

**Controles avanzados y referencia de funciones.**

## Indice

<b>Nº- 9 Controles avanzados y referencia de funciones.....</b>	<b>1</b>
1. Control Calendario.....	1
1.1 Formato del control.....	4
1.2 Restringir fechas.....	6
2. Control Addrotator.....	8
2.1 La clase AdRotator.....	8
3. Páginas con varias vistas.....	12
3.1 Control "MultiView" .....	14
4. El asistente o control Wizard.....	19
5. Dibujar con ASP.NET.....	25
5.1 Dibujo básico.....	25
5.2 Dibujar una imagen personalizada.....	27
6. Funciones aritméticas.....	29
Ejemplo con ABS.....	31
3.6 Funciones de fechas.....	42
Now.....	43
<b>Ejercicios.....</b>	<b>54</b>
Ejercicio 1.....	54
Ayudas y comentarios:.....	57
Ejercicio 2.....	59

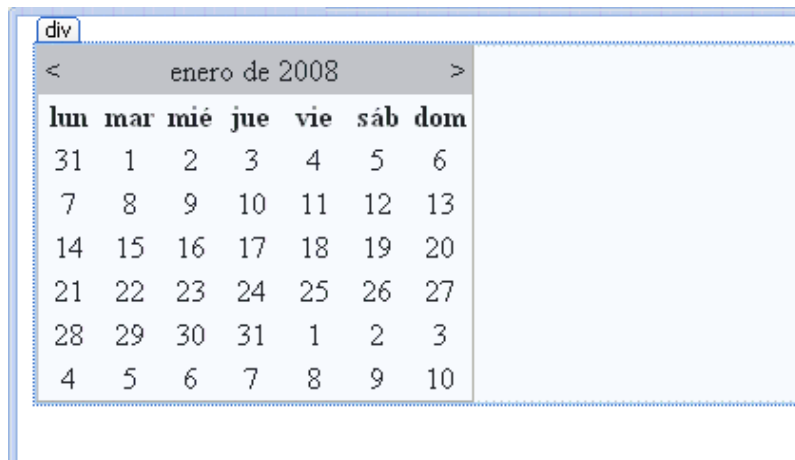
## Nº- 9 Controles avanzados y referencia de funciones

Hasta ahora hemos visto controles sencillos, vamos a tratar ahora con controles avanzados, que nos va a proporcionar funcionalidades realmente avanzadas. Los eventos que atenderán serán mucho mas complejos y nos van a proporcionar muchas mas opciones que las que hemos visto hasta ahora. Así que vamos allá.

### 1. Control Calendario.

El control calendario quizás sea el mas espectacular por la gran cantidad de comandos que tiene y la buena representación gráfica que tiene, ya que es prácticamente idéntico que el equivalente en una aplicación Windows.

Crea una página en blanco y añades un control de tipo calendario:



Si practicas un poco con él verás como no hace falta añadirle nada: nos muestra el mes actual y responde a los eventos si seleccionamos los meses anterior y posterior. Recuperaremos la fecha seleccionada por el usuario en la propiedad "SelectedDate" que será de tipo "DateTime", amplía la página para poner en un "label" la opción que seleccione el usuario:

## Controles avanzados y referencia de funciones.

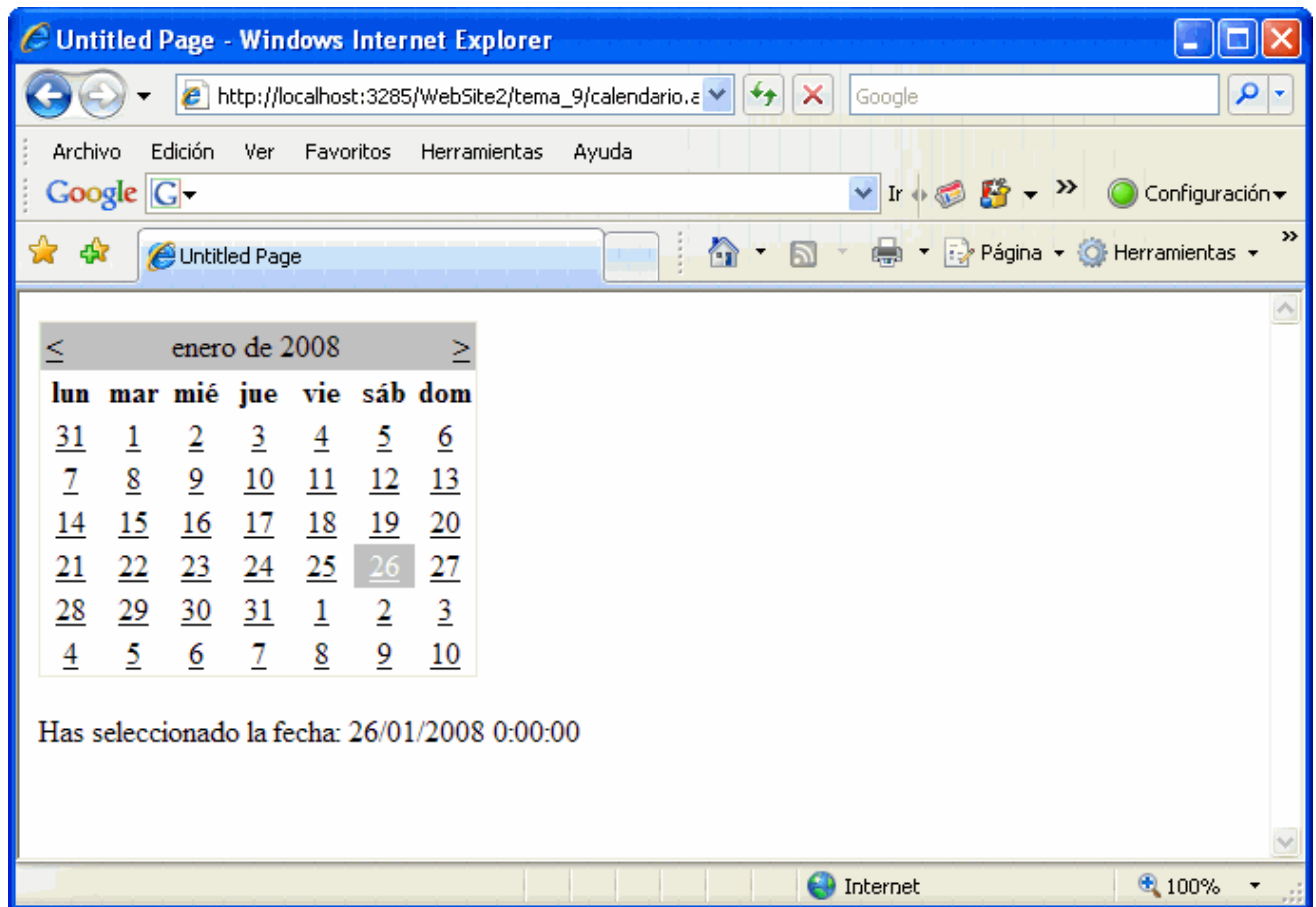
< enero de 2008 >						
lun	mar	mié	jue	vie	sáb	dom
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3
4	5	6	7	8	9	10

Has seleccionado la fecha: Label

Ahora pondremos en el evento adecuado que nos escriba la fecha seleccionada.. ¿cual puede ser el evento adecuado? Pues el que detecte que ha habido un cambio:

```
1
2 Partial Class tema_9_calendario
3     Inherits System.Web.UI.Page
4
5     Protected Sub Calendar1_SelectionChanged(ByVal sender As Object, ByVal e As System.EventArgs)
6         Label1.Text = Calendar1.SelectedDate.ToString
7     End Sub
8 End Class
9
```

Al ejecutar la página nos indicará la fecha seleccionada:



Fíjate que sencillo va a ser ahora que el usuario introduzca una fecha o un rango de fechas. Podemos indicarle el modo de selección con la propiedad "SelectionMode", si está a "Day" nos permite seleccionar un día, "DayWeek" nos seleccionará una semana y "DayWeekMonth" un mes. Fíjate como selecciono una semana:

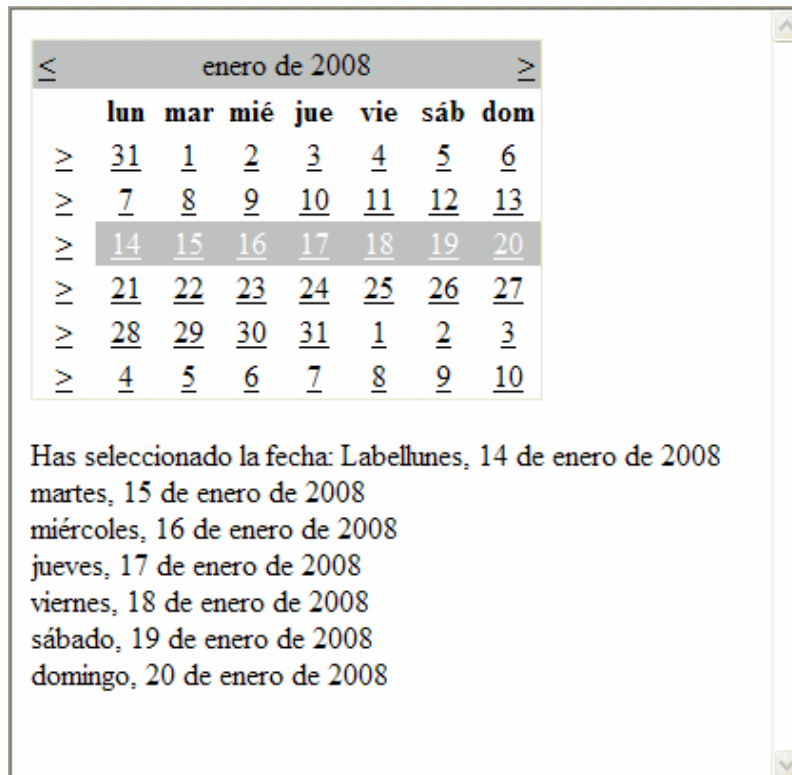


Cuando seleccionamos varias fechas debemos recorrerlas en nuestro código para poder procesarlas, así que en el caso de la selección múltiple pondremos en nuestro código un sencillo bucle "For each" que como sabes es para recorrer colecciones, ya que en este caso una selección múltiple nos va a devolver una colección de fechas:

## Controles avanzados y referencia de funciones.

```
Protected Sub Calendar1_SelectionChanged(ByVal sender As Object, By
    Dim fecha As DateTime
    For Each fecha In Calendar1.SelectedDates
        Label1.Text &= fecha.ToLongDateString & "<br />"
    Next
End Sub
```

Esta vez el método utilizado para la conversión ha sido "ToLongDateString" para que me extraiga solo la fecha y no fecha y hora como antes, ya que el tipo de datos "DateTime" almacena ambos datos\_



### 1.1 Formato del control

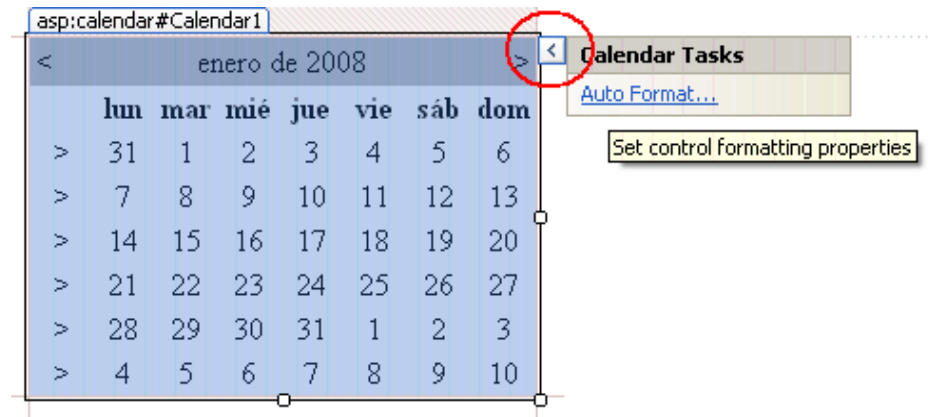
Disponemos de multitud de estilos para personalizar nuestro control:

Miembro	Descripción
DataHeaderStyle	Estilo para la sección que muestra los días de la semana
DayStyle	Estilo estándar para los días
NextPrevStyle	Estilo para para los controles de navegación de mes adelante/atrás
OtherMonthDayStyle	Estilo para los días que no son del mes actual que aparecen habitualmente en la primera y última semana del mes visible
SelectedDayStyle	Estilo para los días seleccionados
SelectorStyle	Estilo para la selección de dias/meses

## Controles avanzados y referencia de funciones.

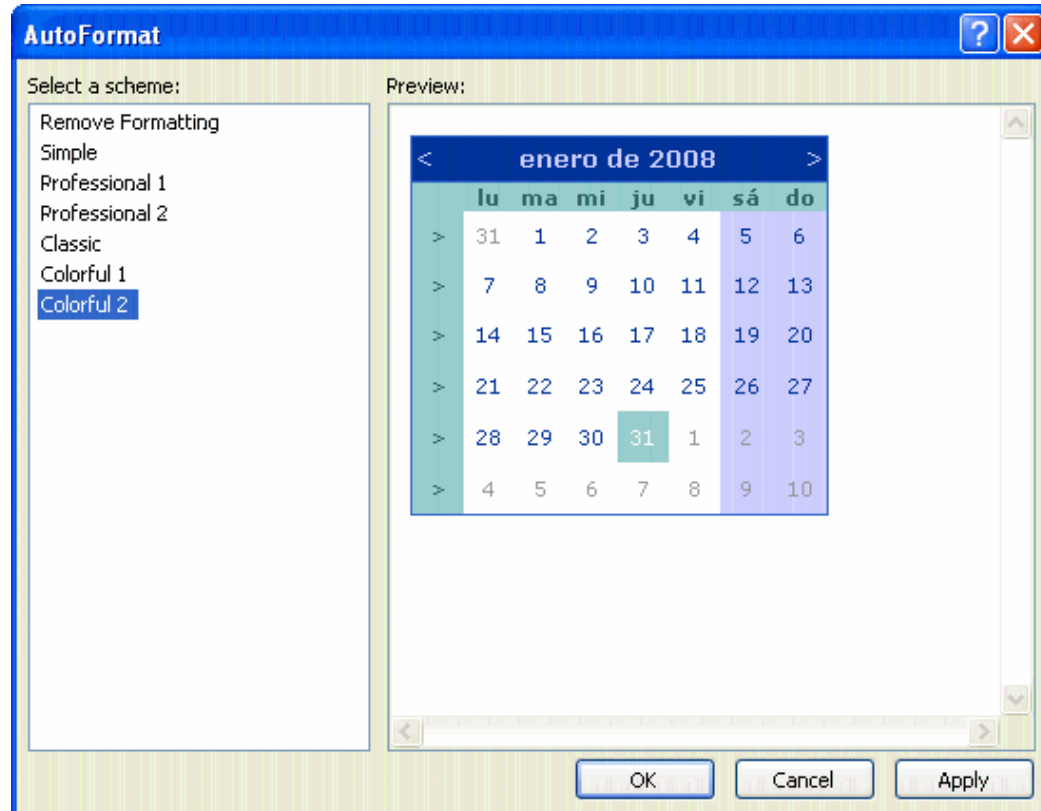
TitleStyle	Estilo para el título de la selección
TodayDayStyle	Estilo para el día seleccionado como hoy. En los calendarios siempre se muestra de forma distinta al día actual
WeekEndDayStyle	Estilo para los días del fin de semana

Por si fuera poco tenemos una pantalla para seleccionar estilos completos del control calendario, para esto pulsamos la flecha que aparece a la derecha y en la parte superior del control:



Has seleccionado la fecha: Label

Para que nos muestre el formato automático:



## Controles avanzados y referencia de funciones.

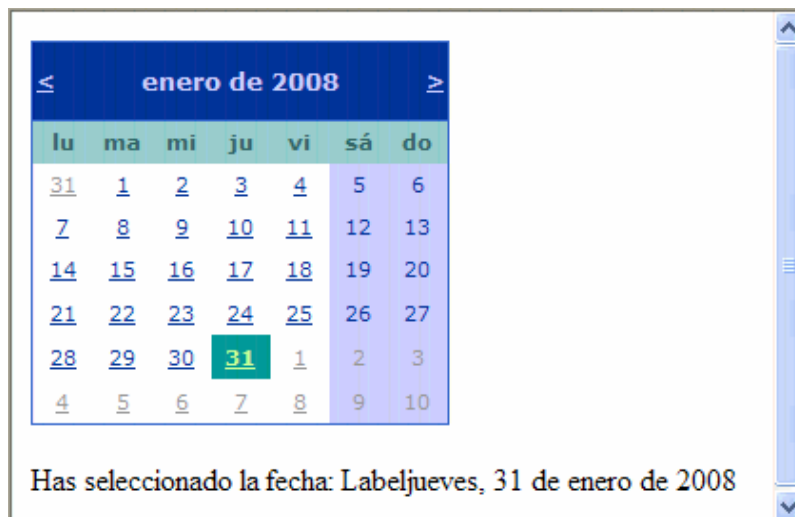
Practica un poco con las opciones que tiene porque son muchas: "ShowDayHeader, ShowTitle, ShowGridLines", ... Las propiedades que acaban en "Text" nos permitirán establecer el texto que se muestra en el control

### 1.2 Restringir fechas

En ocasiones queremos restringir alguna fechas porque el usuario no debe seleccionarlás, esto es una opción verdaderamente avanzada que supera incluso a las prestaciones de los calendarios de Windows, veamos como hacerlo. Para realizar estas restricciones las debemos activar cuando se esté construyendo el calendario, es decir cuando se esté haciendo el proceso de "Render" que es justo antes de mostrarse. Veamos estas instrucciones dentro de ese evento del calendario:

```
Protected Sub Calendar1_DayRender(ByVal sender As Object, ByVal e As Syst
    If e.Day.IsWeekend Then
        e.Day.IsSelectable = False
    End If
End Sub
```

Recuerda que "e" es un parámetro de entrada que aporta información adicional del control. En este caso información adicional del calendario cuando se está disparando el evento de "DayRender" es decir, se está generando y justo se va a escribir en la página. El resultado es esto, donde no podemos seleccionar los días del fin de semana:



Ciertamente potente pero te habrá desconcertado un poco la cantidad de métodos y propiedades de este control, pues si, tiene unas cuantas que le da una versatilidad total. Para tu tranquilidad te voy a describir las propiedades mas interesantes del "e.Day" que es una instancia de la clase "CalendarDay" :

Miembro	Descripción
Date	El objeto DateTime que representa esta fecha.
IsWeekEnd	Es cierto si los días son sábados o domingos
IsToday	Indica si es el día de hoy



## Controles avanzados y referencia de funciones.

IsOtherMonth	Es cierto si se ha pulsado en un día de la primera o última fila que corresponden a otro mes
IsSelectable	Indica si ese día se puede seleccionar o no.

Este evento es muy pero que muy interesante. Fíjate que podemos consultar en una base de datos los días del mes que hay eventos y así destacar aquí los que tienen, dejándolos como seleccionables, o destacando esos días que hay eventos e incluso escribiéndolos:

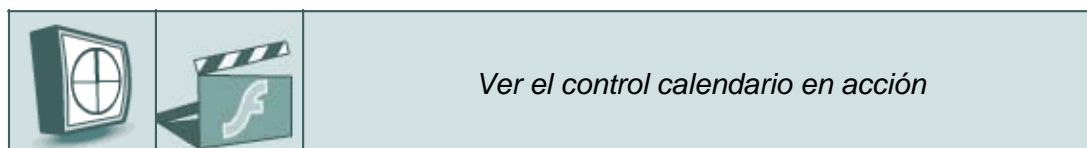
≤ marzo de 2008 ≥						
lu	ma	mi	ju	vi	sá	do
25	26	27	28	29	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19 Cumpleaños de Jose	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

Esto lo hemos conseguido simplemente comprobando si el día y mes es el que queremos y luego cambiando el color de la celda. Además le hemos añadido un control de tipo label a ese día. Con lo cual ya sabes también como añadir controles a este control, por lo tanto podemos añadir pequeñas imágenes (en lugar de label, img) con iconos de eventos. El código quedaría:

```
Protected Sub Calendar1_DayRender(ByVal sender As Object, ByVal e As Syst
    If e.Day.IsWeekend Then
        e.Day.IsSelectable = False
    End If
    If e.Day.Date.Day = 19 And e.Day.Date.Month = 3 Then
        ' Es mi cumpleaños!
        e.Cell.BackColor = Drawing.Color.Aqua
        Dim texto As New Label()
        texto.Text = "<br />Cumpleaños de Jose"
        e.Cell.Controls.Add(texto)
    End If
End Sub
```

Tenemos además dos eventos importantes, aunque uno ya lo has visto: SelectionChanged y VisibleMonthChanged. Que se producen cuando se selecciona una fecha o cuando se pulsa al mes siguiente. En ambos casos podemos querer realizar una consulta a una base de datos para actualizar los eventos de ese día por ejemplo, así que basta con escribir el código en esos eventos para actualizar los datos.

Ya solo queda practicar un poco, revisa las propiedades de "Caption", "CaptionAlign", "CellPadding", "CellSpacing", "Show..." son sencillas de utilizar y te van a ser muy útiles para dar formatos.



## 2. Control Addrotator

Este control es muy sencillo y su funcionamiento es casi tan antiguo como la Web, ya que simplemente nos va a mostrar un gráfico en pantalla de varios posibles. Por ejemplo en una página de propaganda y que van cambiando los anuncios... son de una lista predefinida y que muestra aleatoriamente con un campo de este tipo. Es un control sencillo pero que nos aportará mas experiencia así que a por él.

Podríamos hacer este proceso de una forma relativamente sencilla porque podemos con pocas líneas escoger un número aleatorio en nuestro "Page\_Load" y con él leer el nombre de un fichero de una lista pero vamos a hacerlo con este control por practicar y porque si queremos esto ya está hecho!. Para empezar, este control necesita un fichero XML con esta estructura en la que hay ya un ejemplo:

```

1  <?xml version="1.0" encoding="utf-8" ?>
2  <Advertisements>
3
4  <Ad>
5      <ImageUrl>fotos/foto_1.jpg</ImageUrl>
6      <NavigateUrl>http://www.microsoft.com</NavigateUrl>
7      <AlternateText>Mi Web</AlternateText>
8      <Impressions>1</Impressions>
9  </Ad>
10
11 </Advertisements>
    
```

Estas propiedades "ImageUrl", ... obviamente no las he puesto porque sí, sino porque son las propiedades que necesita definir este control y que son:

Elemento	Descripción
ImageUrl	La imagen que se va a mostrar.
NavigateURL	La página web que visitará cuando el usuario haga clic en la imagen
AlternateText	Texto que se mostrará al pasar el ratón por encima y cuando no se pueda mostrar la imagen
Impressions	Un valor numérico con la importancia de la imagen. Por ejemplo, una con el valor 10 se mostrará mas a menudo que otra con valor 5
KeyWord	Una palabra clave que identifica un grupo de imágenes. Se puede utilizar para filtrar por grupos.

### 2.1 La clase AdRotator

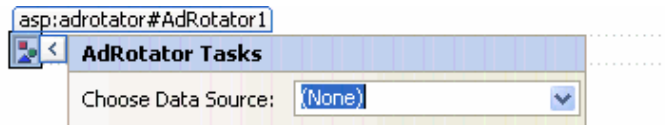
La clase AdRotator tiene muy pocas propiedades, debemos especificar el fichero de las imágenes a mostrar y el tipo de ventana que se abrirá cuando se haga clic, valor éste que puede ser:

Destino (Target)	Descripción
_blank	El enlace abre una nueva página sin marcos
_parent	El enlace se abre en la página "padre" de la actual (si tiene marcos)

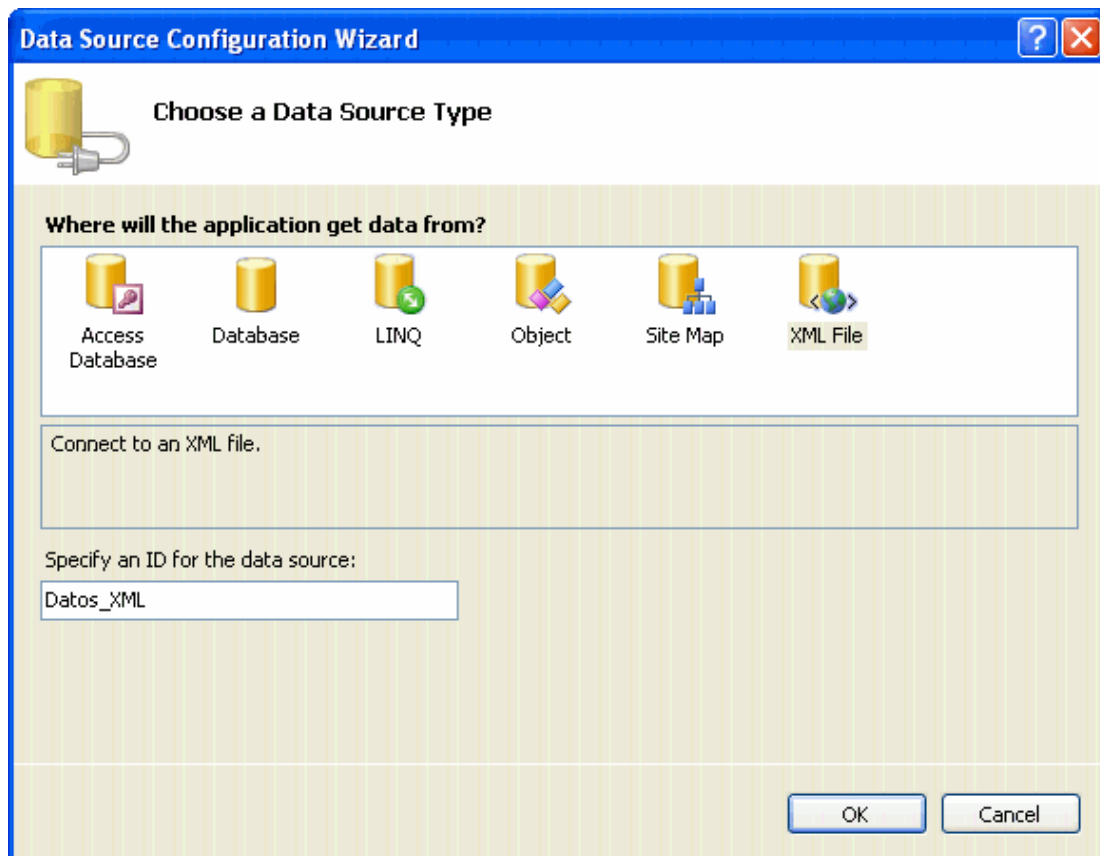
## Controles avanzados y referencia de funciones.

_self	Se abre en el marco actual
_top	Se abre en la página actual pero a pantalla completa, sin marcos

Además podemos añadir la propiedad "KeywordFilter" que hemos visto antes, por ejemplo, porque durante este día queremos mostrar las imágenes relativas a navidades. Vamos a crear una página en blanco y añadimos este control:

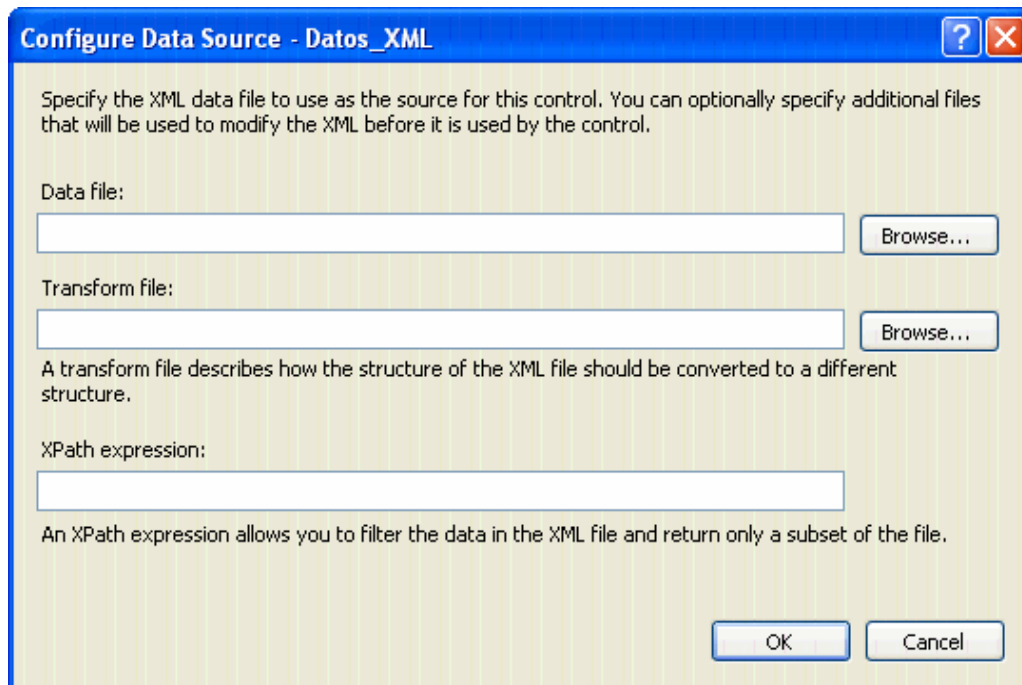


De momento nos solicita un origen de datos, es decir, dónde está la lista de las imágenes o fichas que debe mostrar. Pulsamos en el desplegable y seleccionamos Nuevo "Data Source" u origen de datos. Esta pantalla nos va a ser muy familiar cuando trabajemos con bases de datos. De momento un control que necesita enlazarse con un origen de datos, como es el fichero XML con las fichas a mostrar, nos solicita este origen de datos. Seleccionaremos en este caso un fichero "XML":

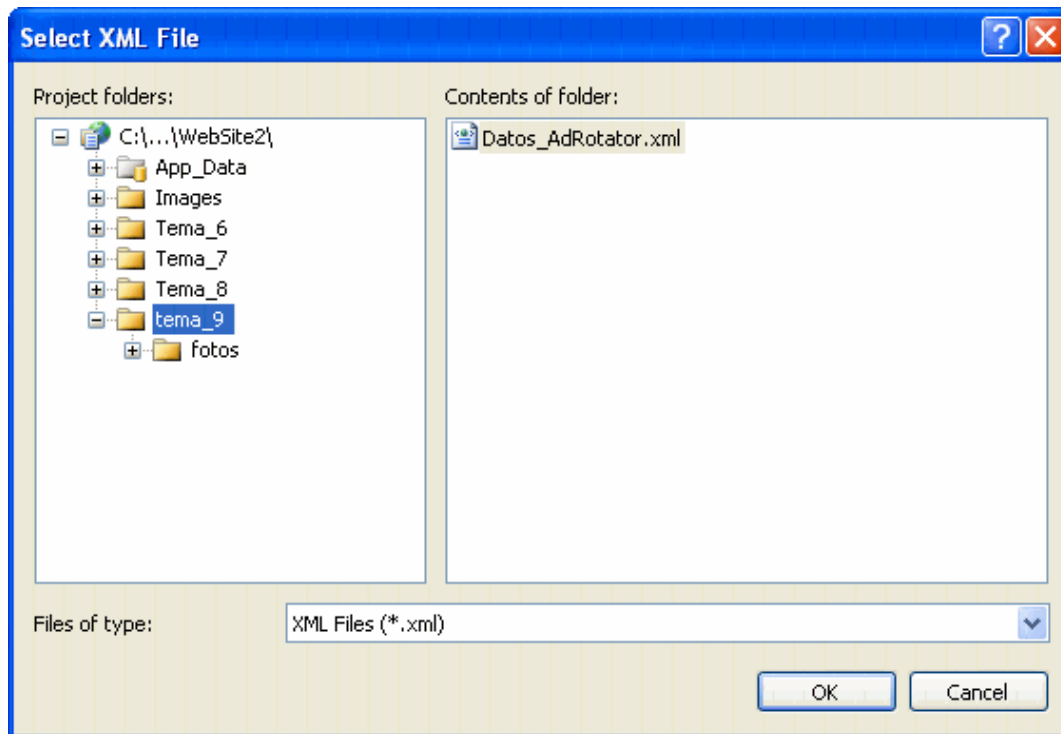


Y le ponemos debajo un nombre para tener definido este origen de datos. A continuación y dependiendo del origen de datos nos solicitará valores para configurarlo, en este caso nos solicita:

## Controles avanzados y referencia de funciones.



Pulsamos en el botón para buscar el fichero XML que hemos creado antes:

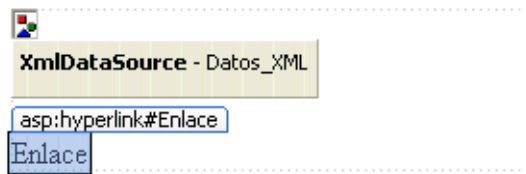


Lo seleccionamos y pulsamos en "Ok" y una vez mas para terminar con el control, ya que esto es todo lo que necesita. Si ejecutamos la página veremos que todo funciona y nos muestra una imagen, que será la misma porque en el fichero XML sólo hemos puesto una entrada.

Como eventos tenemos solo uno destacable que es "AdRotator.AdCreate" que se da cuando se ha elegido la imagen y ficha a mostrar y justo se va a mandar al navegador, similar al "render" del calendario. Así que podremos modificar ahora alguna propiedad de esta control, por ejemplo podemos poner un control de tipo

## Controles avanzados y referencia de funciones.

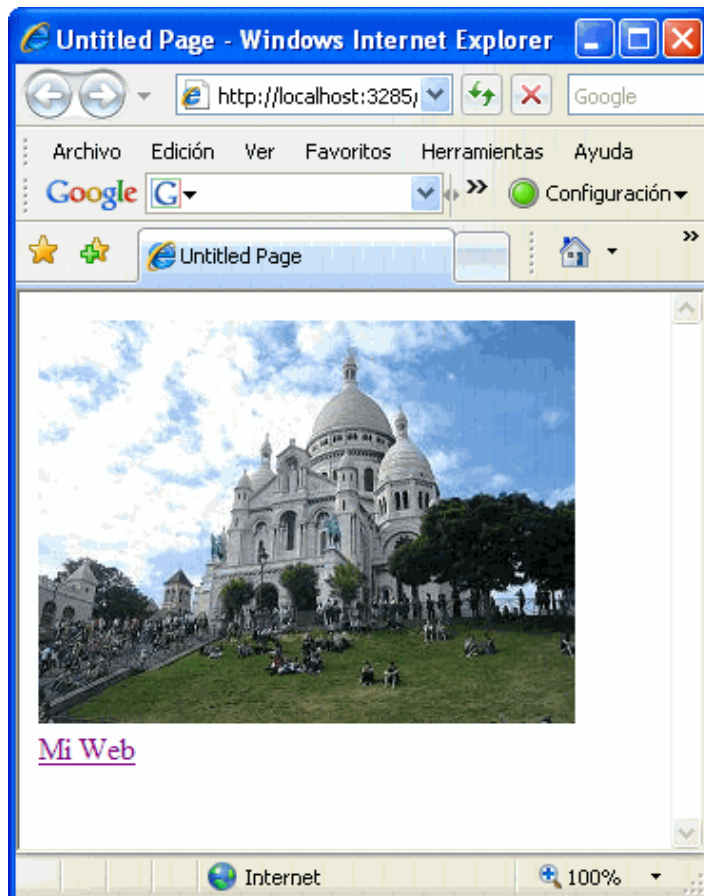
"hipervínculo" debajo con los enlaces y textos del control:



Y escribir en ese hipervínculo los datos de la imagen que se va a mostrar ahora:

```
Protected Sub AdRotator1_AdCreated(ByVal sender As Object, ByVal e As System.Web  
    'Escribimos que el hiperenlace va a ser el mismo que el de la foto  
    Enlace.NavigateUrl = e.NavigateUrl  
  
    'Escribos el texto  
    Enlace.Text=e.AlternateText|  
End Sub
```

Recuerda que el objeto "e" de todos los controladores de eventos de los controles tiene las propiedades del control que está manejando, por tanto tiene las propiedades del AdRotator. Si ejecutamos la página:



Todo funciona perfectamente. Sigamos con mas cosas avanzadas de los controles, pasemos a trabajar con paneles....

### 3. Páginas con varias vistas

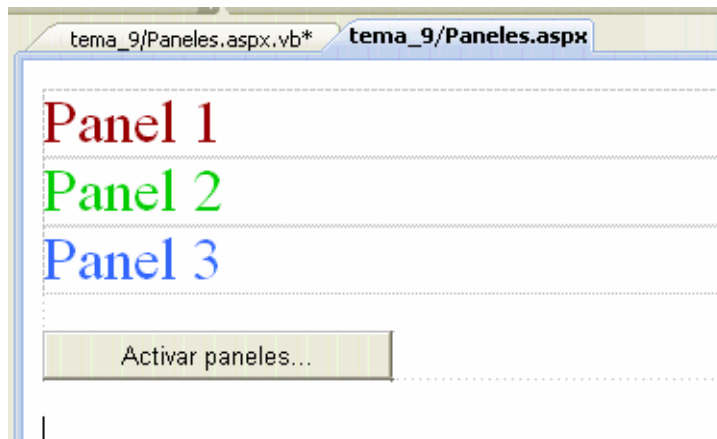
En un sitio web es normal moverse por páginas con distintos formatos, por ejemplo en una tienda en la que vemos el detalle de un producto en otra página. Esto es una estructura interesante porque nos permite dividir el código en las páginas utilizando distintos métodos como vimos en capítulos atrás para pasar información entre páginas.

Sin embargo en ocasiones sería mejor disponer de una misma página para realizar distintas tareas. Por ejemplo para mostrar varias vistas distintas de los mismos datos: en una cuadrícula o tabla y en una lista, permitiendo al usuario cambiar de una vista a otra de una forma sencilla. Necesitaremos entonces páginas dinámicas para realizar estas distintas vistas. La idea es mostrar y ocultar controles dependiendo de la vista seleccionada por el usuario, para esto utilizaremos los "paneles". Los controles de tipo "Panel" pueden contener un grupo de controles luego son idóneos para mostrar u ocultar controles al usuario. Pondremos varios paneles en nuestra página uno encima de otro, como sólo estará uno con la propiedad "visible" activada no hay problema que en el diseño los solapemos. En el código pondríamos así los paneles:

```
<form id="form1" runat="server">
<div>
<asp:Panel ID="panel_1" runat="server"> ... </asp:Panel>
<asp:Panel ID="panel1" visible="false" runat="server"> ... </asp:Panel>
<asp:Panel ID="panel2" visible="false" runat="server"> ... </asp:Panel>
</div>
</form>
```

Cuando ponemos un control como oculto no se envía a la página web. Es decir, no es que se envía al navegador del cliente y luego no se muestra sino que al estar oculto simplemente no se envía. Si el usuario hace clic en algún control para visualizar otro panel ya lo enviará el evento adecuado.

Vamos a poner un control de tipo label de prueba en cada panel y debajo un botón que nos va a permitir movernos por los paneles:



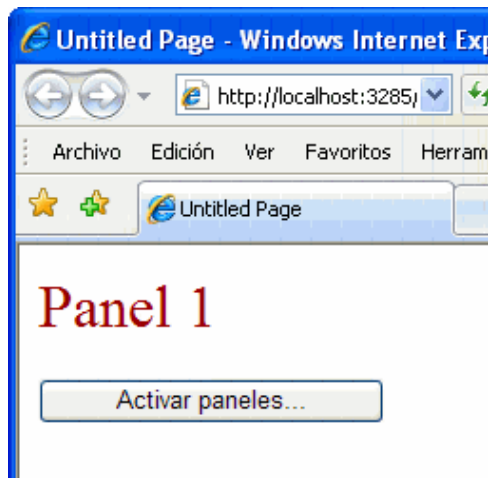
La idea es que inicialmente en el Page\_Load marquemos uno como visible y luego al pulsar el botón vayamos intercambiando el estado visible de ellos. Por ejemplo si pulsamos el botón y está visible el primer panel, lo ocultaremos y haremos visible el segundo:

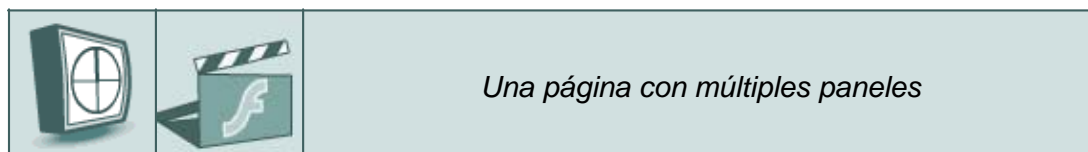
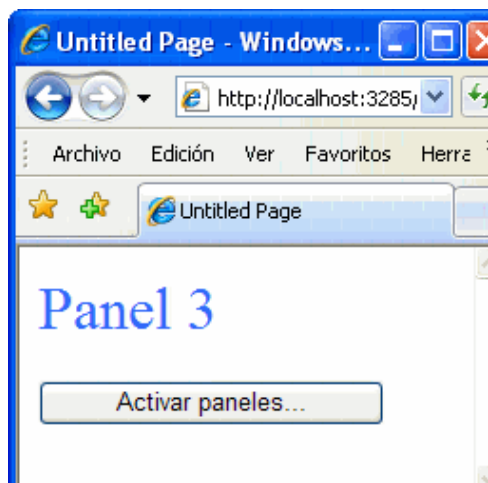
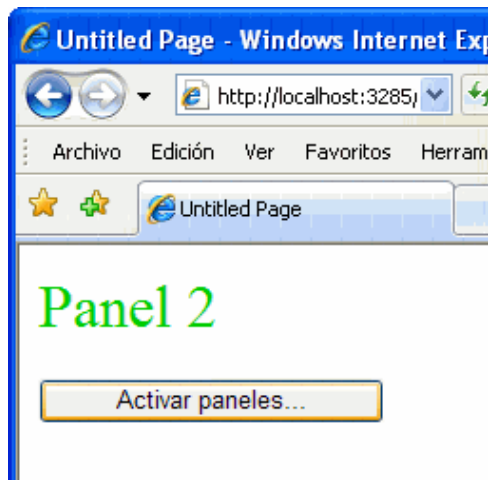
## Controles avanzados y referencia de funciones.

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles MyBase.Load
    'Para la primera carga activamos el primerpanel
    If Not IsPostBack Then
        panel_1.Visible = True
        panel_2.Visible = False
        panel_3.Visible = False
    End If
End Sub

Protected Sub btn_panel_Click(ByVal sender As Object, ByVal e As EventArgs) Handles btn_panel.Click
    If panel_1.Visible = True Then
        panel_1.Visible = False
        panel_2.Visible = True
        panel_3.Visible = False
    ElseIf panel_2.Visible = True Then
        panel_1.Visible = False
        panel_2.Visible = False
        panel_3.Visible = True
    ElseIf panel_3.Visible = True Then
        panel_1.Visible = True
        panel_2.Visible = False
        panel_3.Visible = False
    End If
End Sub
```

y el resultado está en que podemos ver sucesivamente los tres controles:





Esto se parece mucho a los "asistentes" que es cuando nos presenta unas opciones y le damos al botón de avanzar para seguir seleccionando opciones y por fin realizar un proceso, por ejemplo, una consulta a una base de datos. Pero ya ves que es un poco laborioso esto de mantener visibles y ocultos los paneles. Por suerte ASP.NET también nos ayuda con esto, y son los controles: "MultiView" y "Wizard". Como ejemplo mejoraremos nuestro generador de felicitaciones que hicimos capítulos atrás...

### 3.1 Control "MultiView"

Este control permite definir varias vistas y tener solo una visible. La definición sería como:

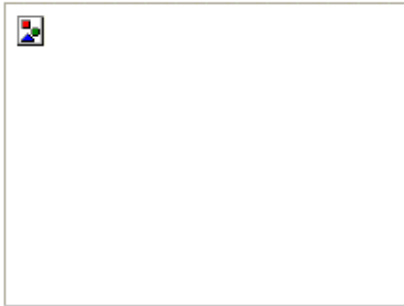
```
<asp:MultiView id="Multivista" runat="server">  
    <asp:View id="Vista1" runat="server">...</asp:View>  
    <asp:View id="Vista2" runat="server">...</asp:View>
```



## Controles avanzados y referencia de funciones.

```
<asp:View id="Vista3" runat="server">...</asp:View>
</asp:MultiView>
```

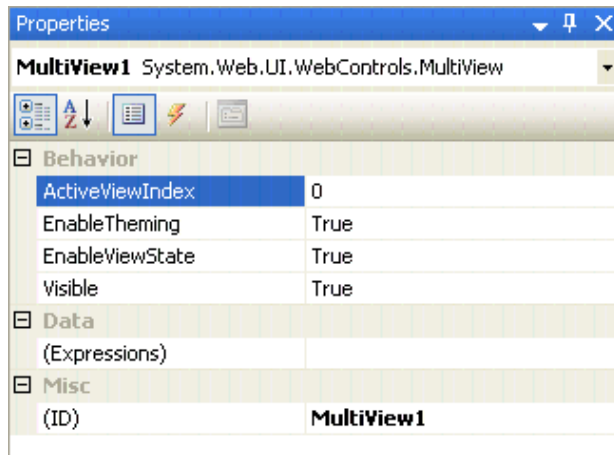
Ya ves que la idea es sencilla: un control principal "MultiView" y luego una serie de vistas "View" anidadas. En nuestro editor primero añadiremos un control "MultiView" y luego añadiremos dentro de él 3 de tipo "View", todo esto en nuestro ejemplo de la tarjeta de felicitación que vimos capítulos atrás:

MultiView1	
<div>View1</div> <div>Elige un color de fondo:</div> <div>Unbound</div> <div>Elige una fuente:</div> <div>Unbound</div> <div>Siguiente &gt;</div>	<div>[Felicitacion]</div> <div></div>
<div>View2</div> <div>Estilo del contorno:</div> <div><input type="radio"/> Unbound</div> <div><input type="checkbox"/> Añadir gráfico</div> <div>&lt; Anterior</div> <div>Siguiente &gt;</div>	
<div>View3</div> <div>Tamaño de letra:</div> <div></div> <div>Texto de la felicitación:</div> <div></div> <div>&lt; Anterior</div> <div>Aplicar</div>	

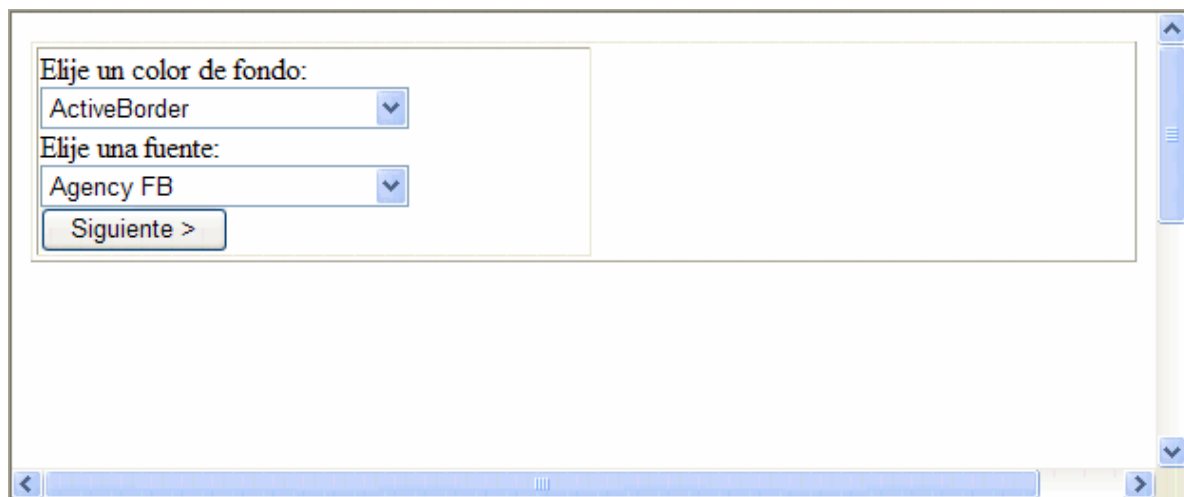
Los cambios han sido muchos y la verdad es que es un poco laborioso pero el resultado merece la pena. He pintado una tabla de una fila y dos columnas, he puesto los controles de la izquierda en la celda de la izquierda y los demás en la derecha. Luego he añadido un control "MultiView" y dentro de él tres de tipo "View". Luego he puesto los controles como ves, en tres grupos dentro de cada vista y finalmente he puesto unos botones de "siguiente", "anterior" y "aplicar" para poder moverse por los tres paneles.

Curiosamente la ejecución de esta página devuelve una página en blanco porque no hay una vista activa dentro del "multiview" así que debemos activar una de ellas en las propiedades de la "multiview":

## Controles avanzados y referencia de funciones.



Poniendo la propiedad "ActiveViewIndex" a 0. Entonces si:



Ahora debemos poner código en el botón siguiente para que muestre la siguiente vista, obviamente solo hay que poner ActiveViewIndex a "0", "1" ó 2

```
Protected Sub btn_siguiente1_Click(ByVal sender As Object, ByVal e As EventArgs)
    MultiView1.ActiveViewIndex = 1
End Sub
```

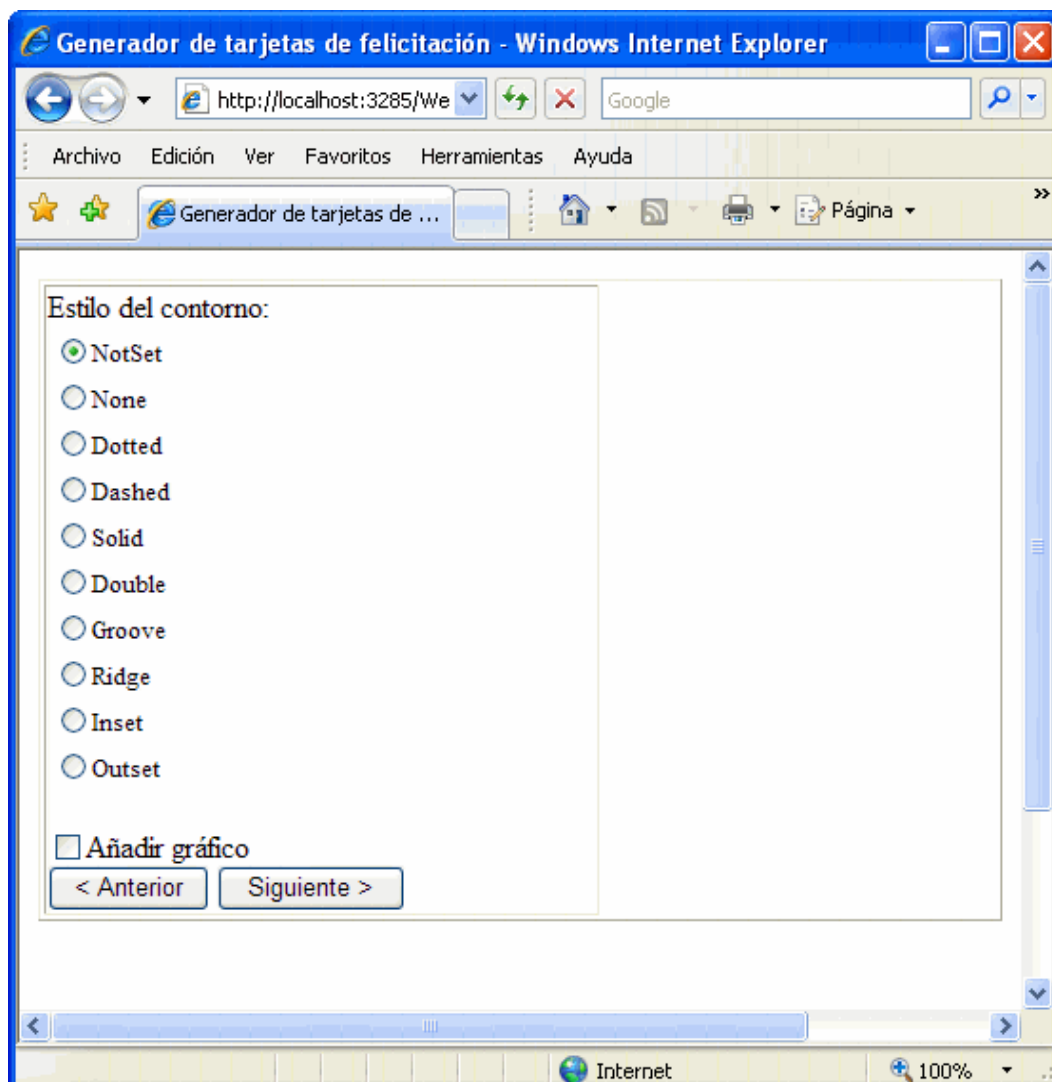
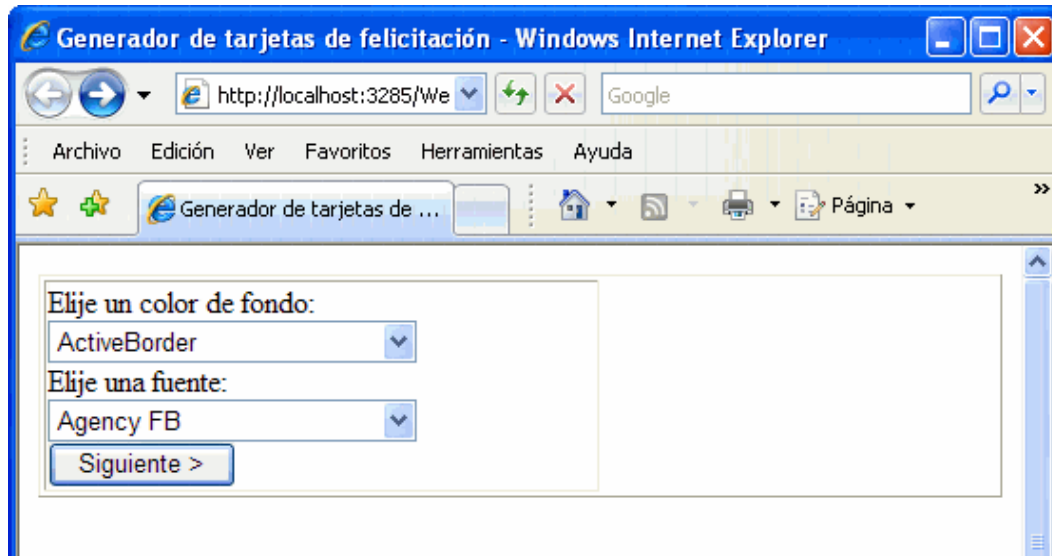
```
Protected Sub btn_siguiente2_Click(ByVal sender As Object, ByVal e As EventArgs)
    MultiView1.ActiveViewIndex = 2
End Sub
```

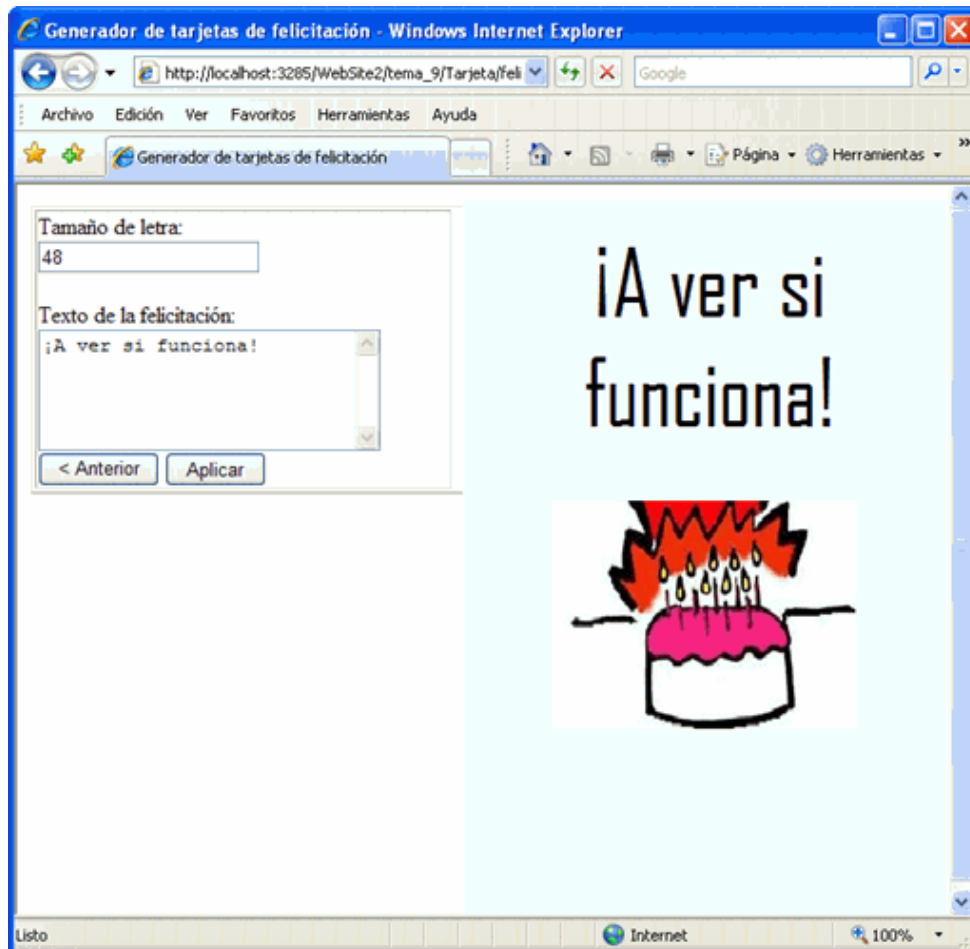
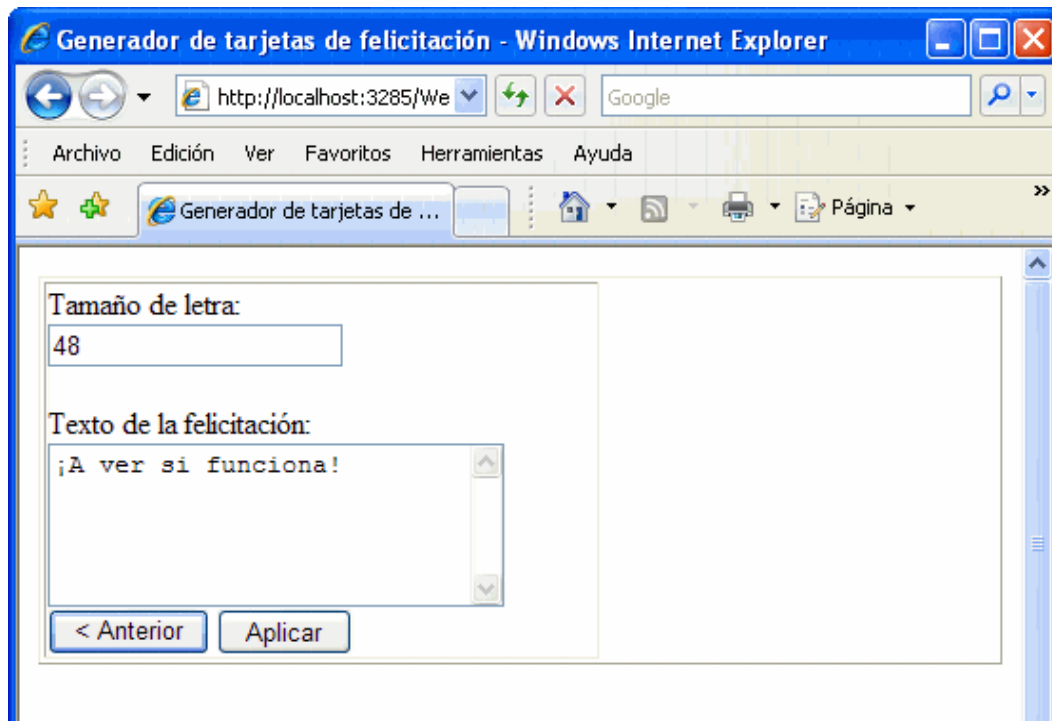
```
Protected Sub btn_anterior1_Click(ByVal sender As Object, ByVal e As EventArgs)
    MultiView1.ActiveViewIndex = 0
End Sub
```

```
Protected Sub btn_anterior2_Click(ByVal sender As Object, ByVal e As EventArgs)
    MultiView1.ActiveViewIndex = 1
End Sub
```

## Controles avanzados y referencia de funciones.

Con lo que nos funcionará la secuencia completa al poner en el último botón de "Aplicar" la ejecución de la tarjeta, mira la secuencia:





## Controles avanzados y referencia de funciones.

Los botones anterior y siguiente nos han funcionado perfectamente y además los datos que hemos escrito en los controles han permanecido porque en realidad no los estamos creando y destruyendo sino que los estamos ocultando cuando no está el índice del panel visible.

Pero aun hay mas ventajas, los botones que hemos puesto a mano los podíamos haber evitado ya que este control multivista dispone de unos opciones para pasar automáticamente de panel:

Nombre del comando	Campo	Descripción
PrevView	PreviousViewCommandName	Se mueve a la vista anterior
NextView	NextViewCommandName	Se mueve a la siguiente vista
SwitchViewByID	SwitchViewByIDCommandName	Se mueve a la vista con el identificador indicado
SwitchViewByIndex	SwitchViewByIndexCommandName	Se mueve a la vista con el índice especificado.

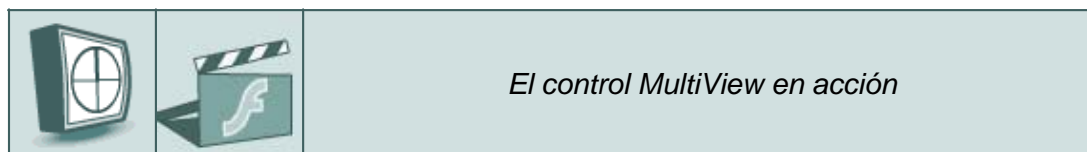
Con lo que simplificamos un poco el código, lo que hemos hecho ha sido utilizar el último método para movernos por las solapas. Pero si son muchas tendríamos que repetir mucho estas instrucciones, una por cada botón de cada vista. El botón de la primera vista debería ser así:

```
<asp:Button ID="boton1" runat="Server" CommandArgument="View2"
            CommandName="SwitchViewByID" Text="Siguiente" />
```

Por tanto podremos poner también para los botones anterior/siguiente:

```
<asp:Button ID="boton1" runat="Server" CommandName="PrevView" Text="< Anterior" />
<asp:Button ID="boton1" runat="Server" CommandName="NextView" Text="> Siguiente" />
```

"NextView" En el primer caso como hemos elegido indicar el "ID" "SwitchViewByID" se lo indicamos en el argumento "CommandArgument" para que vaya al que le estamos indicando. En los ejemplos de abajo no hace falta porque "PrevView" y "NextView" ya le indican directamente a que página debe ir....



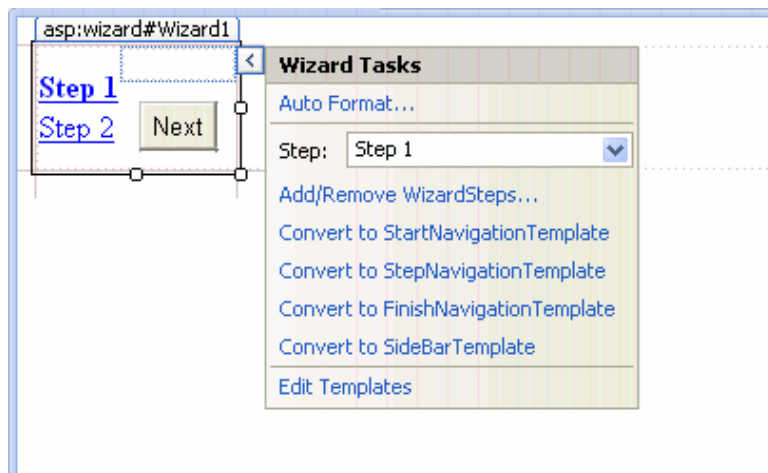
## 4. El asistente o control Wizard

EL asistente es como la multivista de antes pero en plan de lujo ya que incorpora por si mismo la navegación entre los paneles, barras, estilos, plantillas... así que será mas fácil de utilizar para este proceso tan concreto como es el asistente.

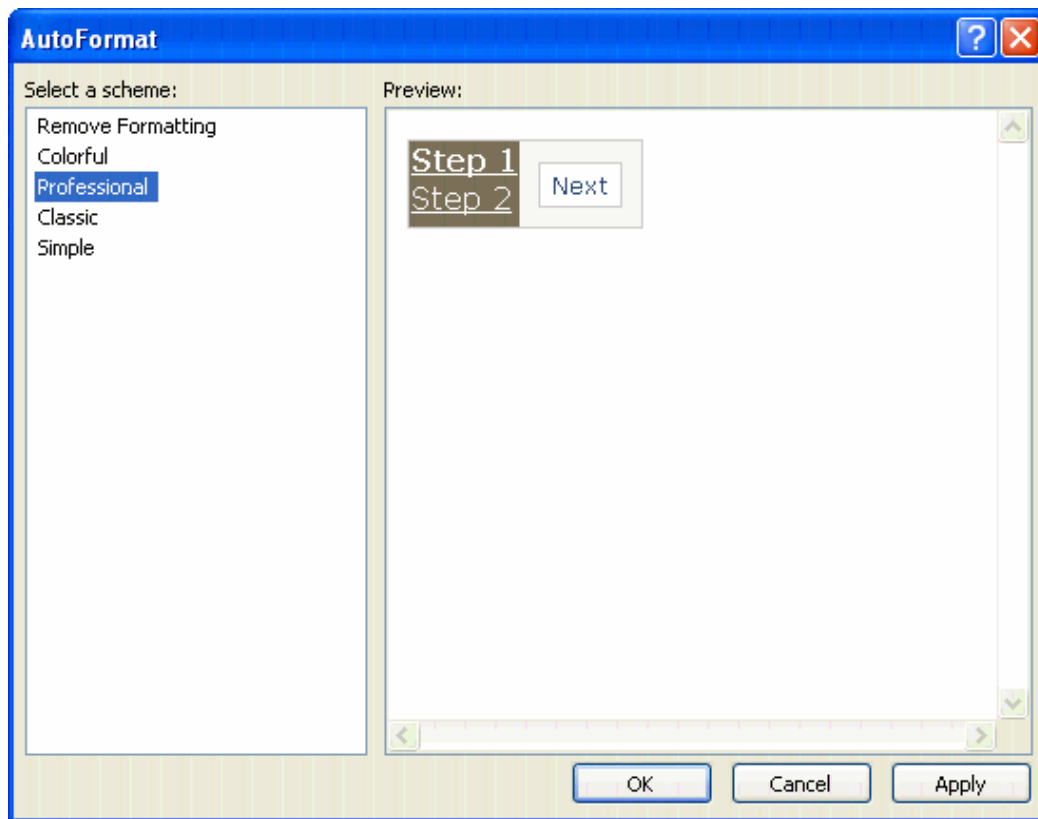
Los asistentes normalmente representan una única tarea que se ha ido configurando por medio de las pantallas intermedias. En nuestro control haremos esta navegación pero con muchas mas posibilidades por ejemplo movernos a una pantalla en particular o saltarnos alguna dependiendo de la selección del usuario, que se le llama navegación no lineal.

## Controles avanzados y referencia de funciones.

Veamos un control de este tipo en una página en blanco:



Podemos ponerle ya un autoformato para que nos muestre un aspecto mas elegante:



Y como hemos visto antes se intuye como serán los tipos de pestañas porque uno ponía "StarNavigationTemplate", otro "StepNavigationTemplate" y el otro "FinishNavigationTemplate". Es decir nos pregunta cuales será el primer panel, los intermedios y el final. ¿para que? Pues muy sencillo, así sabrá en que paneles o pasos debe poner los botones de siguiente y anterior. Obviamente en el primer panel solo pondrá el botón de "Siguiente". Además disponemos de la barra de navegación que nos permitirá movernos por los paneles.

En código sería como esto:

## Controles avanzados y referencia de funciones.

```
<asp:Wizard id="Asistente" runat="server">

    <WizardSteps>

        <asp:WizardStep Title="Pasol" runat="server">

            ...

        </asp:WizardStep>

        <asp:WizardStep Title="Pasol" runat="server">

            ...

        </asp:WizardStep>

        ...

    </WizardSteps>

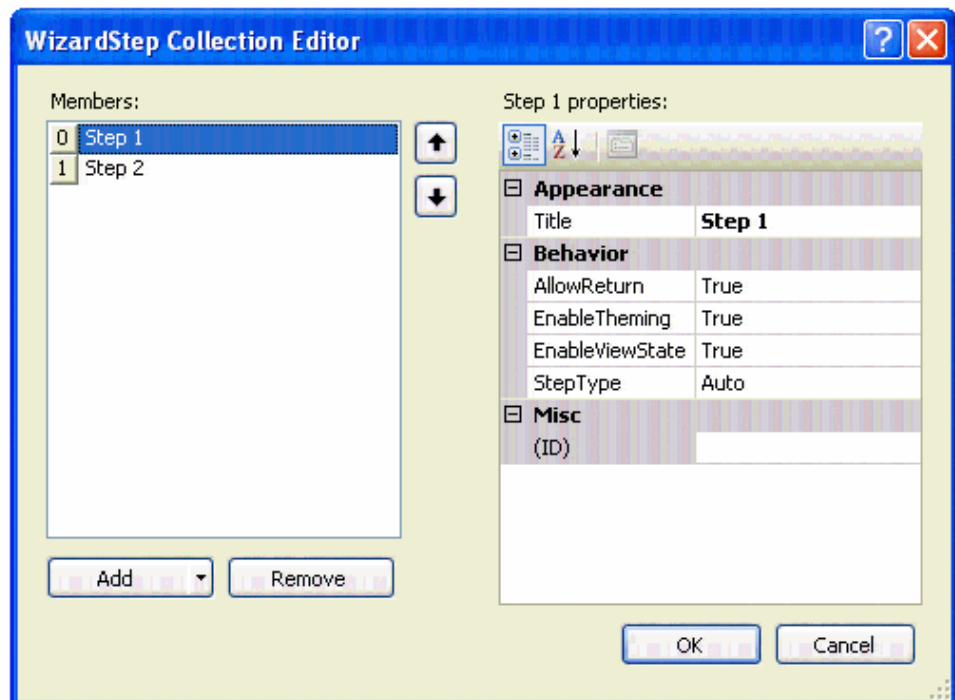
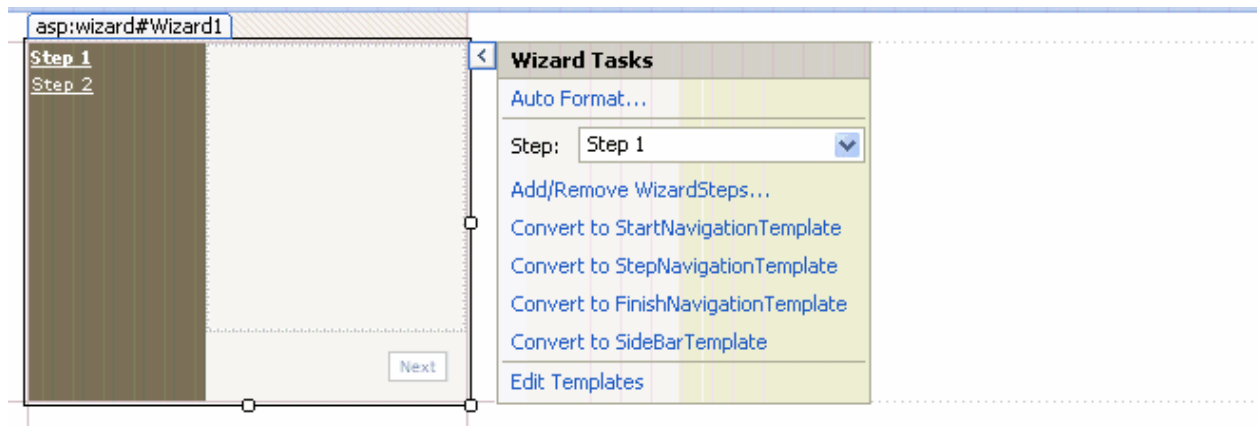
</asp:MultiView>
```

Una sintaxis jerárquica muy parecida a la del "MultiView" con sus vistas que vimos antes. El funcionamiento será muy sencillo, pero vamos a tener en cuenta estas propiedades:

Propiedad	Descripción
Title	Nombre para el paso del asistente, este nombre se utiliza como texto en la barra de navegación
StepType	Indica el tipo de paso que será un valor de la enumeración "WizardStepTye" (recuerda las enumeraciones de colores, ... siempre ha listas de estas constantes para poner tipos o estilos). Este valor determina el tipo de navegación de los botones: "Start" que mostrará un botón "Siguiete", "Step" mostrará "Siguiete" y "Anterior". "Finish" solo mostrará el botón "Anterior", "Complete" oculta los botones y la barra de navegación. El valor predeterminado es "auto" que pondrá los botones según sea el número de panel o paso estemos editando.
AllowReturn	Indica si el usuario puede retroceder a este paso. Si es "False" una vez que ha pasado por este paso no pondrá volver a él.

Si nos fijamos en el editor, veremos que ya hay dos pasos creados "Setp1" y "Setp2", si pulsas en la opción "Add/Remove WizardSteps", te mostrará la página para añadir pasos:

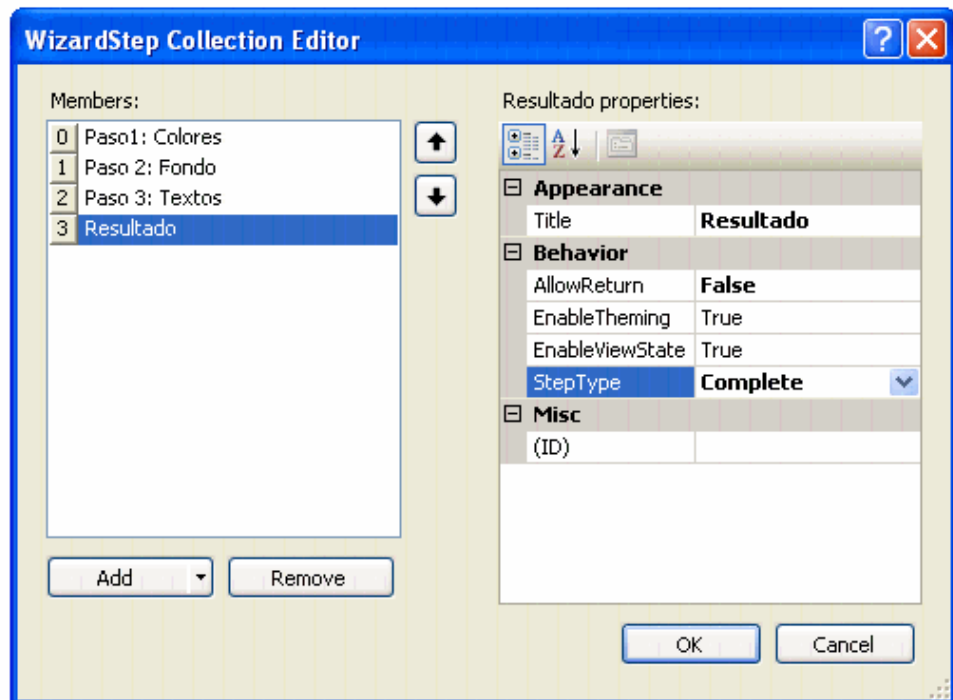
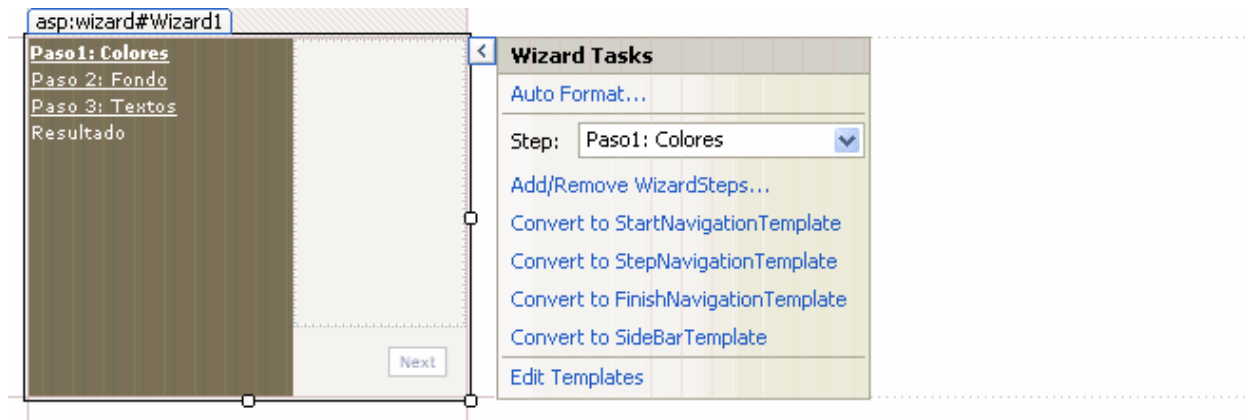
## Controles avanzados y referencia de funciones.



Como ves debajo tienes los botones de "Add". Así que vamos a crear 4 pasos, tres para las distintas opciones de nuestro generador de tarjetas y el cuarto para escribir la tarjeta de felicitación resultante. Está claro que esta última será del tipo "Complete" para que no nos muestre botones ya que hemos dado por finalizado el asistente. Añadimos pues 4 pasos:



## Controles avanzados y referencia de funciones.



Ahora dentro de cada paso pondremos los controles que queremos mostrar. Así que poco a poco los vamos colocando:

## Controles avanzados y referencia de funciones.

Este control no realiza "postback" hasta que no termina el último botón de "Terminar" así que no hay "postback" automáticos. Nos queda saber algo importante y es que eventos tenemos para controlar este "asistente", veamos cuales son:

Evento	Descripción
ActiveStepChanged	Sucede cuando pasamos de una pantalla a otra en el asistente.
CancelButtonClick	Sucede cuando se pulsa el botón "Cancelar" Este botón no se muestra por defecto pero lo podemos activar mediante la propiedad "Wizard.DisplayCancelButton"
FinishButtonClick	Se activa cuando se pulsa el botón "Finish"
NextButtonClick y PreviousButtonClick	Se dan cuando se pulsan en los botones de siguiente y anterior.
SideBarButtonClick	Cuando se pulsa un botón de la barra de pasos lateral.

Tenemos dos formas de funcionar, una que nos permite flexibilidad para volver atrás en los datos en la que activaremos "AllowReturn=True" donde realizaremos el proceso al final del asistente y otra en la que podremos realizar algunos cambios según las opciones solicitadas. Como detectamos el evento del paso de página podemos meter código en cualquier paso, en esos casos es mejor desactivar el paso atrás ya que la pantalla se ha configurado con las nuevas opciones, pero esto son solo recomendaciones que tu juzgarás en tu asistente.

Está claro donde tendremos que generar nuestra tarjeta: en el evento "FinishButtonClick" ya que es el último paso así que en el código:

## Controles avanzados y referencia de funciones.

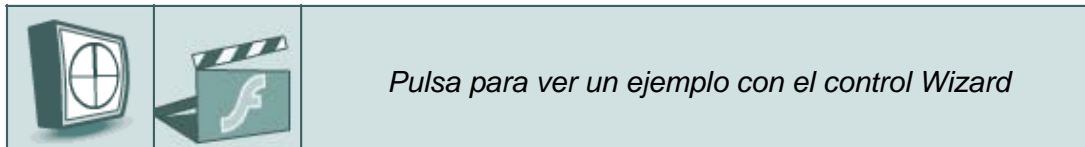
```
Protected Sub Wizard1_FinishButtonClick(ByVal sender As Object, ByVal e As Sys
    ' Actualizamos el color
    panel_tarjeta.BackColor = Color.FromName(lista_colores.SelectedItem.Text)

    ' Actualizamos el tipo de letra
    Felicitacion.Font.Name = Lista_fuentes.SelectedItem.Text
    ...    ...
```

En cuanto a los estilos tienes muchos para aplicar, hemos utilizado el generador de estilos para poner uno pero tienes a tu disposición un buen número de opciones para poner los estilos de todos los elementos que conforman un asistente. Como esto es fácil, lo dejo para que lo practiques tu y así puedas poner los botones en español!.

Para terminar veamos como cancelar la navegación. ¿para que? pues porque no ha escrito un valor obligatorio y no debe seguir al siguiente paso, así que lo pondremos en el cuadro de texto del tamaño de la fuente ya que no debe estar en blanco. Aunque podría poner un control de validación mejor veamos como cancelar la navegación para que el usuario meta el valor:

```
Protected Sub Wizard1_NextButtonClick(ByVal sender As Object, _
    ByVal e As System.Web.UI.WebControls.WizardNavigationEventArgs) Handles Wizard1.NextButtonClick
    'Vemos en que paso está y si el cuadro de texto está en blanco:
    If e.NextStepIndex = 1 AndAlso txt_tam_fuente.Text = "" Then
        'Cancelamos la navegación!
        e.Cancel = True
        lb_info.text = "No puedes continuar hasta que no hayas un tamaño de fuente"
    End If
End Sub
```



Te recuerdo lo de otras veces "e" es un parámetro que contiene valores adicionales del control del que esté manejando el evento. En este caso es un Wizard así que tendrá eventos de este control.

## 5. Dibujar con ASP.NET

Si has realizado el curso de VB.NET de esta misma plataforma o conoces este lenguaje, sabrás que .NET tiene multitud de objetos para dibujar. Como no podía ser menos en ASP.NET podremos utilizar algunos de ellos para cargar imágenes, manipularlas y en definitiva, crear gráficos dinámicos. Todo esto lo haremos con GDI+ que es el motor gráfico de .NET

### 5.1 Dibujo básico

Para crear un gráfico primero tendremos que en memoria un "bitmap" que es un gráfico digamos vacío. (una foto o un logotipo es un gráfico bitmap, o mapa de bits). Este gráfico vacío será nuestro escenario para dibujar, para esto utilizaremos la clase Bitmap contenida en "System.Drawing":

## Controles avanzados y referencia de funciones.

'Creamos en memoria un mapa de bits (bitmap) donde realizaremos nuestros dibujos.

' Tiene un tamaño de 400 píxeles de ancho x 100 de alto:

```
Dim imagen as new bitmap (400,100)
```

El siguiente paso es crear el contexto para esta imagen, este contexto ya se referirá al objeto imagen que hemos creado para pintar luego en él con los objetos del GDI+. Este contexto representa un objeto "System.Drawing.Graphics". Si queremos crear un objeto "graphics" de otro objeto bitmap ó mapa de bits existente utilizaremos el método compartido:

```
Dim g as Graphics = Graphics.FromImage(Imagen)
```

Ahora viene la parte interesante. Con los métodos de la clase "graphics" podremos dibujar: líneas, arcos, figuras geométricas, imágenes, ... Veamos una lista de métodos que tenemos en esta clase. Los que empiezan con la palabra "Fill" dibujan regiones sólidas excepto "DrawString()" que dibuja texto con la fuente especificada.

Métodos	Descripción
DrawArc()	Dibuja un arco que representa una parte de una elipse especificada por dos coordenadas, una altura y una anchura
DrawBezier() y DrawBeziers()	Dibuja una curva Bezier definida por los puntos de control
DrawClosedCurve()	Dibuja una curva cerrada uniendo los puntos
DrawCurve()	Dibuja una curva
DrawEllipse()	Dibuja una elipse definida por el rectángulo especificado por las coordenadas de anchura y altura
DrawIcon() y DrawIconUnstretched()	Dibuja el icono representado por un objeto "icon" y opcionalmente lo ajusta a un rectángulo
DrawImage() y DrawImageUnscaled()	Dibuja la imagen representada por un objeto "image" y opcionalmente lo ajusta a un rectángulo
DrawLine() y DrawLines()	Dibuja una o mas líneas, cada línea conecta los dos puntos especificados en las coordenadas
DrawPie()	Dibuja una porción de "tarta" de un gráfico de tartas, especificado como una elipse con altura y anchura y dos líneas radiales
DrawPolygon()	Dibuja un polígono multicara definido por una matriz de puntos
DrawRectangle() y DrawRectangles()	Dibuja un rectángulo, Cada rectángulo está definido por un par de coordenada, una anchura y una altura.
DrawString()	Escribe una cadena con la fuente especificada
DrawPath()	Dibuja una figura compleja definida por el objeto "Path"
FilledClosedCurve()	Dibuja una curva, las conecta por sus puntos y la rellena
FillEllipse()	Rellena el interior de una elipse
FillPie()	Rellena el interior de una "tarta" (porción de un gráfico de tipo tarta)

## Controles avanzados y referencia de funciones.

FillPolygon()	Rellena el interior de un polígono
FillRectangle() y FillRectangles()	Rellena el interior de uno o mas rectángulos
FillPath()	Rellena el interior de una figura compleja definida por el objeto Path

Cuando llamamos a un método de la clase gráfica necesitaremos varios parámetros para indicar las coordenadas de los píxeles. Por ejemplo, para un rectángulo, necesitamos especificar la ubicación de la esquina superior izquierda y su altura y anchura. Por ejemplo:

```
'Dibuja un rectángulo comenzando por la posición (0,0)
```

```
'con 300 píxeles de ancho y 50 de alto:
```

```
g.FillRectangle (Brushes.Yellow, 0, 0, 300, 50)
```

ASP.NET dibuja en memoria el rectángulo y hasta no realizar el proceso de "renderizado" no se envía a la página web. Además necesitaremos definir un pincel (Pen) o una brocha (Brush). para dibujar. Estos dos objetos también están definidos en el espacio de nombres de "System.Drawing". Los métodos que dibujan líneas necesitan un lápiz (Pen) y los que dibujan zonas con rellenos necesitan una brocha para él (brush).

Podemos crear nuestras propias brochas y pinceles aunque ya tenemos incorporados muchos objetos listos para utilizarse. Por ejemplo "Brushes.Yellow" devuelve un objeto "brush" que rellena una figura sólida con color amarillo.

Una vez que se ha completado la imagen se envía al navegador utilizando el método Image.Save(). No es que salve en ningún sitio sino que le decimos que ya está dibujada y que la mande al navegador.

```
' Render the image to the HTML output stream.
```

```
image.Save(Response.OutputStream, System.Drawing.Imaging.ImageFormat.Gif)
```

Por último podemos liberar la imagen y el contexto gráfico que hemos creado para liberar la memoria y los recursos:

```
g.Dispose()
```

```
image.Dispose()
```

El GDI+ es realmente extenso y merece un curso dedicado a él solo ya que la potencia que tiene para dibujar es enorme. Aquí solo quiero mostrarte algunas cosas para que lo conozcas y pueda ser un punto de partida de este campo tan amplio.

## 5.2 Dibujar una imagen personalizada

Con este grupo de objetos del GDI+ es sencillo crear una página con un rectángulo y un gráfico. Este es el código que utilizaremos:

## Controles avanzados y referencia de funciones.

```
Imports System.Drawing
Partial Class tema_9_grafico
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
        ' Creamos un mapa de bits de 400 de ancho x 100 de alto
        Dim imagen As New Bitmap(400, 100)
        ' Obtenemos el contexto.
        Dim g As Graphics = Graphics.FromImage(imagen)
        ' Dibujamos un rectángulo sólido con un borde rojo
        g.FillRectangle(Brushes.LightYellow, 0, 0, 400, 100)
        g.DrawRectangle(Pens.Red, 0, 0, 399, 99)

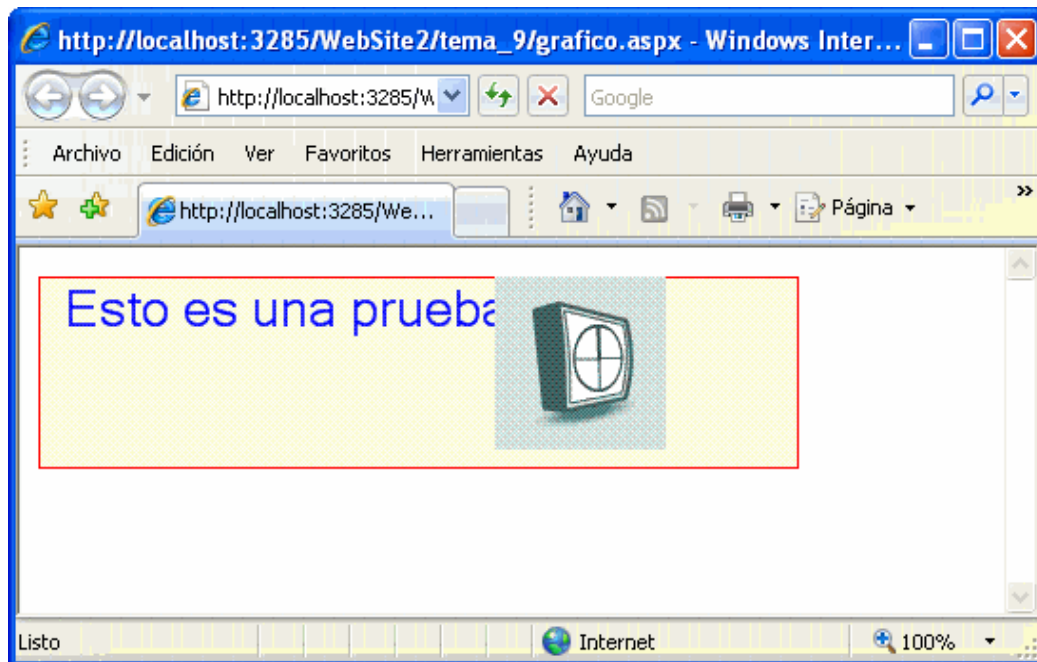
        ' Dibujamos un texto
        Dim fuente As New Font("Alba Super", 20, FontStyle.Regular)
        g.DrawString("Esto es una prueba.", fuente, Brushes.Blue, 10, 0)
        ' Pegamos en el contexto una imagen
        Dim dibujo As System.Drawing.Image
        dibujo = Image.FromFile(Server.MapPath("imagen.gif"))
        g.DrawImageUnscaled(dibujo, 240, 0)
        ' Grabamos la imagen para mandarla al navegador:
        imagen.Save(Response.OutputStream, System.Drawing.Imaging.ImageFormat.Gif)
        ' Liberamos la memoria y el contexto:
        g.Dispose()
        imagen.Dispose()
    End Sub

End Class
```

El código es fácil de entender, como siempre la complejidad está en conocer los objetos disponibles. Primero hemos creado un rectángulo relleno de amarillo y dentro de él hemos dibujado otro de color rojo. Luego hemos definido una fuente o tipo de letra para de texto y hemos escrito de forma gráfica un texto de prueba. Luego hemos añadido un gráfico y finalmente lo hemos "renderizado" o construido y enviado al cliente en formato "GIF":

```
image.Save(Response.OutputStream, System.Drawing.Imaging.ImageFormat.Gif)
```

La fuente que hemos indicado es del servidor ya que el código se ejecuta en el servidor, así que si queremos utilizar algún tipo de letra especial no hace falta que las instalemos en los clientes porque el servidor construye un gráfico con todo lo que le indicamos y se lo envía como una imagen de formato GIF.



Cuando enviamos una imagen al navegador debemos tener en cuenta el formato que le estamos mandando ya que existen varios formatos y cada uno con sus ventajas e inconvenientes. Vemos los formatos:

- JPG (utilizado en las cámaras de fotos por ejemplo). Es un buen formato para imágenes con muchos colores, por ejemplo, las fotografías. Pero es un formato "con pérdida" ya que consigue reducir su tamaño quitando algo de información a la imagen. Aun así es el que debemos utilizar para fotografías
- GIF. Tiene como límite que solo puede utilizar 256 colores, pero son mas que suficientes para imágenes sencillas y logotipos. Junto con el anterior componen prácticamente todos los formatos de la web.
- PNG. Es el mejor formato, permite mejor compresión que el JPG, pero no lo soportan todos los navegadores por lo que uso todavía no es muy extendido

## 6. Funciones aritméticas

Continuando con el repaso de las funciones mas importantes en ASP.NET, veremos ahora las aritméticas.

En esta versión .NET cambia todo lo conocido anteriormente en las otras versiones de Visual Basic y ASP. Así que si vienes de las versiones anteriores "cambia el chip" y atento a las siguientes funciones. Es mucho mas fácil de estudiar y además seguimos con la filosofía de las clases de .NET obligatorias de utilizar por otra parte.

En .NET disponemos de una clase llamada **Math** que proporciona constantes y métodos estáticos para operaciones trigonométricas, logarítmicas y otras funciones matemáticas comunes.

Los miembros de esta clase son:

Campos públicos	Descripción
E	Representa la base logarítmica natural, especificada por la constante, <b>e</b>

## Controles avanzados y referencia de funciones.

PI	Representa la relación entre la longitud de la circunferencia de un círculo y su diámetro, especificada por la constante $\pi$
<b>Métodos públicos</b>	<b>Descripción</b>
Abs	Sobrecargado. Devuelve el valor absoluto de un número especificado
Acos	Devuelve el ángulo cuyo coseno es el número especificado
Asin	Devuelve el ángulo cuyo seno es el número especificado
Atan	Devuelve el ángulo cuya tangente corresponde al número especificado
Atan2	Devuelve el ángulo cuya tangente es el cociente de dos números especificados
BigMul	Calcula el producto completo de dos números de 32 bits
Ceiling	Devuelve el número entero más pequeño mayor o igual que el número especificado.
Cos	Devuelve el coseno del ángulo especificado.
Cosh	Devuelve el coseno hiperbólico del ángulo especificado.
DivRem	Sobrecargado. Devuelve el cociente de dos números y pasa también como parámetro de salida el resto de la división.
Exp	Devuelve e elevado a la potencia especificada.
Floor	Devuelve el número entero más grande menor o igual que el número especificado
IEEERemainder	Devuelve el resto de la división de dos números especificados
Log	Sobrecargado. Devuelve el logaritmo de un número especificado.
Log10	Devuelve el logaritmo en base 10 de un número especificado.
Max	Sobrecargado. Devuelve el mayor de dos números especificados.
Min	Sobrecargado. Devuelve el menor de dos números.
Pow	Devuelve un número especificado elevado a la potencia especificada.
Round	Sobrecargado. Devuelve el número más próximo al valor especificado.
Sign	Sobrecargado. Devuelve un valor que indica el signo de un número.
Sin	Devuelve el seno del ángulo especificado.
Sinh	Devuelve el seno hiperbólico del ángulo especificado
Sqrt	Devuelve la raíz cuadrada de un número especificado.
Tan	Devuelve la tangente del ángulo especificado.
Tanh	Devuelve la tangente hiperbólica del ángulo especificado

Como ves he cambiado un poco al contar las cosas ahora estamos viendo una "clase" y sus "miembros", esta forma de llamar las cosas la veremos en profundidad cuando descubramos la "Programación orientada a objeto". También nos encontramos que algunos métodos están "sobrecargados".

La palabra sobrecargado está dentro ya de la programación orientada a objeto y su definición es muy sencilla, veras...La sobrecarga como vimos en el capítulo anterior consiste en crear mas de un procedimiento o función



## Controles avanzados y referencia de funciones.

dentro de la misma clase con el mismo nombre y distintos argumentos. Veamos un ejemplo, el primero, la función ABS. Esta es una sencilla función que devuelve el valor absoluto de un número especificado, es decir el mismo valor sin signo. Extraemos de la ayuda de .NET lo siguiente, que son la "Lista de sobrecargas":

- Devuelve el valor absoluto de un decimal:  
Overloads Public Shared Function Abs(Decimal) As Decimal
- Devuelve el valor absoluto de punto flotante de precisión doble:  
Overloads Public Shared Function Abs(Double) As Double
- Devuelve el valor absoluto de un entero de 16 bits con signo:  
Overloads Public Shared Function Abs(Short) As Short
- ...

Es decir es una función sobrecargada porque está declarada varias veces admitiendo argumentos distintos: decimal, double, short...

¿Complicado? no, es sólo un poco de práctica, ya iremos haciendo programas para practicar con estas funciones. Vamos ver unos cuantos ejemplos de estas funciones matemáticas para que conozcas cómo las podemos utilizar en nuestro código.

### Ejemplo con ABS

```
Private m_longBase As Double

Private m_shortBase As Double

Private m_leftLeg As Double

Private m_rightLeg As Double

'Simplemente es procedimiento donde entran cuatro variables
'y se las asigna a las privadas definidas en valor absoluto

Public Sub Nuevo(ByVal longbase As Double, ByVal shortbase As Double,
                  ByVal leftLeg As Double, ByVal rightLeg As Double)

    m_longBase = Math.Abs(longbase)

    m_shortBase = Math.Abs(shortbase)

    m_leftLeg = Math.Abs(leftLeg)

    m_rightLeg = Math.Abs(rightLeg)

End Sub
```

### Ejemplo con la función Max

La función Max devuelve el mayor de dos números especificados, veamos su funcionamiento con distintos tipos de datos:

## Controles avanzados y referencia de funciones.

Sub Page\_Load

```
Dim str As String = "{0}: El mayor de {1,3} y {2,3} es {3}."
```

```
Dim xByte1 As Byte = 1
```

```
Dim xByte2 As Byte = 51
```

```
Dim xShort1 As Short = - 2
```

```
Dim xShort2 As Short = 52
```

```
Dim xInt1 As Integer = - 3
```

```
Dim xInt2 As Integer = 53
```

```
Dim xLong1 As Long = - 4
```

```
Dim xLong2 As Long = 54
```

```
Dim xSingle1 As Single = 5F
```

```
Dim xSingle2 As Single = 55F
```

```
Dim xDouble1 As Double = 6.0
```

```
Dim xDouble2 As Double = 56.0
```

```
Dim xDecimal1 As [Decimal] = 7D
```

```
Dim xDecimal2 As [Decimal] = 57D
```

```
' Los siguientes tipos no se pueden utilizar:
```

```
' SByte, UInt16, UInt32, y UInt64.
```

```
Response.Write("Muestra el mayor de dos valores:")
```

```
Response.Write(str, "Byte ", xByte1, xByte2, Math.Max(xByte1, xByte2))
```

```
Response.Write(str, "Int16 ", xShort1, xShort2, Math.Max(xShort1, xShort2))
```

```
Response.Write(str, "Int32 ", xInt1, xInt2, Math.Max(xInt1, xInt2))
```

```
Response.Write(str, "Int64 ", xLong1, xLong2, Math.Max(xLong1, xLong2))
```

```
Response.Write(str, "Single ", xSingle1, xSingle2, Math.Max(xSingle1, xSingle2))
```

```
Response.Write(str, "Double ", xDouble1, xDouble2, Math.Max(xDouble1, xDouble2))
```

```
Response.Write(str, "Decimal", xDecimal1, xDecimal2, Math.Max(xDecimal1, xDecimal2))
```

```
,
```

```
Response.Write("No están soportados los siguientes tipos")
```

## Controles avanzados y referencia de funciones.

```
Response.Write("SByte, UInt16, UInt32, y UInt64.")

End Sub

'

'La salida de esté programa produce:

'

'Muestra el mayor de dos valores:

'

'Byte : El mayor de 1 y 51 es 51.
'Int16 : El mayor de -2 y 52 es 52.
'Int32 : El mayor de -3 y 53 es 53.
'Int64 : El mayor de -4 y 54 es 54.
'Single : El mayor de 5 y 55 es 55.
'Double : El mayor de 6 y 56 es 56.
'Decimal: El mayor de 7 y 57 es 57.

'
```

### Ejemplo con la función Round

Esta función muestra el redondeo al entero mas próximo:

```
Math.Round(3.44, 1) 'Devuelve 3.4.
Math.Round(3.45, 1) 'Devuelve 3.4.
Math.Round(3.46, 1) 'Devuelve 3.5.
```

### Ejemplo con la función Pow

Esta función devuelve un número elevado a una potencia. Lógicamente necesita dos parámetros, la base y el exponente.

La base es "x" y el exponente "y". En esta tabla muestra los resultados si x=0 o "NaN" (valor no numérico), NegativeInfinity (infinito negativo) o PositiveInfinity (infinito positivo)

Valores del parámetro	Devuelve
x o y es igual a Double.NaN	NaN.

## Controles avanzados y referencia de funciones.

$x$ es igual a Double.NegativeInfinity	NegativeInfinity si $y$ es un entero impar, en caso contrario, PositiveInfinity.
$y$ es igual a Double.NegativeInfinity	0.
$x$ es igual a Double.PositiveInfinity	0 si $y$ es igual a NegativeInfinity; en caso contrario, PositiveInfinity.
$y$ es igual a Double.PositiveInfinity	PositiveInfinity.

Y aplicado en una función:

```
Public Function GetHeight() As Double

    Dim x As Double = GetRightSmallBase()

    GetHeight = Math.Sqrt(Math.Pow(m_rightLeg, 2) - Math.Pow(x, 2))

End Function
```

## Funciones matemáticas derivadas

A continuación, se muestra una lista de funciones matemáticas que pueden derivarse de las funciones matemáticas que hemos visto antes:

Función	Funciones derivadas equivalentes
Secante (Sec(x))	$1 / \cos(x)$
Cosecante (Csc(x))	$1 / \sin(x)$
Cotangente (Ctan(x))	$1 / \tan(x)$
Seno inverso (Asin(x))	$\arcsin(x) = \arctan(x / \sqrt{1 - x^2})$
Coseno inverso (Acos(x))	$\arccos(x) = \arctan(\sqrt{1 - x^2} / x) + \arctan(1)$
Secante inversa (Asec(x))	$2 * \arctan(1) - \arctan(\text{Sign}(x) / \sqrt{x^2 - 1})$
Cosecante inversa (Acsc(x))	$\arctan(\text{Sign}(x) / \sqrt{x^2 - 1})$
Cotangente inversa (Acot(x))	$2 * \arctan(1) - \arctan(x)$
Seno hiperbólico (Sinh(x))	$(\exp(x) - \exp(-x)) / 2$
Coseno hiperbólico (Cosh(x))	$(\exp(x) + \exp(-x)) / 2$
Tangente hiperbólica (Tanh(x))	$(\exp(x) - \exp(-x)) / (\exp(x) + \exp(-x))$
Secante hiperbólica (Sech(x))	$2 / (\exp(x) + \exp(-x))$
Cosecante hiperbólica (Csch(x))	$2 / (\exp(x) - \exp(-x))$
Cotangente hiperbólica (Coth(x))	$(\exp(x) + \exp(-x)) / (\exp(x) - \exp(-x))$
Seno hiperbólico inverso (Asinh(x))	$\log(x + \sqrt{x^2 + 1})$
	$\log(x + \sqrt{x^2 - 1})$

## Controles avanzados y referencia de funciones.

Coseno hiperbólico inverso (Acosh(x))	
Tangente hiperbólica inversa (Atanh(x))	<b>Log</b> ((1 + x) / (1 - x)) / 2
Secante hiperbólica inversa (AsecH(x))	<b>Log</b> (( <b>Sqrt</b> (-x * x + 1) + 1) / x)
Cosecante hiperbólica inversa (Acsch(x))	<b>Log</b> (( <b>Sign</b> (x) * <b>Sqrt</b> (x * x + 1) + 1) / x)
Cotangente hiperbólica inversa (Acoth(x))	<b>Log</b> ((x + 1) / (x - 1)) / 2

### Format

Devuelve una cadena con el formato que especifiquen las instrucciones contenidas en una expresión **String** de formato

### Sintaxis

**Function Format( ByVal Expression As Object, Optional ByVal Style As String = "" ) As String**

La tabla siguiente muestra los nombres de formato numérico predefinidos. Éstos pueden usarse por nombre como argumento de estilo para la función **Format**:

Nombre de formato	Descripción
<b>General Number, G o g</b>	Muestra el número sin separadores de miles.
<b>Currency, C o c</b>	Muestra el número con separadores de miles, en su caso; también muestra dos dígitos a la derecha del separador de decimales. El formato de salida dependerá de la configuración regional.
<b>Fixed, F o f</b>	Muestra al menos un dígito a la izquierda y dos a la derecha del separador de decimales.
<b>Standard, N o n</b>	Muestra el número con separador de miles, al menos un dígito a la izquierda y dos a la derecha del separador de decimales.
<b>Percent</b>	Muestra el número multiplicado por 100 con un signo de porcentaje (%) a la derecha; siempre muestra dos dígitos a la derecha del separador de decimales.
<b>P o p</b>	Muestra el número con separador de miles multiplicado por 100 con un signo de porcentaje (%) a la derecha y separado por un solo espacio; siempre muestra dos dígitos a la derecha del separador de decimales.
<b>Científico</b>	Utiliza notación científica estándar y proporciona dos dígitos significativos.
<b>E o e</b>	Utiliza notación científica estándar y proporciona seis dígitos significativos.
<b>D o d</b>	

## Controles avanzados y referencia de funciones.

	Muestra el número como una cadena que contiene el valor del número en formato Decimal (base 10). Esta opción sólo se admite para tipos integrales ( <b>Byte</b> , <b>Short</b> , <b>Integer</b> , <b>Long</b> ).
<b>X o x</b>	Muestra el número como una cadena que contiene el valor del número en formato Hexadecimal (base 16). Esta opción sólo se admite para tipos integrales ( <b>Byte</b> , <b>Short</b> , <b>Integer</b> y <b>Long</b> ).
<b>Yes/No</b>	Muestra <b>No</b> si el número es 0; de lo contrario, muestra <b>Yes</b> .
<b>True o False</b>	Muestra <b>False</b> si el número es 0; de lo contrario, muestra <b>True</b> .
<b>On/Off</b>	Muestra <b>Off</b> si el número es 0; de lo contrario, muestra <b>On</b> .

La siguiente tabla identifica caracteres que puede usar para crear formatos de número definidos por el usuario. Éstos pueden usarse para generar el argumento de estilo correspondiente a la función **Format**:

Carácter	Descripción
Ninguno	Muestra el número sin formato alguno.
(0)	<p>Marcador de posición de dígito. Muestra un dígito o un cero. Si la expresión tiene un dígito en la posición donde aparece el cero en la cadena de formato, éste se mostrará así; de lo contrario, aparecerá un cero en esa posición.</p> <p>Si el número tiene menos dígitos que ceros (a cualquier lado del separador decimal) en la expresión de formato, se mostrarán ceros iniciales o finales. Si el número tiene más dígitos a la derecha del separador decimal que ceros en la expresión de formato, se redondeará el número a tantos decimales como ceros haya. Si el número tiene más dígitos a la izquierda del separador decimal que ceros en la expresión de formato, se redondeará el número a tantos decimales como ceros haya.</p>
(#)	<p>Marcador de posición de dígito. Muestra un dígito o nada. Si la expresión tiene un dígito en la posición donde aparece el carácter # en la cadena de formato, se muestra; de lo contrario, no aparece nada en esa posición.</p> <p>Este símbolo funciona como el marcador de posición de dígito <b>0</b>, salvo que los ceros iniciales y finales no se mostrarán si el número contiene menos dígitos que caracteres # a cualquiera de los lados del separador decimal en la expresión de formato.</p>
(.)	<p>Marcador de posición decimal. El marcador de posición decimal determina cuántos dígitos se mostrarán a la izquierda y derecha del separador decimal. Si la expresión de formato sólo contiene caracteres # a la izquierda de este símbolo, los números inferiores a 1 empezarán con un separador decimal. Para mostrar un cero inicial con números fraccionarios, use el cero como el primer marcador de posición digital a la izquierda del separador decimal. El uso del punto o la coma como separador decimal depende de la configuración regional en cada caso. El mismo carácter utilizado como marcador decimal en virtud del formato de salida dependerá del formato de número reconocido por su sistema. Por tanto, deberá usar el punto como marcador decimal en sus formatos, incluso aunque su configuración regional utilice la coma como separador decimal. La cadena con formato se mostrará con el formato correcto para la configuración regional correspondiente.</p>

## Controles avanzados y referencia de funciones.

(%)	Marcador de posición de porcentaje. Multiplica la expresión por 100. El carácter de porcentaje (%) se inserta en la misma posición en la que aparece en la cadena de formato.
(,)	<p>Separador de miles. El separador de miles separa las unidades de millar de las centenas con un número que presente cuatro o más dígitos a la izquierda del separador decimal. Se especificará un uso estándar del separador de miles si el formato contiene un separador de miles rodeado de marcadores de posición de dígito (<b>0</b> ó <b>#</b>). Un separador de miles situado inmediatamente a la izquierda del separador decimal (se especificará si se trata de un decimal o no) o como el carácter más a la derecha de la cadena significa "reducir el número dividiéndolo por 1000 y redondeándolo en caso necesario".</p> <p>Por ejemplo, puede utilizar el formato de cadena "##0, ." para representar 100 millones como 100,000. Los números menores que 1,000 pero mayores o iguales que 500 se muestran como 1; los menores que 500 se muestran como 0. Dos separadores de miles adyacentes en esta posición se reducen por un factor de 1 millón, más otro factor adicional de 1000 por cada separador adicional.</p> <p>En el caso de los separadores múltiples en cualquier posición que no sea inmediatamente a la izquierda del separador decimal o la posición más a la derecha de la cadena, se interpretará que simplemente especifican el uso de un separador de miles. El uso del punto o la coma como separador de miles depende de la configuración regional en cada caso. El mismo carácter utilizado como separador de miles en virtud del formato de salida dependerá del formato de número reconocido por su sistema. Por tanto, deberá usar la coma como marcador de miles en sus formatos, incluso aunque su configuración regional utilice el punto como marcador de miles. La cadena con formato se mostrará con el formato correcto para la configuración regional correspondiente.</p>
(:)	Separador de hora. En ciertas configuraciones regionales, pueden usarse otros caracteres para representar el separador de hora. Este separador horario separa horas, minutos y segundos cuando se da formato a valores horarios. El carácter real utilizado es el especificado como separador de hora en la configuración de su sistema.
(/)	Separador de fecha. En ciertas configuraciones regionales, pueden usarse otros caracteres para representar el separador de fecha. Este separador separa el día, mes y año cuando se da formato a los valores de fecha. El carácter real utilizado es el especificado como separador de fecha en la configuración de su sistema.
(E- E+ e- e+)	Formato científico. Si la expresión de formato contiene al menos un marcador de posición de dígito ( <b>0</b> o <b>#</b> ) a la izquierda de <b>E-</b> , <b>E+</b> , <b>e-</b> o <b>e+</b> , el número se mostrará en formato científico y se insertará <b>E</b> o <b>e</b> entre el número y su exponente. El número de marcadores de posición digitales a la izquierda determina el número de dígitos en el exponente. Use <b>E-</b> o <b>e-</b> para colocar un signo menos junto a los exponentes negativos. Use <b>E+</b> o <b>e+</b> para colocar un signo menos junto a los exponentes negativos y un signo más junto a los positivos. También deberá incluir marcadores de posición digitales a la derecha de este símbolo para obtener un formato correcto.
- + \$ ( )	Caracteres literales. Estos caracteres se mostrarán exactamente como se escriben en la cadena de formato. Para mostrar un carácter distinto de los listados, precédase de una barra invertida (\) o escríbase entre comillas (" ").

## Controles avanzados y referencia de funciones.

(\)	<p>Muestra el siguiente carácter de una cadena de formato. Para mostrar un carácter dotado de un significado especial como carácter literal, éste debe ir precedido de una barra invertida (\). La barra invertida en sí no aparecerá. El uso de una barra invertida equivale a incluir el siguiente carácter entre comillas. Para mostrar una barra invertida, úsense dos (\\).</p> <p>Los caracteres de formato de fecha y los caracteres de formato de hora (<b>a</b>, <b>c</b>, <b>d</b>, <b>h</b>, <b>m</b>, <b>n</b>, <b>p</b>, <b>q</b>, <b>s</b>, <b>t</b>, <b>w</b>, <b>y</b>, / y :), los caracteres de formato numérico (<b>#</b>, <b>0</b>, <b>%</b>, <b>E</b>, <b>e</b>, la coma y el punto) y los caracteres de formato de cadena (<b>@</b>, <b>&amp;</b>, <b>&lt;</b>, <b>&gt;</b> y <b>!</b>) son ejemplos de caracteres que no se pueden mostrar como caracteres literales.</p>
("ABC")	Muestra la cadena entre comillas (" "). Si se desea insertar una cadena en el argumento de estilo desde el código, deberá usar <b>Chr(34)</b> para incluir el texto ( <b>34</b> es el código de caracteres correspondiente a las comillas dobles ("")).

### Ejemplo

La tabla siguiente contiene algunas muestras de expresiones de formato correspondientes a números. En estos ejemplos se presupone que la configuración regional del sistema es Inglés (Estados Unidos). La primera columna contiene las cadenas de formato correspondientes al argumento *Style* de la función **Format**; las otras columnas contienen el formato de salida resultante si los datos con formato contienen el valor asignado en los encabezados de columna.

Formato ( <i>Style</i> )	"5" con formato como	"-5" con formato como	"0.5" con formato como
Zero-length string ( " " )	5	-5	0.5
0	5	-5	1
0.00	5.00	-5.00	0.50
#,##0	5	-5	1
\$#,##0;(\$#,##0)	\$5	(\$5)	\$1
\$#,##0.00;(\$#,##0.00)	\$5.00	(\$5.00)	\$0.50
0%	500%	-500%	50%
0.00%	500.00%	-500.00%	50.00%
0.00E+00	5.00E+00	-5.00E+00	5.00E-01
0.00E-00	5.00E00	-5.00E00	5.00E-01

La tabla siguiente identifica los nombres de formatos de fecha y hora predefinidos. Éstos pueden usarse por nombre como argumento de estilo para la función **Format**:

Nombre de formato	Descripción
-------------------	-------------



## Controles avanzados y referencia de funciones.

<b>General Date o G</b>	Muestra una fecha o una hora. En el caso de números reales, muestra una fecha y una hora; por ejemplo, 4/3/93 05:34 PM. Si no existe parte fraccional, muestra sólo una fecha; por ejemplo, 4/3/93. Si no existe parte entera, muestra sólo una hora; por ejemplo, 05:34 PM. El formato de fecha depende del valor <b>LocaleID</b> del sistema.
<b>Long Date o D</b>	Muestra una fecha de acuerdo con el formato de fecha larga vigente en su sistema.
<b>Medium Date</b>	Muestra una fecha usando el formato medio que corresponda a la versión de idioma que use la aplicación host.
<b>Short Date o d</b>	Muestra una fecha de acuerdo con el formato de fecha corta vigente en su sistema.
<b>Long Time o T</b>	Muestra una hora de acuerdo con el formato de fecha larga vigente en su sistema; e incluye horas, minutos y segundos.
<b>Medium Time</b>	Muestra la hora en formato de 12 horas utilizando horas y minutos y la especificación a.m./p.m.
<b>Short Time o t</b>	Muestra una hora con el formato de 24 horas, por ejemplo, 17:45.
<b>f</b>	Muestra la fecha larga y la hora corta de acuerdo con el formato vigente en su sistema.
<b>F</b>	Muestra la fecha larga y la hora larga de acuerdo con el formato vigente en su sistema.
<b>g</b>	Muestra la fecha corta y la hora corta de acuerdo con el formato vigente en su sistema.
<b>M, m</b>	Muestra el mes y el día de una fecha dada.
<b>R, r</b>	Da formato a la fecha y la hora como Hora media de Greenwich (GMT)
<b>s</b>	Da formato a la fecha y la hora como un índice ordenable.
<b>u</b>	Da formato a la fecha y la hora como un índice GMT ordenable.
<b>U</b>	Da formato como GMT a la fecha larga y la hora larga.
<b>Y, y</b>	Da formato a la fecha especificando el año y el mes

En la siguiente tabla se muestran los caracteres que se pueden utilizar para crear formatos de fecha y hora definidos por el usuario. A diferencia de versiones anteriores de Visual Basic, estos caracteres de formato distinguen mayúsculas de minúsculas.

<b>Carácter</b>	<b>Descripción</b>
(:)	Separador de hora. En ciertas configuraciones regionales, pueden usarse otros caracteres para representar el separador de hora. Este separador horario separa horas, minutos y segundos cuando se da formato a valores horarios. El carácter real utilizado como separador de hora en los resultados con formato viene determinado por el valor <b>LocaleID</b> del sistema.

## Controles avanzados y referencia de funciones.

<b>(/)</b>	Separador de fecha. En ciertas configuraciones regionales, pueden usarse otros caracteres para representar el separador de fecha. Este separador separa el día, mes y año cuando se da formato a los valores de fecha. El carácter real utilizado es el especificado como separador de fecha en la configuración local de su sistema.
<b>(%)</b>	Se utiliza para indicar que el carácter siguiente debe leerse como formato de una sola letra sin tener en cuenta las posibles letras finales. También se emplea para indicar que un formato de una sola letra se lea como formato definido por el usuario.
<b>d</b>	Muestra el día como un número sin cero a la izquierda (por ejemplo, 1). Usa <b>%d</b> si es el único carácter en el formato numérico definido por el usuario.
<b>dd</b>	Muestra el día como un número con cero a la izquierda (por ejemplo, 01).
<b>ddd</b>	Muestra el día de forma abreviada (por ejemplo, Dom).
<b>dddd</b>	Muestra el día de forma completa (por ejemplo, Domingo).
<b>M</b>	Muestra el mes como un número sin cero a la izquierda (por ejemplo, enero se representa como 1). Use <b>%M</b> si es el único carácter en el formato numérico definido por el usuario.
<b>MM</b>	Muestra el mes como un número con cero a la izquierda (por ejemplo, 01/12/01), doce de enero de 2001.
<b>MMM</b>	Muestra el mes en forma abreviada (por ejemplo, Ene).
<b>MMMM</b>	Muestra el mes en forma completa (por ejemplo, Enero).
<b>gg</b>	Especifica la era histórica (por ejemplo, A.D.).
<b>h</b>	Muestra la hora como un número sin ceros a la izquierda y en formato de doce horas (por ejemplo, 1:15:15 PM). Usa <b>%h</b> si es el único carácter en el formato numérico definido por el usuario.
<b>hh</b>	Muestra la hora como un número con ceros a la izquierda y en formato de doce horas (por ejemplo, 01:15:15 PM).
<b>H</b>	Muestra la hora como un número sin ceros a la izquierda y en formato de doce horas (por ejemplo, 1:15:15 PM). Usa <b>%H</b> si es el único carácter en el formato numérico definido por el usuario.
<b>HH</b>	Muestra la hora como un número con ceros a la izquierda y en formato de doce horas (por ejemplo, 01:15:15).
<b>m</b>	Muestra los minutos como un número sin ceros a la izquierda (por ejemplo, 12:1:15). Usa <b>%m</b> si es el único carácter en el formato numérico definido por el usuario.
<b>mm</b>	Muestra los minutos como un número con ceros a la izquierda (por ejemplo, 12:01:15).
<b>s</b>	Muestra los segundos como un número sin ceros a la izquierda (por ejemplo, 12:15:5). Usa <b>%s</b> si es el único carácter en el formato numérico definido por el usuario.
<b>ss</b>	Muestra los segundos como un número con ceros a la izquierda (por ejemplo, 12:15:05).

## Controles avanzados y referencia de funciones.

<b>F</b>	Muestra fracciones de segundos. Por ejemplo, <b>ff</b> muestra centésimas de segundo, mientras que <b>ffff</b> muestra diez milésimas de segundo. Puede utilizar hasta siete símbolos <b>f</b> en el formato definido por el usuario. Use <b>%f</b> si es el único carácter en el formato numérico definido por el usuario.
<b>T</b>	Usa el reloj de doce horas; muestra una A mayúscula para cualquier hora entre medianoche y mediodía, y una P mayúscula para cualquier hora entre mediodía y medianoche. Utilice <b>%t</b> si éste es el único carácter del formato numérico definido por el usuario.
<b>tt</b>	Usa el reloj de doce horas y muestra la leyenda AM en mayúsculas para cualquier hora entre medianoche y mediodía; y PM en mayúsculas para cualquier hora entre mediodía y medianoche.
<b>y</b>	Muestra el año sin cero inicial. Utilice <b>%y</b> en caso de que éste sea el único carácter en su formato numérico definido por el usuario.
<b>yy</b>	Muestra el año en formato numérico de dos dígitos sin cero inicial, si procede.
<b>yyy</b>	Muestra el año en formato numérico de cuatro dígitos.
<b>yyyy</b>	Muestra el año en formato numérico de cuatro dígitos.
<b>z</b>	Muestra el desplazamiento de zona horaria sin cero a la izquierda (por ejemplo, -8). Use <b>%z</b> si es el único carácter en el formato numérico definido por el usuario.
<b>zz</b>	Muestra el desplazamiento de zona horaria con un cero a la izquierda (por ejemplo, -08).
<b>zzz</b>	Muestra el desplazamiento completo de zona horaria (por ejemplo, -08:00).

## Ejemplo

A continuación se muestran algunos ejemplos de formatos de hora y fecha definidos por el usuario y correspondientes al 7 de diciembre de 1958, a las ocho horas, cincuenta minutos y treinta y cinco segundos de la tarde (Diciembre 7, 1958, 8:50 PM, 35 segundos):

Formato	Muestra
M/d/yy	12/7/58
d-MMM	7-Dic
d-MMMM-yy	7-Diciembre-58
d MMMM	7 Diciembre
MMMM yy	Diciembre 58
hh:mm tt	08:50 PM
h:mm:ss t	8:50:35 P
H:mm	20:50
H:mm:ss	20:50:35
M/d/yyyy H:mm	12/7/1958 20:50

### Ejemplo

En este ejemplo se muestran los diversos usos de la función `Format` para dar formato a valores tanto con formatos `String` como otros definidos por el usuario. Para el separador de fecha (/), hora (:) e indicadores de a.m./p.m. (t y tt), el formato de salida que muestre su sistema dependerá de la configuración regional que use el código. Cuando las horas y fechas se muestren en el entorno de desarrollo, se utilizará el formato de fecha y hora corta de la configuración regional del código.

```
Dim MyDateTime As Date = #1/27/2001 5:04:23 PM#

Dim MyStr As String

' Devuelve la hora del sistema en formato de hora larga.
MyStr = Format(Now(), "Long Time")

' Devuelve la fecha del sistema en formato de fecha largo
MyStr = Format(Now(), "Long Date")

' También devuelve la fecha del sistema con formato definido por el usuario
' Utilizando el caracter D.
MyStr = Format(Now(), "D")

' Devuelve el valor de MyDateTime con formato date/time definido por el usuario

MyStr = Format(MyDateTime, "h:m:s")           ' Devuelve "5:4:23".
MyStr = Format(MyDateTime, "hh:mm:ss tt")     ' Devuelve "05:04:23 PM".
MyStr = Format(MyDateTime, "dddd, MMM d yyyy") ' Devuelve "Sábado,
' 27 de Enero de 2001.

MyStr = Format(MyDateTime, "HH:mm:ss")        ' Devuelve "17:04:23"
MyStr = Format(23) ' Returns "23".

' Formatos numéricos definidos por el usuario

MyStr = Format(5459.4, "##,##0.00")           ' Devuelve "5,459.40".
MyStr = Format(334.9, "###0.00")              ' Devuelve "334.90".
MyStr = Format(5, "0.00%")                    ' Devuelve "500.00%".
```

## 3.6 Funciones de fechas

Cambios para los programadores de VB6, Visual Basic .NET reemplaza **Date** y **Time** por las propiedades **Today** y **TimeOfDay**, que utilizan la estructura **DateTime** de ocho bytes. Esto se corresponde con el tipo de datos **Date** en Visual Basic .NET. Ahora veremos estas dos funciones junto a otras de tratamiento de fechas...

## Today

Devuelve o establece un valor **Date** que contiene la fecha actual de acuerdo con el sistema.

El tipo de datos **Date** incluye componentes de tiempo. Al devolver la fecha del sistema, **Today** los establece todos en 0, de tal forma que el valor devuelto representa medianoche (00:00:00). Al establecer la fecha del sistema, **Today** omite los componentes de hora.

Para obtener acceso a la fecha actual del sistema como un valor **String**, utilice la propiedad **DateString**.

Utilice la propiedad **TimeOfDay** para obtener o establecer la hora actual del sistema

## Ejemplo

```
Dim MyDate As Date  
  
MyDate = Today
```

## Now

Devuelve un valor **Date** que contiene la fecha y la hora actuales de acuerdo con el sistema. Utilice la propiedad **Today** para establecer la fecha del sistema. Utilice la propiedad **TimeOfDay** para establecer la hora del sistema

## Ejemplo

En este ejemplo se utiliza la propiedad **Now** para mostrar la fecha y hora de sistema actual.

```
Dim ThisMoment As Date  
  
ThisMoment= Now ' Establece la fecha y hora actual del sistema
```

## DateString

Devuelve o establece un valor **String** que representa la fecha actual de acuerdo con el sistema.

**DateString** siempre devuelve la fecha del sistema como "MM-dd-yyyy", es decir, utilizando el nombre de mes abreviado. Los formatos aceptados para configurar la fecha son "M-d-yyyy", "M-d-y", "M/d/yyyy" y "M/d/y".

Si se intenta establecer **DateString** con un valor no válido, se produce un error **InvalidCastException**.

Para obtener o establecer la hora actual del sistema como un valor **String**, utilice la propiedad **TimeString**. Para obtener acceso a la fecha actual del sistema como un valor **Date**, utiliza la propiedad **Today**

## Ejemplo

En este ejemplo se utiliza la propiedad **DateString** para mostrar la fecha de sistema actual.

```
Response.Write("La fecha actual es: " & DateString)
```

## TimeOfDay

Devuelve o establece un valor **Date** que contiene la hora del día actual de acuerdo con el sistema

## Ejemplo

En este ejemplo se utiliza la propiedad **TimeOfDay** para mostrar la hora de sistema actual.

El tipo de datos **Date** incluye componentes de fecha. Al devolver la hora del sistema, **TimeOfDay** los establece todos en 1, de tal forma que el valor devuelto representa el primer día del año 1. Al establecer la hora del sistema, **TimeOfDay** omite los componentes de fecha.

Para obtener acceso a la hora actual del sistema como un valor **String**, utilice la propiedad **TimeString**.

```
Dim MyTime As Date  
  
MyTime = TimeOfDay ' Devuelve la hora actual del sistema.
```

## TimeString

Devuelve o establece un valor **String** que representa la hora del día actual de acuerdo con el sistema.

**TimeString** devuelve siempre la hora del sistema como "HH:mm:ss", que es un formato de 24 horas. Si intentamos establecer **TimeString** con un valor no válido, se produce un error **InvalidCastException**.

Para obtener o establecer la fecha actual del sistema como un valor **String**, utiliza la propiedad **DateString**. Para obtener acceso a la hora actual del sistema como un valor **Date**, utiliza la propiedad **TimeOfDay**

## Ejemplo

En este ejemplo se utiliza la propiedad **TimeString** para mostrar la hora de sistema actual.

```
Response.Write("La hora actual es: " & TimeString)
```

## Year

Devuelve un valor **Integer** entre 1 y 9999 que representa el año.

## Ejemplo

En este ejemplo se utiliza la función **Year** para obtener el año de una fecha especificada. En el entorno de desarrollo, el literal de fecha se muestra en formato corto de fecha con los valores de configuración regional del código correspondiente.

```
Dim MyDate As Date

Dim MyYear As Integer

MyDate = #2/12/1969#      ' Asigna una fecha.

MyYear = Year(MyDate)    ' Asigna el año 1969
```

## Month

Devuelve un valor **Integer** entre 1 y 12 que representa el mes del año

## Ejemplo

En este ejemplo se utiliza la función **Month** para obtener el mes de una fecha especificada. En el entorno de desarrollo, el literal de fecha se muestra en formato corto de fecha con los valores de configuración regional del código correspondiente.

```
Dim MyDate As Date

Dim MyMonth As Integer

MyDate = #2/12/1969#      ' Asigna una fecha.

MyMonth = Month(MyDate)   ' Asigna el valor 2
```

## Day

Devuelve un valor **Integer** entre 1 y 31 que representa el día del mes

## Ejemplo

En este ejemplo se utiliza la función **Day** para obtener el día del mes de una fecha especificada. En el entorno de desarrollo, el literal de fecha se muestra en formato corto estándar (por ejemplo "12/02/1969") con los valores de configuración regional del código correspondiente.

```
Dim MyDate As Date

Dim MyDay As Integer

MyDate = #2/12/1969#      'Asigna una fecha utilizando el formato estándar corto.

MyDay = Microsoft.VisualBasic.DateAndTime.Day(MyDate)  ' MyDay contiene 12.
```

**Day** se califica para distinguirlo de la enumeración **System.Windows.Forms.Day**

## WeekDay

Devuelve un valor **Integer** que contiene un número que representa el día de la semana.

El valor devuelto por la función **Weekday** corresponde a los valores de la enumeración **FirstDayOfWeek**; es decir, 1 indica domingo y 7 indica sábado.

Si *DayOfWeek* es inferior a 0 o mayor que 7, se produce un error **ArgumentException**.

**Nota** **Weekday** utiliza la configuración de calendario actual de la propiedad **CurrentCulture** de la clase **CultureInfo** en el espacio de nombres **System.Globalization**. Los valores **CurrentCulture** predeterminados están determinados por la configuración del Panel de control.

## Sintaxis

```
Function Weekday( ByVal DateValue As DateTime,  
Optional ByVal DayOfWeek As FirstDayOfWeek  
= FirstDayOfWeek.Sunday ) As Integer
```

## Argumentos

### *DateValue*

Requerido. Valor **Date** del cual se desea determinar el día de la semana.

### *DayOfWeek*

Opcional. Valor elegido de la enumeración **FirstDayOfWeek** que especifica el primer día de la semana. Si no se especifica ningún valor, se utiliza **FirstDayOfWeek.Sunday**.

El argumento *DayOfWeek* puede tener uno de los siguientes valores:

Valor de enumeración	Valor	Descripción
<b>FirstDayOfWeek.System</b>	0	Primer día de la semana especificado en la configuración del sistema
<b>FirstDayOfWeek.Sunday</b>	1	Domingo (predeterminado)
<b>FirstDayOfWeek.Monday</b>	2	Lunes (de acuerdo con la norma ISO 8601, sección 3.17)
<b>FirstDayOfWeek.Tuesday</b>	3	Martes
<b>FirstDayOfWeek.Wednesday</b>	4	Miércoles
<b>FirstDayOfWeek.Thursday</b>	5	Jueves
<b>FirstDayOfWeek.Friday</b>	6	Viernes
<b>FirstDayOfWeek.Saturday</b>	7	Sábado



## Ejemplo

En este ejemplo se utiliza la función **Weekday** para obtener el día de la semana de una fecha especificada.

```
Dim MyDate As Date

Dim MyWeekDay As Integer

MyDate = #2/12/1969# ' Asigna una fecha

MyWeekDay = Weekday(MyDate) ' MyWeekDay contiene 4

' MyDate representa a miércoles
```

## WeekDayName

Devuelve un valor **String** que contiene el nombre del día de la semana especificado.

```
Public Function WeekdayName( _

    ByVal WeekDay As Integer, _

    Optional ByVal Abbreviate As Boolean = False, _

    Optional ByVal FirstDayOfWeekValue As FirstDayOfWeek = FirstDayOfWeek.System _

) As String
```

## Parámetros

### *WeekDay*

Requerido. **Integer**. Designación numérica del día de la semana, entre 1 y 7; 1 indica el primer día de la semana y 7 indica el último día de la semana. Las identidades del primer y último día dependen de la configuración de *FirstDayOfWeekValue*.

### *Abbreviate*

Opcional. Valor **Boolean** que indica si se abrevia el nombre del día de la semana. Si se omite, el valor predeterminado es **False**, que significa que el nombre del día de la semana no se abrevia.

### *FirstDayOfWeekValue*

Opcional. Valor elegido de la enumeración **FirstDayOfWeek** que especifica el primer día de la semana. Si no se especifica ningún valor, se utiliza **FirstDayOfWeek.System**.

## Configuración

El argumento *FirstDayOfWeekValue* puede tener uno de los siguientes valores:

Valor de enumeración	Valor	Descripción
<b>FirstDayOfWeek.System</b>	0	Primer día de la semana especificado en la configuración del sistema (valor predeterminado)
<b>FirstDayOfWeek.Sunday</b>	1	Domingo

## Controles avanzados y referencia de funciones.

<b>FirstDayOfWeek.Monday</b>	2	Lunes (de acuerdo con la norma ISO 8601, sección 3.17)
<b>FirstDayOfWeek.Tuesday</b>	3	Martes
<b>FirstDayOfWeek.Wednesday</b>	4	Miércoles
<b>FirstDayOfWeek.Thursday</b>	5	Jueves
<b>FirstDayOfWeek.Friday</b>	6	Viernes
<b>FirstDayOfWeek.Saturday</b>	7	Sábado

### Comentarios

La cadena devuelta por **WeekdayName** no depende únicamente de los argumentos de entrada, sino también de los valores de la Configuración regional especificados en el Panel de control de Windows.

Si *WeekDay* es menor que 1 o mayor que 7, o si *FirstDayOfWeekValue* es menor que 0 o mayor que 7, se produce un error **ArgumentException**.

**Nota** **WeekdayName** utiliza la configuración de calendario actual de la propiedad **CurrentCulture** de la clase **CultureInfo** en el espacio de nombres **System.Globalization**. Los valores **CurrentCulture** predeterminados están determinados por la configuración del Panel de control

### MonthName

Devuelve un valor **String** que contiene el nombre del mes especificado.

```
Public Function MonthName( _  
    ByVal Month As Integer, _  
    Optional ByVal Abbreviate As Boolean = False _  
) As String
```

### Parámetros

#### *Month*

Requerido. **Integer**. Designación numérica del mes, entre 1 y 13; 1 indica el mes de enero y 12 indica el mes de diciembre. Se puede utilizar el valor 13 con un calendario de 13 meses. Si el sistema está utilizando un calendario de 12 meses y *Month* es 13, **MonthName** devuelve una cadena vacía.

#### *Abbreviate*

Opcional. Valor **Boolean** que indica si se va a abreviar el nombre del mes. Si se omite, el valor predeterminado es **False**, que significa que el nombre del mes no se abrevia.

### Excepciones o errores

Tipo de excepción	Número de error	Condición
-------------------	-----------------	-----------

ArgumentException	5	<i>Month</i> es menor que 1 o mayor que 13.
-------------------	---	---

## Comentarios

La cadena devuelta por **MonthName** no depende únicamente de los argumentos de entrada, sino también de los valores de la Configuración regional especificados en el Panel de control de Windows.

Si *Month* es inferior a 1 o mayor que 13, se produce un error **ArgumentException**.

**Nota** **MonthName** utiliza la configuración de calendario actual de la propiedad **CurrentCulture** de la clase **CultureInfo** en el espacio de nombres **System.Globalization**. Los valores **CurrentCulture** predeterminados están determinados por la configuración del **Panel de control**.

## Ejemplo

En este ejemplo se utiliza la función **MonthName** para determinar el nombre del mes, a partir del entero dado. El valor Boolean determinará si se muestra el nombre completo (**False**) o el nombre abreviado (**True**).

```
Dim MyMonth As Integer

Dim Name As String

MyMonth = 4

Name = MonthName(MyMonth, True) ' "True" devuelve un valor abreviado.

Response.Write(Name)           ' Name contiene "Abr".
```

## DatePart

Devuelve un valor **Integer** que contiene el componente especificado de un valor **Date** dado.

```
Public Overloads Function DatePart( _
    ByVal Interval As DateInterval, _
    ByVal DateValue As DateTime, _
    Optional ByVal FirstDayOfWeekValue As FirstDayOfWeek = VbSunday, _
    Optional ByVal FirstWeekOfYearValue As FirstWeekOfYear = VbFirstJan1 _
) As Integer
```

O bien

```
Public Overloads Function DatePart( _
    ByVal Interval As String, _
    ByVal DateValue As Object, _
```

## Controles avanzados y referencia de funciones.

```
Optional ByVal DayOfWeek As FirstDayOfWeek = FirstDayOfWeek.Sunday, _  
Optional ByVal WeekOfYear As FirstWeekOfYear = FirstWeekOfYear.Jan1 _  
) As Integer
```

### Parámetros

#### *Interval*

Requerido. Valor de enumeración **DateInterval** o expresión **String** que representa la parte del valor de fecha u hora que se desea devolver.

#### *DateValue*

Requerido. Valor **Date** que se desea evaluar.

#### *FirstDayOfWeekValue*

Opcional. Valor elegido de la enumeración **FirstDayOfWeek** que especifica el primer día de la semana. Si no se especifica ningún valor, se utiliza **FirstDayOfWeek.Sunday**.

#### *FirstWeekOfYearValue*

Opcional. Valor elegido de la enumeración **FirstWeekOfYear** que especifica la primera semana del año. Si no se especifica ningún valor, se utiliza **FirstWeekOfYear.Jan1**.

### Configuración

El argumento *Interval* puede tener uno de los siguientes valores:

Valor de enumeración	Cadena	Parte del valor de fecha u hora devuelta
<b>DateInterval.Day</b>	d	Día del mes (del 1 al 31)
<b>DateInterval.DayOfYear</b>	y	Día del año (del 1 al 366)
<b>DateInterval.Hour</b>	h	Hora
<b>DateInterval.Minute</b>	n	Minuto
<b>DateInterval.Month</b>	m	Mes
<b>DateInterval.Quarter</b>	q	Trimestre
<b>DateInterval.Second</b>	s	Segundo
<b>DateInterval.Weekday</b>	w	Día de la semana (del 1 al 7)
<b>DateInterval.WeekOfYear</b>	ww	Semana del año (de la 1 a la 53)
<b>DateInterval.Year</b>	yyyy	Año

El argumento *FirstDayOfWeekValue* puede tener uno de los siguientes valores:

Valor de enumeración	Valor	Descripción
<b>FirstDayOfWeek.System</b>	0	Primer día de la semana especificado en la configuración del sistema
<b>FirstDayOfWeek.Sunday</b>	1	Domingo (predeterminado)

## Controles avanzados y referencia de funciones.

<b>FirstDayOfWeek.Monday</b>	2	Lunes (de acuerdo con la norma ISO 8601, sección 3.17)
<b>FirstDayOfWeek.Tuesday</b>	3	Martes
<b>FirstDayOfWeek.Wednesday</b>	4	Miércoles
<b>FirstDayOfWeek.Thursday</b>	5	Jueves
<b>FirstDayOfWeek.Friday</b>	6	Viernes
<b>FirstDayOfWeek.Saturday</b>	7	Sábado

El argumento *FirstWeekOfYearValue* puede tener uno de los siguientes valores:

Valor de enumeración	Valor	Descripción
<b>FirstWeekOfYear.System</b>	0	Primera semana del año especificada en la configuración del sistema
<b>FirstWeekOfYear.Jan1</b>	1	Semana en la que se encuentra el 1 de enero (predeterminado)
<b>FirstWeekOfYear.FirstFourDays</b>	2	Semana que contiene al menos cuatro días del nuevo año (de acuerdo con la norma ISO 8601, sección 3.17)
<b>FirstWeekOfYear.FirstFullWeek</b>	3	Primera semana completa del nuevo año

## Excepciones o errores

Tipo de excepción	Número de error	Condición
ArgumentException	5	<i>Interval</i> no es válido.
InvalidCastException	13	<i>DateValue</i> no se puede convertir a <b>Date</b> .

## Comentarios

Se puede utilizar la función **DatePart** para evaluar un valor de fecha u hora y devolver un componente específico. Por ejemplo, se podría utilizar **DatePart** para calcular el día de la semana o la hora actual.

Si se elige **DateInterval.Weekday** para el argumento *Interval*, el valor devuelto es coherente con los valores de la enumeración **FirstDayOfWeek**. Si elige **DateInterval.WeekOfYear**, **DatePart** utiliza las clases **Calendar** y **CultureInfo** del espacio de nombres **System.Globalization** para determinar la configuración actual.

El argumento *FirstDayOfWeekValue* afecta a los cálculos que utilizan los valores **DateInterval.Weekday** y **DateInterval.WeekOfYear** para *Interval*. El argumento *FirstWeekOfYearValue* afecta a los cálculos que especifican **DateInterval.WeekOfYear** para *Interval*.

Si algún argumento tiene un valor no válido, se produce un error **ArgumentException**. Si el argumento *DateValue* tiene un valor que no puede convertirse a un valor **Date** válido, se produce un error **InvalidCastException**.

## Controles avanzados y referencia de funciones.

Puesto que todos los valores **Date** se basan en una estructura **DateTime**, sus métodos proporcionan opciones adicionales para recuperar partes de fecha u hora. Por ejemplo, se puede obtener el valor de fecha completo de una variable **Date**, con el valor de hora establecido en medianoche, como se muestra a continuación:

```
Dim CurrDatTim As Date = Now           'Fecha y hora actuales
Dim LastMidnight As Date = CurrDatTim.Date    ' Medianoche.
```

### Ejemplo

En este ejemplo se utiliza la función **DatePart** sobre una fecha para mostrar el trimestre del año en que se produce.

```
Dim FirstDate, Msg As String    'Declara variables.
Dim SecondDate As Date
FirstDate = "1/1/2005"
SecondDate = CDate(FirstDate)
Msg = "Trimestre: " & DatePart(DateInterval.Quarter, SecondDate)
Response.Write(Msg)
```

### Hour

Devuelve un valor **Integer** entre 0 y 23 que representa la hora del día.

### Ejemplo

En este ejemplo se utiliza la función **Hour** para obtener la hora de una hora especificada. En el entorno de desarrollo, el literal de hora se muestra en formato de hora corto con los valores de configuración regional del código correspondiente.

```
Dim MyTime As Date
Dim MyHour As Integer
MyTime = #4:35:17 PM#    ' Asigna una hora
MyHour = Hour(MyTime)    ' MyHour contiene 16.
```

### Minute

Devuelve un valor **Integer** entre 0 y 59 que representa el minuto de la hora

## Ejemplo

En este ejemplo se utiliza la función **Minute** para obtener el minuto de una hora especificada. En el entorno de desarrollo, el literal de hora se muestra en formato de hora corto con los valores de configuración regional del código correspondiente.

```
Dim MyTime As Date

Dim MyMinute As Integer

MyTime = #4:35:17 PM#      ' Asigna una hora.

MyMinute = Minute(MyTime)  ' MyMinute contiene 35.
```

[Pulsa aquí para descargar los ejemplos de este tema](#)

# Ejercicios

**Hola, los ejercicios de este tema son de envío obligatorio. Si tienes alguna duda, ya sabes mándame una tutoria...**

## Ejercicio 1

Crea una página con un control multivista donde podamos definir una tarjeta de presentación donde en los pasos pondremos:

Paso 1: lectura de nombre, apellidos, ciudad (con un control de cuadro de lista) y fecha de nacimiento con un control de tipo calendario:

The screenshot shows a web browser window titled "Página sin título - Microsoft Internet Explorer proporcionado por Metzeler AP...". The address bar shows "http://localhost:1353/Enunciados". The page contains a form with the following fields:

- Nombre:** Text input field containing "Jose".
- Apellidos:** Text input field containing "Rodriguez".
- Ciudad:** Dropdown menu with options: Madrid, Barcelona, Sevilla, Valencia. "Madrid" is selected.
- Fecha de solicitud:** Calendar control showing "febrero de 2008". The date "27" is selected.

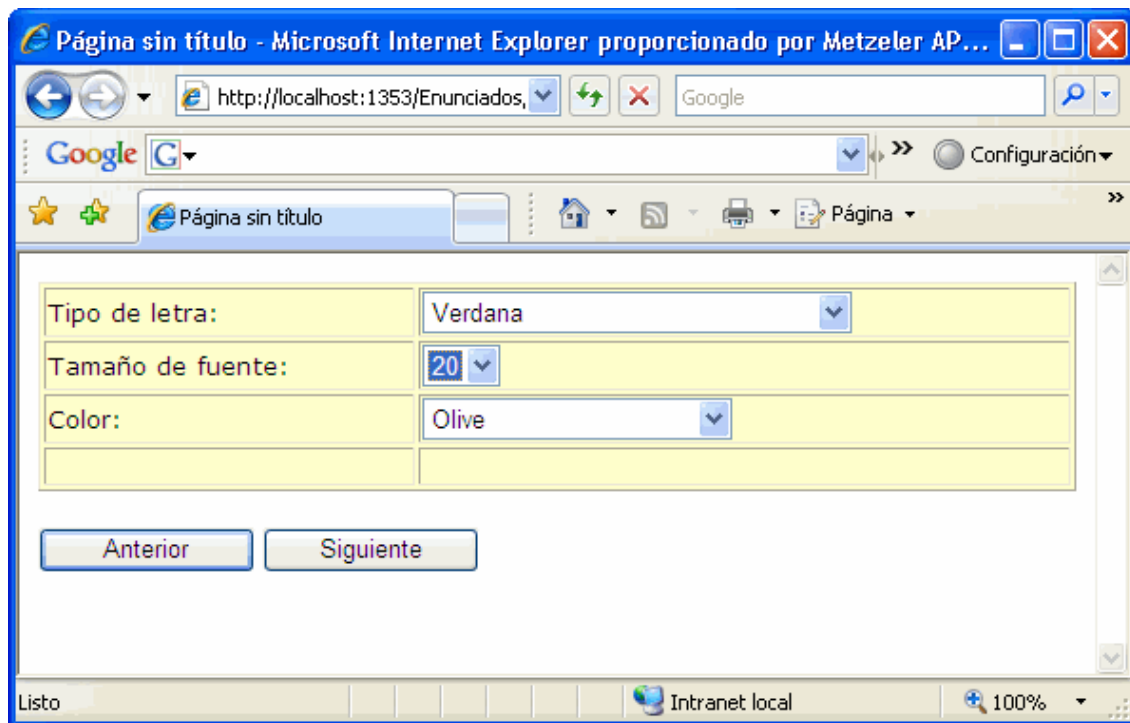
At the bottom of the form is a button labeled "Siguiente".

febrero de 2008						
lu	ma	mi	ju	vi	sá	do
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	1	2
3	4	5	6	7	8	9

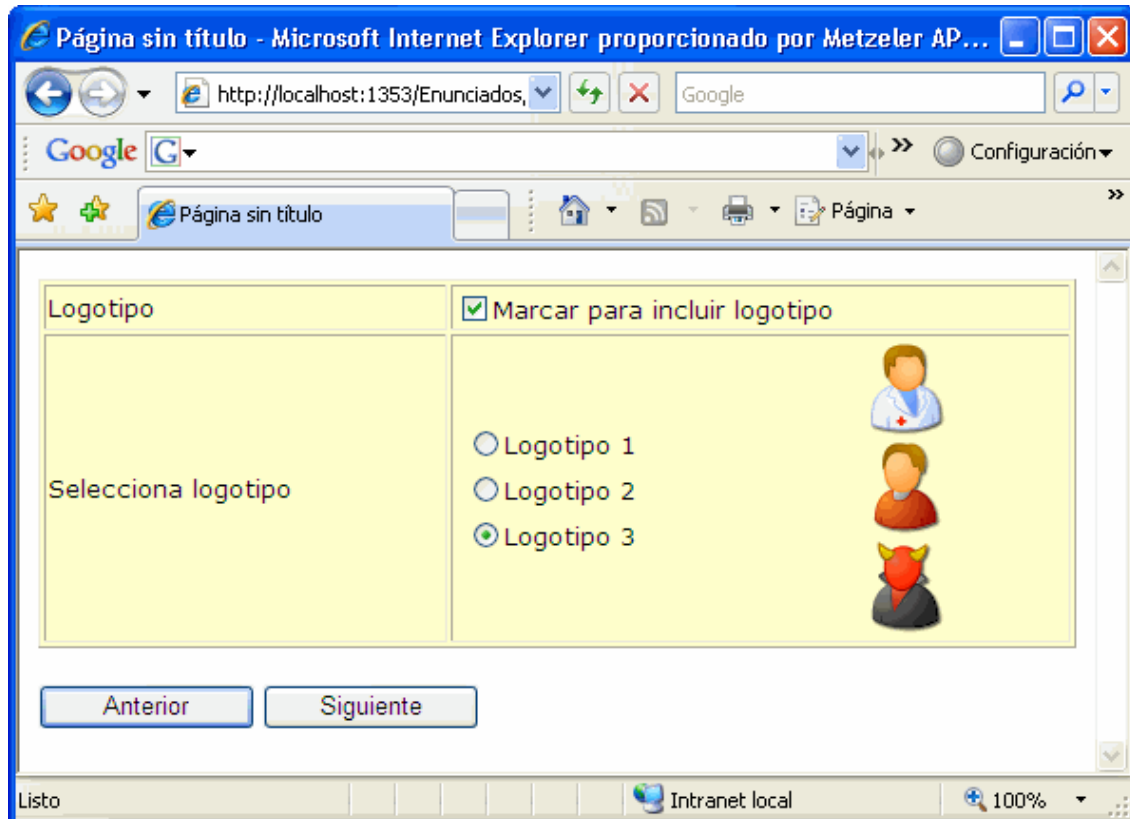


## Controles avanzados y referencia de funciones.

Paso 2: Tipo de letra, tamaño y color

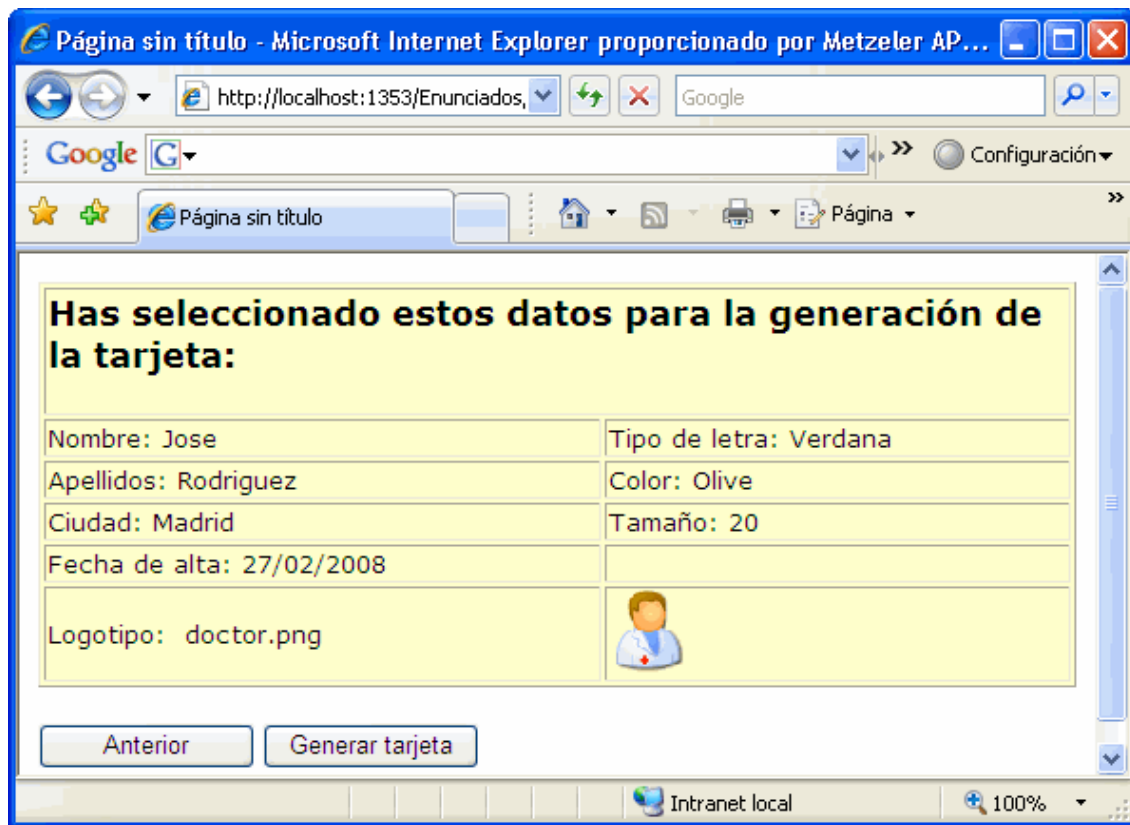


Paso 3: Definir si queremos poner un logotipo y elegir uno:

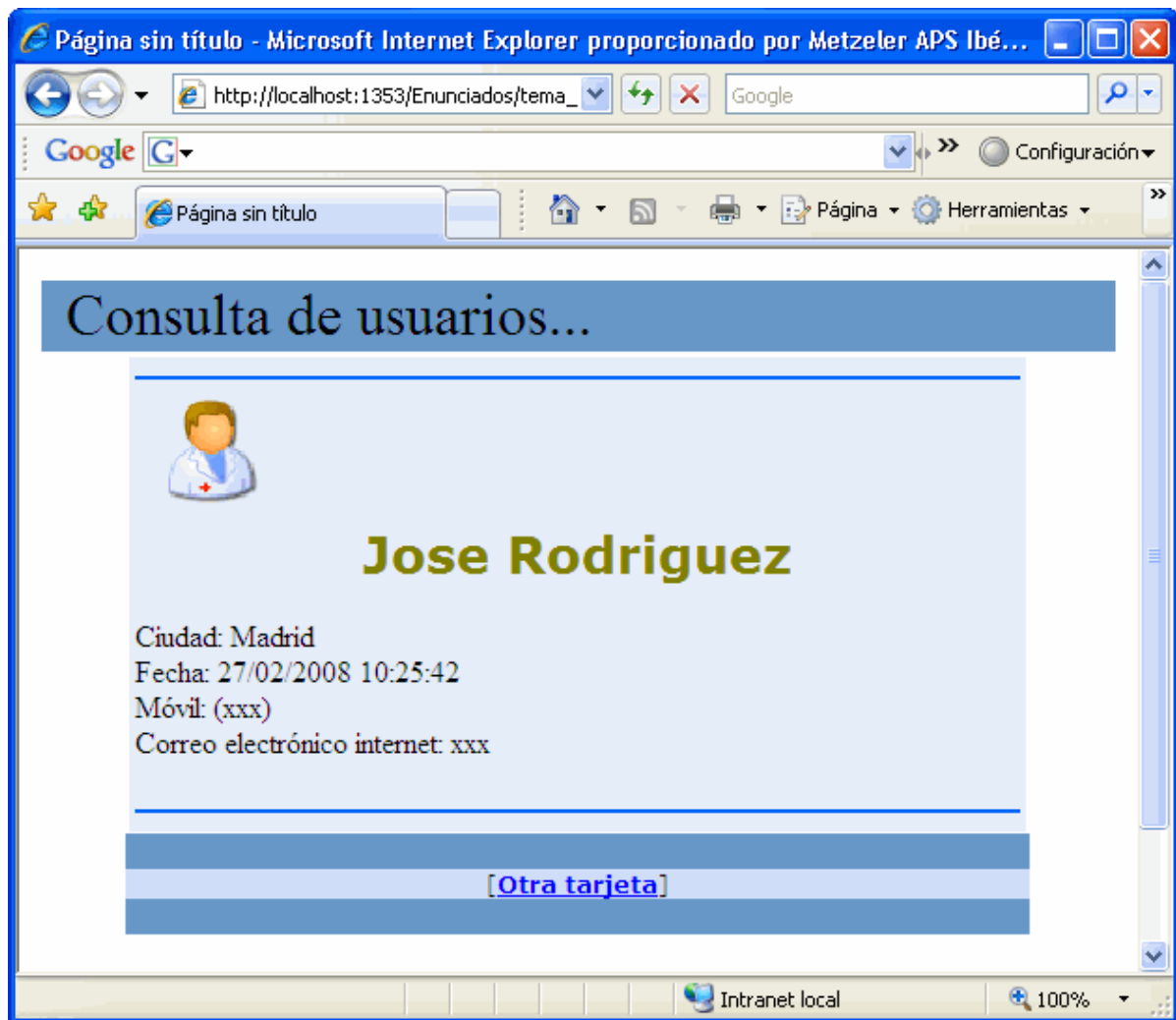


Paso 4: Mostrar un resumen de todo lo seleccionado:

Controles avanzados y referencia de funciones.



Y por fin generar una tarjeta de visita:



### Ayudas y comentarios:

-Los datos de las ciudades, carga de las fuentes, colores y tamaños y lo crearemos en el evento "Load" de la página.

-Debes crear 4 vistas con el control MultiView, por ejemplo:

## Controles avanzados y referencia de funciones.

asp:MultiView#Multivista

Multivista																																											
Vista1																																											
Nombre	<input type="text"/> Campo obligatorio																																										
Apellidos	<input type="text"/> Campo obligatorio																																										
Ciudad	Sin enlazar																																										
Fecha de solicitud:	<div> <div>&lt;</div> <div>febrero de 2008</div> <div>&gt;</div> </div> <table border="1"> <thead> <tr> <th>lu</th> <th>ma</th> <th>mi</th> <th>ju</th> <th>vi</th> <th>sá</th> <th>do</th> </tr> </thead> <tbody> <tr> <td>28</td> <td>29</td> <td>30</td> <td>31</td> <td>1</td> <td>2</td> <td>3</td> </tr> <tr> <td>4</td> <td>5</td> <td>6</td> <td>7</td> <td>8</td> <td>9</td> <td>10</td> </tr> <tr> <td>11</td> <td>12</td> <td>13</td> <td>14</td> <td>15</td> <td>16</td> <td>17</td> </tr> <tr> <td>18</td> <td>19</td> <td>20</td> <td>21</td> <td>22</td> <td>23</td> <td>24</td> </tr> <tr> <td>25</td> <td>26</td> <td>27</td> <td>28</td> <td>29</td> <td>1</td> <td>2</td> </tr> </tbody> </table>	lu	ma	mi	ju	vi	sá	do	28	29	30	31	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	1	2
lu	ma	mi	ju	vi	sá	do																																					
28	29	30	31	1	2	3																																					
4	5	6	7	8	9	10																																					
11	12	13	14	15	16	17																																					
18	19	20	21	22	23	24																																					
25	26	27	28	29	1	2																																					

- La generación se hará en una página independiente a la que se mandarían los datos mediante un postback a otra página con el último botón (propiedad postbackurl).

- Los campos de nombre y apellidos son obligatorios, así que debes poner controles de validación

- Los gráficos de los logos descárgalos de estas imágenes y las incluyes en la aplicación web para luego

mostrarlos:



- Los datos del postback en la página destino los puedes recoger así, según el tipo de control que sea:

- Cuadro de texto:

```
lb_nombre.Text = CType((PreviousPage.FindControl("txt_nombre")), TextBox).Text
```

- Cuadro de lista:

```
lb_ciudad.Text = CType((PreviousPage.FindControl("lista_ciudades")), ListBox).SelectedValue
```

- Botones de opción:

```
Dim nombre_fichero As String = CType((PreviousPage.FindControl("opcion_logo")), RadioButton)
```

La página de destino puedes crearla con FrontPage o SharePoint Designer y pegarla para luego ponerle etiquetas:

## Consulta de usuarios...

A user card template with a light blue background and a thin blue border. The card contains a small square icon with a red, green, and blue pixelated pattern in the top left corner. Below the icon, the text "[lb\_nombre]" is centered. Further down, the text "Ciudad: [lb\_ciudad]" is followed by "Fecha: [lb\_fecha]", "Móvil: (xxx)", and "Correo electrónico internet: xxx" on separate lines. At the bottom of the card, there is a blue horizontal bar with the text "[Otra tarjeta]" in white, underlined, and centered.

Así te será mas fácil diseñarla.

## Ejercicio 2

Repite lo mismo pero para un control de tipo asistente o "wizard". Puedes utilizar todo lo anterior y lo único que debes cambiar es la edición de la plantilla del último elemento para poner los botones para poder asignarle el "postback" a la página que escribe los datos:

[Datos Personales](#)  
[Formato de la tarjeta](#)  
[Logotipo](#)  
[Final...](#)

**Has seleccionado estos datos para la generación de la tarjeta:**

Nombre: eee	Tipo de letra: Browallia New
Apellidos: ee	Color: Menu
Ciudad: Madrid	Tamaño: 24
Fecha de alta: 27/02/2008	
Logotipo: doctor.png	

[Anterior](#) [Generar tarjeta](#)

Para activar cualquier página puedes utilizar: Wizard1.ActiveStepIndex = 1