

Controles Web, clases y eventos. Estado de la aplicación

Indice

Nº 7 Controles Web, clases y eventos. Estado de la aplicación.....	1
1. Introducción a los controles de servidor de ASP.NET o controles Web.....	1
1.1 Clases de controles Web.....	2
2. Clases de controles Web.....	4
2.1 Unidades.....	6
2.3 Enumeraciones.....	7
2.4 Colores.....	7
2.5 Fuentes.....	8
2.6 El enfoque.....	9
2.7 El botón "default".....	10
2.8 El control <asp:label>.....	10
2.9 Control de cuadro desplegable <asp:dropdownlist>.....	15
Mejoras de la aplicación web.....	22
5. Estado de la aplicación.....	25
5.1 Administración del estado.....	25
5.2 Estado de la aplicación.....	26
5.3 Transferir información entre páginas.....	30
5.4 Query string.....	34
5.5 Cookies.....	39
5.6 Sesiones.....	42
5.7 Estado de la aplicación (Application).....	52
¿Que tipo de almacenamiento de estado debo utilizar?.....	54
Ejercicios.....	55
Ejercicio 1.....	55
Ejercicio 2.....	56
Ejercicio 3.....	58

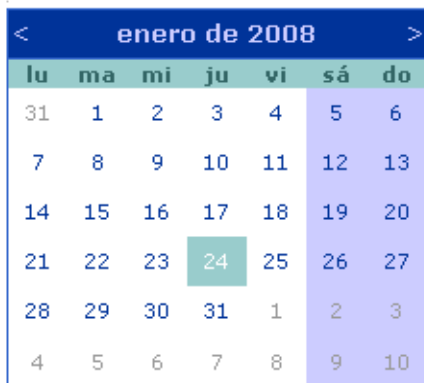
Nº 7 Controles Web, clases y eventos. Estado de la aplicación

1. Introducción a los controles de servidor de ASP.NET o controles Web

Una vez vistos todos los controles HTML veamos ahora los controles Web. ¿por qué dos tipos de controles? Ya lo hemos comentado en otro momento, por un lado están los controles básicos y estándar de HTML que con una pequeña modificación, básicamente poniéndoles un identificador ID y que se ejecuten en el servidor (`runat="server"`) nos han permitido mejorar y ampliar sus funciones. Para los que quieran controlar formularios muy sencillo les será suficiente. Pero la verdadera potencia empieza ahora con los controles "de verdad" los que proporciona ASP.NET y que son los que proporcionan una potencia y programación realmente potente.

Tenemos ya una ventaja y es que conocemos muchos objetos importantes y además el funcionamiento de la programación orientada a eventos.

¿Por qué necesitamos mas controles? Muy sencillo nuestro objetivo es crear páginas web que sean cada vez mas potentes y parecidas a las aplicaciones de Windows. De momento hemos ganado al utilizar los controles HTML del lado del servidor captando en ellos la interacción del usuario con los eventos, aunque de forma un poco rudimentaria y demás, con objetos muy básicos. Imagina el control calendario de una aplicación Windows: es sencillo de manejar y ofrece muchas posibilidades tanto gráficas como de programación, pues bien, este control no existe en los HTML y además sería imposible de simular. Sin embargo tenemos un control de servidor de ASP.NET que nos ofrece casi las mismas prestaciones y aspecto para este control así:



< enero de 2008 >						
lu	ma	mi	ju	vi	sá	do
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

Por tanto vemos la intención de estos controles Web ya que nos van a intentar ofrecer una programación y aspecto similares a las aplicaciones Windows. Los controles HTML además se correspondían directamente con las etiquetas HTML estándar, luego había una traducción sencilla a HTML estándar. En los controles Web no tenemos esa restricción porque no intentan generar el control equivalente en HTML sino que generalmente el código HTML necesario para ofrecer la funcionalidad completa, sin límite de su equivalente, por que sencillamente no existe: existe un control HTML de tipo botón en HTML estándar: `<input type="submit">` pero no existe el control Web del calendario. Así que es mucho mas versátil y:

- **Proporciona una interfaz de usuario mucho mas potente.** Un control Web se programa como un objeto pero no necesariamente corresponde con un elemento HTML. Los calendarios o tablas generarán tanto HTML como necesiten, pero no necesitaremos conocer nada de ese código ya que lo generará de forma automática.
- **Proporciona un modelo de objetos consistente.** Por ejemplo un texto en HTML se puede escribir con `<textarea>`, `<input type="text">` o `<input type="password">`, ahora pasaremos a utilizar un solo control de tipo `TextBox` que dependiendo de sus propiedades admitirá multilínea (como los `textarea`), introducción de contraseñas (tipo=`password` que oculta los caracteres) y muchas mas opciones.
- **Controlan la salida que generan de forma automática.** los controles de servidor de ASP.NET detectan el tipo de navegador que hace la solicitud y ajusta el HTML para que el aspecto final sea igual en todos.
- **Proporciona características de alto nivel.** Nos va a proporcionar en los controles distintas propiedades, métodos y eventos como los ofrecen sus equivalentes en aplicaciones Windows

La finalidad de utilizar los anteriores controles HTML es la de realizar una traducción rápida de páginas ya realizadas de HTML a ASP.NET, ya que simplemente le cambiábamos la ubicación de ejecución al servidor y ya desde ahí podíamos seguir trabajando. A partir de ahora solo utilizaremos controles ASP.NET que serán los que utilicemos ya siempre en el desarrollo de aplicaciones Web.

1.1 Clases de controles Web

Si has creado anteriormente aplicaciones Web seguramente te será muy familiar la utilización de los controles básicos como etiquetas, cuadros de texto, botones, ... la idea de ASP.NET es la de trasladar esa programación a las páginas Web variando muy poco su filosofía. En la siguiente tabla tenemos una lista de los controles mas utilizados y el código HTML que van a generar:

Clase	Etiqueta HTML que genera:
Label	<code></code>
Button	<code><input type="button"></code> ó <code><input type="submit"></code>
TextBox	<code><input type="text"></code> , <code><input type="password"></code> ó <code><textarea></code>
CheckBox	<code><input type="checkbox"></code>
RadioButton	<code><input type="radio"></code>
Hyperlink	<code><a></code>
LinkButton	<code><a></code> con una imagen <code></code>
ImageButton	<code><input type="image"></code>
Image	<code></code>
ListBox	<code><select size="X"></code> , X es el número de filas visibles
DropDownList	una lista o tabla con múltiples etiquetas <code><input type="checkbox"></code>
RadioButtonList	una lista o tabla con múltiples etiquetas <code><input type="radio"></code>
BulletedList	<code></code> para listas numeradas y <code></code> para listas con indicadores gráficos (bullets)

Panel	<div>
Table, TableRow y TableCell	<table>, <tr> y <td> ó <th>

Nos da igual lo que nos genere ASP.NET, en algunos controles es una etiqueta HTML estándar pero en la mayoría de los casos serán varias etiquetas lo que será necesario para obtener el mismo aspecto que el control.

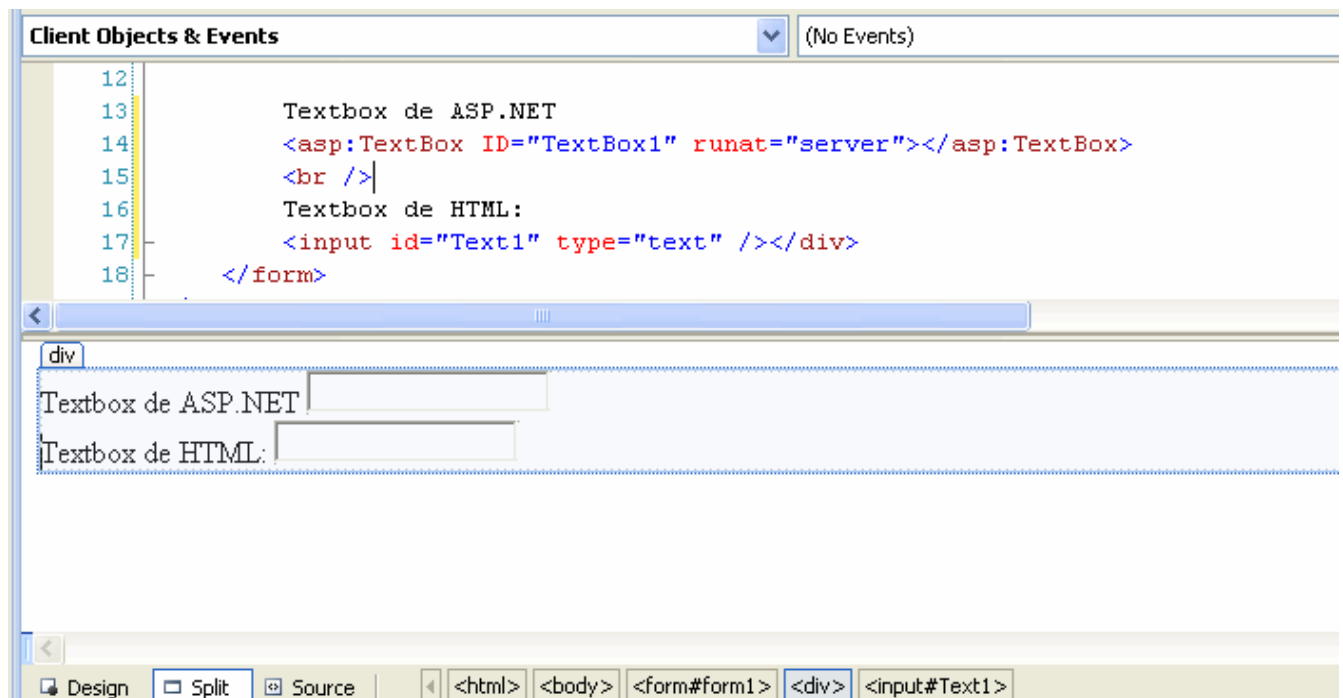
Los controles de servidor tienen una sintaxis especial, siempre utilizan el prefijo "asp:" seguida del nombre de la clase finalizando la etiqueta de definición con "/" en lugar de con una etiqueta de cierre </fin>. Por ejemplo, un control de cuadro de texto sería:

```
<asp:TextBoxID="txt" runat="server" />
```

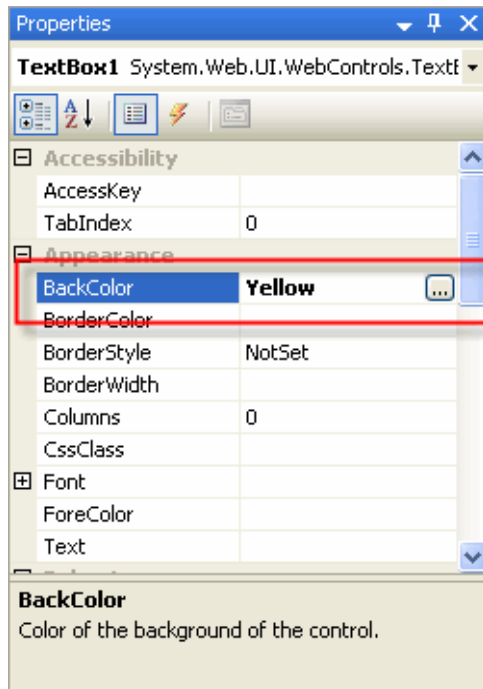
Luego cuando se procesa en el IIS por ASP.NET devolverá al navegador HTML estándar con:

```
<input type="text" id="txt" name="txt" />
```

Es un caso sencillo porque la traducción corresponde con una sola etiqueta HTML pero así lo entendemos mejor. Por ejemplo, pinta con el IDE un control de tipo cuadro de texto (textbox) a ver si es cierto esto que digo y luego pinta debajo uno pero no de ASP.NET sino de los que hemos utilizado antes de tipo HTML:



Fíjate que gráficamente son idénticos pero en el código se declaran de forma diferente y nos ofrecerán muchas posibilidades de programación. En la definición de los controles ASP.NET podemos añadir tantas propiedades como queramos. La ventaja es que no las tendremos que escribir ya en el código sino que las pondremos con la ventana de propiedades. Nunca escribas ya en el código HTML, dejemos que el IDE lo escriba por nosotros. Por ejemplo vamos a poner estas propiedades en el control: que el fondo sea de color amarillo, el texto sea "Hola", que sea de sólo lectura y multilínea de 5 líneas. Todo esto desde el IDE:



El IDE le pondrá estas propiedades al control a medida que las vayamos seleccionando:

```
<asp:TextBoxID="txt" BackColor="Yellow" Text="Hola"ReadOnly="True"
    TextMode="MultiLine" Rows="5" runat="server" />
```

Si ejecutamos la página veremos que ASP.NET lo envía al navegador con este código:

```
<textarea name="txt" rows="5" cols="20" readonly="readonly" id="txt"
    style="background-color:Yellow;">Hola</textarea>
```

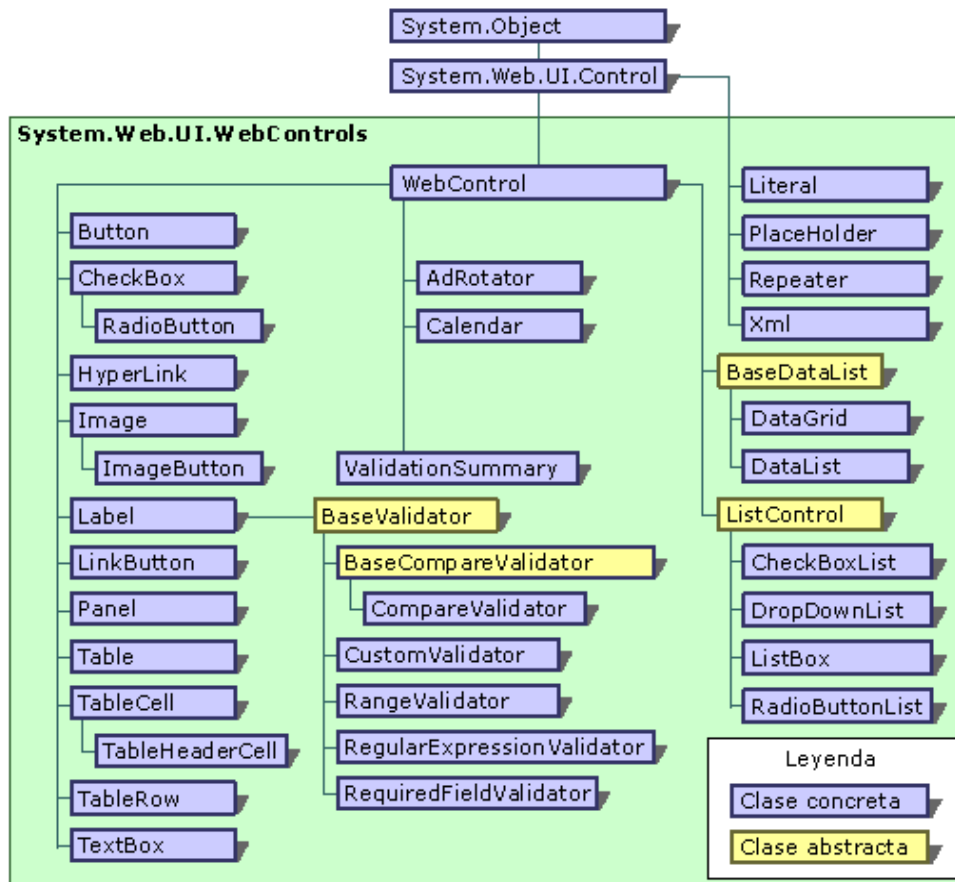
Ya ves que ya no necesitamos saber HTML, pondremos el control, le asignaremos unas propiedades y ASP.NET lo traducirá a las etiquetas HTML necesarias.

Nota: Así como en los ficheros de configuración (web.config) escritos en XML, hay que tener cuidado con las mayúsculas y minúsculas, en la escritura de los controles Web no hay que tener este cuidado. Por ejemplo la propiedad text="Hola" es igual que tEXt="hola"

2. Clases de controles Web

Las clases de control Web se encuentran en el espacio de nombres "System.Web.UI.WebControls" que jerárquicamente son:

Controles Web, clases y eventos. Estado de la aplicación



Vemos que todos los controles se heredan de la clase base "WebControl", veamos algunos detalles de esta clase. Esta clase define la funcionalidad básica y tareas como el enlace a datos y otras propiedades interesantes, veamos algunas de ellas:

Propiedad	Descripción
AccessKey	Especifica la tecla de acceso directo al control. Por ejemplo, si le asignamos la "P", si el usuario pulsa las teclas Alt+P este control tendrá el enfoque.
BackColor, ForeColor y Bordercolor	Colores para el control, en primer plano, el segundo y el contorno
BorderWidth	Tamaño del borde del control
BorderStyle	Valores para el borde del control: sólido, ninguno, línea discontinua...
Controls	Nos proporciona una colección de los controles contenidos en este control. Cada objeto proporciona un objeto genérico del tipo System.Web.UI.Control
Enabled	Si se pone a "false" el control estará visible pero no tendrá el enfoque ni se podrá modificar
EnableViewState	Si se pone a "false" no almacenará su estado, así que cada vez que se visite la página se restablecerá a su valor inicial.
Font	Especifica la fuente utilizada, del tipo System.Web.UI.WebControls.FontInfo
Height y Width	Especifica la anchura y altura del control.

Page	Proporciona una referencia al objeto de la página (Page) del tipo System.Web.UI.Page
Parent	Proporciona una referencia al control que contiene el actual, si no hay ninguno devolverá una referencia a la página
TabIndex	Número que permite ordenar los tabuladores. Al pulsar el tabulador cambia el enfoque al control que indique este número.
ToolTip	Al pasar el ratón por encima muestra un texto de ayuda para el usuario.
Visible	Si es "false" el control estará oculto y no se enviará a la página web del usuario.

2.1 Unidades

Nos vamos a detener un momento para comentar cómo son las unidades de medida en ASP.NET. Todos las propiedades de los controles que necesitan medidas o posiciones utilizarán una estructura de tipo "Unit" que combina un valor numérico con el tipo de medida: píxeles, porcentajes, ... Esto significa que cuando establecemos estas propiedades en un control debemos asegurarnos de añadir px (píxel) o % (para porcentajes) al valor de la medida para indicarle la unidad de medida. Por ejemplo para un control de tipo panel con 200 píxeles de ancho y una altura equivalente al 40% del navegador:

```
<asp:Panel Height="300px"Width="50%"id="panel" runat="server" />
```

Si queremos hacer asignaciones en nuestro código utilizaremos los métodos necesarios de Unit, por ejemplo:

```
'Para una anchura de 300 píxeles:
```

```
panel.Height= Unit.Pixel(300)
```

```
'Para un porcentaje del 50%:
```

```
panel.Width=Unit.Percentage (50)
```

También podemos hacer una declaración para crear de nuestro tipo de unidad:

```
'Creamos un objeto de unidad Unit
```

```
Dim Miunidad as New Unit (300,UnitType.Pixel)
```

```
'Asignamos la unidad a varios controles o propiedades:
```

```
panel.Height=Miunidad
```

```
panel.Width=Miunidad
```

"Unit" es una sencilla estructura para expresar las medidas en un formato determinado. Permite poner mediante "UnitType", como has visto antes, el tipo de unidad de medida que puede ser: píxel, punto, pica, pulgada, milímetro, centímetro, porcentaje y Em-Ex que son relativas a la altura del elemento "font" padre.

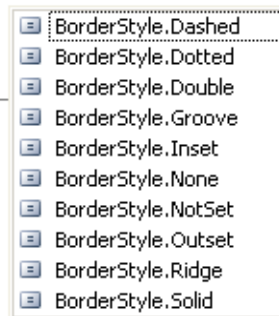
2.3 Enumeraciones

Las enumeraciones se utilizan mucho en .NET como un grupo de constantes relacionadas, por ejemplo, los colores de Windows: White, Blue, ... son constantes que están asociadas con el número del color, así es mucho mas sencillo indicarle el valor de esa enumeración que no un valor numérico que no dice nada.

Por ejemplo, cuando ponemos un contorno a un control podemos definir cómo será este contorno: sólido, discontinuo, ... en lugar de indicarle un valor numérico que indique esta propiedad utilizaremos un valor de la enumeración de contornos que ya tiene definida .NET. Por ejemplo:

```
Private Sub test()  
    txt.BorderStyle=  
End Sub
```

Class



Al escribir que queremos poner la propiedad del contorno y poner el signo de igual "=" nos muestra la lista o enumeración de posibilidades de esta propiedad, son valores constantes pero están agrupadas en una enumeración.

En el código .aspx el valor de una enumeración lo pondremos directamente con el texto:

```
<asp:Label BorderStyle="Dashed"Text="Contorno" ID="label1" runat="server" />
```

2.4 Colores

Las propiedades donde necesitemos asignar un color se referirán al objeto "Color" del espacio de nombres "System.Drawing". Podemos crear colores de varias formas:

- Utilizando los valores **ARGB** (alpha, red, green, blue) o lo que es lo mismo (transparencia, rojo, verde, azul). Cada valor es de 1 byte por lo tanto tendrán valores desde 0 hasta 255 cada uno de ellos. El primero de ellos es el valor de la transparencia del color, pondremos un 255 para que sea opaco.
- Usando **un nombre de color predefinido de .NET**. Podemos utilizar cualquier color de los 140 definidos por .NET
- Utilizando **un nombre de color HTML.**, podemos utilizar este valor como una cadena de caracteres utilizando la clase "ColorTranslator"

Como hemos dicho que el color está en otro espacio de nombres distinto al que tienen nuestros controles web tendremos que importarlo a nuestra página para poder utilizarlo directamente:

```
import System.Drawing
```

Veamos unos ejemplos de colores:

Controles Web, clases y eventos. Estado de la aplicación

```
'Con los valores de ARGB:

Dim transparencia as Integer= 255, rojo as Integer=0

Dim verde as Integer=0, azul as Integer=0

control.ForeColor=Color.FromArgb (transparencia, rojo, verde, azul)

'Utilizandoun nombre de .NET:

control.ForeColor=Color.Salmon

'Utilizandoun color de una página HTML

control.ForeColor =ColorTranslator.FromHtml ("blue")
```

La definición dentro de la sintaxis del control será por el nombre o por el número hexadecimal:

```
<asp:TextBoxForeColor="red" Text="Hola" id="txt" runat="server" />

<asp:TextBoxForeColor="#ff50ff" Text="Hola" id="txt" runat="server" />
```

2.5 Fuentes

Las fuentes las trataremos de forma parecida a la anterior, pero utilizando el objeto FontInfo que está definido en el espacio de nombres "System.Web.UI.WebControls". Cada objeto "fontInfo" tiene distintas propiedades que definirán su nombre, tamaño y estilo:

Propiedad	Descripción
Name	Una cadena indicando el nombre del tipo de letra, por ejemplo "Arial"
Names	Matriz de nombres de fuentes. El navegador utilizará la primera instalada que coincida con un nombre de la matriz
Size	Tamaño de la fuente según el objeto FontUnit
Bold, Italic, StrikeOut, Underline	Valores de tipo "booleano" para indicar si es negrita, subrayada, cursiva o tachada

En el código podemos asignar los valores directamente:

```
Cuadro_texto.Font.Name="Verdana"

Cuadro_texto.Font.Bold=True
```

El tamaño lo podemos establecer con FontUnit:

```
Cuadro_texto.Font.Size=FontUnit.Small o con un

valor absoluto:
```

```
Cuadro_texto.Font.Size=FontUnit.Point (14)
```

En el código de la página tendremos que indicarlo de esta forma si queremos escribirlo a mano en lugar de

con el IDE:

```
<asp:TextBoxFont-Name="Verdana"Font-Size="40" Text="hola"
    Id="Cuadro_texto" runat="server" />
```

ó

```
<asp:TextBoxFont-Name="Verdana"Font-Size="Large"
    Text="hola" Id="Cuadro_texto" runat="server" />
```

El sentido de la propiedad "Names" es proporcionar una lista de fuentes, así el navegador mostrará la primera que vea que tiene instalado el equipo, así que con Names sería:

```
<asp:TextBoxFont-Names="Verdana,Arial,Tahoma"Font-Size="40"
    Text="hola" Id="Cuadro_texto" runat="server" />
```

2.6 El enfoque

El enfoque es el control que tiene el cursor activo en ese momento. Habitualmente en las aplicaciones Windows lo manejas con el **tecla del tabulador**. En las páginas web también tenemos esta posibilidad y por tanto instrucciones para controlar el enfoque.

Los controles HTML no tenían esta posibilidad pero en los controles Web disponemos del método "Focus()" que afectarán solo a los controles que sean para introducir datos, lógico, sería una tontería pasar el enfoque a texto o imágenes. Lo lógico será que vaya pasando por los controles que permiten introducir datos: cuadros de texto, botones, listas, ...

Así que cuando **se termine de cargar la página podemos poner el método "focus()" en el control que queremos que tenga el enfoque, así le facilitamos al usuario la introducción de datos.**

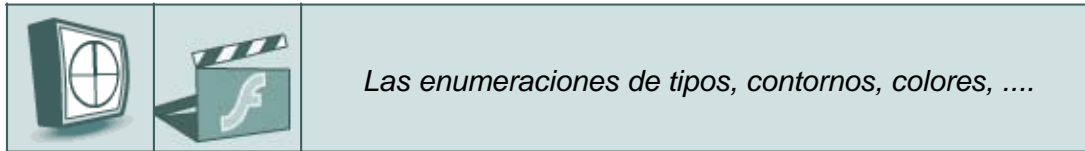
En el código del control lo pondremos así:

```
<form DefaultFocus="TextBox2" runat="server">
```

Otra forma de establecer lo enfoques es poniendo las teclas de acceso directo, como ya hemos comentado anteriormente. Si ponemos a un control la tecla "alt + a", ese control recogerá el enfoque cuando se pulse esa combinación de teclas, por ejemplo:

```
<asp:Label AccessKey="2" AssociatedControl ID="Textbox2" runat="server" text="hola" />
<asp:TextBoxrunat="server" ID="TextBox2">
```

En este caso además lo hemos aplicado a una etiqueta Label, que aunque no permite la entrada de datos es la única excepción que si permite que recoja el enfoque, pero se lo asigna al control siguiente. Es decir las dos instrucciones anteriores están sincronizadas, si se pulsa la tecla de acceso directo "alt+2" que está definida en la etiqueta "Label", el control "textbox2" activará el enfoque.



2.7 El botón "default"

Los programadores de aplicaciones Windows conocerán de sobra las funciones de este botón y que te describo ahora. En los formularios siempre hay un botón activo para cuando el usuario pulse la tecla "intro" se haga esa operación. Por ejemplo, tenemos varios controles por los que el usuario se mueve con la tecla del tabulador y cuando termina de introducir los datos pulsa "enter" o "intro". En ese caso queremos hacer un "submit" de los datos para enviarlos al servidor, así que se disparará el evento clic del botón de enviar, la página se envía y se dispara "button1.click"

Para definir un botón "default" o predeterminado estableceremos la propiedad `HtmlForm.DefaultButton` con el ID o identificador del botón que queramos:

```
<form DefaultButton="cmd_enviar" runat="server">
```

Los tipos de botones que pueden utilizar esto son: `Button`, `LinkButton` e `ImageButton`, pero nunca los controles HTML que, ya sabemos, son muy inferiores en prestaciones.

2.8 El control <asp:label>

Comenzaremos con el mas sencillo de los controles: el control de etiqueta `<asp:label>`. Este control proporciona una alternativa para mostrar texto en nuestra página web. Es fundamental para separar el código HTML del ASP ya que podremos escribir el contenido de variables en la posición de la página que queramos...

Nota Esta insistencia que hago de que debemos separar el código HTML del ASP.NET me la agradecerás con el tiempo. Sobre todo si eres como yo "un reciclado" es decir hemos pasado de trabajar con la antigua tecnología ASP a esta nueva ASP.NET.

La idea de utilizar estas etiquetas es la siguiente

1. Tenemos un formulario que va a consulta la edad de una persona.
2. Hacemos el formulario y la página de destino para realizar la consulta
3. Ponemos en el `Form_Load` el código necesario para acceder a la base de datos y recuperamos su edad.
4. Escribimos el resultado... ¿pero dónde lo escribimos?

Ahí, en ese punto es donde entran los controles de formulario de ASP.NET. En lugar de escribirlo inmediatamente como sabemos hacer con el "response.write" lo que hacemos es asignar el valor de la edad a una variable. Luego en la posición que yo quiera de la página web podremos una etiqueta ASP.NET y le asignaremos ese valor. ¿Lo has entendido? Lee otra vez esta última parte, es vital que la entiendas. De esta forma podremos escribir datos en cualquier parte de la página: insertando controles de formulario con valores que le hemos asignado en la parte inicial de la página que es donde se encuentra el código ASP.NET. Tranquilo veamos detalles de esta etiqueta y luego haremos un ejemplo. Esto ya lo tenemos visto de antes pero ahora quiero enumerar y hacer ejemplos sencillos con los controles de ASP.NET

Atributos del control <asp:label>

Estos son los atributos o propiedades mas importantes del control "label":

Atributo o propiedad	Resultado
BackColor	Establece el color de fondo de la etiqueta
ForeColor	Establece el color del texto
Height	Establece la altura en píxeles de la etiqueta
ID	Identificador único para el control
Text	Establece el texto que queremos mostrar
Visible	Indicasi el control está visible o no en la página. Los valores son True (verdadero) o False (falso)
Width	Establece la anchura en píxeles de la etiqueta

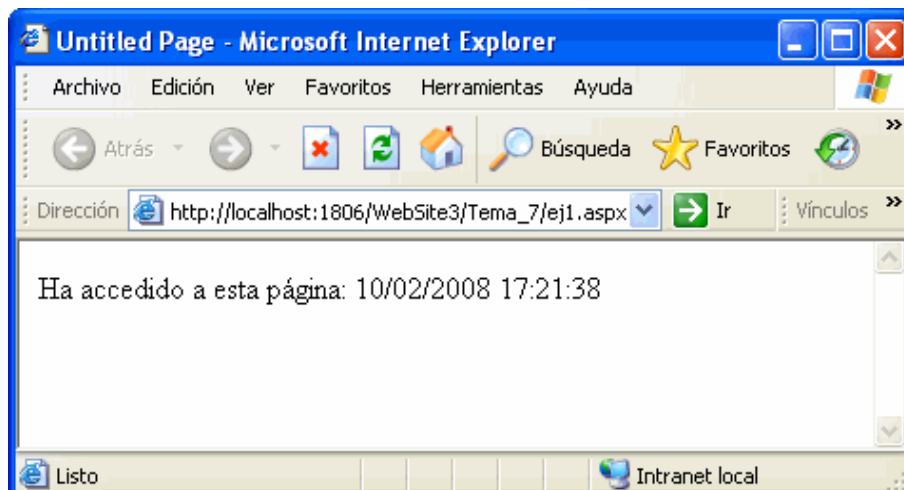
Ahora veremos un par de ejemplo del control ASP.NET <asp:label>

Ejemplo 1

Crea una página ej1.aspx con un control de tipo Label (de la sección de controles "Standar" no de los de "HTML") y ponle este código en el evento Load de la página:

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
    Label1.Text = "Ha accedido a esta página: " & DateTime.Now
End Sub
```

Ejecutamos la página y:



No hay nada nuevo ya, hemos puesto un objeto y al cargar la página le hemos asignada un texto en su propiedad "Text".

Ejemplo 2

Para crear un control lo mínimo obligatorio es:

```
<asp:label id="mensaje" runat="server">Hola</asp:label>
```

Por un lado la etiqueta de asp:label para que el servidor sepa de que control se trata, el "runat=server" para saber que es de servidor (siempre lo es) y un texto predefinido que tiene por ejemplo "hola". AL final como siempre en HTML se le indica que termina la etiqueta </asp:label> o simplemente al final la declaración un ">".

Ahora queremos que el texto aparezca en otro color:

```
<asp:label id="mensaje" forecolor="red" runat="server">Hola</asp:label>
```

Le hemos añadido uno de los atributos o propiedades vistas antes para mostrar el texto en color.

Por último podemos poner el texto que queremos escribir en el atributo Text que hemos visto antes. En este caso como el valor a escribir está dentro hay que terminar la línea de esta forma:

```
<asp:label id="mensaje" runat="server" text="Hola" />
```

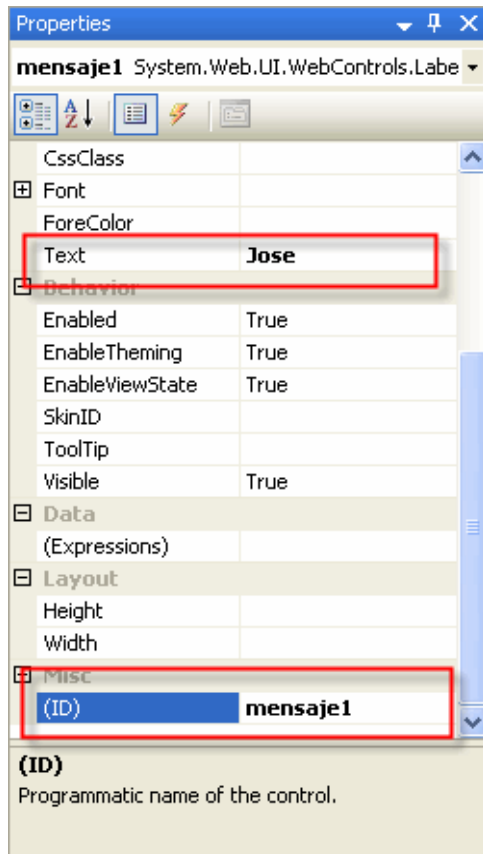
Es mejor escribirlo así porque si vamos a estar cambiando el contenido de esta etiqueta queda más lógico que sea la del valor de la propiedad "Text". Omitimos la etiqueta de cierre y le ponemos un "/" al final. Esta notación para poner la etiqueta de cierre es la que veremos en este curso y que seguramente ya habías apreciado.

Ahora veamos otro ejemplo:

1. Crea o reutiliza la página anterior para escribir dos etiquetas de tipo "label" que se llamarán "mensaje1" y "mensaje2" y con un texto predeterminado en cada una de ellas. Estos datos los debes poner en el IDE en la pantalla de propiedades. Primero pinta los "label" y le escribes un texto a continuación. Esas etiquetas serán las que hagamos variables:

```
Label, esto es un ejemplo para ver alguna variable en pantalla. Hoy estamos a:  
Label A esperar al fin de semana.
```

Y le ponemos los nombres y dos textos predeterminados. A la primero le pondremos "Jose" y a la segunda "lunes":



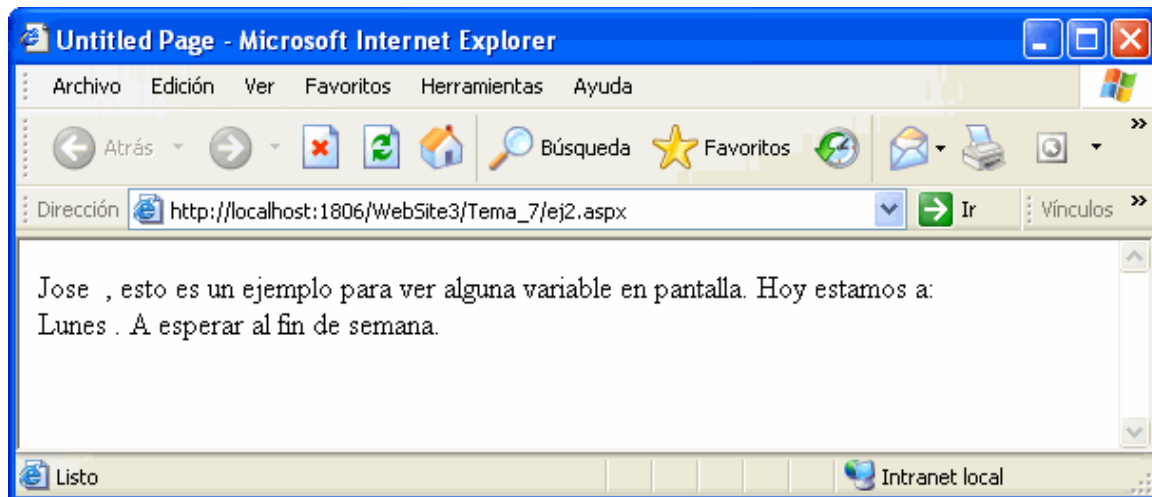
Si vemos el HTML, tendrá:

```
<form id="form1" runat="server">
  <div>

    <asp:Label ID="mensaje1" runat="server" Text="Jose"></asp:Label>
    &nbsp; , esto es un ejemplo para ver alguna variable en pantalla. Hoy estamos a:<br />
    <asp:Label ID="mensaje2" runat="server" Text="Lunes"></asp:Label>
    . A esperar al fin de semana.<br />

  </div>
</form>
```

2. Graba la página y vamos a verla en el navegador:

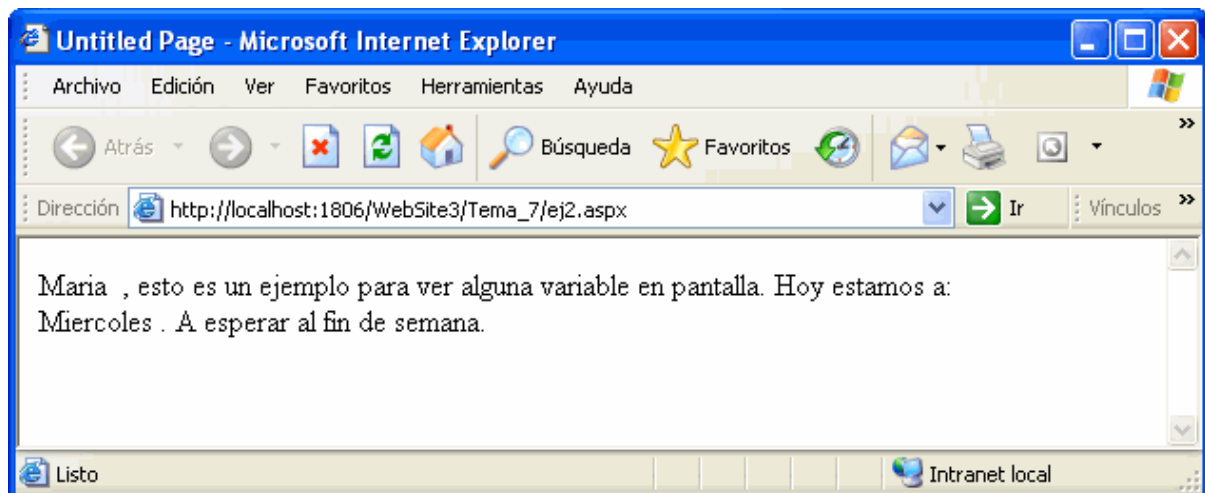


Espero que previeras el resultado ya que es sencillo... Por un lado simplemente hemos escrito un texto fijo. A continuación un control de servidor "mensaje1" con un valor "Jose". Luego hemos escrito mas texto estándar y por fin otro control de servidor "mensaje2".

¿Y dónde está la gracia de esto?. Pues ya ves, sólo con cambiar el contenido de las variables o controles "mensaje1" y "mensaje2" la página ya deja de ser estática para ser dinámica. Veamos si es cierto. Vamos a poner el famoso procedimiento que se ejecuta al cargar la página y le vamos a poner un valor a estas variables... escribe lo siguiente en la parte superior de la página:

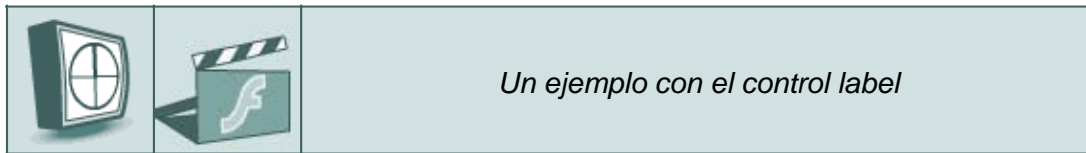
```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles  
    mensaje1.Text = "Maria"  
    mensaje2.Text = "Miercoles"  
End Sub
```

Ejecutamos la página ahora y:



Ha sucedido lo esperado, el texto estático permanece estático pero el variable (las dos etiquetas ASP) se ha actualizado...





Sigamos con mas controles.

2.9 Control de cuadro desplegable <asp:dropdownlist>

Este control, bastante mas complejo y completo que el anterior es uno de los mejores para demostrar la utilidad de poder ejecutar controles en el servidor.

Recuerda primero el código HTML de un control desplegable:

```
<selectsize=1 name="D1">

  <optionvalue="Enero">Enero</option>

  <optionvalue="Febrero">Febrero</option>

  <optionvalue="Marzo">Marzo</option>

  <optionvalue="Diciembre">Diciembre</option>

</select>
```

La definición en HTML comienza con un <select> y termina con un </select>. Luego cada uno de los elementos que producen la lista se indican con <option> para obtener este ejemplo:

Ahora la forma de crear esto con ASP.NET es la siguiente:

```
<asp:dropdownlist id="lista"runat="server">

  <asp:listitem>Enero</asp:listitem>

  <asp:listitem>Febrero</asp:listitem>

  <asp:listitem>Marzo</asp:listitem>

  <asp:listitem>Dociembre</asp:listitem>

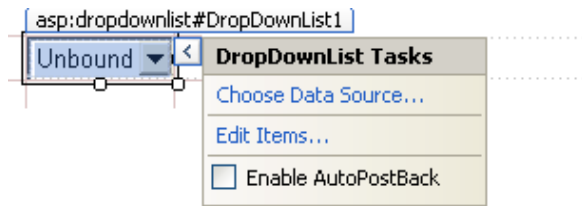
</asp:dropdownlist>
```

La diferencia sintáctica es mucha pero la filosofía es parecida. Una etiqueta indica que es un control de servidor de este tipo <asp:dropdownlist> y luego tiene una lista de elementos llamados <asp:listitem>. Así que tenemos tres cosas fundamentales:

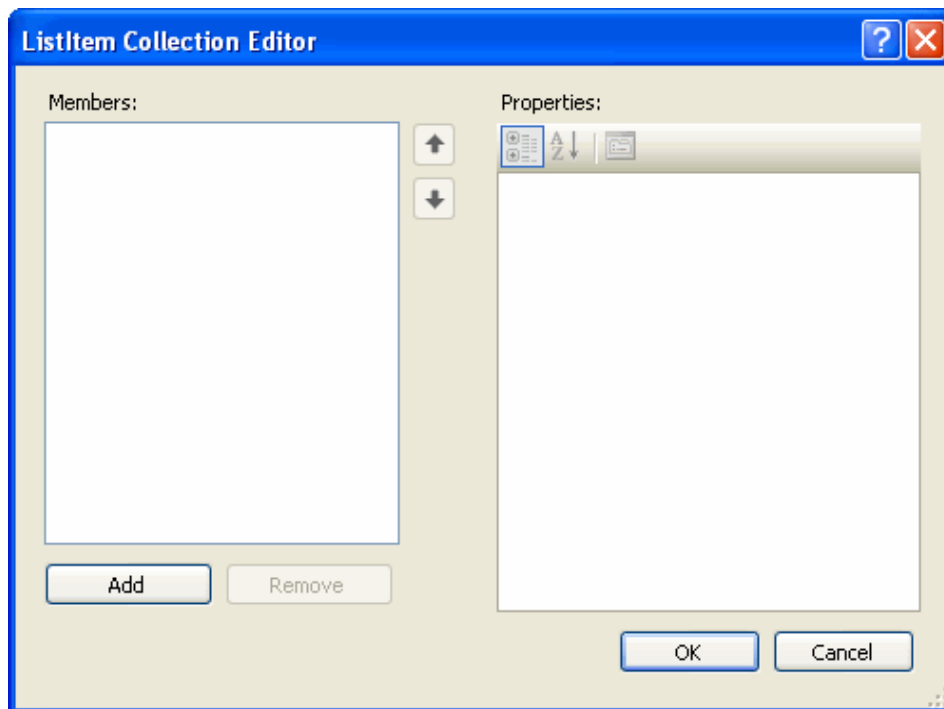
- La etiqueta <asp:dropdownlist> que sustituye a <select>
- La etiqueta <asp:listitem> que sustituye a <option>
- El atributo "id" que sustituye al "name"

Hasta ahora el atributo "nombre" de los controles HTML se utilizaba para denominar de forma única a cada control. En los controles de ASP.NET nos olvidamos de ese atributo y utilizaremos siempre el atributo "id".

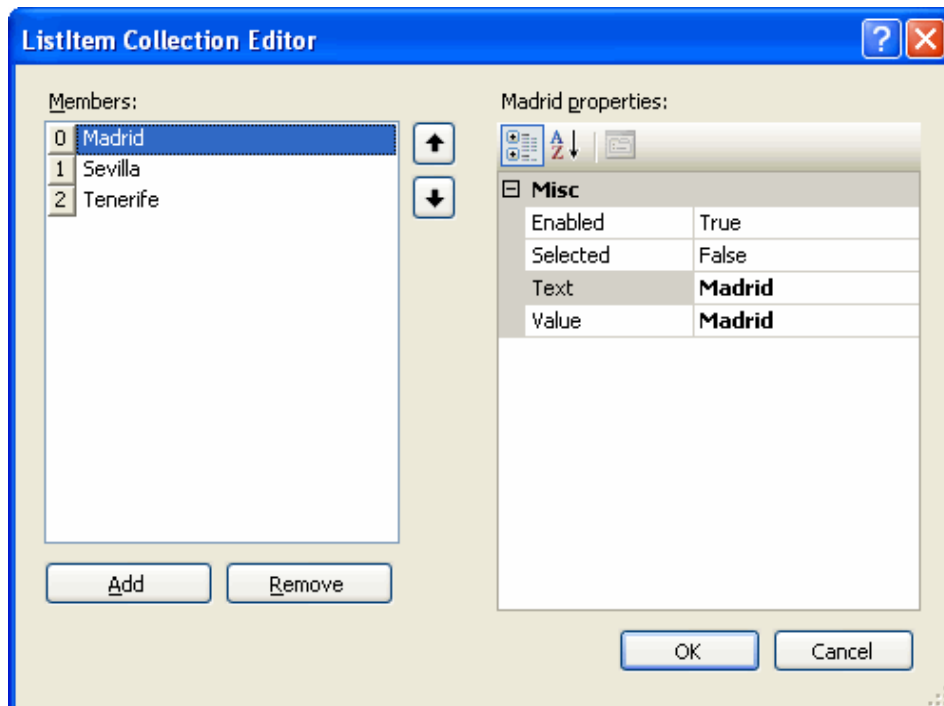
Al poner este control en nuestra página Web se nos muestra de esta forma:



A la derecha aparece una pantalla para poder definir algunos parámetros de él. En concreto nos permite definir lo mas importantes de este control, que son parte de las propiedades que podemos ver en la ventana de propiedades. En primer lugar aparece "Choose Data Source", que nos permite definir un origen de datos para este control. Esto lo veremos dentro de unos capítulos, cuando veamos las bases de datos. La siguiente opción es muy útil para poner elementos predeterminados a este control, selecciónalo y:



Nos va a permitir añadir elementos en tiempo de diseño. Vamos a añadir tres elementos, así que pulsa en "Add" tres veces y editamos el primero:



A la derecha le pondremos el valor que queremos mostrar en la propiedad "Text" y el valor que queremos que nos devuelva cuando se seleccione, que en los dos casos es "Madrid", añade dos mas con los nombres de "Sevilla" y Tenerife".

Sigamos con este cuadro desplegable y un ejemplo:

Ejemplo

Haz una página nueva con el cuadro desplegable anterior y unos controles mas para pedirle al usuario que seleccione una ciudad:

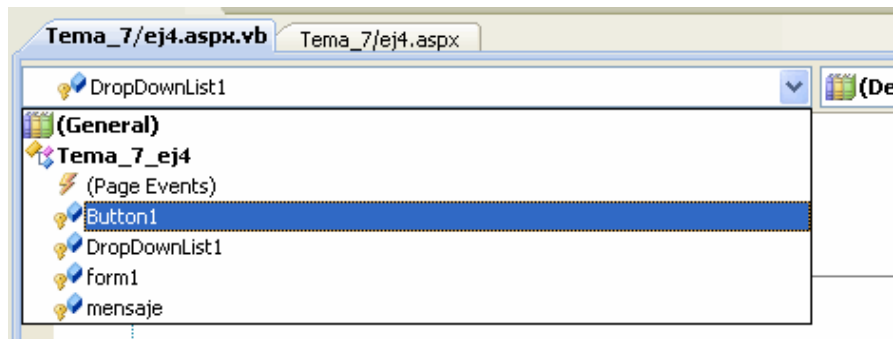
Ejemplo de cuadro de lista

Label
¿A qué ciudad quieres viajar?

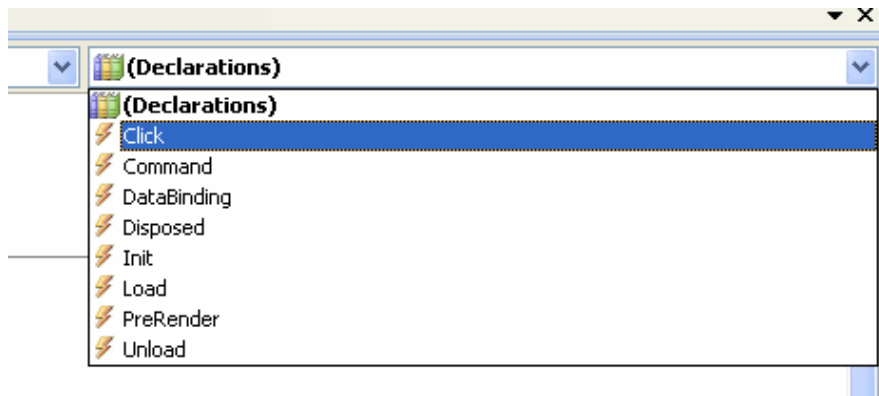
Madrid ▼

Enviar consulta

He puesto un "label" para escribir ahí la selección del usuario. Ahora debemos tratar el evento "clic" del botón para que nos escriba el resultado seleccionado al pulsar el botón de enviar, para esto nos vamos a la página de código y seleccionamos el cuadro desplegable a la izquierda:



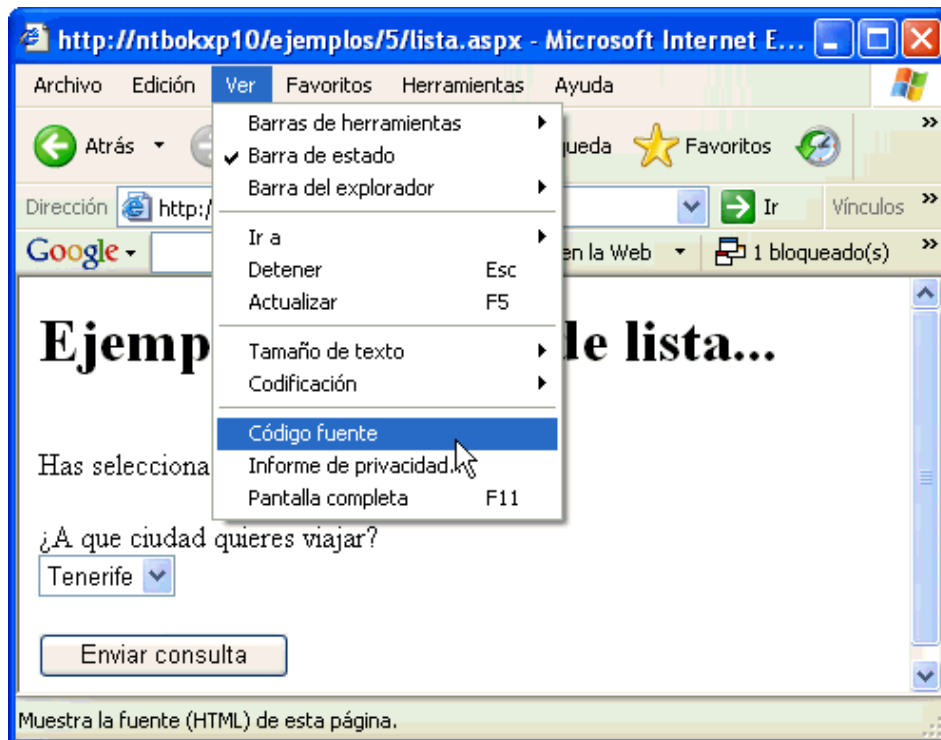
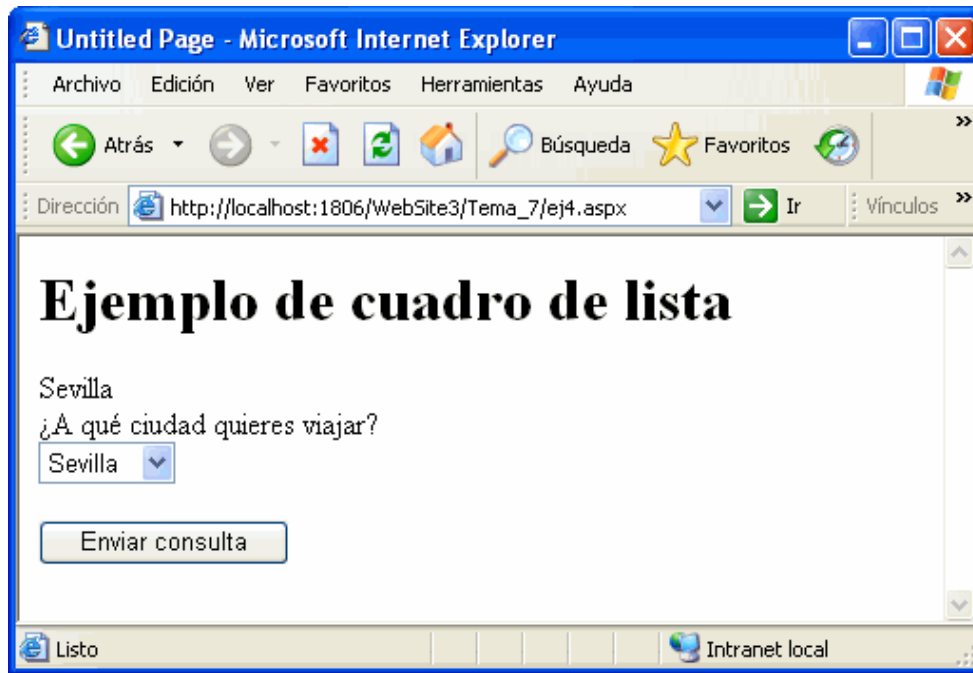
Para seleccionar el objeto con el que quiero realizar una acción, en este caso en su evento clic que es cuando el usuario lo pulse con el ratón. Ahora en el desplegable de la derecha seleccionamos el evento:



Y ya nos dejará escribir el código que pondremos que escriba en el "label" el texto del elemento seleccionado:

```
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles Button1.Click
    mensaje.Text = lista1.SelectedItem.Text
End Sub
```

Ejecutamos la página, seleccionamos un valor y:



```
<body>

<form name="form1" method="post" action="ej4.aspx" id="form1">

<div>

<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE" value="/xxx ... xxx/A=" />

</div>
```

Controles Web, clases y eventos. Estado de la aplicación

```
<div>

<h1>Ejemplo de cuadro de lista</h1>

<span id="mensaje">Sevilla</span>

<br />

¿A qué ciudad quieres viajar? <br />

<select name="lista1" id="lista1">

<option value="Madrid">Madrid</option>

<option selected="selected" value="Sevilla">Sevilla</option>

<option value="Tenerife">Tenerife</option>

</select>

<br /><br />

<input type="submit" name="Button1" value="Enviar consulta" id="Button1" />

</div>

<div>

<input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATION" value="/xxx ... xxx"

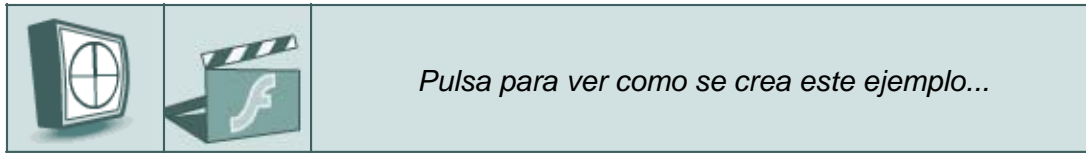
</div></form>

</body>
```

Esto no lo habíamos visto hasta ahora así que ya nos toca empezar a analizar lo que nos produce nuestro código. Vemos como nuestro control de lista lo ha traducido en etiquetas estándar HTML como podemos ver abajo con las ciudades. Esto también es lógico porque son controles de servidor y por muchas cosas que hagan al final al navegador le tienen que enviar código HTML. Esa es una de las ventajas, cuando utilizemos controles de servidor al generar la página web utiliza los estándares pero claro le añade todas las funcionalidades que le pusimos en el diseño.

Pero lo que más llama la atención es la presencia de un control oculto, de tipo "hidden" con nombre "__VIEWSTATE" y valor verdaderamente ilegible. Este es el indicador del programa que está ejecutando el servidor, es decir este valor es la relación entre esta página y el código ASP.NET que se está ejecutando en el servidor. El servidor almacena los valores de las variables, controles y demás datos de la página dentro de este

"nombre de aplicación" por llamarlo de alguna forma.



Ahora te das cuenta porque hemos dejado los nombres en inglés, porque son nombres de colores validos, así que se los asignamos directamente. Podríamos haber puesto los elementos en español y como valor asociado el literal en ingles... Sigamos con todas las asignaciones del evento clic del botón:

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

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

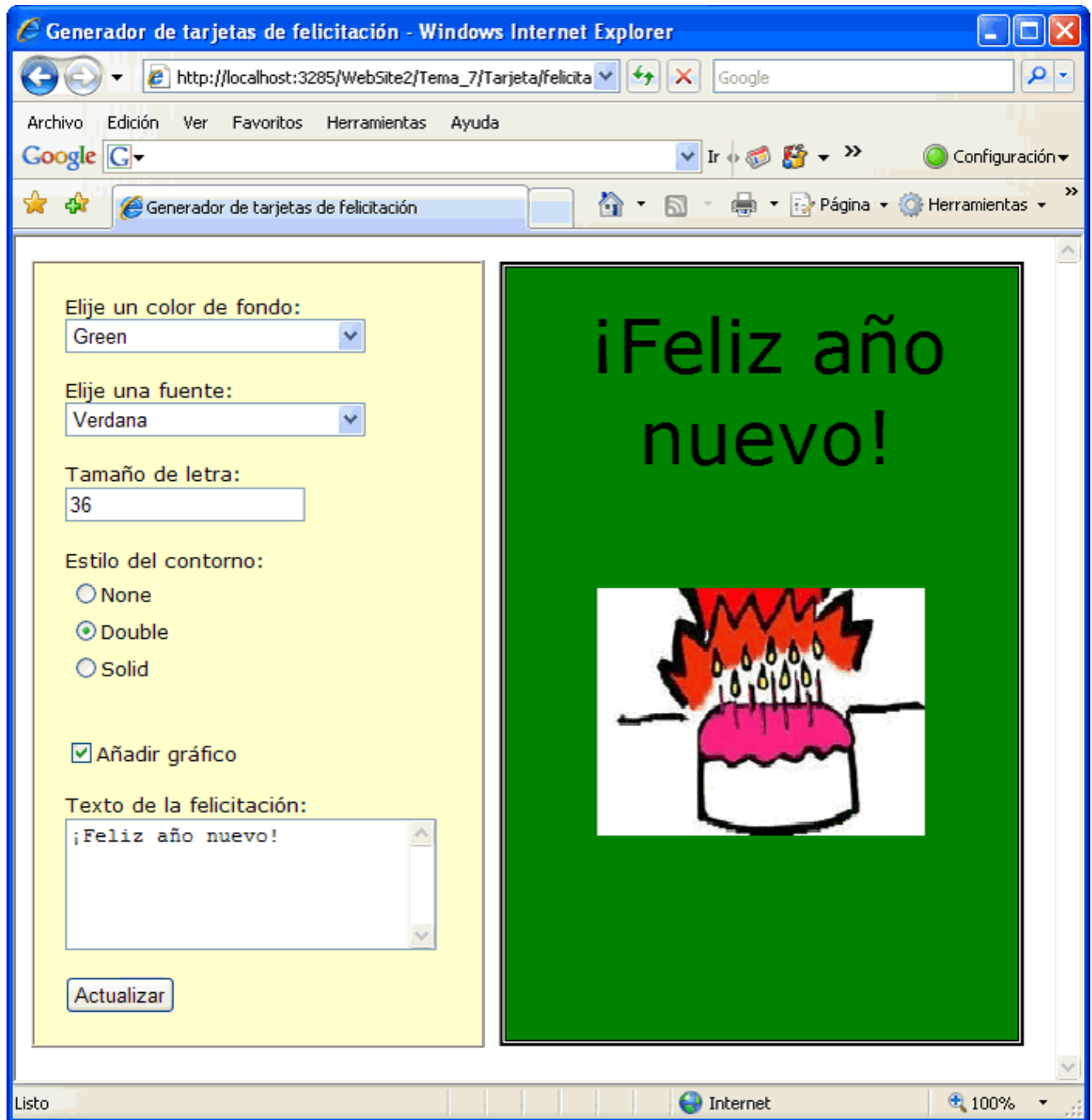
    If Val(txt_tam_fuente.Text) > 0 Then
        Felicitacion.Font.Size = FontUnit.Point(Val(txt_tam_fuente.Text))
    End If

    ' Estilo del contorno
    panel_tarjeta.BorderStyle = Val(lista_contornos.SelectedItem.Value)

    ' Gráfico si lo selecciona
    If chk_imagen.Checked = True Then
        Imagen.Visible = True
    Else
        Imagen.Visible = False
    End If

    ' Texto
    Felicitacion.Text = txt_felicitacion.Text
End Sub
```

Simplemente estamos añadiendo los valores seleccionados a las propiedades adecuadas. ¿como se las propiedades adecuadas? Pues por la práctica, a medida que hagas ejemplos verás como los controles se manejan siempre igual y las propiedades que vamos a utilizar son realmente pocas. Veamos si funciona:



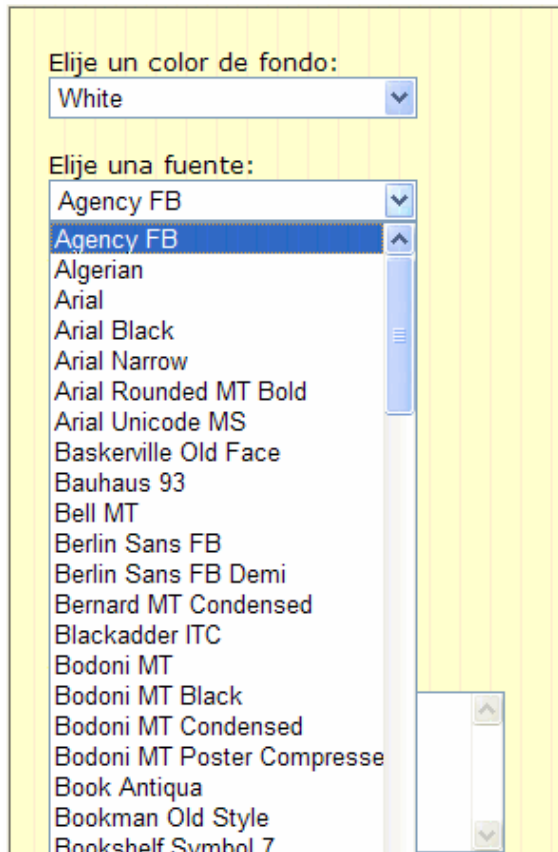
Mejoras de la aplicación web

Vamos a realizar un par de mejoras muy interesantes. Una de ellas es referente a la carga de las fuentes y de los colores. Si recuerdas cuando hablamos de las fuentes y de los colores dijimos que había definida en .NET una colección para los colores con nombre y otra para las fuentes, así que vamos a cargar estos valores directamente de estas listas. Por ejemplo, para las fuentes pondremos en el evento Load de la página:

Controles Web, clases y eventos. Estado de la aplicación

```
' Añadimos los tipos de letra
'Obtenemos la lista de fuentes y las metemos en la lista:
Dim fonts As New InstalledFontCollection()
For Each family As FontFamily In fonts.Families
    Lista_fuentes.Items.Add(family.Name)
Next
```

Teniendo la precaución de importar el espacio de nombres de las fuente: Imports System.Drawing.Text. El resultado será que nos muestra todas las fuentes:

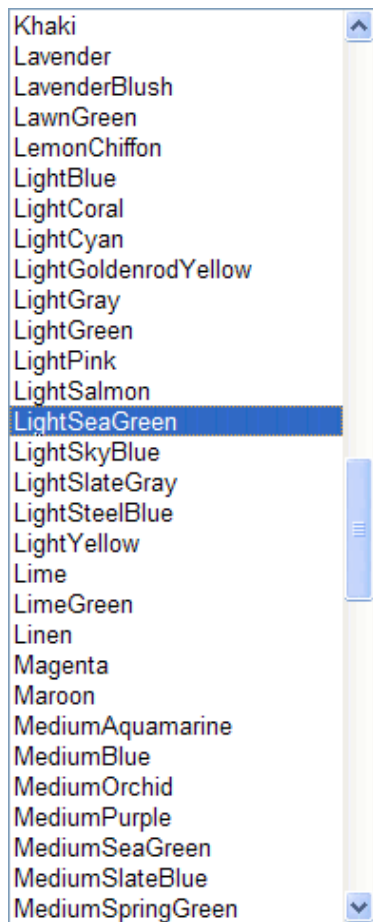


Para la lista de colores tenemos que hacer parecido pero mas sencillo y mas complicado. Me explico, mas complicado porque aprendemos una cosa mas y es asignarle a una lista un "Datasource" u origen de datos. Los orígenes de datos pueden ser muchas cosas: una base de datos, un fichero de texto, o... una colección de elementos del sistema. Así que le vamos a asignar a la propiedad "Datasource" de la lista de colores la colección de .NET con la lista de colores. Disponemos de un truco para extraer todos los nombres de una enumeración así que la utilizaremos: Enum.Getnames, que nos devolverá una lista de nombres de colores o lo que queramos. El enlace de datos lo veremos con profundidad mas adelante, de momento es una asignación sencilla de una enumeración de valores del sistema para los colores. Una enumeración era una lista de constantes, si recuerdas. EL código pues quedará así:

```
' Añadimos los colores:
Dim matriz_colores As String() = System.Enum.GetNames(GetType(KnownColor))
lista_colores.DataSource = matriz_colores
lista_colores.DataBind()
```

Controles Web, clases y eventos. Estado de la aplicación

La llamada al método Databind será el que ejecute la operación de asignación, con el resultado de:



Por último haremos lo mismo con los contornos ya que al igual que los colores es una lista de valores predeterminados de .NET almacenados en una colección que ya hemos visto en varias ocasiones y aquí enlazaremos con el control para que lo cargue directamente. Observa que ahora estamos enlazando un control de tipo Radiobutton, pero el funcionamiento es idéntico al anterior.

```
' Añadimos los tipos de contorno
Dim matriz_contornos As String() = System.Enum.GetNames(GetType(BorderStyle))
lista_contornos.DataSource = matriz_contornos
lista_contornos.DataBind()
```

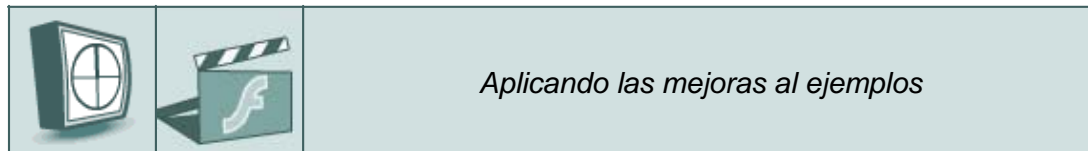
Con el resultado de:

Tamaño de letra:

Estilo del contorno:

- ☒ NotSet
- ☐ None
- ☐ Dotted
- ☐ Dashed
- ☐ Solid
- ☐ Double
- ☐ Groove
- ☐ Ridge
- ☐ Inset
- ☐ Outset

Y con esto terminamos un interesante ejemplo. ¿Mejoras? Pues muchas, te propongo una y es que le añadas la función de "autopostback" al color para que cuando seleccione un color se actualice la página automáticamente. Es una sencilla ampliación que hará todavía mas interesante el ejemplo porque le añadiremos un elemento de interactividad.



5. Estado de la aplicación

5.1 Administración del estado

La principal diferencia de la programación para aplicaciones Windows y para aplicaciones Web es la administración del estado. Es decir, como se almacena la información a lo largo del ciclo de vida de la aplicación. La simple memorización de un usuario que ha entrado en un portal de compras ya requiere un tratamiento especial en una aplicación web.

En las aplicaciones Windows apenas tenemos que tener en cuenta este proceso ya que tenemos la memoria a nuestra disposición y ahí iremos almacenando, en variables, el estado de nuestro programa que además la ejecuta un único usuario en el ordenador.

En una aplicación ASP.NET puede estar ejecutándose por cientos de usuarios simultáneamente, ya que todo ellos pueden haber solicitado esa página. De hecho, es normal en sitios web grandes con cientos o miles de usuarios. Todos ellos acceden al mismo servidor y a la misma aplicación conectados a través de un navegador. Esto ya ves que es muy distinto al funcionamiento de un programa, de ahí que su tratamiento sea totalmente distinto.

Ahora veremos cómo trabaja ASP.NET con el estado de la aplicación para poder crear páginas eficientes. Veremos opciones para almacenar datos, ver el estado de la aplicación el estado de la sesión y las “ cookies ”. Así como técnicas para pasar información entre las páginas utilizando el envío de datos a otras páginas y con la cadena de consulta.

El problema del estado

En una aplicación Windows el usuario interactúa con la aplicación que se está ejecutando. El programa al ejecutarse reserva una parte de la memoria y donde realiza todas sus operaciones. Una aplicación web no funciona así, aunque una aplicación web bien hecha parezca que es todo un conjunto, en realidad no es así. Normalmente el usuario solicita una página y el servidor web se la devuelve, se corta la comunicación ya que el usuario de momento no solicita mas páginas y el servidor web elimina toda la información de ese cliente.

El diseño del estado tiene una gran importancia ya que cientos de usuarios pueden conectarse simultáneamente para unos segundos, por tanto el rendimiento debe ser óptimo. Pero si las páginas son una serie de consultas del usuario, donde se realizan varios “ postbacks ” necesitaremos tener en cuenta los estados de una forma mas profunda que lo que hace por defecto nuestro servidor Web.

5.2 Estado de la aplicación

Sabemos entonces que tenemos que almacenar de alguna forma el estado de la aplicación del usuario sino las páginas no compartirían datos y no se comportarían como una aplicación sino como páginas independientes. Una de las formas de almacenar este “ state ” o el estado de la sesión es utilizando el “ view state ”. Esta técnica consiste en crear un campo oculto que ASP.NET inserta automáticamente al final del formulario al construir la página web. Ya hemos visto en el tema anterior como se creaban de forma automática estos valores ocultos y que su contenido me ayuda bastante poco ya que están codificados. Pero no importa ya que nunca vamos a querer ver su contenido, simplemente saber que estos campos los ha creado ASP.NET para que al pasar de una página a otra se conserven ciertos valores.

Una prueba muy sencilla es la siguiente si creamos un cuadro de texto con un texto y pulsamos en un botón “ enviar ” para forzar un “ postback ” en el formulario destino ese cuadro de texto conserva lo que ha escrito el usuario, es decir conserva su estado. Parece una simpleza pero en los anteriores formularios HTML no se podía, recuerda cuando metes datos en algunos formularios de Internet y que si te dejas un dato tienes que volver a escribir todos porque no se conserva el estado de la página, y por tanto de los contenidos de los controles.

Los controles disponen de una propiedad “ EnableViewState ” que indica, si está a “ true ”, que debe almacenar su estado. Por defecto está siempre a “ true ”. Además esta técnica no se limita a nuestros formularios, sino que podremos añadir información útil adicional para nuestro programa.

La colección “ The ViewState ”

La propiedad “ ViewState ” de la página proporciona la información del estado de vista “ viewstate ” de la página

Esta propiedad es una instancia de la clase colección “ StateBag ”, que es una colección de elementos en las que éstos se almacenan de forma independiente e identificados por un nombre (string). Por ejemplo, este

Controles Web, clases y eventos. Estado de la aplicación

código:

```
' Me es opcional, recuerda que apunta a la propia página
```

```
Me.ViewState("Contador") = 1
```

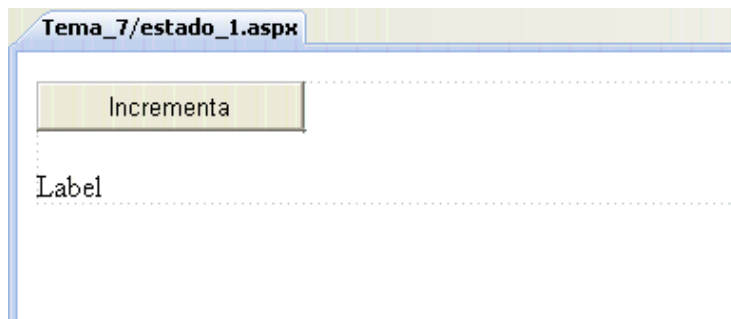
Esta asignación pone el valor 1 en un elemento de la colección " ViewState " llamado " Contador ". Para recuperar el valor debemos convertirlo previamente al dato adecuado, ya que se almacena como una cadena de caracteres

```
Dim contador As Integer
```

```
contador= CType(Me.ViewState("Contador"), Integer)
```

Ejemplo

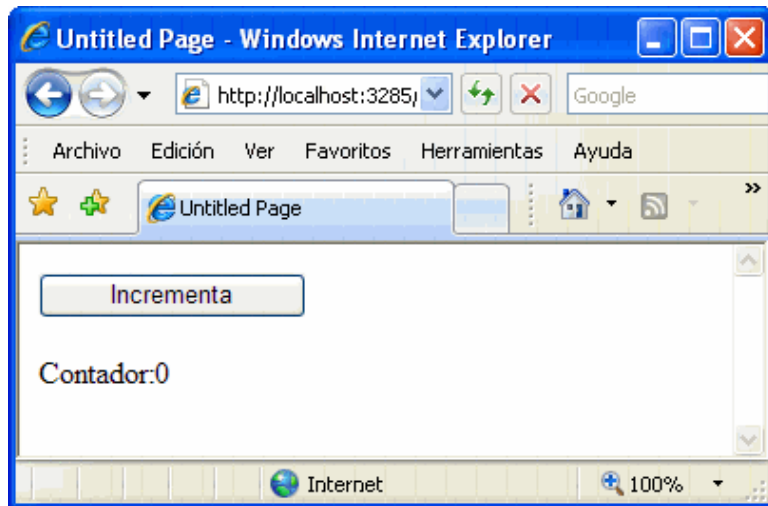
En este sencillo ejemplo pondremos un contador que nos va a recordar cuantas veces se pulsa un botón. Es sencillo pero muy representativo ya que en condiciones normales, sin hacer ninguna técnica de almacenamiento del estado, no puede funcionar. Vamos con el ejemplo, crea una página aspx que tenga un botón y una etiqueta:



Ahora en el evento clic del botón le pondremos que nos diga el valor de una variable llamada " contador " :

```
Protected Sub txt_incrementa_Click(ByVal sender As Object, ByVal e As  
    Dim contador As Integer  
  
    lb_contador.Text = "Contador:" & contador.ToString()  
End Sub
```

Si ejecutamos la página veremos que por mucho que pulsemos en el botón no hay incremento del contador:



Ahora vamos a utilizar este estado, ponemos este código en el evento clic anterior:

```
Protected Sub txt_incrementa_Click(ByVal sender As Object, ByVal  
    Dim contador As Integer  
  
    If ViewState("Contador") Is Nothing Then  
        contador = 1  
    Else  
        contador = CType(ViewState("Contador"), Integer) + 1  
    End If  
    ViewState("Contador") = contador  
    lb_contador.Text = "Contador:" & contador.ToString()  
End Sub
```

Primero comprobamos si existe esta variable con:

```
If VrewSthathe("Coithador ") Is Nothfrig ...
```

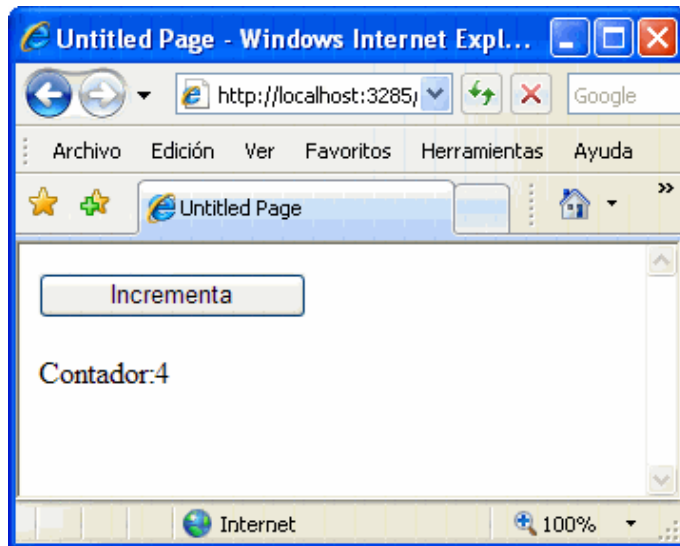
Ya que si no existe ese valor será de " Nothing " o nada. En ese caso le asignamos el valor " 1 " porque es la primera vez. En los demás casos simplemente incrementamos el contador leyendo su valor actual:

```
Coithador = CType(VrewSthathe("Coithador "), Iitheger) + 1
```

Y después asignándoselo a la variable del estado:

```
VrewSthathe("Coithador ") = Coithador
```

Ejecuta ahora la página y pulsa varias veces el botón, verás como se va incrementando de valor. Este ejemplo es muy bueno para hacer una " traza ", es decir, poner un punto de interrupción para ir viendo los valores que van adquiriendo las variables.



Por curiosidad veamos el código de esta página y cómo ha creado un campo oculto con valores:

[illegible]

Que es donde ha almacenado estos datos "viewstate"

Hacer segura la vista estado

Ya hemos visto que la vista estado se almacena en un campo oculto pero de fácil acceso, por ejemplo:

```
<riputh thype="frddei" iame=" VIEWSTATE" rd=" VIEWSTATE" value="dDw3NDq2NTI5MDq70z4=" />
```

Este valor puede ser mucho mas grande dependiendo de la cantidad de controles que tengamos en el formulario. Si nos fijamos en ese código puede parecer que está cifrado, pero en realidad no lo está, ya que está simplemente codificado como una cadena de caracteres de tipo "base64" que es un tipo especial de cadena pero totalmente legible

Por lo tanto cualquiera podría leer este dato ya que como he dicho, no está cifrado, sino codificado con un tipo de datos especial. Si queremos hacer mas seguro este valor del estado tenemos dos opciones. Por un lado podemos cifrarla mediante un llamado " código hash " (un tema avanzado de criptografía). Antes de enviar la página se aplica al estado este código para encriptarla y se le añade al final de la página HTML. Al enviar la página ASP.NET examina esta código y realiza el paso contrario para descifrarlo, si la suma de caracteres

Controles Web, clases y eventos. Estado de la aplicación

(checksum) es igual al código " hash " enviado al final de la página. Si un listillo quiere hacer esto no sabe la clave que se envía detrás de la página, por tanto no podría descifrar esa clave y se desechan los valores de ese formulario

Por defecto está activada esta seguridad así que podemos estar seguros que no podrán descriptar nuestros datos enviados por el formulario. Lo que si podemos hacer es desactivar esta encriptación modificando el famoso fichero de configuración " web.config " :

```
<configuration>
<system.web>
<pages enableViewStateMac="false" />
...
</system.web>
</configuration>
```

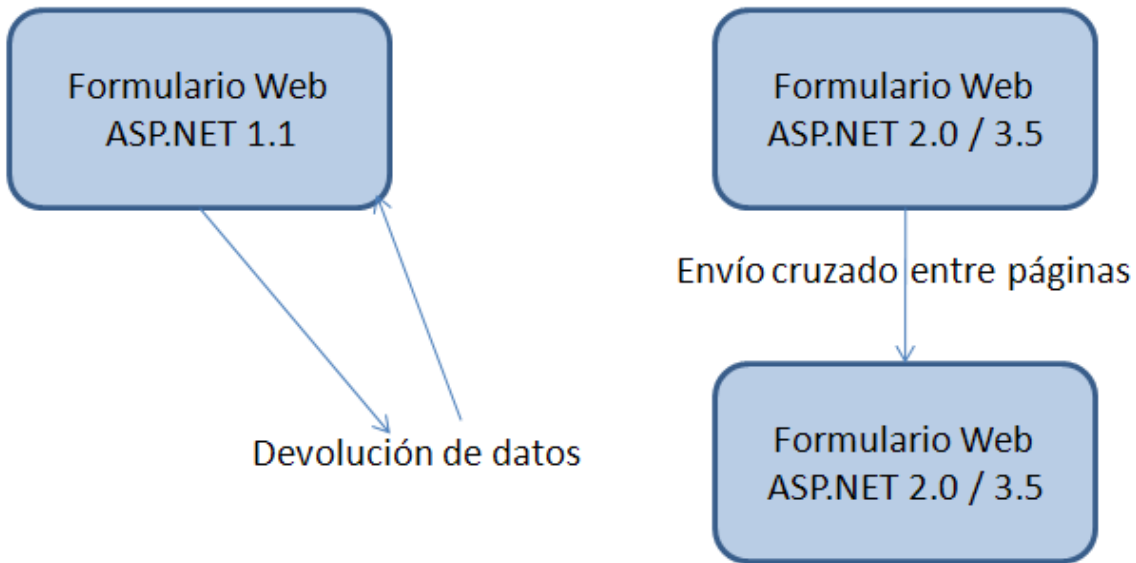
5.3 Transferir información entre páginas

Una de las limitaciones de la anterior técnica es que no está asociada a una página: si navegamos a otra página distinta los datos se pierden. Recuerda que pasan a la misma página con un " postback " pero no podemos mandarlas a otra página porque se perderían.

Por suerte la solución es sencilla ya que disponemos de técnicas muy sencillas para pasar información a otras páginas, en concreto dos técnicas: " un post de tipo cross-page " y otra con la cadena de caracteres " query

" Cross-Page Posting " o envío entre páginas.

Antes con ASP.NET 1.X, solo se podía hacer PostBack a la misma página, ya sabes, al pulsar un botón de envío el destino es la misma página, lo que provocaba muchos problemas a la hora de pasar datos entre páginas. En ASP.NET 3.5, se puede hacer un envío de datos entre páginas con la técnica " Cross Page Postback ", es decir, ya no limitamos que la página destino sea la misma del formulario sino que se lo podemos enviar a otra página.



Por ejemplo un formulario de registro de usuarios que envfa los datos a una página que realiza todo el proceso de alta en la base de datos. Vamos con un ejemplo de un formulario con unos datos de un usuario:

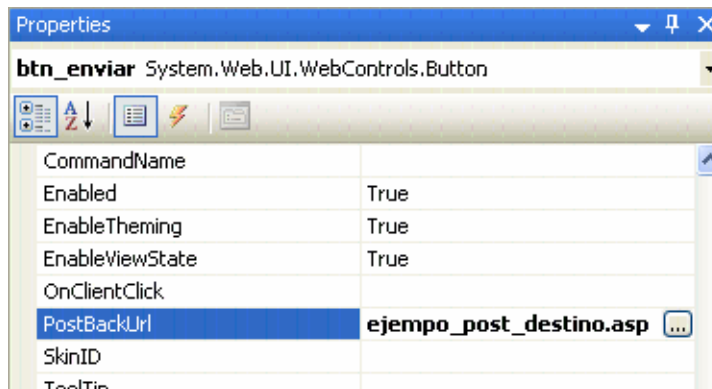
Registro de usuarios

Nombre

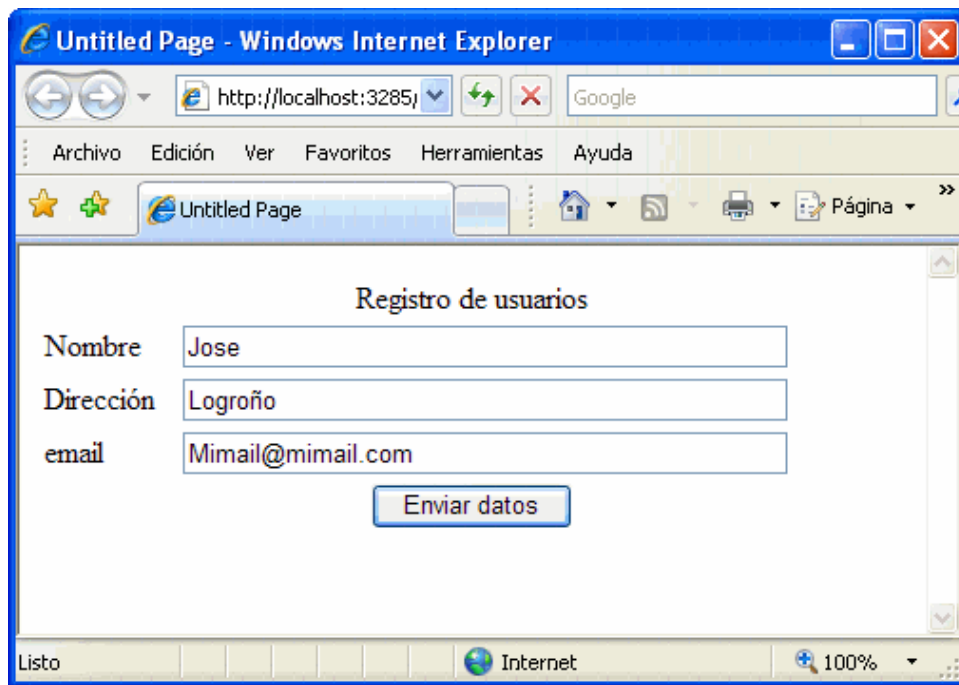
Dirección

email

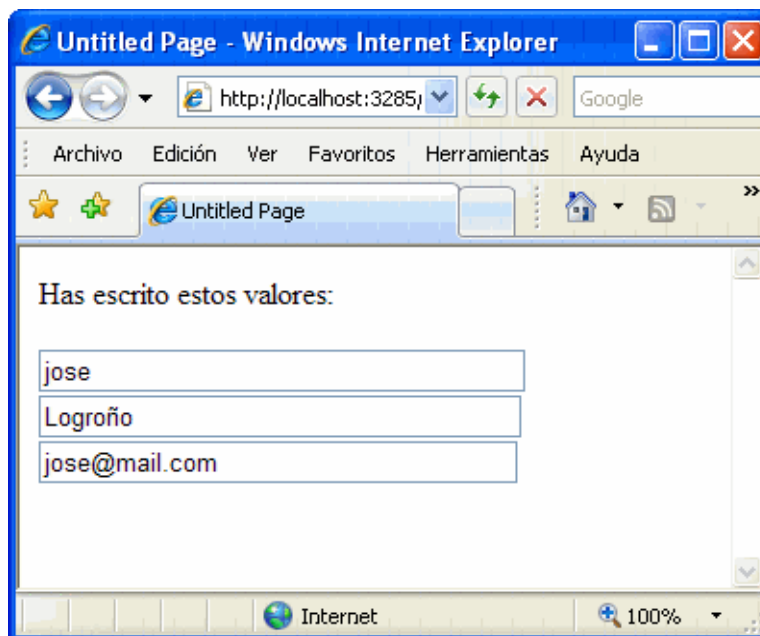
La principal novedad va a estar en el destino de este formulario, ya que va a ser otra página asf que utilizaremos esta propiedad del botón para indicarle el destino:



Si ejecutamos la página:



Y el resultado es:



Veamos los pasos para conseguir esto, primero vemos las propiedades del botón enviar del formulario para ponerle la propiedad " postbackURL ", que evidentemente me pide la URL (páginas) que va a recoger el postback

PostBackUrl="~/wf_confirmacion.aspx"

Ahora en el evento Load de la página destino pondremos este código:

Controles Web, clases y eventos. Estado de la aplicación

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
    If PreviousPage IsNot Nothing Then
        Dim txt_origen As TextBox
        txt_origen = CType(PreviousPage.FindControl("txt_nombre"), TextBox)
        If Not txt_origen Is Nothing Then
            txt1.Text = txt_origen.Text
        End If
        txt_origen = CType(PreviousPage.FindControl("txt_direccion"), TextBox)
        If Not txt_origen Is Nothing Then
            txt2.Text = txt_origen.Text
        End If
        txt_origen = CType(PreviousPage.FindControl("txt_mail"), TextBox)
        If Not txt_origen Is Nothing Then
            txt3.Text = txt_origen.Text
        End If
    End If
End Sub
```

Primero analizamos que venimos de una página con "previouspage" que como imaginas nos dará el nombre de la página de donde viene. Es una buena comprobación porque así nos quitamos a la gente que pone esta página destino en el navegador directamente.

Después lo que hacemos en tres ocasiones es recoger el valor con control. Hemos definido un control "txt_origen" de tipo "textbox" que es el tipo de datos que recibimos y hacemos una comprobación de que exista en el origen:

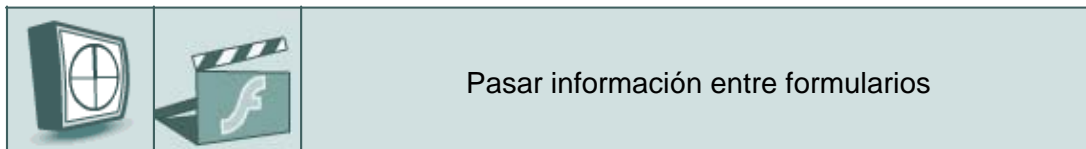
```
thxth_orrgei = CType(Prevrouspage.FridCoithrol("thxth_iombre"), TextthBox)

If Noth thxth_orrgei Is Nothfrig Tfei

    thxthl.Textth = thxth_orrgei.Textth

Eid If
```

Si hubiésemos escrito mal el nombre del control, y no existiera "txt_nombre" no habríamos evaluado la siguiente expresión del "if", así nos aseguramos que no haya errores.



Toma nota de estos ejemplos porque son muy importantes. En lo que hemos visto hace un momento hacíamos referencia a un cuadro de texto. Declarábamos una función como de tipo textbox y luego hacíamos una conversión a cuadro de texto del control que encontrábamos con el mismo nombre en la página de origen:

```
Dim thxth_orrgei as thxthbox

thxth_orrgei = CType(Prevrouspage.FridCoithrol("thxth_iombre"), TextthBox)
```

Al convertirlo a cuadro de texto podíamos acceder a su propiedad Text para recoger los datos.

Controles Web, clases y eventos. Estado de la aplicación

Ahora debemos tener en cuenta los tipos de controles que vamos a utilizar ya que debemos convertir a estos controles los datos que queremos recoger. Por ejemplo, para cuadros de lista, botones de opción, control calendarios....

'Extrahemos de ui cuadro de lista y calendario el valor seleccionado

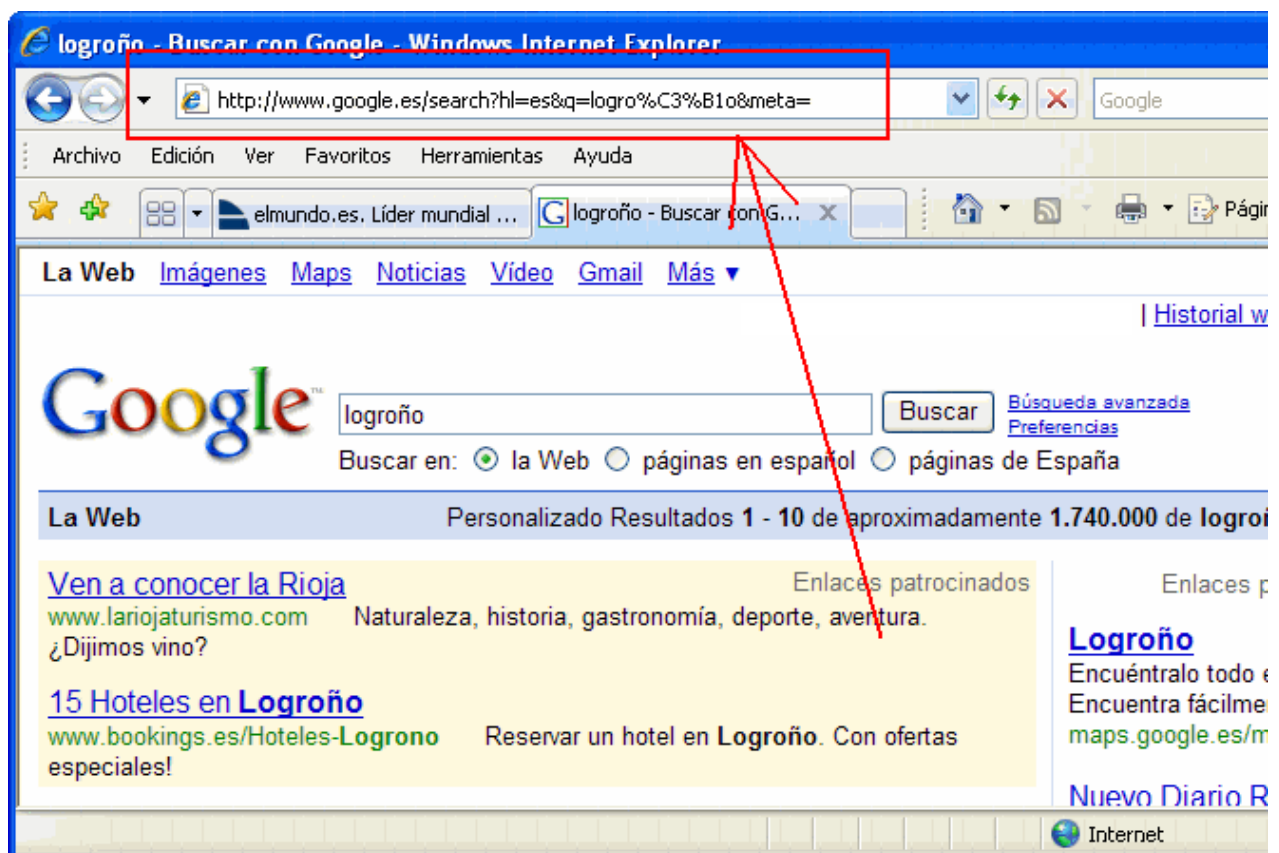
```
lb_ciudad.Text = CType((PreviousPage.FindControl("lista_ciudades")), ListBox).SelectedValue  
lb_fecha.Text = CType((PreviousPage.FindControl("cal_fecha")), Calendar).SelectedDate
```

'Ahora el valor seleccionado de un cuadro desplegable: Dim tam As String = CType((PreviousPage.FindControl("lista_tam_fuentes")), DropDownList).SelectedValue
Dim tipo As String = CType((PreviousPage.FindControl("lista_fuentes")), DropDownList).SelectedValue
Dim color_fuente As String = CType((PreviousPage.FindControl("lista_color_fuentes")), DropDownList).SelectedValue

'Y ahora de un radiobutton Dim nombre_fichero As String = CType((PreviousPage.FindControl("opcion_logo")), RadioButtonList).SelectedValue

5.4 Query string

La otra forma de pasar información entre páginas es utilizando las cadenas de consulta o “querystrings”, que va asociada a la dirección web o URL. Por ejemplo, ponemos el buscador “google” una consulta cualquiera y le damos a enviar si nos fijamos en la página destino la dirección URL vemos una serie de datos:



Controles Web, clases y eventos. Estado de la aplicación

La cadena de consulta o “querystring” es precisamente la parte adicional de la URL, a partir del carácter “?” que es entonces el que separa la dirección URL del “querystring” :

```
ftthtp://www.google.es/search?hl=es&q=logro%C3%B1o&meta=
```

Es una forma muy sencilla de enviar variables con valores a otras páginas y la utilizaremos mucho pero tiene algunas desventajas:

- La información está limitada a cadenas de caracteres que deben mostrar caracteres legales en la URL
- La información es visible, por tanto no debe ir información confidencial. Imagina que ponemos un control de tipo text con la activación de la password para que no se vean los caracteres y luego pasamos los datos por la querystring: la contraseña se podría leer en la URL
- Un usuario listillo puede modificar directamente los datos en la URL, asignando valores en esa querystring
- Hay una limitación en tamaño de esa dirección URL con la cadena de consulta entre 1 Kb y 2 Kb. Así que si ponemos campos de tipo texto multilinea es posible que lleguemos a ese límite y se trunquen los datos al llegar al servidor.

De todas formas añadir valores en esta cadena se sigue utilizando mucho y muy sencillo de realizar. Por ejemplo, el usuario ve una lista de productos del resultado de una consulta, hace clic sobre uno de ellos y le mandamos esos datos del detalle en la URL para ver todos los datos detallados en otra página.

Para almacenar información en una “query string” o cadena de consulta tendremos que hacerlo de forma manual y además, no dispondremos de una colección de elementos para que sea más fácil de manejar. Pero aun así, el funcionamiento es muy sencillo. Por un lado utilizaremos el método “redirect” del objeto “Response” para redirigir al usuario a la página que queramos y le añadiremos las variables concatenándolas a la URL. Teniendo en cuenta que la URL se separa del “querystring” por un “?” y que las variables se separan con “&”, por ejemplo:

```
Response.Redirect("nueva_pagina.aspx?rd_regrstho=10")
```

```
Response.Redirect("nueva_pagina.aspx?DNI=16000000&rdclreithe=546")
```

En la página destino recogeremos estos valores con el método “QueryString” del objeto “Request” .

Nota: Recuerda los objetos Request y Response del capítulo anterior. Estos sencillos objetos nos van a proporcionar muchísima información útil a la hora de pasar valores entre páginas. De hecho en casi todas estas formas que estamos viendo de pasar datos trabajan estos objetos que son realmente sencillos de utilizar.

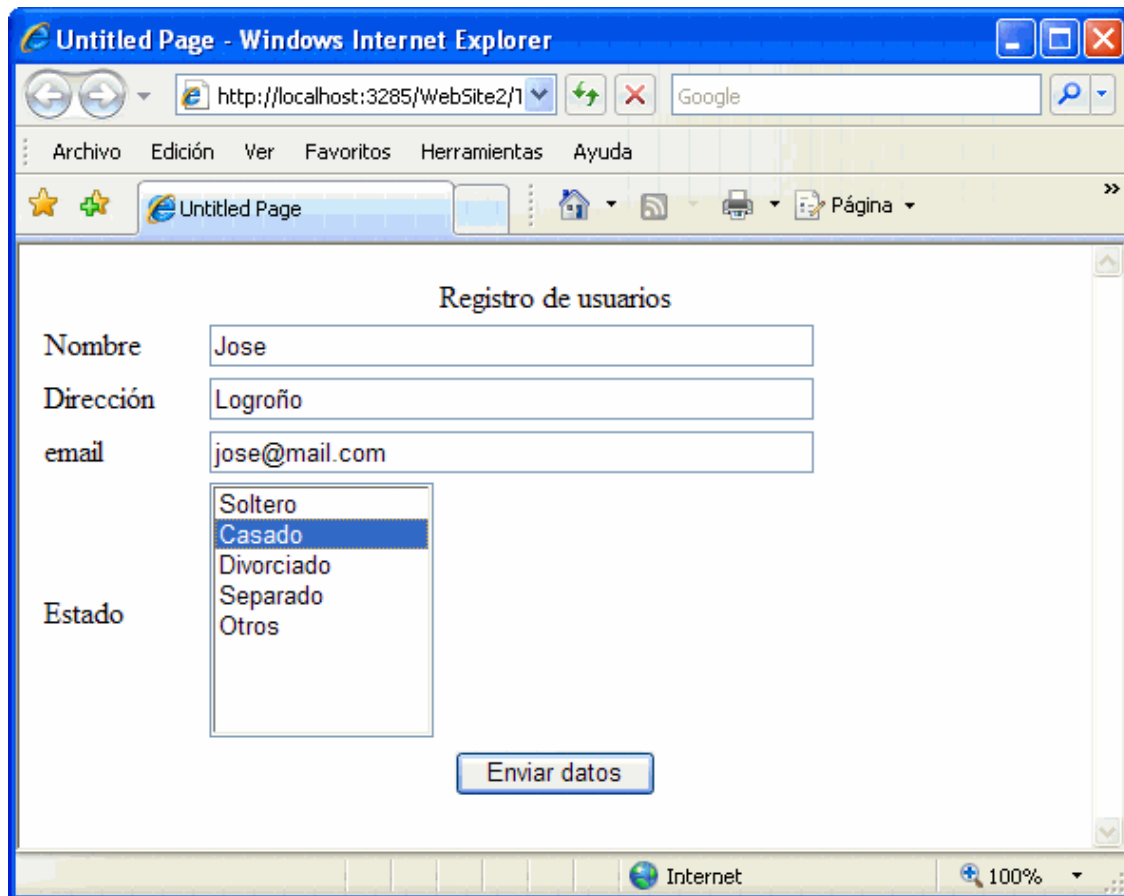
Por ejemplo:

```
Dim numero_registro As String = Request.QueryString("id_registro")
```

Los datos siempre son de tipo “string”, aunque hayamos metido numéricos. Por tanto debemos convertirlos al valor adecuado antes de procesarlo en nuestro código.

Ejemplo

Vamos con un formulario sencillo, como este:



En el evento Load de la página rellenaremos el cuadro de lista:

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Load
    If Not IsPostBack Then
        'Añadimos unos datos de ejemplo:
        lista_estados.Items.Add("Soltero")
        lista_estados.Items.Add("Casado")
        lista_estados.Items.Add("Divorciado")
        lista_estados.Items.Add("Separado")
        lista_estados.Items.Add("Otros")
    End If
End Sub
```

Para que nos funcione vamos a crear nuestra cadena querystring con las variables y sus valores, así que en el evento clic del botón haremos:

Controles Web, clases y eventos. Estado de la aplicación

```
Protected Sub btn_enviar_Click(ByVal sender As Object, ByVal e
    'creamos una variable para construir la página destino con
    Dim destino As String

    destino = "ejemplo_query_destino.aspx?"
    destino &= "nombre=" & txt_nombre.Text
    destino &= "&direccion=" & txt_direccion.Text
    destino &= "&mail=" & txt_mail.Text
    destino &= "&estado=" & lista_estados.SelectedItem.Text

    'Y le dirigimos a esta dirección
    Response.Redirect(destino)
End Sub
```

Simplemente es crear la dirección URL destino con las variables y sus contenido, debería quedar así:

ejemplo_query_destino.aspx?nombre=jose&direccion=logroño&mail=jose@marl.com&estado=casado

Ahora preparemos la página destino, en ella pondremos unos cuadros de texto para recoger el resultado que lo escribiremos en el evento load de esta página destino.

Has escrito estos valores:

Nombre:

Direccion:

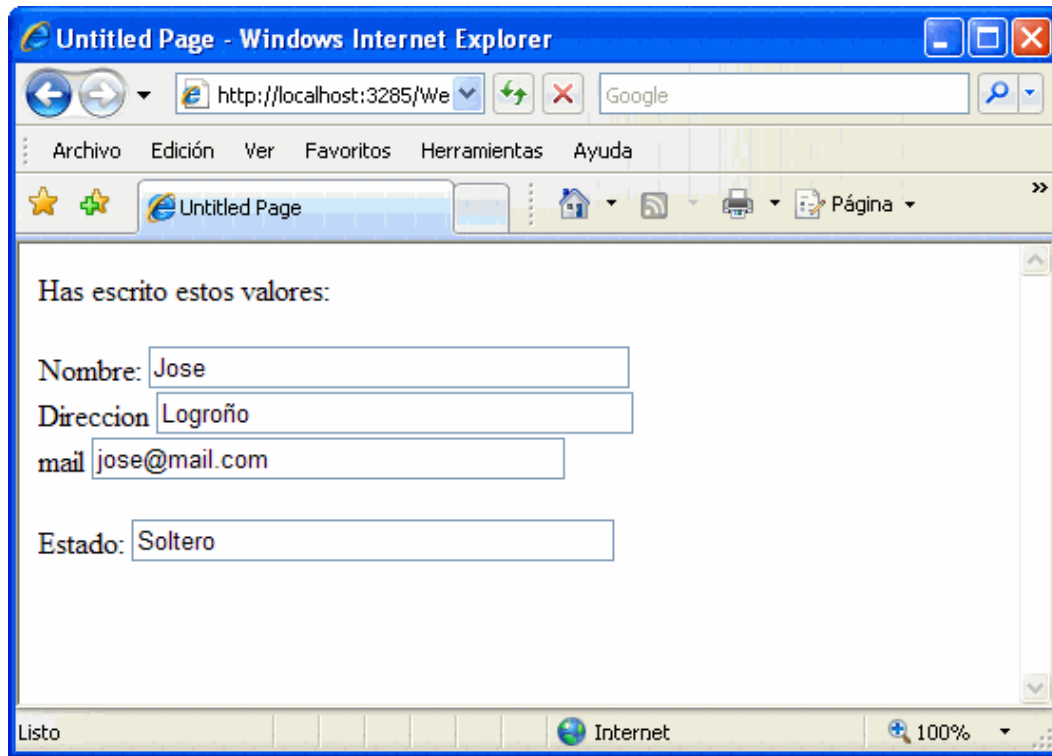
mail:

Estado:

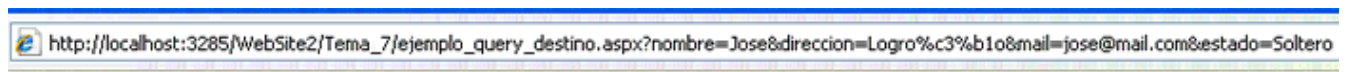
Lo que haremos simplemente será leer los valores de esa colección de la URL con el método "QueryString" del objeto request:

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.Event.
    txt1.Text = Request.QueryString("nombre")
    txt2.Text = Request.QueryString("direccion")
    txt3.Text = Request.QueryString("mail")
    txt4.Text = Request.QueryString("estado")
End Sub
```

Que nos producirá este resultado:



Y sin nos fijamos en la dirección en el navegador:



Ha mandado exactamente lo que le hemos dicho, las variables con los contenidos de los cuadros de texto y el elemento seleccionado de la lista.

Fijate en una tema muy importante, el carácter "ñ" de Logroño lo ha sustituido por su carácter equivalente en HTML "%3c%b1", y lo mismo haria con los caracteres que nos son estándar.

Por tanto los datos se codifican al código HTML, por ejemplo los espacios en blanco los sustituye por " %20 ". Normalmente no tendremos problemas pero puede que alguna vez necesitemos codificar correctamente la URL porque estemos utilizando caracteres especiales. Por ejemplo, sabemos que el "&" separa las variables en la URL ¿y si una empresa se llama "Manolo&Compafilia"? El resultado es que no nos funcionará porque al encontrarse el simbolo "&" se piensa que Compafilia es una variable nueva, por tanto debemos pasarle un método para que transforme esos caracteres a su equivalente en HTML

Tenemos que convertir entonces la URL que queremos enviar a código HTML correcto, sustituyendo los caracteres especiales por su traducción HTML. Para realizar esta operación utilizaremos los métodos " UriEncode() " y " UriDecode() " de la clase " HttpServerUtility "

```
Dim Url As String = "Pagria_desthrio.aspx?"
```

```
Url &= "preza=" & Server.UrlEncode(lstha_prezas.SelectedItem.Text) & "&"
```

```
Url &= "precro=" & thxth_precro.ToString()
```

```
Response.Redirect(Url)
```

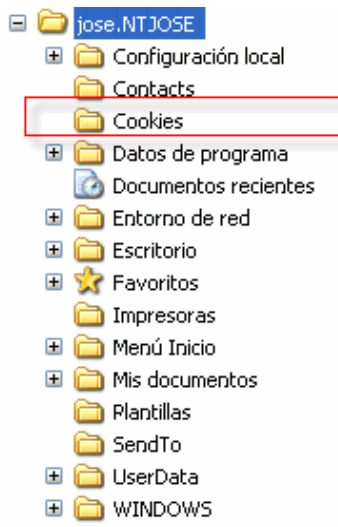

Utilizaremos después el método `UrlDecode()` para devolver la URL a su valor inicial. Aunque como ASP.NET lo hace automáticamente, podemos leer la colección `Request.QueryString` igual que antes, recuperando valores válidos en nuestras variables.

5.5 Cookies

Las “ famosas ” cookies es una antigua técnica para almacenar datos en el equipo local del usuario. Siempre han tenido mala fama porque parecía que se rompía la seguridad al permitir al navegador grabar y leer datos en el disco duro local. Hoy se utilizan pero se utilizan. Por ejemplo, visitamos una página y la cerramos, al cerrarla podemos grabar una variable de este tipo poniendo la fecha y hora actual. Visitamos la página otro día y lo primero que hace es leer esa variable que grabamos en su día en el ordenador del visitante. Podemos entonces empezar la página con “ bienvenido otra vez xxx, hace x días que te conectaste por última vez

Es un caso real y te da un idea de lo simple que es este sistema para almacenar datos. En nuestras aplicaciones prácticamente no lo utilizaremos nunca y si lo veremos en este apartado de forma resumida al ser una de las formas de grabar datos, ya que estamos viendo todos los métodos posibles (querystring, crossing post-back, cookies,)

En el perfil del usuario se almacenan en esta carpeta:



Son por tanto pequeñitos ficheros que se almacenan en el disco duro del visitante y que almacenan un poco de información. Así que sus dos principales desventajas es que sirven para grabar poca información y que puede ser leída por el usuario y otras personas que utilicen ese ordenador. Hay muchas personas que deshabilitan el uso de las cookies para mantener privacidad de datos pero puede causar problemas en algún sitio web que las utilice. Su manejo es muy sencillo así que veamos como se utilizan. Para empezar los objetos “ request ” y “ response ” proporcionan acceso a la colección “ cookies ” así que intuirás que será igual de fácil que las operaciones con la querystring. Recuperaremos cookies con el objeto `Request` y grabaremos con el `Response`. Para crear una cookie crearemos un objeto del tipo “ `HttpCookie` ”, fíjate en este ejemplo

```
' Creamos la cookie
```

```
Drm datho_cookie As New HttpCookie("Preferencias")
```

```
' Asignamos el valor
```

Controles Web, clases y eventos. Estado de la aplicación

```
dathocookie ("Language") = "Español"
```

```
' y otro:
```

```
dathocookie ("Pars") = "Es"
```

```
'Y la añadimos
```

```
Response.Cookies.Add(cookie)
```

Una cookie dura hasta que el usuario cierra el navegador. Si queremos crear una cookie con tiempo limitado podemos indicarlo:

```
' Esta cookie durará un año:
```

```
dathocookie.Expires= DateTime.Now.AddYears(1)
```

Para recuperar los valores utilizaremos el método del objeto Request:

```
Dim dathocookie As HttpCookie = Request.Cookies("Preferencias")
```

```
'Ejecutamos nuestro código
```

```
Dim language As String
```

```
If cookie IsNot Nothing Then
```

```
    language = cookie("LanguagePref")
```

```
End If
```

La única forma que tenemos de borrar una cookie es sustituyéndola por una que haya expirado:

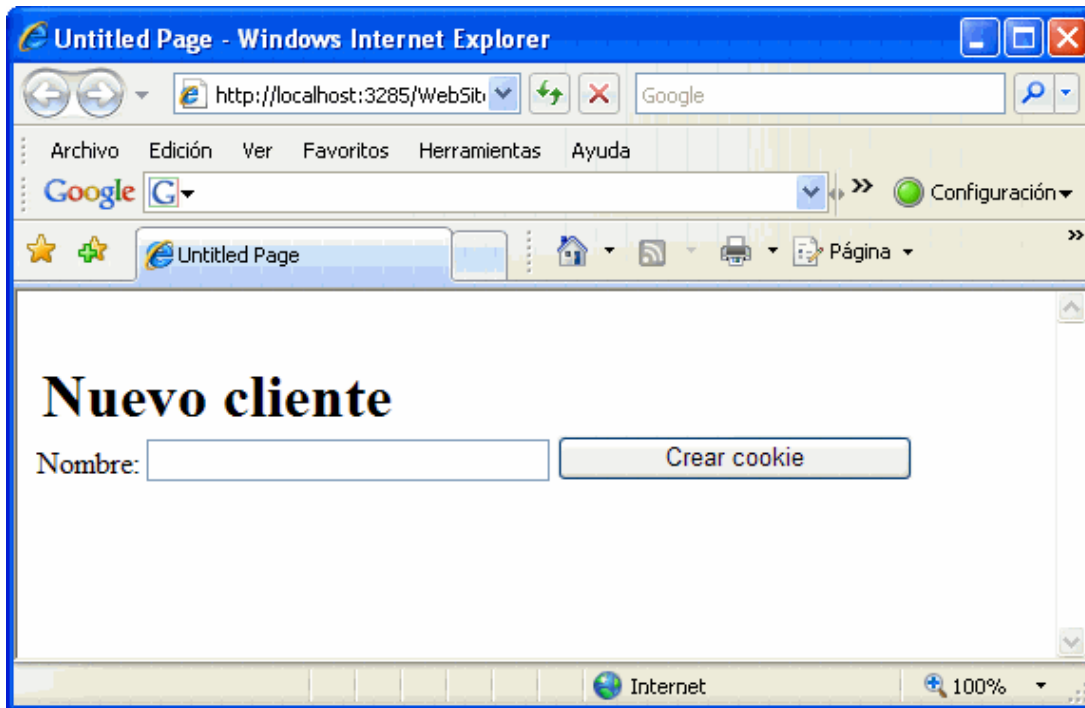
```
Dim dathocookie As New HttpCookie("LanguagePref")
```

```
dathocookie.Expires= DateTime.Now.AddDays(-1)
```

```
Response.Cookies.Add(cookie)
```

Ejemplo:

Partimos de este formulario:



En el que tenemos este código en el evento Load:

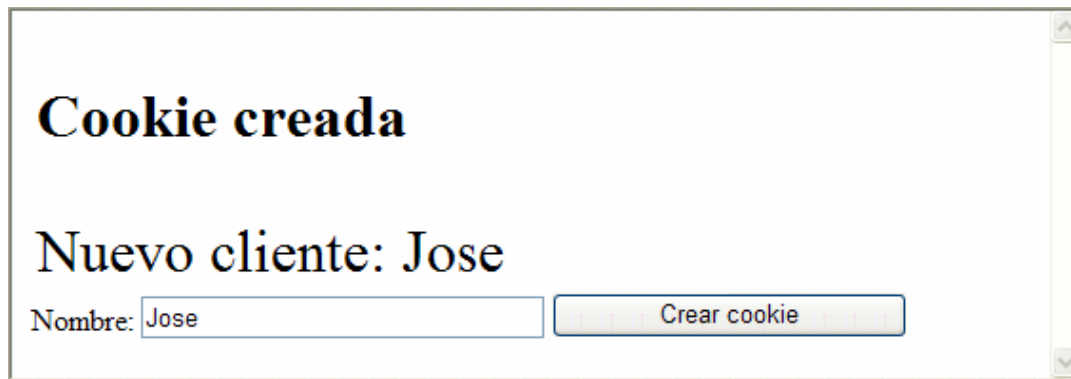
```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim dato_cookie As HttpCookie = Request.Cookies("Preferencias")
    If dato_cookie Is Nothing Then
        Label1.Text = "<b>Nuevo cliente</b>"

    Else
        Label1.Text = "<b>Cookie encontrada.</b><br /><br />"
        Label1.Text &= "Bienvenido, " & dato_cookie("Nombre")
    End If
End Sub
```

Es decir, si no hay ninguna cookie con el nombre "Nombre" en la sección "Preferencias" escribimos "Nuevo cliente" ya que nunca ha tenido una cookie nuestra guardada. Veamos el evento clic del botón por la creación de la cookie:

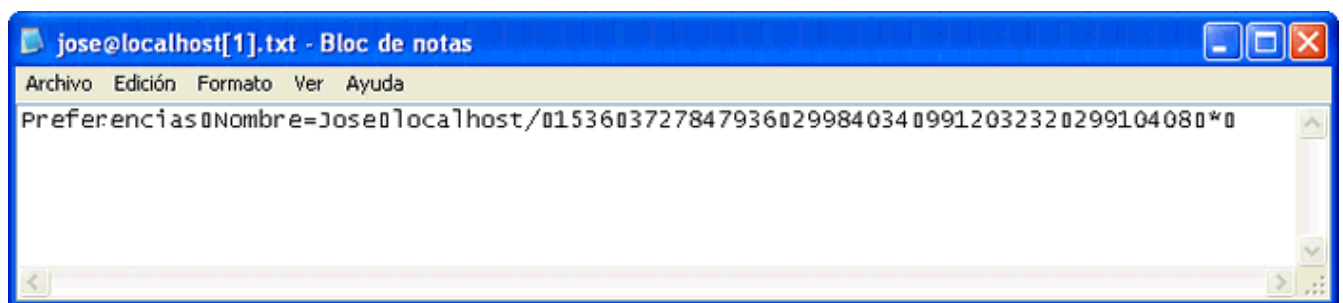
```
Protected Sub btn_crear_Click(ByVal sender As Object, ByVal e As System.E
    Dim dato_cookie As HttpCookie = Request.Cookies("Preferencias")
    If dato_cookie Is Nothing Then
        dato_cookie = New HttpCookie("Preferencias")
    End If
    dato_cookie("Nombre") = txt_cookie.Text
    dato_cookie.Expires = DateTime.Now.AddYears(1)
    Response.Cookies.Add(dato_cookie)
    Label1.Text = "<b>Cookie creada</b><br /><br />"
    Label1.Text &= "Nuevo cliente: " & dato_cookie("Nombre")
End Sub
```

Primero la creamos, luego le asignamos el "nombre" y finalmente le ponemos una vigencia de un año. Finalmente con "response.cookies.add" la añadimos a nuestra colección y se la mostramos en pantalla.



The image shows a web browser window with a form. At the top, it says "Cookie creada". Below that, it says "Nuevo cliente: Jose". There is a text input field labeled "Nombre:" containing the text "Jose". To the right of the input field is a button labeled "Crear cookie".

Para otra vez que visite la página se ejecutará el "Page_Load " que tiene el código para leerla y mostrarla. Si te fijas en la carpeta anterior verás que se ha creado un fichero de texto que al abrirlo nos muestra:



5.6 Sesiones

Ahora vamos con una de las opciones mas interesantes para el almacenamiento de estados y variables en nuestra aplicación web: los estados de sesión y de aplicación. Ya comentamos algo resumido hace no mucho pero ahora lo vamos a ver detenidamente. Digamos que tendremos la opción de crear " varias globales " en memoria y que todas las páginas serán capaces de leerlas. Esto suena muy bien y es muy útil, y si, es fácil y muy práctico, así que vamos a ver lo que es.

Una sesión se inicia cuando un usuario solicita una página y se termina cuando sale de nuestra aplicación web. Así que vamos a poder almacenar datos importantes de esa sesión que va a ser totalmente distinta de la sesión de otro usuario, así que tranquilamente nuestro servidor web puede estar tratando con 100 ó 500 sesiones de usuarios simultáneas, cada una con sus datos y variables.

El estado de aplicación es lo mismo pero sus datos son visibles para todos los usuarios. Por ejemplo un contador de visitas lo declaramos a nivel de aplicación en lugar de a nivel de sesión. Vamos a detallar todo esto y a ver ejemplos:

Estado de la sesión

Tenemos por tanto la necesidad de crear valores persistentes y visibles por las páginas, y además, valores que pueden ser objetos complejos , por tanto no nos vale lo visto hasta ahora. Además por temas de seguridad no queremos utilizar los sistemas conocidos porque no son totalmente inviolables: querystring, cookies,

El estado de la sesión es una de las características mas potentes de ASP.NET ya que nos va a permitir entre otras cosas almacenar esos datos en zonas de memoria del servidor y nunca se transmitirán a otras páginas.

Será desde cada página donde haremos la lectura de estas variables. Cada cliente tiene su propia sesión con su propia colección de información

Identificador de la sesión

Asp.NET identifica cada sesión con un valor único de 120 bits, es lo que llama "track". Utiliza un algoritmo para calcular este valor y es único para cada usuario, este identificador es la única parte que se transmite entre el cliente y el servidor web. El cliente al identificarse con su número de sesión (TD) puede recuperar todos los objetos previamente almacenados en memoria y los coloca en una colección especial de fácil acceso por el código. Puesto que si necesitamos este TD podemos acompañarlo con nuestras páginas de dos formas:

- Utilizando "cookies". El TD de la sesión se transmite en una cookie especial llamada "ASP.NET_SessionId" que ASP.NET crea automáticamente.
- Utilizando URL modificadas. En este caso el TD se transmite con un modificador especial, así las sesiones seguirían funcionando en equipos que no tengan las cookies activadas

Lógicamente la utilización de los estados no es "gratis" para el servidor, ya que debe almacenar un pequeño registro en memoria de cada conexión que realizan los clientes para poder almacenar su número de sesión. Este requerimiento adicional de memoria es pequeño pero hay que tener en cuenta que puede crecer con miles de usuarios simultáneos en nuestra web.

Utilizar el estado "session"

Podemos interactuar con la sesión utilizando la clase "System.Web.SessionState.HttpSessionState" que lo proporciona la página web ASP por medio del objeto ya incorporado "Session". Para añadir un objeto de tipo "session". Es decir, una variable que va a estar visible por todas las páginas de la sesión simplemente debemos hacer:

```
Sessroi("IifoDathaSeth") = dsIifo
```

Y para recuperar el valor en una página

```
dsIifo = CType(Sessroi("IifoDathaSeth"), DathaSeth)
```

Estas variables y el estado de la sesión es global para toda nuestra aplicación web del usuario que está conectado. Pero se destruirán si se produce alguno de estos sucesos:

- Si un usuario cierra y vuelve a abrir el navegador
- Si un usuario accede a la misma página pero desde otra ventana,
- Si la sesión finaliza por que ha expirado el tiempo (timeout). Este dato es variable y si recuerdas se podía poner en las propiedades de las aplicaciones en la configuración del TTS
- Si llamamos al método "Session.Abandon()" para finalizar intencionadamente una sesión. Por ejemplo cuando el usuario pulsa en el típico "desconectar" o "cerrar sesión"

En los dos primeros casos los datos quedan en memoria sin poder acceder a ellos hasta que finalizan por exceso de tiempo o "timeout". Veamos los miembros que tenemos en las sesiones:

Miembro	Descripción
Count	Proporciona el número de elementos en la colección de sesiones actuales
TsCookieless	Identifica si la sesión está siendo controlada por una cookie o por un identificador en la URL

Controles Web, clases y eventos. Estado de la aplicación

TsNewSession	Identifica si la session se ha creado solo para la petición actual
Mode	Devuelve una lista mostrándonos como ASP.NET almacena la información de la session.. Este forma de almacenamiento se define en el fichero de configuración " web.Config
SessionTD	Identificador de la session actual del cliente
Timeout	Número de minutos de vigencia en memoria de la session. Podemos cambiar este tiempo cuando queramos en nuestro programa. Recuerda que hay páginas que visitas y al volver al rato te dice " su sesión ha expirado, debe volver a registrarse ", muy comunes en las páginas de los bancos.
Abandon()	Cancela la session actual y libera la memoria.
Clear()	Elimina todos los datos de la session actual pero no su identificador. Por lo tanto es útil para borrar todas las asignaciones que hemos hecho en memoria en la sesión actual del usuario.

Ejemplo

En este ejemplo crearemos una variable de tipo session con varios datos, construiremos una página con este aspecto:

Un label arriba y debajo una tabla con un cuadro de lista donde meteremos una serie de elementos y a la derecha en la otra columna un botón y otro control de tipo label para escribir el resultado.

En lugar de utilizar valores sencillos vamos a crear un objeto: un CD de música con tres argumentos: nombre, grupo y precio. Así que en nuestra página definimos el objeto:

```
Public Class CD
    Public Titulo As String
    Public Grupo As String
    Public Precio As Double

    Public Sub New(ByVal titulo As String, _
        ByVal grupo As String, ByVal precio As Double)
        Me.Titulo = titulo
        Me.Grupo = grupo
        Me.Precio = precio
    End Sub
End Class
```

Controles Web, clases y eventos. Estado de la aplicación

Como ves hemos utilizado un constructor (New) para crear objetos de este tipo. Los vamos a crear en la creación de la página:

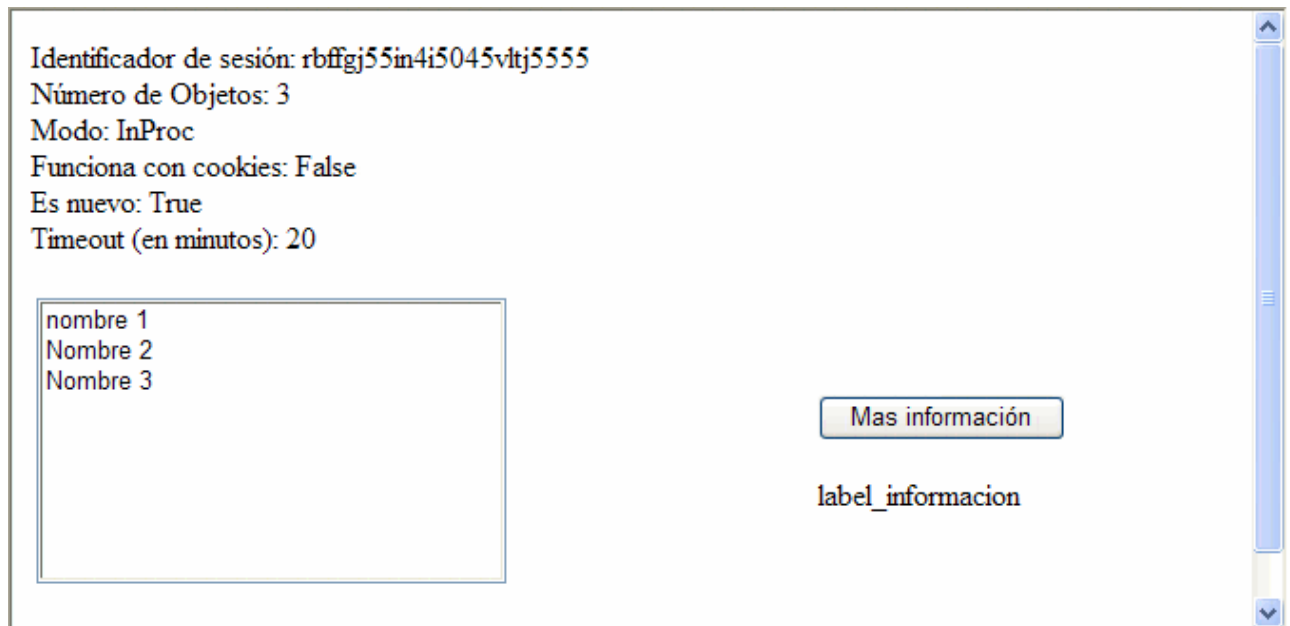
```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    If Me.IsPostBack = False Then
        'Creamos el objeto
        Dim CD_datos1 As New CD("nombre 1", "Grupo 1", 24)
        Dim CD_datos2 As New CD("Nombre 2", "Grupo 2", 16.75)
        Dim CD_datos3 As New CD("Nombre 3", "Grupo 3", 30.2)

        'Añadimos los objetos a la sesión:
        Session("CD1") = CD_datos1
        Session("CD2") = CD_datos2
        Session("CD3") = CD_datos3
    'Añadimos los Cd's al cuadro de lista.
        lista_cds.Items.Add(CD_datos1.Titulo)
        lista_cds.Items.Add(CD_datos2.Titulo)
        lista_cds.Items.Add(CD_datos3.Titulo)
    End If

    'Mostramos información de la session, es interesante para depuración
    label_session.Text = "Identificador de sesión: " & Session.SessionID
    label_session.Text &= "<br />Número de Objetos: "
    label_session.Text &= Session.Count.ToString()
    label_session.Text &= "<br />Modo: " & Session.Mode.ToString()
    label_session.Text &= "<br />Funciona con cookies: "
    label_session.Text &= Session.IsCookieless.ToString()
    label_session.Text &= "<br />Es nuevo: "
    label_session.Text &= Session.IsNewSession.ToString()
    label_session.Text &= "<br />Timeout (en minutos): "
    label_session.Text &= Session.Timeout.ToString()

End Sub
```

Si es la primera vez que se carga la página creamos los tres objetos CD con los parámetros que ves, unos de prueba. Luego mediante "Session" los almacenamos en memoria y los añadimos al cuadro de lista para que seleccione uno el usuario. Debajo escribimos ya información sobre el estado de la sesión del usuario, así que la página ejecutada tendrá ya este aspecto:



Ahora en el evento clic donde se obtiene mas información ejecutaremos lo siguiente:

```
Protected Sub btn_info_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles btn_info.Click
    If lista_cds.SelectedIndex = -1 Then
        label_informacion.Text = "No se ha seleccionado elemento"
    Else
        'Contruimos el nombre según el índice del elemento seleccionado: CD1, CD2, CD3
        Dim Clave As String
        Clave = "CD" & (lista_cds.SelectedIndex + 1).ToString()

        'Recuepramos el objeto seleccionado de la memoria con la sesión
        Dim CD_datos As CD = CType(Session(Clave), CD)

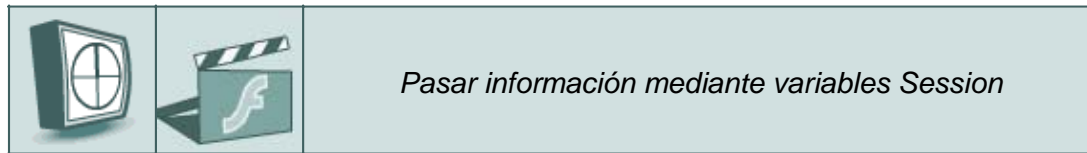
        'Finalmente mostramos la información del objeto:
        label_informacion.Text = "Name: " & CD_datos.Titulo
        label_informacion.Text &= "<br />Grupo: "
        label_informacion.Text &= CD_datos.Grupo
        label_informacion.Text &= "<br />Precio: " & CD_datos.Precio.ToString
    End If
End Sub
```

Utilizaremos el índice del cuadro de lista para saber el elemento que ha seleccionado, así si el índice es 1:

```
Clave =
"CD" & (lista_cds.SelectedIndex + 1).ToString()
```

Nos devolverá CD1, CD2 ó CD3. Y por tanto podemos recuperar los valores de Session ("CD1")

Con lo que hemos conseguido almacenar una serie de valores en memoria y luego recuperarlos. En este caso ha sido en la misma página pero podía haber sido en cualquier otra porque la ventaja de estas variables sesión es que están accesibles en toda la sesión del usuario, y por tanto, en todas las páginas que visita.



Otro ejemplo

Este es muy interesante porque nos va a dar una idea de lo importante que pueden ser estas variables. Vamos a tener una pantalla de inicio de sesión. Si el usuario es correcto le llevará a un menú. ¿Qué pasa si el usuario es un listo y va al menú directamente sin pasar por el registro? Pues en condiciones normales si le funcionaria pero para eso vamos a hacer un control de esta forma. Mostramos la página de login, si es correcto crearemos una variable de tipo "session" con su número de empleado y su nombre y le mandaremos a la página con el menú de opciones de su empresa. En esa página comprobaremos que existen esas variables de tipo session: si existe bien, ha pasado por el login y es correcto y sino existe quiere decir que ha escrito la página directamente, en ese caso le mandaremos a la página de login para que repita la operación de identificarse.

Creamos una página de inicio de sesión:

td.style2	
Nombre de usuario:	<input type="text"/>
Contraseña:	<input type="password"/>
<input type="button" value="Inicar Sesion"/>	

Ojo con el cuadro de texto de la contraseña, debes poner la propiedad de este cuadro de texto "TextMode" a "Password" para que no aparezca cuando se escriba. La segunda página será un menú de opciones:

Bienvenido:	asp:label#Label1
1. Contabilidad 2. Compras 3. Logistica 4. Ventas	

Ahora comprobaremos el usuario en el evento clic del botón de la página de inicio de sesión:

```
Protected Sub btn_inicio_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    'Haríamos la comprobación buscando el usuario en una base de datos
    'pero aquí lo haremos sencillo:

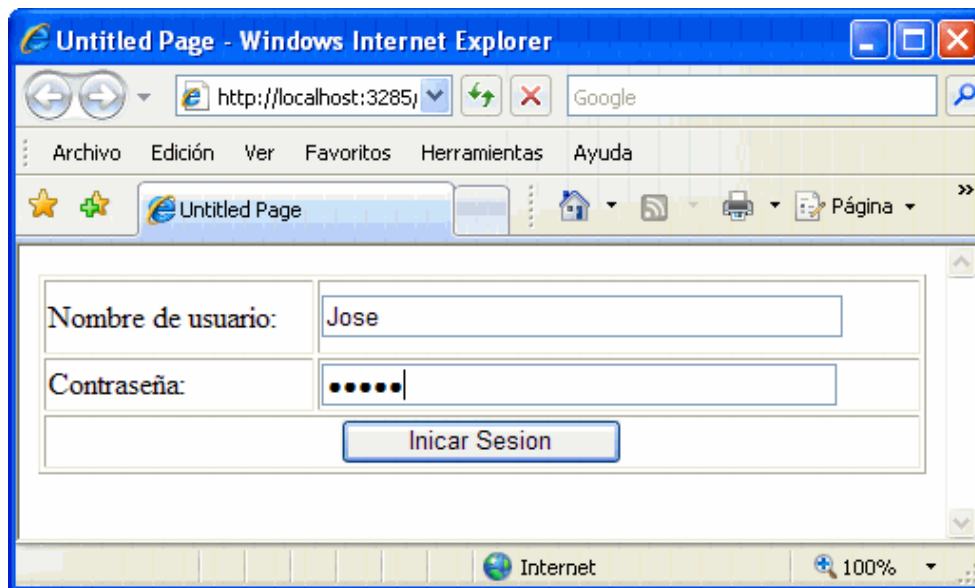
    If txt_usuario.Text = "Jose" And txt_contrasena.Text = "12345" Then
        Session("usuario") = "Jose"
        Response.Redirect("ejemplo_login_menu.aspx")
    End If
End Sub
```

Controles Web, clases y eventos. Estado de la aplicación

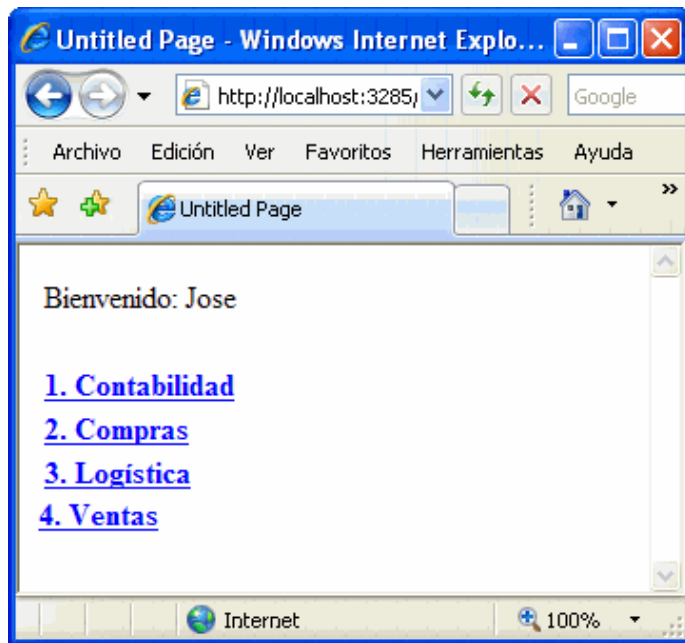
Como ves, si el usuario y contraseña coinciden declaramos una variables de tipo sesión y le mandaremos al menú de opciones. En la página del menú comprobaremos si existe esta variable sesión, sino existe le devolveremos a la pagina de inicio de sesión:

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    If Session("usuario") = "" Then
        Response.Redirect("ejemplo_login.aspx")
    Else
        Label1.Text = Session("usuario")
    End If
End Sub
```

Ejecutamos la página de ejemplo:



Y si los datos son correctos nos mostrará el menú con el nombre del usuario:



Como si existe la página se muestra. Si un listillo quiere poner la página del menú directamente no le funcionará porque al no existir la variable de tipo "session" que se creó al validar el usuario y contraseña se le redirigiremos a la pantalla de inicio de sesión.

Este ejemplo es muy importante porque esta técnica se utiliza mucho y es muy segura ya que al estar las variables en memoria nadie las puede ver.

Configuración de la sesión

Podemos configurar el funcionamiento de la sesión utilizando el fichero "web.config", contenido en el mismo directorio virtual que las páginas aspx. La configuración de este fichero nos va a permitir establecer valores avanzados como el tiempo de vigencia (timeout) de las sesiones y el estado:

```
<?xml versio="1.0" encoding="utf-8" ?>

<configuration>

  <system.web>

    <!-- Other settings omitted. -->

    <sessionState>

      cookieless="UseCookies" cookieName="ASP.NET_SessionId"

      regenerateExpiredSessionId="false"

      timeout="20" mode="InProc"

      stateCollectionMode="ThreadPool"

      networkTimeout="10"
```

Controles Web, clases y eventos. Estado de la aplicación

```
sqlCoiecthroiSthrrig="data source=127.0.0.1;Iithegrathed Securthy=SSPI"

sqlCommaidTrmeouth="30"

allowCusthomSqlDathabase="false"

custhomProvvrder=" "

/>

</systhem.web>

</coifrgurathroi>
```

Expiración de las sesiones

Por defecto ASP.NET permite la reutilización de los identificadores de sesión. Por ejemplo si hacemos una solicitud y nuestra " query string " contiene una sesión caducada ASP.NET crea una nueva sesión con ese identificador. El problema es que si ese identificador se ha visto ya en alguna página podemos perder seguridad ya que alguien podría acceder identificándose con ese TD y por lo tanto estar varios usuarios compartiendo una misma sesión.

Para evitar esta inconsistencia podemos incluir la propiedad opcional regenerateExpiredSessionId asignándola a True para que fuerce a regenerar siempre el identificador.

Tiempo de vigencia

Otro valor importante de la configuración de la session es establecer el tiempo de vigencia, especificando en minutos el tiempo de validez en memoria de la sesión:

```
<sessroiSthathe thrmeouth="20" ... />
```

En el caso de una Intranet, donde el tiempo de uso es elevado ese valor debe ser mucho mas alto para no forzar a identificarse cada cierto tiempo. Por el contrario en sitios web de acceso masivo impactará en el rendimiento si dejamos un tiempo demasiado alto ya que si son muchos usuarios simultáneos serán muchas sesiones de larga duración las que debe mantener en memoria.

De todas forma si en alguna sección de nuestro web quisiéramos variar el tiempo de vigencia, lo podemos hacer con la propiedad del objeto " Session " :

```
Session.Timeout = 10
```

Modo

Existen distintos modos de funcionamiento para el tratamiento de las sesiones que hace ASP.NET.

1. InProc: Es el modo por defecto. El estado de la sesión se almacena en la memoria del servidor web y es el que ofrece mejor rendimiento, pero como desventajas se pueden destacar que no se persiste si reinicias la aplicación web o a través varios servidores(Web Farm).
2. State Server: El estado de la sesión se almacena en un servicio llamado ASP.NET State Service y el estado de la sesión persiste aunque reinicies la aplicación o a través de varios servidores (Web Farm) pero ofrece menor rendimiento que el modo InProc
3. SQL Server: El estado de la sesión se almacena en una base de datos de SQL Server. En las mismas condiciones de Hardware, ofrece menor rendimiento que State Server pero ofrece una mejor integridad de los datos y reporting. Modo SQLServer, que almacena el estado de sesión en una base de datos de SQL Server. Este modo garantiza que el estado de

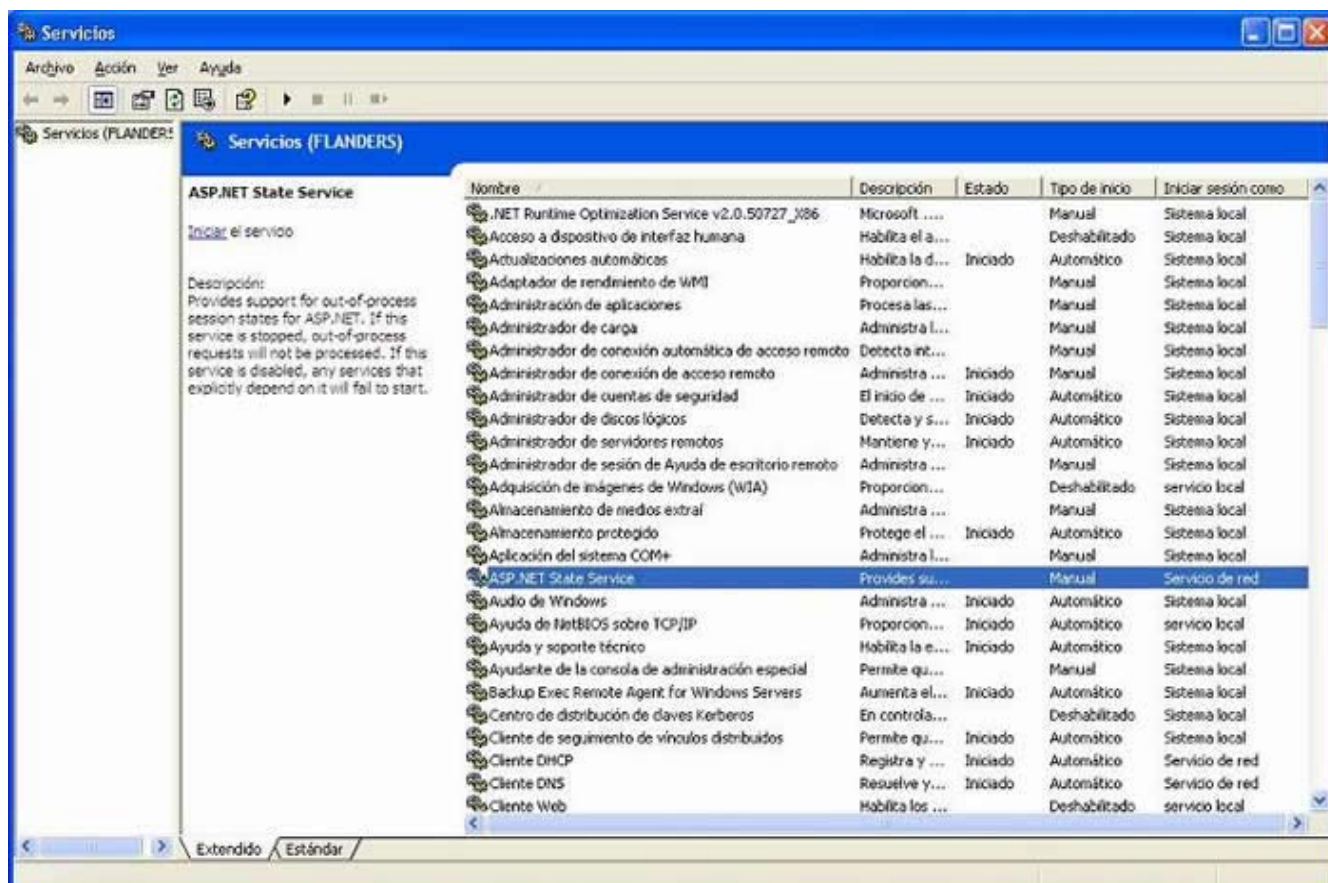
Controles Web, clases y eventos. Estado de la aplicación

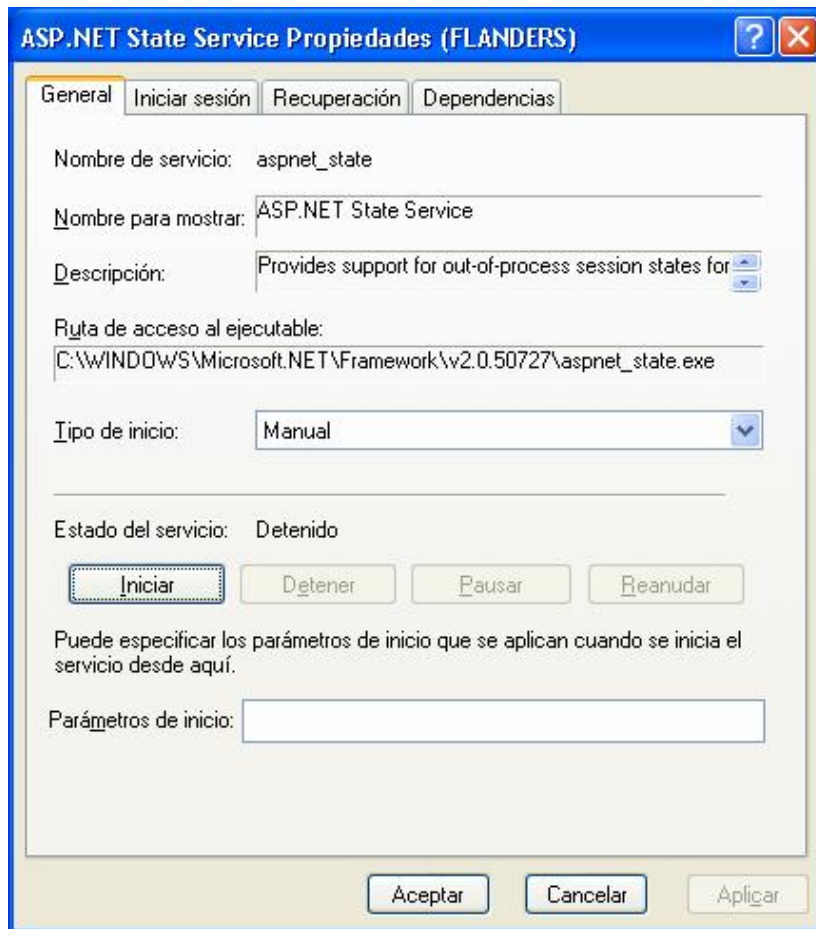
sesión se mantiene si se reinicia la aplicación Web y que esté disponible también para varios servidores Web en una batería de servidores Web

4. Custom: Permite especificar tu propio proveedor de almacenamiento de la sesión, eso si, necesitas implementarlo tú.
5. Off: Deshabilitar el estado de la sesión. Si en tu aplicativo no usas sesión, deshabilitalo para un mejor rendimiento.

Modo StateServer

Con este valor ASP.NET utiliza un servicio de Windows y además necesitaremos especificar un valor para stateConnectionString que identifica la dirección IP del equipo que está ejecutando el servicio StateServer y su puerto. Normalmente se utiliza el mismo servidor pero dado que puede ser un sitio web de muchos accesos simultáneos y para no impactar en la carga del equipo podemos “ sacarlo ” a otro servidor de esta forma. En el panel de control, herramientas administrativas tenemos los servicios de Windows:





SQLServer

Este sistema utiliza SQL Server para almacenar la información de los estados conectándose al servidor mediante el atributo `sqlConnectionString`. Obviamente necesitaremos un servidor SQL Server en nuestra red. La configuración de la cadena de conexión es la habitual en ADO.NET (que veremos mas adelante). Este método es muy avanzado ya que necesita configurar varias cosas como SQL Server, ejecutar procedimientos almacenados y alguna cosa mas por lo que considero que escapa a nuestras pretensiones del curso de inicio a ASP.NET. De todas formas os pongo un enlace por si alguno queréis mas información de este sistema y que por supuesto yo os ayudaré en tutorias si hace falta.

[http://msdn2.microsoft.com/es-es/library/ms178586\(VS.80\).aspx](http://msdn2.microsoft.com/es-es/library/ms178586(VS.80).aspx)

5.7 Estado de la aplicación (Application)

Asi como la sesión almacena el estado del usuario, el estado "application" almacena objetos globales para todos, es decir todos los usuarios conectados tienen acceso a los objetos de alcance "application". Este estado se basa en la clase `System.Web.HttpApplicationState` proporcionada por el objeto `Application`. El funcionamiento es prácticamente igual al que hemos visto en la sesión. ¿utilidades? Pues por ejemplo un contador con el número de visitas.

Como se debe incrementar con cada usuario podemos crear un fichero de configuración del tipo "global.asax" que como recuerdas puede definir también variables de tipo sesión y almacenar

procedimientos.

O podemos utilizar el "Page_Load" para realizar este contador, por ejemplo

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

    Dim Counter As Integer = CType(Application("Counter"), Integer)

    Counter += 1

    Application("Counter") = Counter

    lblCounter.Text = Counter.ToString()

End Sub
```

Este objeto no se utiliza mucho porque normalmente no necesitamos declarar objetos a nivel global. Además la eterna cuestión en la concurrencia: ¿qué pasa si dos usuarios acceden a la vez a la misma variable para modificarla? No es tan descabellado que suceda esto para incrementar el contador de visitas por ejemplo. En estos casos de concurrencia tendríamos que bloquear esa variable para que lo la modifique nadie con los métodos "Lock" y "Unlock" :

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

    ' Activa acceso exclusivo

    Application.Lock()

    Dim Counter As Integer = CType(Application("Counter"), Integer)

    Counter += 1

    Application("Counter") = Counter

    ' Libera la variable

    Application.Unlock()

    lblCounter.Text = Counter.ToString()

End Sub
```

Y los otros usuarios deben esperar hasta que esté desbloqueada para poder acceder a ellas. Así que como ves es poco eficaz, antes se utilizaba porque no había otra forma de almacenar variables globales de configuración de una base de datos, por ejemplo. Ahora disponemos de otras formas para almacenar estos valores como son los sencillos ficheros web.config, que son más flexibles y fáciles de utilizar. Así que no seguimos con este tipo de almacenamientos de estado porque este Application es poco eficaz y en casos de concurrencia problemático.

¿Que tipo de almacenamiento de estado debo utilizar?

La respuesta no es clara, depende del momento y del tipo de proceso. EL querystring es muy sencillo y funciona bien pero no podemos valores personales. En este caso parece que los estados "viewstate" codificados son mejor solución. Ya la utilización de variables de tipo "Session" nos facilitarán otro tipo de labores para tener datos globales en nuestro web. Así que será una combinación de ellos según el tipo de proceso que queramos realizar. La experiencia nos dirá que sistema nos ha ido mejor en cuanto a eficacia y sencillez, pero una pequeña tabla con un análisis de ellos tampoco nos viene mal para recordar:

"ViewState"QueryStringCookiesSessionApplicationTipos de datos permitidosTodos los tipos de datos de ASP.NETCantidad limitada de "string"Datos "string"Todos los tipos de datos de ASP.NET

Todos los tipos de datos de ASP.NET

AlmacénCampo oculto en la páginaValores en la URLEquipo del clienteMemoria del servidor, servicio del servidor o SQL ServerMemoria del servidorDuraciónDurante el "postback" a otra páginaCuando el usuario cierra el navegador o visita otra páginaEstablecido por el clienteTimeout predefinido a 20 minutos y modificableTiempo de vida de la aplicación normalmente hasta el reinicio.ÁmbitoLimitado a la página actualLimitado a la página destinoToda la aplicación ASP.NETToda la aplicación ASP.NETToda la aplicación ASP.NETSeguridadLimitada, hay que encriptarla con la propiedad adecuada.Visible y fácilmente modificableInsegura y modicable por el usuarioMuy seguro porque no se transmiten al clienteMuy seguro porque no se transmiten al clienteImplicaciones de rendimientoLento si es mucha información, pero no afecta al rendimiento del servidorNingunoNingunoLento con mucha información para cada cliente.Lento con mucha información para cada cliente.Uso habitualValores de páginasDetalles de consultas y catálogosPreferencias de personalización del sitio webUna cesta de la compra en una tiendaAlmacenamiento de datos globales.

Ejercicios

Ejercicio 1

Crea un generador de tablas. Que pida los siguientes datos en pantalla y que la genere. Pondremos un control de tabla en la página: `<asp:Table id="tabla" runat="server" />`

El objeto a crear es un TableRow con las TableCell o celdas correspondientes y escribe dentro la fila y columna que son. Los datos de solicitud se pondrán en una tabla que estará a su vez dentro de un control de tipo "Panel".

asp:Panel#Panel1

Número de fila:

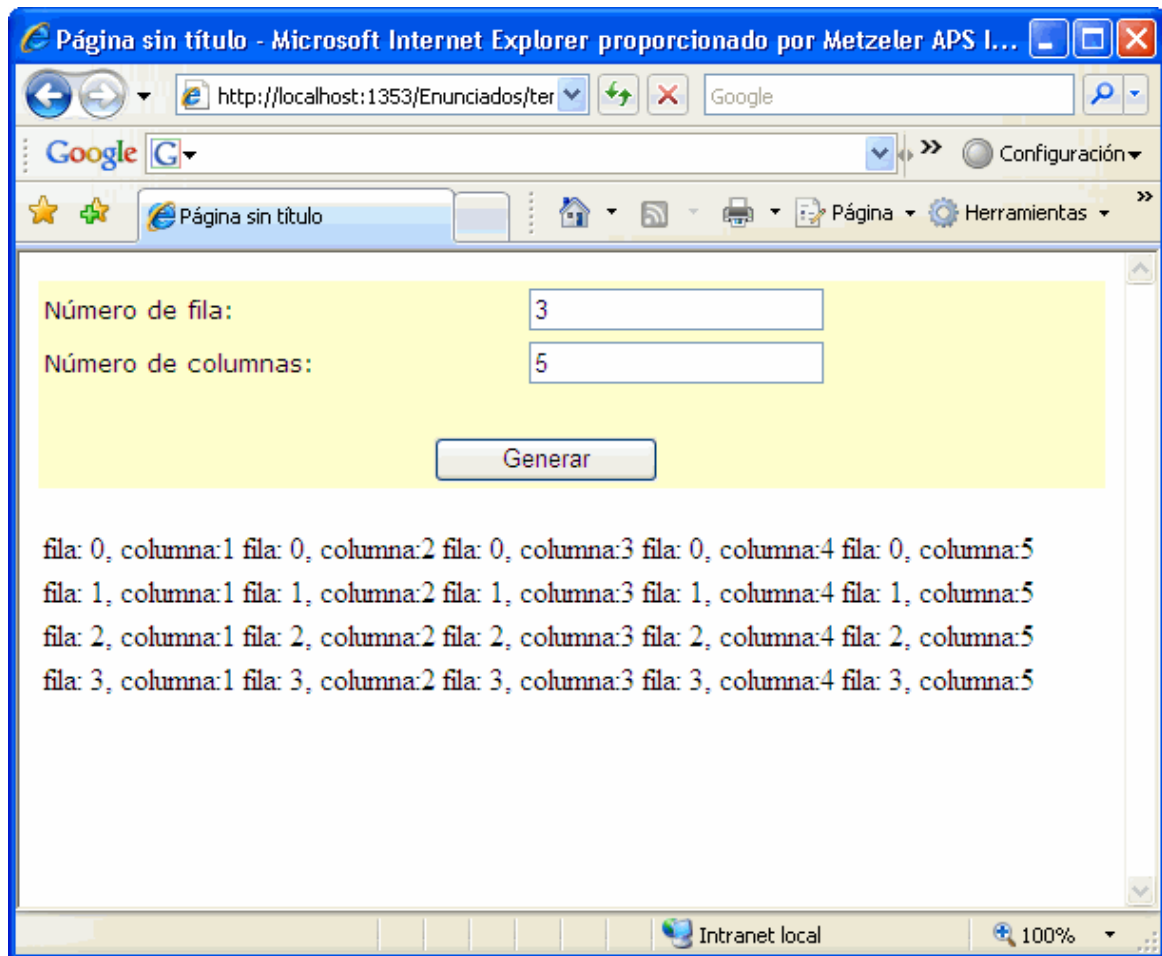
Número de columnas:

Generar

asp:Table#Table1

###		

Que dará como resultado:



Inicialmente las celdas no tienen contorno, ahora seguiremos con la práctica.

Ejercicio 2

En el evento adecuado haz la carga de las colecciones de contornos, colores de los contornos y color de la tabla. Tendrás que importar:

```
Imports System.Drawing
Imports System.Drawing.Text
```

Para poder acceder a las fuentes y los colores. Por ejemplo la carga de colores será:

```
Añadimos los colores:
Dim matriz_colores As String() = System.Enum.GetNames(GetType(KnownColor))
lista_colores.DataSource = matriz_colores
lista_colores.DataBind()
```

La carga de las fuentes la tienes en el ejemplo que hicimos en el capítulo.

Controles Web, clases y eventos. Estado de la aplicación

asp:Panel#Panel1

Número de fila:

Número de columnas:

Tipo de contorno: Grosor controno:

Color del contorno:

Tipo de letra: Tamaño de fuente:

Color de fondo de la tabla:

###

Ejecutando nos debería dar:

Página sin título - Microsoft Internet Explorer proporcionado por Metzeler APS Ibérica S.A.

http://localhost:1353/Enunciados/tema_7/Ejercicio2.aspx

Número de fila:

Número de columnas:

Tipo de contorno: Grosor controno:

Color del contorno:

Tipo de letra: Tamaño de fuente:

Color de fondo de la tabla:

fila: 0, columna:1	fila: 0, columna:2	fila: 0, columna:3	fila: 0, columna:4
fila: 1, columna:1	fila: 1, columna:2	fila: 1, columna:3	fila: 1, columna:4
fila: 2, columna:1	fila: 2, columna:2	fila: 2, columna:3	fila: 2, columna:4
fila: 3, columna:1	fila: 3, columna:2	fila: 3, columna:3	fila: 3, columna:4

Listo

Intranet local

100%

La opción del contorno es nueva y algo compleja, no la implementes y luego te fijas en el código. Ya que debemos sacar el valor al que corresponde cada elemento de la enumeración.

Ponle un contorno fijo con: `celda_nueva.BorderStyle=BorderStyle.Solid`

Ejercicio 3

Pon controles de validación para que no se pueda meter mas que valores numéricos entre el 1 y el 10 en las solicitudes. Pon el "autopostback" en el cuadro desplegable del color de fondo de la tabla y un botón para que el panel de solicitud de datos se pueda mostrar u ocultar. El botón tendrá un texto dependiendo de si está visible u oculto el panel:

Ponemos el botón:

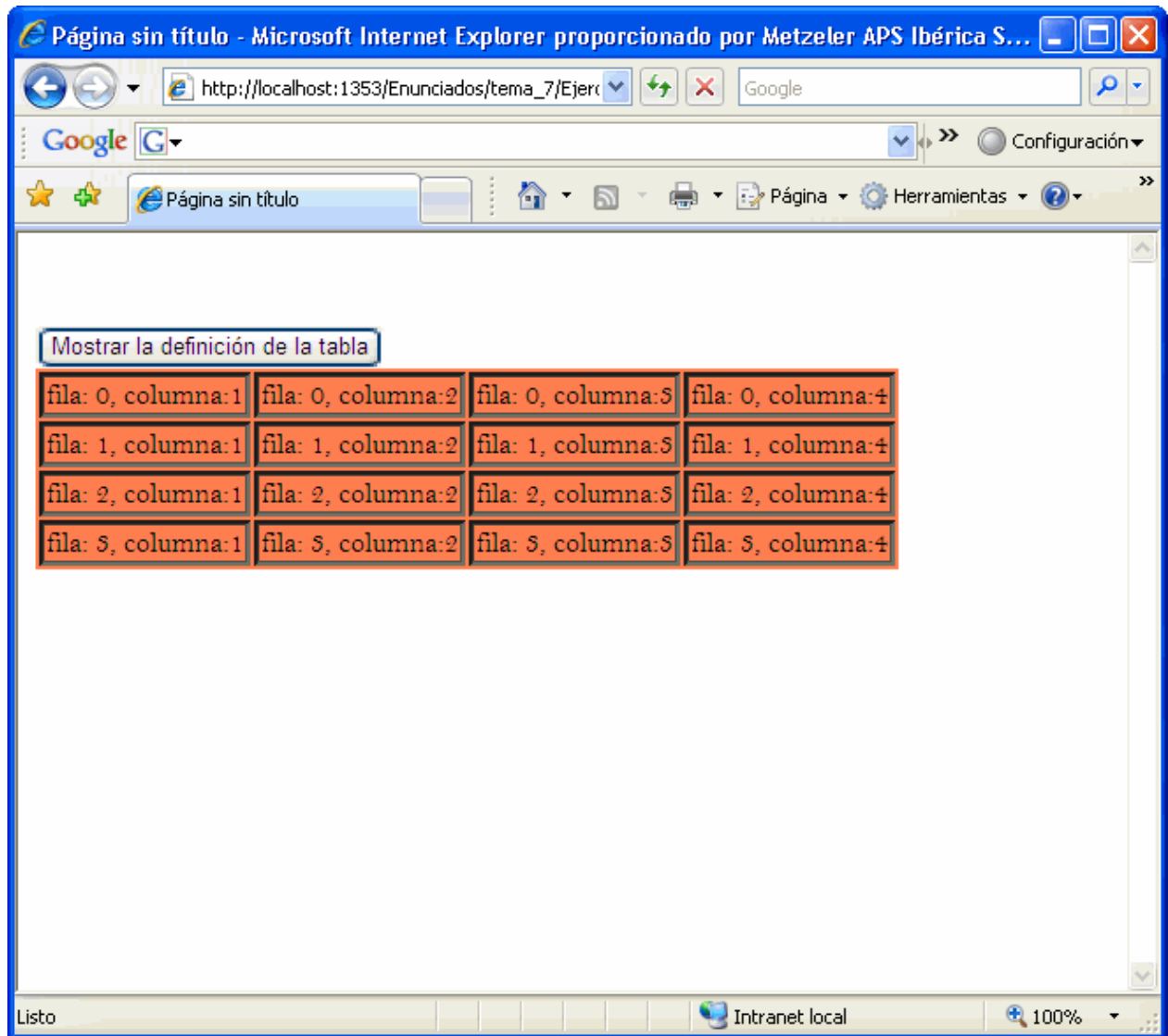
The screenshot shows a web application interface for configuring a table. The form includes the following controls:

- Número de fila:** Text input with value 3.
- Número de columnas:** Text input with value 4.
- Tipo de contorno:** Dropdown menu with value 'Inset'.
- Grosor controno:** Spin box with value 3.
- Color del contorno:** Dropdown menu with value 'ControlDarkDark'.
- Tipo de letra:** Dropdown menu with value 'Bell MT'.
- Tamaño de fuente:** Spin box with value 12.
- Color de fondo de la tabla:** Dropdown menu with value 'Coral'.
- Generar:** Button to generate the table.

Below the form, there is a button labeled 'Ocultar definición de tabla'. Below this button is a generated table with 4 rows and 4 columns. Each cell in the table contains text indicating its row and column indices, such as 'fila: 0, columna:1'.

fila: 0, columna:1	fila: 0, columna:2	fila: 0, columna:3	fila: 0, columna:4
fila: 1, columna:1	fila: 1, columna:2	fila: 1, columna:3	fila: 1, columna:4
fila: 2, columna:1	fila: 2, columna:2	fila: 2, columna:3	fila: 2, columna:4
fila: 3, columna:1	fila: 3, columna:2	fila: 3, columna:3	fila: 3, columna:4

Y al pulsarlo se oculta y cambiamos el texto del botón:



Recuerda que pusimos la tabla dentro de un panel, luego con mostrar/ocultar éste nos valdría. Además como necesitaremos pintar la tabla después de ese clic del botón seguramente tengas que definir en un procedimiento todo el proceso de generación de la tabla que pusiste en el "Load" de la página.

Finalmente controlaremos los errores de los cuadros de texto.

Página sin título - Microsoft Internet Explorer proporcionado por Metzeler APS Ibérica S.A.

http://localhost:1353/Enunciados/tema_7/Ejercicio3.aspx

Google

Google

Página sin título

Número de fila: wfref El dato no está en el rango (1-10)

Número de columnas: El dato no está en el rango (1-10)

Tipo de contorno: NotSet Grosor controno: 1

Color del contorno: ActiveBorder

Tipo de letra: 3 of 9 Barcode Tamaño de fuente: 8

Color de fondo de la tabla: ActiveBorder

Generar

Ocultar definición de tabla

Intranet local 100%

Nota: Para probar la validación utiliza solo el botón de Generar