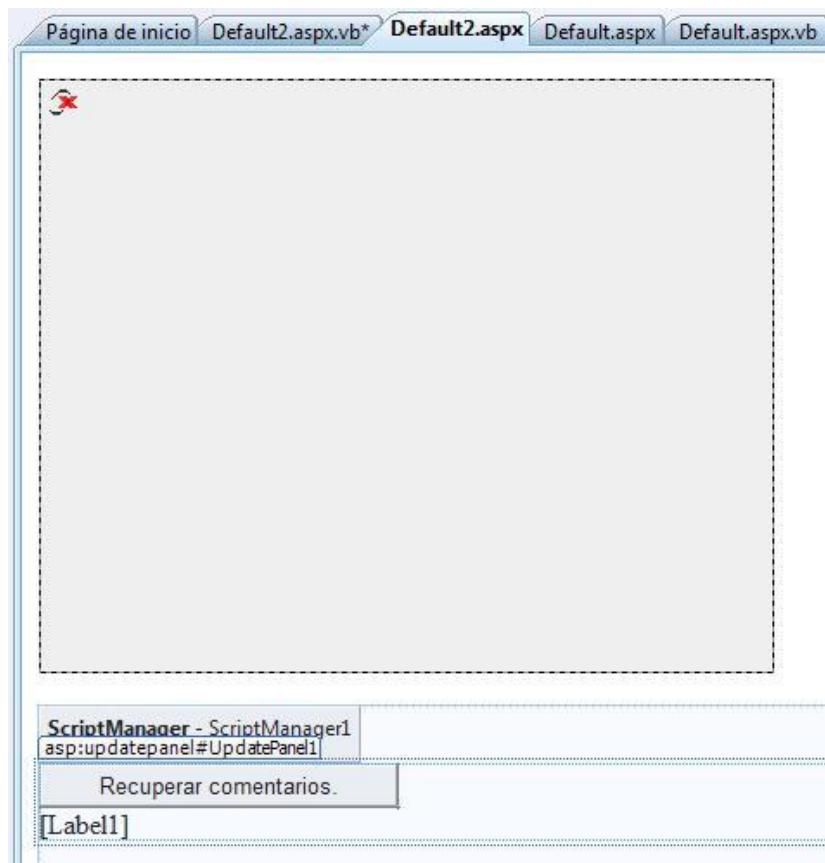
End Sub
End Class
```

Es decir cargamos el contenido de la Label con los comentarios. Como podrán ver al ejecutar la aplicación si estamos viendo el video y presionamos el botón de “Ver comentarios” el contenido de la página se recarga, es decir debemos comenzar de nuevo a ver el video (esta situación es poco agradable para el visitante del sitio)

Ahora crearemos otra página y utilizaremos ASP.NET AJAX para no tener que refrescar toda la página sino la Label donde deben aparecer los comentarios.

Creamos una segunda página (Default2.aspx), agregamos también el código suministrado por youtube.com.

Debemos primero insertar un objeto de la clase ScriptManager y un objeto de la clase UpdatePanel. Todo lo contenido en el control UpdatePanel se refrescará sin tener que recargar la página por completo. Disponemos entonces dentro del UpdatePanel1 la Label y el botón para “Recuperar comentarios”, la interface visual creada es:



Para el evento click del botón realizamos el mismo código hecho para la página anterior:

```

Partial Class Default2
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(ByVal sender As Object,
        ByVal e As System.EventArgs) Handles Button1.Click
        Me.Label1.Text =
            "1 - Muy buen video. <br> 2 - Correcto.<br> 3 - De lo mejor."
    End Sub
End Class

```

Si ejecutamos la página Default2.aspx veremos que la experiencia del usuario visitante será mejor. Proberemos de iniciar el video y luego presionar el botón “Ver comentarios”, como observaremos el video no se detiene y luego de un corto tiempo tenemos en la página los comentarios dispuestos en dicho video.

La idea básica es dividir su página web en una o más regiones distintas, cada una de las que se envuelve dentro de un UpdatePanel invisibles en tiempo de ejecución. Cuando un evento ocurre en un control que se ubicado dentro de un UpdatePanel, y este evento normalmente desencadena una página completa postback, el UpdatePanel intercepta el evento y realiza una llamada asincrónica. Aquí está un ejemplo de cómo esto sucede:

1. El usuario hace clic en un botón dentro de un UpdatePanel.
2. El UpdatePanel intercepta en el lado del cliente el clic.

Ahora, ASP.NET AJAX realiza una llamada al servidor en lugar de una página completa postback.

3. En el servidor, se ejecuta su ciclo de vida en forma normal, con todos los eventos habituales.
4. ASP.NET AJAX recibe todo el HTML y se actualiza cada UpdatePanel y los cambios que no están dentro de un UpdatePanel se ignoran.

